

Sylvain Besençon

Les temps de la sécurité informatique

**Une anthropologie du soin du
protocole cryptographique
OpenPGP**



DOI : <https://doi.org/10.51363/unifr.lth.2025.052>

Sylvain Besençon  <https://orcid.org/0000-0001-9321-2237>

© Sylvain Besençon, 2025



This work is published under the Creative Commons Attribution 4.0 International license (CC BY 4.0): <https://creativecommons.org/licenses/by/4.0/>

THÈSE DE DOCTORAT
PRÉSENTÉE DEVANT LA FACULTÉ DES LETTRES ET DES SCIENCES HUMAINES
DE L'UNIVERSITÉ DE FRIBOURG (SUISSE)

**Les temps de la sécurité informatique :
une anthropologie du soin
du protocole cryptographique OpenPGP**

Sylvain Georges Besençon

Ballaigues, VD – Suisse

Identifiant de la clé OpenPGP
9584 92A9 80DC C8FE F5C5
EE91 56F7 ACD4 B1FD E56F



SOUS LA DIRECTION DE

Prof. Dr. DAVID BOZZINI

Université de Fribourg, Suisse

ET DE

Dr. FRANCESCA MUSIANI

Centre national de la recherche scientifique (CNRS), France

APPROUVÉ PAR LA FACULTÉ DES LETTRES ET DES SCIENCES HUMAINES SUR PROPOSITION DE
PROF. DR. DAVID BOZZINI, DR. FRANCESCA MUSIANI ET PROF. DR. JÉRÔME DENIS

FRIBOURG, LE 11 JANVIER 2024

LE DOYEN PROF. DR. DOMINIK SCHÖBI

*Tout ce que tu touches,
Tu le changes.
Tout ce que tu changes,
Te change.
La seule vérité permanente
Est le Changement. [...]*

*Octavia Estelle Butler, La parabole du semeur
(2020 [1993]. Vauvert : Édition Au Diable Vauvert.
Traduction : Philippe Rouard)*

L'illustration de la couverture est un dessin de Léa Kissling (@lea_kissling_officielle).

Déclaration sur l'honneur

N.B. Cette attestation ne remplace pas l'attestation officielle signée de ma main et déposée en même temps que ma thèse. Il s'agit d'un exemple didactique de signature numérique grâce à OpenPGP.

Pour vérifier cette signature, il faut télécharger ma clé publique sur un serveur de clé (par exemple <https://keys.openpgp.org>) et utiliser une implémentation d'OpenPGP pour vérifier cette signature (<https://www.openpgp.org/software/>).

Identifiant de la clé OpenPGP :

sylvain.besencon@unifr.ch
9584 92A9 80DC C8FE F5C5
EE91 56F7 ACD4 B1FD E56F

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA256

Par ma signature électronique, j'atteste avoir rédigé personnellement cette thèse sans aide extérieure non autorisée, n'avoir utilisé que les sources et moyens autorisés, et mentionné comme telles les citations et paraphrases. Cette thèse n'a pas déjà été présentée devant une autre faculté.

J'ai pris connaissance que le comportement scientifique délictueux selon les directives de l'Université de Fribourg* est sanctionné.

Nom : Besençon

Prénom : Sylvain Georges

Lieu et date : Fribourg, le 15 septembre 2023

-----BEGIN PGP SIGNATURE-----

iHUEARYIAB0WISVhJKpgNzI/vXF7pFW96zUsf3lbwUCZQOD2wAKCRBW96zUsf3l
b6/nAQcwBBR+PTOLi29c8GCK5/jW0z2affDtxcu/HxKU5Z5aFgEArjRVEsVPz9u1
YsqPM6j38F9SHcUn06AnsPArhrTYBQs=
=HI1h

-----END PGP SIGNATURE-----

Résumé

Cette thèse en anthropologie sociale analyse les pratiques de soin autour du protocole cryptographique OpenPGP. OpenPGP est un protocole de chiffrement asymétrique utilisé principalement pour assurer la confidentialité du courrier électronique et authentifier des données numériques. Au cœur de nombreuses controverses depuis des années, notamment liées à son ancienneté et sa complexité d'usage, il est néanmoins toujours utilisé car difficilement remplaçable pour certaines tâches. Cette thèse analyse comment de nombreuses personnes agissent afin de faire durer dans le temps la sécurité que cherche à offrir ce protocole, ce qui implique non seulement de s'accommoder de l'héritage du passé, mais aussi de (p)réparer l'avenir.

Basée sur une étude ethnographique de plusieurs années auprès d'équipes de développement et de recherche, cette thèse analyse la production sociale de la sécurité informatique en l'analysant au prisme du temps. Plus précisément, elle questionne les trois dimensions temporelles de la sécurité informatique que sont la transformation, le rythme et la durée. Elle propose une réflexion sur le vieillissement des technologies du numérique et sur les pratiques de soin à court et long termes des normes de sécurité informatique.

Cette thèse s'insère dans les champs d'études des sciences et des techniques et de l'anthropologie de la sécurité auxquels elle contribue en proposant plusieurs notions novatrices. Tout d'abord, dans une perspective inspirée de l'anthropologie de la sécurité et en particulier des approches constructivistes de la sécurisation, l'analyse propose les notions de « processus d'(in)sécurisation » et de « projet-sécurité ». Les processus d'(in)sécurisation soulignent la constante oscillation entre sécurité et insé-

curité. Cela amène à considérer la sécurité informatique non pas en tant que pure propriété technique mais en tant que processus relationnel. La notion de « projet-sécurité » permet d'approfondir cette dimension en axant l'analyse autour de ce qui est fait au nom de la sécurité. La sécurité peut en effet être considérée comme un projet, un objectif vers lequel on tend, tout en sachant qu'il ne pourra jamais être atteint de manière définitive. Finalement, la thèse suggère également la notion de « résilience prospective » afin de rendre compte de la constante projection dans l'avenir qu'implique le travail de soin.

Cette approche constructiviste, processuelle et relationnelle de la sécurité est finalement renforcée par une perspective transversale inspirée des approches féministes des sciences et des techniques. En effet, ce travail propose de reconsidérer la manière dont on parle de sécurité informatique : plutôt que d'utiliser un champ lexical viril et guerrier pour parler de sécurité de manière dichotomique, cette thèse l'appréhende à travers le vocabulaire du soin, de la collaboration et du compromis.

Après l'introduction (chapitre 1) et une section méthodologique (chapitre 2), le troisième chapitre présente différentes approches de la sécurité, tant du point de vue des personnes qui cherchent à la produire que des scientifiques qui décrivent leur travail. Suivent ensuite trois parties de chacune deux chapitres qui forment le cœur de l'analyse : la première analyse l'influence du temps en tant que facteur de transformation (première partie) et questionne le vieillissement des technologies numériques. La deuxième partie est tournée vers les rythmes de la sécurité, qu'il s'agisse de situations d'urgence ou de lentes négociations. La troisième partie est quant à elle tournée vers l'avenir et explore les durées de la sécurité. Une conclusion revient sur les différents apports et souligne cinq valeurs centrales analysées dans cette thèse : la collaboration, la compatibilité avec le passé, les opportunités de transformation, le soin, et le compromis.

Remerciements

Merci à mon Pruneau Soulmète d'avoir partagé avec moi l'amour des chiffres, des acronymes et des cryptiques cartes postales.

Merci à mon Corbeau de m'avoir inspiré de ses glauques et néanmoins solaires croassements.

Merci à mon Baraki, grand maître des potions herbeuses et fin psychologue.

Merci à mon Petit Poulet pour sa grande sagesse et son énergie de lion.

Merci à Léane pour sa science inébranlable et sa joie ineffable, et à ses parents Cécile et Martel pour leur accueil inconditionnel et leur amitié si inspirante.

Merci à Masette (aka Jipépé) de m'avoir tenu compagnie durant toutes ces années.

Merci à David et Francesca, pour leur soutien, leur expertise, leurs précieux conseils, leur patience, leur confiance, la liberté qu'il et elle m'ont accordée, leur amitié.

Merci à toutes les personnes qui ont partagé avec moi leur temps, leurs connaissances et leur amour (ou haine) d'OpenPGP, et particulièrement à Anastasiia, aux différents Daniels, Damian, Heiko, Hernani, Holger, Justus, Lara, Markus, Neal, Nora, aux deux Pauls, Phil, Robin, Sebastian, Sofia, Stephen, Vincent, Werner, Wiktor. Merci à elles et eux de faire en sorte qu'OpenPGP survive, encore et toujours.

Merci à Phil d'avoir inventé PGP, et à Joe de l'avoir poussé à publier le code en open source dans l'urgence.

Merci à Ben de m'avoir aidé à m'orienter (concrètement et métaphoriquement) dans les sombres couloirs du CCC et des universités suédoises.

Merci à Léa (@lea_kissling_officielle) pour ses dessins.

Merci à mes collègues de hier et d'aujourd'hui.

Merci aux membres du GRED de leur soutien.

Merci au personnel administratif et logistique de l'Université de Fribourg, sans qui la recherche et l'enseignement ne seraient pas concevables.

Merci à mon maître Jedi. Que la force soit avec lui.

Merci aussi à Qnu, Jeanne et Jean.

Merci à Clem, Chloé, Élodie, Estelle, Felsenkirch, Filippo, France, Francine, iCoco, Giorgio, Matthieu., Merci à mes géniteurices, mes consanguines et leur descendance. Merci aussi aux porteuses de prénoms en A qui m'ont soutenu vaillamment : Carlota, Chiara, Fiona, Francesca, Krittika, Nina, Laura, Louella (la surfeuse), Luca, Melina, Selena, Olga, et le couple Jüggä et Veronika.



Merci aussi aux abeilles pour le bon miel et les jolies fleurs. Sans elles, je n'aurais pas non plus terminé. (Source : photo personnelle.)

Table des matières

DÉCLARATION SUR L'HONNEUR.....	2
RÉSUMÉ.....	3
REMERCIEMENTS.....	5
TABLE DES MATIÈRES.....	7
GLOSSAIRE.....	11
LISTE DES ABRÉVIATIONS.....	15
1. INTRODUCTION.....	19
1.1. La sécurité informatique, la cryptographie et OpenPGP.....	21
Fonctionnement de la cryptographie asymétrique.....	25
Les usages d'OpenPGP.....	26
1.2. L'émergence des revendications de droit à la vie privée numérique et l'imaginaire « crypto »...	29
1.3. OpenPGP, une chose fragile et frêle plutôt qu'un solide cadenas.....	34
1.4. Prendre soin de la sécurité informatique.....	43
1.5. Interroger le mouvement, le rythme et la durée de la sécurité.....	48
Panorama de la communauté OpenPGP.....	52
2. ETHNOGRAPHIE DE L'ÉCOSYSTÈME OPENPGP.....	57
Les mots pour le dire : norme, protocole, code, programme.....	61
2.1. Du village à l'écosystème OpenPGP.....	63
2.2. Les différentes communautés d'OpenPGP.....	67
Quatre communautés de pratique de l'écosystème OpenPGP.....	68
2.3. La première phase de ma recherche : <i>lightshows</i> , ateliers de crocheting et discussions assourdissantes.....	71
2.3.1. Les conférences de sécurité informatique.....	72
2.3.2. Une exploration par tâtonnements.....	77
2.3.3. Les données produites lors de la phase exploratoire.....	78
2.4. La deuxième phase : visioconférences et après-midi « piscine ».....	80
2.4.1. Du côté de l'IETF.....	81
2.4.2. Du côté de chez Sequoia.....	83
2.4.3. Les entretiens.....	86
2.4.4. L'éthique de la recherche.....	88

2.4.5. Les données produites lors de la deuxième phase.....	91
2.5. La troisième phase : désécriture et air iodé.....	94
2.5.1. Les processus d'analyse.....	96
2.5.2. De crises en crises : la banalité du quotidien.....	98
3. ALICE ET BOB AUX PAYS DES BOGUES.....	103
3.1. De la CIA aux multiples traductions successives.....	107
La triade CIA.....	107
3.1.1. Premier niveau de traduction : établir un protocole de sécurité.....	109
3.1.2. Deuxième niveau de traduction : du protocole au code.....	114
3.1.3. Troisième niveau de traduction : du code aux usages.....	118
3.2. Des processus d'(in)sécurisation et de soin au cœur de la machine.....	121
3.2.1. Les processus d'(in)sécurisation.....	123
3.2.2. La sécurité comme objet de sollicitude.....	126
3.3. La sécurité comme projet : une attention tournée vers le devenir.....	130
3.3.1. Le projet-sécurité pour penser les temps de la sécurité informatique.....	133
3.3.2. Le projet-sécurité et le temps.....	138
3.3.3. Note sur l'inquiétude d'Alice et Bob.....	143
PREMIÈRE PARTIE : L'HÉRITAGE DU PASSÉ EN MOUVEMENT.....	145
4. ENLISEMENT ET ANCRAGE INFRASTRUCTURELS.....	149
4.1. L'importance infrastructurelle d'OpenPGP.....	152
4.1.1. Les infrastructures numériques.....	156
Les neuf propriétés des infrastructures du point de vue d'OpenPGP.....	157
4.1.2. Le Web of Trust et l'héritage du passé.....	164
4.1.3. Travailler avec le passé, travailler en dépit du passé : remplacer SHA-1.....	167
4.2. L'enlèvement infrastructurel.....	169
4.2.1. L'usure et la lenteur.....	171
4.2.2. Faut-il remplacer OpenPGP et créer une nouvelle norme ?.....	172
4.3. Ancrage et opportunités.....	175
4.3.1. Symbole et obligation morale.....	176
4.3.2. Évolution plutôt que révolution.....	177
4.3.3. Ancrage et innovation.....	179
Exemples d'innovation dans l'infrastructure du Web of Trust.....	179
5. HÉRITER, FAIRE VIEILLIR ET TRANSMETTRE LE CODE.....	185
5.1. Composer le code source.....	189
Anatomie d'un module écrit en RUST : exemple de la fonction parse_iso8601.....	192
5.1.1. La méticulosité des ordinateurs.....	195
5.1.2. Le choix du langage informatique.....	197
5.1.3. Le choix d'une librairie cryptographique.....	201
5.2. La complexité du code, de l'incompréhension au dialogue.....	203
5.2.1. L'inévitable complexité et les petits poils qui poussent sur le code.....	204
5.2.2. Le passé comme source d'incompréhension.....	207
5.3. Du vieillissement à l'ancrage.....	210
DEUXIÈME PARTIE : LES RYTHMES DU PRÉSENT.....	215
Faire et défaire la sécurité en identifiant et patchant des failles : le paradigme de la sécurité par l'insécurité.....	219

6. EFAIL ET LE TEMPS DE L'URGENCE.....	225
6.1. EFAIL, un (contre-)exemple extraordinaire.....	228
6.1.1. La période de gestation d'EFAIL.....	228
6.1.2. L'accouchement d'EFAIL et la stupéfaction générale.....	232
6.1.3. La controverse d'EFAIL dans la peur, l'incompréhension et la colère.....	239
Les principaux points contentieux de la controverse d'EFAIL.....	241
6.2. L'incertitude comme moteur de la controverse, et le sentiment d'urgence comme combustible	245
6.2.1. L'assignation des responsabilités des différentes actrices.....	247
6.2.2. Le savoir et le non savoir dans la gestion de la controverse.....	250
7. LENTEUR ET CONSENSUS À L'IETF.....	253
7.1. Le groupe de travail OpenPGP en prise avec EFAIL.....	256
7.1.1. L'identification d'un des problèmes d'EFAIL : la <i>chunk size</i> de l'AEAD.....	258
L'AEAD et l'authentification des données chiffrées.....	260
7.1.2. Un vieux problème inscrit dans l'histoire d'OpenPGP.....	262
7.1.3. Négocier les propriétés de l'AEAD, encore et toujours.....	265
7.2. Prendre soin du groupe de travail OpenPGP.....	268
7.2.1. La refonte du groupe de travail.....	270
7.2.2. Nouveau souffle, nouvel essoufflement, et nouvelles solutions.....	275
7.2.3. Trois sources de blocages.....	278
7.2.4. La bureaucratie au service des normes techniques.....	282
7.3. La persistance d'OpenPGP.....	284
7.3.1. Un état de panne compensé.....	286
7.3.2. Le temps long des vulnérabilités : à propos du point commun entre la Belle au bois dor-	288
mant et les vulnérabilités.....	
7.3.3. La direction du soin : vers la résilience prospective.....	291
TROISIÈME PARTIE : LA DURÉE DE L'AVENIR.....	293
8. PENSER LES FRAGILITÉS À VENIR POUR PANSER L'AVENIR.....	297
8.1. L'innovation est capitale... de même que la maintenance.....	300
8.1.1. L'exemple de la création de PGP pour nuancer l'importance de l'innovation.....	300
8.1.2. « Être à l'avant-garde du progrès »... en changeant un paramètre par défaut.....	303
8.2. OpenPGP en route vers l'avenir.....	307
8.2.1. Documenter le code pour préparer l'avenir.....	309
8.2.2. Prototypes et tâtonnements.....	314
La norme <i>de facto</i> de la confidentialité persistante.....	314
8.3. Le régime futuriste de la maintenance informatique.....	318
9. ÉQUIPER OPENPGP FACE AUX ORDINATEURS QUANTIQUES.....	323
9.1. Une bombe à retardement hors du commun ?.....	325
9.1.1. La catastrophe annoncée de la cryptographie post-quantique.....	327
9.1.2. Une menace si exceptionnelle ?.....	330
9.2. Des préparations sérieuses, dans l'incertitude et les tâtonnements.....	333
9.2.1. S'accorder sur les priorités.....	334
9.2.2. Bricoler et tâtonner en attendant mieux.....	336
9.3. La résilience prospective comme horizon.....	342
La résilience prospective pour penser la sécurité.....	342
9.3.1. Connaître pour être sûr-e ? La préparation de l'avenir.....	343
9.3.2. Vers une nouvelle norme dans les attentes et les besoins des utilisatrices.....	345

9.3.3. Surfer sur la vague de l'innovation.....	348
10. CONCLUSION : PENSER LE COMPROMIS, PANSER LA SÉCURITÉ.....	353
10.1. Les trois dimensions temporelles de la sécurité informatique.....	356
10.2. Mouvement : vieillir n'est pas mourir, c'est survivre.....	358
10.3. Rythme : une construction fragile prête à s'enflammer.....	362
10.4. Vers la résilience prospective.....	365
10.5. Communauté, Compatibilité, Customisation, Care et Compromis.....	368
BIBLIOGRAPHIE.....	373

Glossaire

Algorithme de chiffrement

Un algorithme de chiffrement, aussi appelé **primitive cryptographique** ou simplement **chiffre**, correspond aux opérations qu'il faut réaliser pour chiffrer/déchiffrer une information. Les algorithmes peuvent être très simples et faits de tête, comme la cryptographie par substitution monoalphabétique, qui remplace une lettre par une autre ; ou très complexe, comme les algorithmes de chiffrement asymétriques qui n'utilisent pas la même clé pour chiffrer que pour déchiffrer.

Authentification

En cryptographie, l'authentification est un processus permettant de vérifier l'intégrité et la légitimité d'une information chiffrée. L'AEAD (*Authenticated Encryption with Associated Data*) est par exemple un mode de chiffrement authentifié par bloc : chaque bloc d'une taille donnée est assigné à une valeur qui est vérifiée lors du déchiffrement. Si la valeur du chiffrement et du déchiffrement ne correspondent pas, cela signifie que les données ont été corrompues, soit de manière accidentelle, soit à des fins offensives. On utilise typiquement des fonctions de hachage pour ces tâches.

Autocrypt

Autocrypt est une norme ouverte et gratuite qui cherche à offrir une alternative au → **Web of Trust** en simplifiant l'échange des clés et le chiffrement. Autocrypt contourne certaines étapes fastidieuses de la recherche et de la vérification des clés publiques en attachant celles-ci directement aux emails envoyés. Dès le deuxième message échangé, la conversation est chiffrée sans qu'aucune action ne soit nécessaire. La version 1.0 a été publiée en 2017.

Chat

Écrit en italique, ce terme ne fait pas référence à un mignon et inoffensif félin qui nous tient compagnie dans notre chaise à bascule, mais à toute sorte de communication en ligne. Les québécois-es utilisent le joli terme de « clavardage ».

Chiffrement / déchiffrement

Le chiffrement est un procédé de cryptographie par lequel on peut rendre des données incompréhensibles pour toute personne qui n'en possède pas la → **clé** de déchiffrement (sous la forme, par exemple, d'un mot de passe). Le chiffrement produit à partir d'un → **texte en clair** (données lisibles) un → **cryptogramme**. Le processus inverse est le déchiffrement. On appelle **chiffres** les → **algorithmes de chiffrement** utilisés pour cette opération.

Chiffrement symétrique et asymétrique

La cryptographie a, depuis des millénaires, utilisé des modes de **chiffrement symétrique** : il s'agit d'utiliser la même clé pour chiffrer et déchiffrer une information. La cryptographie numérique utilise encore ce procédé, par exemple avec le protocole AES.

Le **chiffrement asymétrique** est une invention récente datant des années 1970 rendue possible grâce à l'émergence des ordinateurs et leur puissance de calcul. Grâce à des fonctions mathématiques complexes, le chiffrement asymétrique

utilise une clé différente pour chiffrer et pour déchiffrer une information. La clé publique peut être distribuée largement sans compromettre la confidentialité de l'information ; la clé privée est secrète et permet d'avoir accès à ladite information. Cela facilite grandement l'envoi de message car il n'y a plus besoin de partager un même secret entre l'expéditeuse et le ou la destinataire.

Clé	La clé est une information secrète qui permet de chiffrer et déchiffrer un cryptogramme. Il peut s'agir d'un nombre (pour le chiffre par décalage), d'un mot de passe (pour le chiffre de Vigenère) ou d'une clé en code binaire pour la cryptographie numérique ; dans ce dernier cas, le clé est souvent calculée à partir d'un mot de passe grâce à une fonction de → hachage .
Client (email, IRC)	En informatique, un client est le logiciel qui envoie des demandes à un serveur. Un client email est par exemple un logiciel qui permet d'envoyer et recevoir des emails. Il ne s'agit donc pas de personnes mais de logiciels.
Confidentialité persistante	La confidentialité persistante [<i>forward secrecy</i> en anglais], est une propriété cryptographique qui garantit que la découverte d'une clé privée ne compromet pas la confidentialité des communications passées. Cela signifie que, même en obtenant la clé privée d'un échange, il n'est pas possible de déchiffrer les échanges passés.
Code informatique	Le code informatique est une série d'instructions écrites par des humains destinées à être interprétées par un ordinateur. Le code source est un code écrit par des humains dans un langage informatique donné (C, C++, RUST, JavaScript, etc.) destiné à être compilé en code binaire par un → compilateur pour être interprété par une machine.
Compilateur	Un compilateur est un programme informatique qui traduit le code source écrit par des humains en code binaire interprété par des ordinateurs.
Cryptogramme	Le cryptogramme correspond à ce que l'on obtient après une opération de chiffrement. Le cryptogramme d'un message est la forme chiffrée de ce message, supposée être illisible par quiconque n'en possède pas la → clé . L'inverse est le → texte clair , c'est-à-dire l'information non chiffrée, lisible par toutes.
Cryptoparty	L'organisation CryptoParty est un mouvement populaire mondial visant à enseigner au grand public les bases de la cryptographie pratique. Les Cryptoparties sont des événements populaires visant à promouvoir et expliquer au grand public les bases de la cryptographie pratique. Il s'agit d'un mouvement mondial né en 2012 en Australie. Dans l'esprit des communautés du logiciel libre et gratuit, ces événements sont généralement eux-mêmes libres et gratuits.
Crypto-refresh	Nom du document provisoire servant de brouillon pour de la refonte des spécifications du protocole OpenPGP à l'IETF ; depuis mars 2021, remplace le rfc4880bis.
EFAIL	Nom d'un article scientifique publié en 2018 décrivant une série de vulnérabilités informatiques affectant les protocoles de chiffrement et d'authentification d'email OpenPGP et S/MIME ; par commodité, désigne également la série des dites vulnérabilités.
GIT GitLab GitHub	Git est le logiciel libre et gratuit de gestion de versions le plus populaire dans le domaine du développement logiciel et web. Plusieurs plateformes l'utilisent, notamment GitLab et GitHub, très populaires parmi les équipes de développement du logiciel libre.
Hash / fonction de	Une fonction de hachage est une fonction cryptographique qui associe des va-

hachage	leurs de taille fixe (p. ex. 128 bits ou 256 bits) à des données de taille quelconque (une clé publique, un email, une thèse de doctorat). Les <i>hash</i> sont le produit de cette fonction. Ils sont utilisés notamment pour l'authentification de données numériques.
Librairie logicielle	Ou « bibliothèque logicielle » ; il s'agit d'un ensemble de fonctions informatiques qui peuvent être implémentées dans différents codes informatiques afin de remplir des tâches communes : par exemple, collection de routines, qui peuvent être déjà compilées et prêtes à être utilisées par d'autres programmes informatiques.
Linux	Famille de systèmes d'exploitation open source de type Unix. Les distributions les plus célèbres sont Debian, Ubuntu, Linux Mint, Fedora, Arch.
Mastodon	Mastodon est un réseau social et logiciel de microblogage auto-hébergé, libre, distribué et décentralisé.
Norme	Une norme technique, parfois appelée « standard » par anglicisme, est un référentiel établi par un organisme de normalisation national ou international comme ISO, W3C ou IETF.
Paquet logiciel	En informatique, et surtout dans les communautés du logiciel libre, un paquet est une archive comprenant les informations nécessaires à l'installation d'un logiciel.
Patch	Le mot patch est un anglicisme pour nommer une correction du code informatique (ou, par extension, d'un protocole). Les patches corrigent donc le code en modifiant, ajoutant ou supprimant une ou plusieurs lignes de code. Ces correctifs sont d'abord appliqués localement sur l'ordinateur de la codeureuse, puis assemblés à l'ensemble du code généralement sur une plateforme de partage de code (→ GIT) avant que le logiciel puisse être mis à jour sur l'ordinateur de l'utilisatrice.
Primitive cryptographique	Voir → Algorithme de chiffrement .
Protocole	Un protocole est un ensemble de spécifications et de règles qui précisent la marche à suivre pour effectuer une action spécifique. De très nombreux protocoles sont mobilisés pour utiliser Internet. Les protocoles prennent généralement la forme d'un texte écrit. Certains protocoles, mais pas tous, sont des normes.
rfc4880bis	Voir → crypto-refresh .
Serveur de clés	Un serveur de clés est un serveur sur lequel on peut déposer des clés publiques afin que d'autres puissent trouver ces clés et les utiliser pour communiquer avec les propriétaires des clés. On peut se représenter un serveur de clés comme une espèce d'annuaire téléphonique pour clés publiques. Toutefois, la plupart des serveurs ne vérifient pas la légitimité des clés et n'importe qui peut déposer une clé au nom de quelqu'un. Il est donc important de vérifier que les clés soient bien celles des destinataires. Pour cela, on peut par exemple confirmer l'empreinte de la clé avec cette personne (par exemple par téléphone) ou utiliser le → Web of Trust afin de vérifier les signatures des clés.
Signature électronique	Une signature électronique est une manière d'authentifier la légitimité d'un document numérique. OpenPGP est par exemple utilisé pour signer des emails, mais aussi authentifier, par les développeuses de logiciels libres, des paquets logiciels afin de s'assurer de la légitimité de ces programmes. Un exemple de signature se trouve en page 2 de cette thèse dans la version électronique de la dé-

claration sur l'honneur.

OpenPGP permet également de signer sa propre clé publique, et celles d'autres personnes. Cela sert de base au fonctionnement du → **Web of Trust**.

Slack

Slack est une plateforme de chat utilisée notamment en entreprise, et particulièrement dans le domaine des technologies numériques.

Smartcard

Une *smartcard* est une carte à puce sécurisée qui permet de stocker des clés cryptographiques privées de manière sécurisée. Le principe est que les opérations cryptographiques se passent sur la carte et non dans l'ordinateur. Ainsi, la clé secrète n'est jamais lue par l'ordinateur et encore moins stockée dans sa mémoire ; l'ordinateur peut seulement demander à la clé d'exécuter des opérations de déchiffrement ou de signature. Une *smartcard* peut soit ressembler à une carte à puce du même type que les cartes bancaires, soit prendre la forme d'une clé USB.

Texte en clair

Le texte en clair [*plain text* en anglais] correspond à la forme non chiffrée d'une information. On parle de texte en clair quand un message n'est pas encore chiffré, ou déjà déchiffré. L'inverse est le → **cryptogramme**, c'est-à-dire l'information chiffrée, supposée être illisible sauf pour les personnes qui en possède la clé.

Web of Trust

Le Web of Trust est un concept inventé en même temps que PGP qui permet de tisser une « toile de confiance » entre les utilisatrices d'OpenPGP en signant les clés des unes et des autres. Des *Key Signing Parties* étaient et sont toujours organisées. Il s'agit d'événements lors desquels des utilisatrices d'OpenPGP qui ne se connaissent pas nécessairement se rencontrent et signent les clés entre elles.

Liste des abréviations

AEAD	<i>Authenticated Encryption with Associated Data</i> , un mode de chiffrement authentifié.
AES	<i>Advanced Encryption Standard</i> , algorithme de chiffrement devenu en 2000 une norme technique de l'Institut national des normes et de la technologie des États-Unis d'Amérique (NIST). Cet algorithme est considéré comme l'un des plus sûr et fiable actuellement. Il est notamment utilisé dans OpenPGP comme protocole de chiffrement symétrique par défaut.
BSI	<i>Bundesamt für Sicherheit in der Informationstechnik</i> , l'Office fédéral de la sécurité des technologies de l'information de l'Allemagne.
CCC	<i>Chaos Computer Club</i> , une des plus grande et influente organisation de hackers ; CCC désigne aussi le <i>Chaos Communication Congress</i> , un événement rassemblant des milliers de personnes et ayant lieu chaque année entre le 27 et le 30 décembre.
CERT / CSIRT	<i>Computer Emergency Response Team</i> , centre d'alertes et de réactions aux attaques informatiques, généralement destiné aux entreprises ou aux administrations ; l'acronyme CSIRT (<i>Computer security incident response team</i>) est aussi parfois utilisé.
CIA	<i>Central Intelligence Agency</i> , agence centrale de renseignement des États-Unis d'Amérique.
CIA (la triade)	<i>Confidentialité, Intégrité, Accessibilité</i> , aussi connue sous le nom des trois « A » de la sécurité (Anonymat, Authentification, Accessibilité). Ce sont les trois critères à la base de nombreux systèmes cryptographiques.
CVE	<i>Common Vulnerabilities and Exposures</i> , vulnérabilités et expositions communes ; il s'agit d'un registre des informations publiques relatives aux vulnérabilités de sécurité connues. Le dictionnaire est maintenu par l'organisme MITRE, soutenu par le département de la Sécurité intérieure des États-Unis : https://www.cve.org/ .
DES	<i>Data Encryption Standard</i> , algorithme de chiffrement symétrique par bloc utilisant des clés de 56 bits. Il a été la norme pour le chiffrement symétrique de 1977 à 1999, son usage n'est plus recommandé aujourd'hui et il a été remplacé par AES.
ECDH	<i>Elliptic curve Diffie-Hellman</i> , échange de clés Diffie-Hellman basé sur les courbes elliptiques.
EFF	<i>Electronic Frontier Foundation</i> , une des organisations non gouvernementales les plus réputées en matière de protection des droits numériques.
ETSI	<i>European Telecommunications Standards Institute</i> , institut européen des normes de télécommunications en français.
FAQ	<i>Frequently-asked questions</i> , ou Foire aux questions en français.
GNU	<i>Gnu is Not Unix</i> , système d'exploitation libre.
GPG / GnuPG	<i>GNU Privacy Guard</i> , la première implémentation du standard OpenPGP, après PGP lui-même.

GPGME	<i>GnuPG Made Easy</i> , librairie basée sur GnuPG qui est utilisée dans de nombreux programmes informatiques.
HTML	<i>HyperText Markup Language</i> , le langage de balisage conçu pour représenter les pages web.
HTTP/HTTPS	<i>HyperText Transfer Protocol</i> , le protocole de communication client-serveur développé pour le World Wide Web. Le « S », pour <i>secured</i> , désigne le chiffrement de cette communication.
IETF	<i>Internet Engineering Task Force</i> , organisation qui élabore et promeut des normes Internet, en particulier les normes qui composent la suite de protocoles Internet (TCP/IP).
IESG	<i>Internet Engineering Steering Group</i> , groupe de l'IETF chargé du pilotage de l'activité de production des normes Internet.
IMAP	<i>Internet Message Access Protocol</i> , protocole qui permet d'accéder à ses emails directement sur les serveurs de messagerie.
IPPQT	<i>Infiniment Plus Petit Que Trois</i> , protocole de soutien de thèse.
IRL	<i>In Real Life</i> , dans la vraie vie, c'est-à-dire pas en ligne.
MDC	<i>Modification Detection Code</i> , une fonctionnalité d'OpenPGP permettant la vérification de l'authentification d'un cryptogramme.
MIME	<i>Multipurpose Internet Mail Extensions</i> , norme internet qui étend le format de données des emails pour supporter des textes en différents codage des caractères.
MOOC	<i>Massive Open Online Courses</i> , cours ouverts et gratuits en ligne.
NIST	<i>National Institute of Standards and Technology</i> , Institut national des normes et de la technologie.
NSA	<i>National Security Agency</i> , Agence nationale de la sécurité aux États-Unis d'Amérique.
OCB	<i>Offset codebook mode</i> , mode de fonctionnement de chiffrement authentifié pour les algorithmes de chiffrement par blocs.
OTR	<i>Off-The-Record</i> , protocole de chiffrement à clé publique développé en 2004.
PEP	<i>Pretty Easy Privacy</i> (pEp, ou P≡P), une fondation qui finance notamment les activités de l'équipe de développement de Sequoia. PEP est aussi le nom d'une extension pour Thunderbird et Outlook qui implémente OpenPGP et Autocrypt.
PET	<i>Privacy-Enhancing Technologies</i> , technologies améliorant la confidentialité.
PGP	<i>Pretty Good Privacy</i> , un programme de chiffrement développé en 1991.
PKIX	<i>Public-Key Infrastructure X.509</i> , groupe de travail de l'IETF dédié aux infrastructures à clés publiques.
PQC	<i>Post-Quantum Cryptography</i> , ou cryptographie résistante aux ordinateurs quantiques.
RFC	<i>Request for comments</i> , document publié par l'IETF ; certains d'entre eux sont des normes Internet.
RSA	<i>Rivest-Shamir-Adleman</i> , protocole de chiffrement à clé publique.
RSF	<i>Reporters Sans Frontière</i> , une organisation non gouvernementale active dans la liberté de la presse.
RSS	<i>Really Simple Syndication</i> , famille de formats de données utilisés pour la syndication de contenu Web.
SHA (SHA-1, SHA-256)	<i>Secure Hash Algorithm</i> , fonction de hachage cryptographique conçue par la National Security Agency des États-Unis d'Amérique.

Liste des abréviations

SKS	<i>Synchronizing Key Server</i> , un logiciel utilisé pour les serveurs de clés OpenPGP.
S/MIME	<i>Secure/Multipurpose Internet Mail Extensions</i> , norme de chiffrement et de signature numérique pour les emails au format MIME.
SMTP	<i>Simple Mail Transfer Protocol</i> , protocole de communication utilisé pour transférer les emails vers les serveurs de messagerie électronique.
TAILS	<i>The Amnesic Incognito Live System</i> , distribution GNU/Linux axée sur la sécurité basée sur Debian qui a pour but de préserver la vie privée et l'anonymat. Elle fonctionne sur clé USB. Elle est très utilisée par les journalistes et les activistes.
TCP/IP	<i>Transmission Control Protocol</i> et <i>Internet Protocol</i> . Ce sigle désigne la suite des protocoles Internet, soit l'ensemble des protocoles utilisés pour le transfert des données sur Internet.
TLS	<i>Transport Layer Security</i> , norme de sécurisation par chiffrement du transport de l'information sur Internet.
TOR	<i>The Onion Router</i> , réseau informatique mondial et décentralisé qui permet une navigation anonymisée.
WoT	<i>Web of Trust</i> , permet de vérifier la relation entre une clé publique et une identité numérique.
W3C	<i>World Wide Web Consortium</i> , organisme de standardisation des technologies du Web.
XKCD	xkcd (ou XKCD) est une bande dessinée en ligne créée par Randall Munroe. L'auteur la définit comme « un <i>webcomic</i> qui parle de romance, de sarcasme, de maths et de langage ». Voir par exemple les illustrations 8 p. 105, 9 p. 107, 16 p.164 et 17 p. 175 dans cette thèse, ou le site Internet https://xkcd.com/ .

1. Introduction

Sur la couverture d'un livre de Simson Garfinkel publié en 1995 à propos du programme de chiffrement Pretty Good Privacy, on voit le dessin d'un gros cadenas métallique frappé du sigle PGP (voir illustration 1 ci-dessous). Pretty Good Privacy – abrégé PGP – est un logiciel de chiffrement d'emails publié en 1991 par Philip R. Zimmermann, un activiste anti-nucléaire états-unien passionné de cryptographie. Le livre de Garfinkel comprend une histoire de la cryptographie numérique et en particulier du protocole PGP, ainsi que des explications et conseils sur son usage. Le cadenas est un symbole qui représente l'accès sécurisé à un lieu où seules peuvent pénétrer les personnes habilitées qui en ont la clé. Dans l'iconographie du numérique, il représente la sécurité informatique en général, et particulièrement tout ce qui concerne la cryptographie. Plus précisément, il symbolise à la fois la protection de l'accès à un lieu donné (qu'il s'agisse d'une pièce derrière une porte ou des données numériques), mais aussi la solidité de cette protection. D'ailleurs, sur la couverture du livre de Garfinkel, l'anse du cadenas est marquée de la mention « *hardened* » qui précise la qualité du matériau utilisé : de l'acier trempé. Finalement, en tant qu'objet familier, le cadenas est aussi le symbole de la facilité d'usage : les utilisatrices¹

1 Pour des raisons que j'explique dans le chapitre 2 « Ethnographie de l'écosystème OpenPGP » (p. 57), j'ai eu recours à des néologismes tels qu'« utilisatrice », « développeuse » ou « codeuse » afin d'inclure toutes les personnes concernées indépendamment de leur genre. Ces mots sont accordés au féminin. J'espère que les lectrices se retrouveront dans cette formulation.

n'ont pas à se soucier de sa mécanique interne parfois très complexe et l'utilisent généralement très facilement.

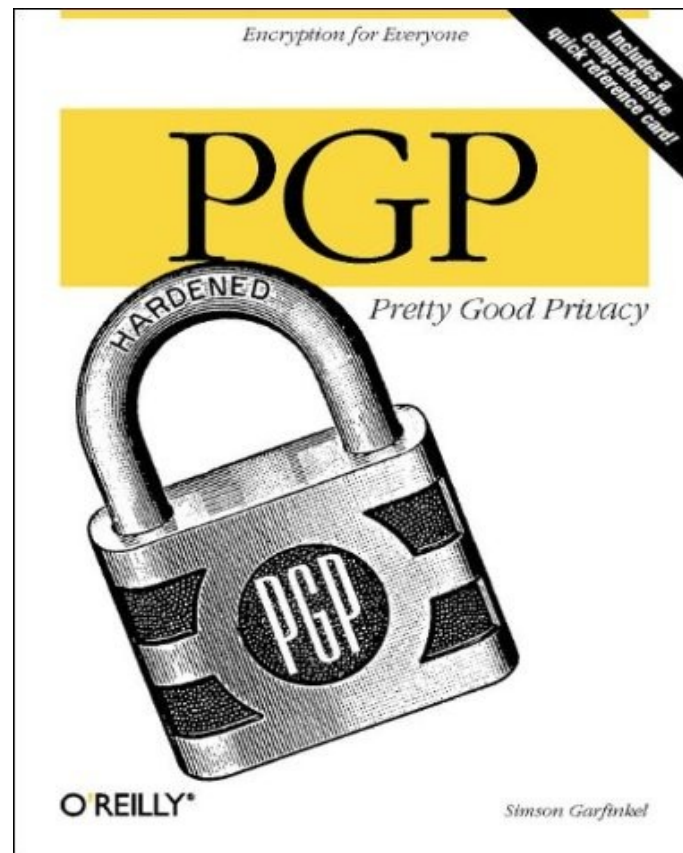


Illustration 1 : Couverture du livre de Simson Garfinkel (1995). Le « cadenas PGP » de la couverture de ce livre symbolise non seulement la capacité du logiciel PGP à protéger l'accès à des informations confidentielles, mais également sa solidité et sa facilité d'usage.

Ce parallèle se vérifie avec l'illustration du livre de Garfinkel : alors que la cryptographie avait (et a encore) la réputation d'être un domaine extrêmement complexe et peu accessible, l'apparence familière de ce cadenas est rassurante. Le sous-titre « *encryption for everyone* » l'affirme d'ailleurs clairement : la cryptographie numérique ne doit plus faire peur car elle est devenue aussi facile à utiliser qu'un simple cadenas. Ces trois qualités représentées par ce cadenas (sécurité, solidité et facilité d'usage) forment la métaphore de ce que j'appelle le cyber-cadenas pour représenter la sécurité informatique. Depuis le livre de Garfinkel (1995), cette métaphore s'est largement popularisée et est désormais omniprésente dans le quotidien des sociétés connec-

tées à Internet. Les utilisatrices de technologies numériques voient (peut-être d'ailleurs sans plus le remarquer) un petit symbole de cadenas des centaines de fois par jour, par exemple dans la barre des navigateurs web quand la connexion est protégée par le chiffrement de HTTPS, dans les formulaires pour le paiement en ligne et sur le bouton « envoyer » de certaines applications de messagerie sécurisées comme Signal.

Fiable, robuste, facile à utiliser : le cadenas semble une métaphore efficace et pleine de promesses pour parler de la sécurité informatique et de la cryptographie à ses utilisatrices, mais qu'en est-il des développeuses de ces technologies ? Comment perçoivent-elles cette sécurité qu'elles façonnent et entretiennent ? Comment s'y prennent-elles pour fabriquer un produit qui se veut sûr, solide et simple en dépit de la mécanique interne d'une grande complexité ? Quelles sont les contraintes de la confection de telles technologies, et comment sont-elles maintenues dans le temps ? Avec cette thèse, j'ai voulu regarder par le trou de la serrure pour voir de quoi la mécanique de ce cyber-cadenas était faite, comment elle était maintenue et comment elle vieillissait. Je me suis tourné vers les ingénieur·e·s qui conçoivent les protocoles derrière cette technologie, vers les codeuses qui implémentent ces protocoles dans des logiciels informatiques, et vers des scientifiques qui ont testé la robustesse de ce cadenas. Au long des pages qui suivent, je montre comment de nombreuses personnes s'affairent à prendre soin d'une technologie vieillissante afin qu'elle puisse encore être utilisée aujourd'hui, l'année prochaine et, peut-être, dans dix ans. Cette thèse est une exploration des modalités des pratiques de soin de la sécurité informatique pour prolonger la durée de vie de ce cyber-cadenas dans le temps.

1.1. La sécurité informatique, la cryptographie et OpenPGP

La sécurité informatique est devenue un sujet récurrent des fils d'actualité, une source d'inquiétude pour la violation de la vie privée et une infrastructure sociotechnique grandissante. De nouvelles vulnérabilités affectant aussi bien les logiciels que

le matériel informatique sont régulièrement dévoilées, des vols de données confidentielles touchant des personnes privées, des sociétés ou des institutions étatiques se font de plus en plus fréquents, et des technologies de surveillance numérique envahissent toutes les sphères des sociétés contemporaines (Lyon 2018; Zuboff 2020). Face à cela, de nombreux discours et pratiques ont vu le jour afin de tâcher de (re-)prendre contrôle des données numériques et soigner l'hygiène numérique, parmi lesquelles la cryptographie tient une place de premier plan (Levy 2001; Bozzini 2016; Di Salvo 2020; Ermoshina 2023).

Le mot cryptographie, du grec κρυπτός, qui signifie « caché », « secret », et γράφειν, « écrire », se comprend comme « la science de la dissimulation » (Bozzini 2016, 278) ou encore, l'« art d'écrire en chiffres ou d'une façon secrète quelconque »². En français, il est courant d'utiliser le mot « chiffrement » comme synonyme de cryptographie, le mot « chiffre » comme synonyme de « protocole cryptographique », et le verbe « chiffrer » à la place de « crypter » (qui ne devrait d'ailleurs pas être utilisé en français). La cryptographie est un art et une science ancienne dont les origines remontent à des millénaires. On en trouve des traces dans les traités militaires et la correspondance privée bien sûr, mais aussi dans des lieux plus étonnants, comme les bateaux de pêche bretons³ et même dans le kâma-sûtra (Kahn 1996; Singh 1999). Depuis quelques décennies, la cryptographie numérique s'est immiscée dans notre quotidien sans même qu'on ne s'en rende compte, tant elle est devenue transparente et efficace. Comme l'a écrit le cryptographe Bruce Schneier sur son blog personnel, la cryptographie joue un rôle central dans l'organisation des sociétés humaines actuelles :

Des emails à la communication mobile, de l'accès sécurisé à Internet à l'argent numérique, la cryptographie est un élément essentiel des systèmes d'information d'aujourd'hui. La cryptographie permet de garantir la responsabilité, l'équité, l'exactitude et la confidentialité. Elle peut prévenir la fraude dans le

2 Selon le Wiktionnaire : <https://fr.wiktionary.org/wiki/cryptographie>.

3 Dès que les chalutiers des terre-neuvas ont été équipés d'un système de radio, les marins ont mis au point un système cryptographique permettant de communiquer avec l'armateur sans que les chalutiers concurrents puissent écouter les informations stratégiques qui étaient communiquées (chargement de la cale, date du retour au port, etc.).

commerce électronique et garantir la validité des transactions financières. Elle peut prouver votre identité ou protéger votre anonymat. Elle peut empêcher les malfaiteurs de modifier votre page Web et empêcher les concurrents industriels de lire vos documents confidentiels. (Schneier 1997, ma traduction)

Actuellement, une personne en Europe utilise ainsi de nombreuses fois par jour différents technologies de chiffrement sans même sans rendre compte et de manière totalement transparente. Certaines observateurices ont montré que les technologies de chiffrement numérique n'ont pas été développées avec le même succès selon le type d'applications et les usages. Arvind Narayanan (2013a; 2013b) a par exemple souligné que les technologies cryptographiques à des fins de vie privée (comme le chiffrement des communications en ligne) a pris beaucoup de retard par rapport aux technologies vouées à des usages commerciaux et économiques qui ont connu un plus grand succès. Toutefois, les révélations d'Edward Snowden en 2013 à propos de plusieurs programmes de surveillance numérique de masse ont été un véritable jalon qui a marqué le développement et surtout l'adoption de la cryptographie à des fins de protection de la vie privée (Bozzini 2016; Cogburn 2016; Hellegren 2017). Plusieurs protocoles et applications mobiles ont vu le jour, notamment le protocole Signal qui est non seulement utilisé dans l'application du même nom, mais aussi dans WhatsApp et d'autres applications de messagerie sécurisée (Ermoshina et Musiani 2019). Actuellement, d'autres applications comme Telegram (en mode secret), Wire, Threema ou DeltaChat permettent d'utiliser un chiffrement de bout en bout.

Le chiffrement de bout en bout signifie que l'information est chiffrée entre les deux points finaux de la communication, ce qui implique en principe que personne d'autre ne peut avoir accès aux informations. Par exemple, quand j'envoie un message sur Signal, ce dernier est chiffré sur mon téléphone jusqu'au téléphone de mon interlocuteurice, et seules nos applications ont la capacité de déchiffrer le message en question. Ce n'est pas le cas de Telegram qui, dans le mode classique, chiffre uniquement la connexion entre les utilisateurices et le serveur central. Cela signifie que le message que j'envoie est chiffré de mon téléphone jusqu'au serveur de Telegram, qui le déchiffre et le re-chiffre jusqu'à la destinataire. Ces applications de messagerie

sécurisée se sont multipliées ces dernières années et de nombreuses options existent, avec des avantages et des inconvénients distincts en termes de protection du contenu (y compris les métadonnées), mais aussi d'interopérabilité, de coûts financiers et de résilience de l'infrastructure en cas de panne ou d'attaque sur le réseau (Ermoshina, Musiani, et Halpin 2016). Un des problèmes récurrents des applications les plus répandues concerne l'effet silo : en effet, il n'est pas possible d'envoyer depuis Signal un message à une personne utilisant Threema ou Telegram. Cela implique soit d'utiliser de nombreuses applications différentes, soit de se restreindre à ne communiquer qu'avec des personnes utilisant l'application en question.

Dans le cas des emails, ce problème n'existe pas car il s'agit d'une infrastructure interopérable fédérée, ce qui signifie que je peux sans autre envoyer un email à n'importe quelle autre adresse email sans me soucier du service utilisé par la destinataire. Concrètement, avec mon adresse @unifr.ch, je peux communiquer avec une adresse @cnrs.fr et cela fonctionnera – du moins dans la plupart des cas, car n'oublions pas que les pannes sont une partie constitutive des technologies. En revanche, par défaut, un email n'est pas protégé par un chiffrement de bout en bout⁴. Contrairement à une impression souvent répandue parmi les utilisatrices non averties (Dechand et al. 2019), les emails comptent ainsi parmi les technologies de communication numérique les moins protégées d'un point de vue de la confidentialité des informations envoyées. Ils sont en cela souvent comparés à une carte postale : quiconque tombe dessus peut lire son contenu, mais aussi toutes sortes de métadonnées : expéditeuse, destinataire, date et encore le sceau de la poste locale. C'est pour pallier ce problème que Pretty Good Privacy (PGP) a été conçu en 1991 par Philip R. Zimmermann : offrir un moyen de chiffrer ses emails de bout en bout.

En concevant PGP 1991, Zimmermann s'est largement inspiré du protocole RSA qui était alors protégé par une licence (Levy 2001). Il avait comme objectif premier d'offrir un moyen de chiffrer et signer électroniquement des emails. Son programme

4 Les emails sont toutefois quand même chiffrés avec le protocole TLS qui assure une connexion sécurisée entre les serveurs, par exemple entre mon client email et le serveur de mon fournisseur d'email. Toutefois, chaque serveur déchiffre et rechiffre le message en question, et peut donc avoir accès au contenu de chaque paquet.

visait à rendre les emails chiffrés illisibles à toutes personnes qui n'en possédaient pas la clé. Il a ainsi développé un protocole qui utilise la cryptographie à clé publique, ou cryptographie asymétrique. Avant d'aller plus loin, il me semble important d'expliquer brièvement le principe du fonctionnement de la cryptographie à clé publique et de spécifier les usages actuels d'OpenPGP dans l'encadré ci-dessous.

Fonctionnement de la cryptographie asymétrique

La cryptographie asymétrique est une invention relativement récente puisqu'elle a été développée dès les années 70 (Diffie et Hellman 1976), contrairement à la cryptographie dite symétrique qui est utilisée depuis des milliers d'années (Singh 1999). Un système cryptographique est dit symétrique si c'est le même mécanisme qui permet de chiffrer et de déchiffrer, par exemple un même mot de passe. Au contraire, la cryptographie asymétrique utilise deux clés différentes, qui sont appelées une paire de clés. Il y a d'un côté la clé privée, secrète, qui permet de déchiffrer les informations, et une clé publique qui est utilisée pour chiffrer. Cette seconde clé est dite publique car elle peut être très largement diffusée sans compromettre la confidentialité du message. Ainsi, la cryptographie asymétrique, dite à clés publiques, permet de chiffrer un message pour une destinataire spécifique.

Le fonctionnement de base est relativement simple à comprendre et le cadenas est une fois de plus souvent utilisé comme métaphore : avoir une paire de clés OpenPGP peut ressembler à posséder un nombre infini de cadenas ouverts et une seule clé pour tous les ouvrir. Il est alors possible de distribuer ces cadenas très largement en les postant notamment sur des serveurs de clés publiques, sur sa page Internet ou dans ses emails, car ces cadenas peuvent être facilement fermés par n'importe qui, mais ouverts seulement grâce à la clé secrète.

En outre, la cryptographie asymétrique permet de signer une information numérique par le procédé inverse : l'information est chiffrée par la clé privée pour être déchiffrée par la clé publique (à laquelle tout le monde peut avoir accès). En étant capable de déchiffrer cette information, la clé publique peut

alors certifier que seule la clé privée était capable de faire cela. Ce procédé est utilisé pour émettre des signatures électroniques qui certifient que l'information a bien été émise par la personne qui prétend le faire. Dans le cas des emails, cette fonction est particulièrement utile car il est très facile de « forger » des emails qui prétendent avoir été envoyés par une certaine adresse⁵. En résumé, OpenPGP permet de faire plusieurs choses. Je peux par exemple chiffrer un email pour que seules mes directeurices de thèses puissent le lire – si tant est qu'il et elle utilisent OpenPGP et que je possède leurs clés publiques. Je peux également signer cet email pour certifier que c'est bien moi, Sylvain Besençon, possesseur présumé de la clé 9584 92A9 80DC C8FE F5C5 EE91 56F7 ACD4 B1FD E56F, qui envoie cet email. Je peux aussi signer un simple texte, comme la version numérique de mon attestation sur l'honneur en début de cette thèse. Dans ce cas, ce sont mes destinataires qui devront posséder ma clé publique afin de vérifier ladite signature. Finalement, OpenPGP permet également de chiffrer de manière symétrique des documents électroniques, en utilisant un mot de passe. Par exemple, je produis ci-dessous un cryptogramme qu'il est possible de déchiffrer avec le mot de passe « **secret** ».

```
-----BEGIN PGP MESSAGE-----  
wy4ECQMIfG1v44X5v4f/sjP7HFzMHkjaX/EDbWt3J3m6sPpkYxkRI5b/02LwlaJR  
0nMBBTw9Ugosa37wmvD8avtcakqzHRQk0/9URbQ5kR0bnrVbN8pte6T9ltAot16U  
FDpVV0i4jAsS6xTapMD6GtYr0meK4N9hg32+WLvGfZEZdoNiVQiM0jQ/iE64ecb/  
i1B//mJGyRf00+EcRlGYdU+7xLEW  
=1Ffj  
-----END PGP MESSAGE-----
```

Les usages d'OpenPGP

Parmi les usages d'OpenPGP, il faut tout d'abord distinguer les usages de la technologie OpenPGP comme **norme cryptographique**, des usages des **programmes informatiques** qui implémentent cette norme. Ce sont deux caté-

5 Le logiciel Zmail permet par exemple de le faire : <https://zmail.sourceforge.net/tutorial.html>. Une explication vulgarisée du fonctionnement des emails forgés peut être consultée sur le site hackersrepublic.org : <https://www.hackersrepublic.org/outils/emails-forges-explications-detections-et-corrections>.

gories distinctes d'utilisatrices : d'un côté, il y a des **développeuses** qui ont besoin d'une norme pour coder un logiciel ; de l'autre, il y a des **utilisatrices finales** qui ont besoin d'un logiciel pour protéger et/ou authentifier des données numériques. Ils et elles vont alors utiliser des applications comme GnuPG, ProtonMail ou Thunderbird pour protéger les informations en question. La plupart des études s'intéressent à ces utilisatrices finales de la technologie (Whitten et Tygar 1999; S. L. Garfinkel et Miller 2005; Sheng et al. 2006; Sousedek 2008; Andersen 2016; Ruoti et al. 2016; Dechand et al. 2019) ; cette thèse se tourne au contraire vers les développeuses, codeuses et ingénier-e-s qui utilisent la norme RFC4880 de l'IETF afin de produire des logiciels de sécurité informatique. Toutefois, pour comprendre leur projet et pourquoi ils et elles ont décidé d'utiliser OpenPGP (et pas un autre protocole) pour produire leur logiciel, il est important de comprendre comment les utilisatrices finales utilisent leurs logiciels.

OpenPGP est principalement connu pour ses capacités de chiffrement asymétrique et c'est dans ce domaine qu'il a forgé sa réputation. Toutefois, des milliers de développeuses l'utilisent régulièrement non pas pour chiffrer, mais pour authentifier des données, et notamment des codes informatiques. En effet, de nombreuses communautés open source, notamment les communautés Linux et la communauté Wikipédia, utilisent OpenPGP pour signer les contributions, et ainsi authentifier leurs auteures. De cette manière, on peut avoir une plus grande confiance dans les contributions et vérifier que les données sont bien celles que la contributeurice a écrites. Certaines communautés open source exigent même que leurs membres aient une paire de clés OpenPGP avant de leur permettre de contribuer⁶.

En résumé, les implémentations OpenPGP sont utilisées pour trois types d'usages : (1) **la cryptographie asymétrique** qui utilise un système de clés publiques pour les communications en ligne, et notamment le chiffrement

6 Voir par exemple le projet Debian : « Lorsqu'ils et elles rejoignent le projet Debian, les développeuses doivent s'identifier en fournissant une clé OpenPGP qui est signée par au moins deux membres existants du projet. Les contributions à l'archive Debian sont signées cryptographiquement en utilisant la clé OpenPGP de la développeuse pour se protéger contre les falsifications. » (<https://wiki.debian.org/OpenPGP>, ma traduction).

des emails, (2) **la cryptographie symétrique** pour chiffrer des documents électroniques avec un mot de passe et (3) **l'authentification** de données numériques, que ce soit une signature d'email, de contribution à un projet open source ou une déclaration sur l'honneur en début de thèse.

Le programme de Zimmermann PGP a été une révolution car c'était le premier logiciel de cryptographie à clé publique disponible pour des citoyennes et citoyens lambda – et gratuitement qui plus est (Singh 1999; Levy 2001; Coleman et Golub 2008). PGP a ainsi ouvert une porte vers la domestication de cette technologie. Quelques années plus tard, en 1998, PGP est devenu la norme⁷ OpenPGP de l'Internet Engineering Task Force (IETF), l'une des organisations les plus importantes en matière de standardisation des technologies Internet. À ce titre, OpenPGP est devenu un protocole libre et fédéré, ce qui signifie que son usage est gratuit, que ses spécifications sont négociées par une communauté ouverte d'ingénieur·e·s.

Je donne de plus amples informations sur l'histoire, le développement et l'usage de PGP au fil des chapitres qui suivent. Toutefois, il me semble nécessaire d'ajouter encore un point crucial pour bien comprendre le contexte actuel d'OpenPGP : pour beaucoup de personnes aujourd'hui, cette technologie n'est plus qu'une relique du passé qui est dépassée d'un point de vue technique, mais aussi architectural. D'ailleurs, au début de ma recherche, de très nombreuses personnes, notamment des cryptographes, m'ont fortement recommandé de ne pas m'intéresser à OpenPGP mais à d'autres tendances beaucoup plus intéressantes : la nouvelle version de TLS (v1.3, publiée en 2018), la compétition pour une nouvelle norme de cryptographie résistante aux ordinateurs quantiques, ou des protocoles plus « modernes », comme Signal. OpenPGP est dès lors très éloigné des nouvelles tendances où la « crypto » doit être extraordinaire, rutilante et exemplaire (par exemple les blockchains ou la cryptographie quantique).

7 J'utilise le mot français « norme » pour traduire le terme anglais « *standard* », et non pas le mot « standard » lui-même qui n'a pas la même signification et connotation en français (Prud'homme, Doray, et Frédéric Bouchard 2015).

1.2. L'émergence des revendications de droit à la vie privée numérique et l'imaginaire « crypto »

Le développement des technologies numériques, et notamment la cryptographie, a fait émerger de nouvelles formes de subversion, répression et résistance dans l'espace public (Tréguer 2015). Après plusieurs scandales, notamment les révélations d'Edward Snowden en 2013 et celles de Christopher Wylie à propos de Cambridge Analytica en 2018, le droit à la vie privée⁸ est devenu une revendication récurrente de nombreuses associations de la société civile, faisant par exemple émerger la notion de citoyenneté numérique (Hintz, Dencik, et Wahl-Jorgensen 2017). Toutefois, les origines de cette mouvance remontent aux années 1970, à partir desquelles tout un imaginaire socioculturel s'est développé autour du diminutif « crypto ». Isadora Hellegren (2017) a proposé une histoire de ce qu'elle a appelé les « discours crypto » qu'elle fait remonter aux débuts de la cryptographie à clé publique, en 1975 et qu'elle divise en trois périodes : les origines (1975-1990), la cristallisation (1990-2000) et la revitalisation (2000-2015) (voir illustration 2 ci-dessous). La période dite des origines est surtout marquée par une revendication des milieux universitaires états-uniens qui voulaient s'approprier la cryptographie en tant que sujet d'étude civil alors qu'elle était réservée jusque là aux domaines militaire et du renseignement.

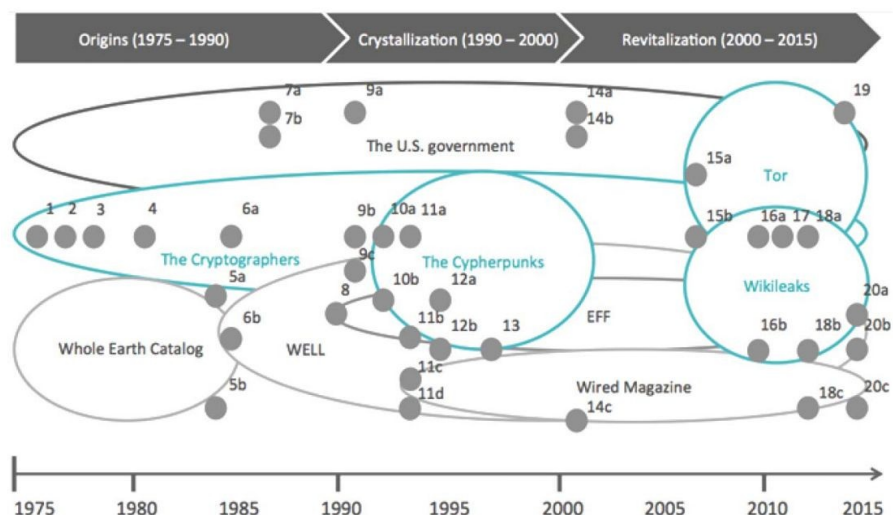


Illustration 2 : Chronologie des discours crypto, établie par Hellegren (2017, p. 11).

8 Le terme anglais *privacy* est difficilement traduisible en français. Il englobe à la fois la notion de respect de la vie privée, la protection des données personnelles, voire, dans certains cas, la revendication d'un droit de contrôle sur les données produites par des entreprises.

Ce n'est pourtant qu'avec la publication de Pretty Good Privacy que la cryptographie a été véritablement accessible à la population civile (événement 9b dans l'illustration 2). En tant que « première solution vraiment utilisable sur ordinateur personnel d'un système cryptographique complet, des signatures numériques au chiffrement » (Levy 2001, 198, ma traduction), Pretty Good Privacy a été une révolution dans le domaine de la protection des communications numériques civiles. Jusqu'au début des années 2000, le programme de Zimmermann et la nouvelle norme OpenPGP suscitaient beaucoup d'admiration et d'espoir, tout en inspirant de nombreux autres projets, notamment le protocole Off-The-Record Communication, publié en 2004. Puis, petit à petit, OpenPGP a perdu de sa superbe, supplanté techniquement et esthétiquement par d'autres programmes.

Parmi les études en sciences sociales qui ont saisi la cryptographie comme objet d'étude, j'aimerais mentionner celles qui ont analysé les discours et les imaginaires que la cryptographie suscitait (West 2018; Hellegren 2020) et la manière dont la cryptographie pouvait inspirer la confiance (Ruh 2018; Dechand et al. 2019; Bruun, Andersen, et Mannov 2020). Ces études ont notamment montré comment la cryptographie, en tant qu'outil sociotechnique, s'est développée afin de répondre à certains besoins mais aussi en tant qu'élément d'une certaine culture de mouvements sociaux numériques. En parallèle, les dernières décennies ont vu émerger un imposant développement de dispositifs de surveillance numérique de plus en plus envahissants dans la vie des citoyen·nes (Diffie et Landau 2010; Lyon 2018; Zuboff 2020). La cryptographie est devenue par conséquent un objet privilégié pour étudier les techniques de résilience face à cette surveillance (Bozzini 2016; Gürses, Kundnani, et Van Hoboken 2016; Ermoshina et Musiani 2022).

En me penchant sur le cas d'étude du protocole OpenPGP, j'ai voulu participer à cette discussion en interrogeant la manière dont les programmes cryptographiques étaient produits et maintenus. L'objectif qui traverse l'entier de cette thèse est de comprendre comment la sécurité informatique est faite et défaite au quotidien, par qui, et à travers quels processus. Plus spécifiquement, j'ai analysé le point de vue et

les pratiques des personnes qui prennent soin au quotidien de technologies liées à OpenPGP afin de comprendre les différentes logiques qui encadrent le travail de sécurité. J'ai observé des pratiques collaboratives autour du code informatique (Besençon 2022), des moments de crises et de controverses lors de révélation de failles informatiques (Besençon et Bozzini 2020), ainsi que des négociations lentes autour des processus de standardisation de cette technologie. À partir de ces observations, une question est devenue de plus en plus prégnante : dans le champ des technologies du numérique où tout semble aller très vite, comment les technologies vieillissent-elles, comment et pourquoi les fait-on durer, et qu'est-ce que cela nous apprend sur l'idée même de sécurité que ces technologies cherchent à produire ? Le cas d'OpenPGP permet de comprendre comment le passé, le présent et l'avenir influencent le développement des technologies de sécurité informatique, et pourquoi ces dimensions sont à leur tour importantes pour comprendre les modalités de soin des infrastructures numériques. Cette thèse est une exploration de la manière dont des communautés de développeuses, de codeuses, d'ingénieur·e·s et d'universitaires ont collaboré, parfois non sans frictions, pour prolonger la durée de la sécurité d'OpenPGP.

Cette thèse est une contribution à la manière d'appréhender de manière pragmatique la sécurité informatique en questionnant non pas ce qu'elle protège mais ce qu'elle engendre. En cherchant à comprendre ce qui était entrepris au nom de la sécurité informatique, j'ai développé l'idée de la sécurité en tant que projet et proposé la notion de projet-sécurité pour comprendre comme l'avenir était une préoccupation centrale dans les pratiques de soin des technologies de sécurité informatique (voir le chapitre 3 « Alice et Bob aux pays des bogues », p. 104). En dialogue avec les études sur la maintenance et le soin des choses, j'ai concentré mon attention sur les temporalités et les rythmes du soin de la sécurité informatique, temporalités qui justement faisaient ressortir certaines spécificités des technologies numériques. Par exemple, alors que ces technologies sont souvent l'objet de fantasmes futuristes, je montre comment elles sont en réalité ancrées, voire enlisées dans un passé de plus

en plus contraignant (voir chapitre 4 « Enlisement et ancrage infrastructurels », p. 150). Partant, je soutiens qu’au lieu de suivre les discours optimistes sur le numérique qui proclament un avenir radieux, plus égalitaire et plus inclusif, les personnes en charge de la maintenance de technologies de sécurité informatique envisagent l’avenir comme plein de menaces et de problèmes (voir les deux chapitres de la dernière partie de cette thèse : « La durée de l’avenir », p. 296). Ainsi, cette thèse suggère que le soin des choses numériques ne s’inscrit pas seulement dans une temporalité axée sur le présent (Denis et Pontille 2022), mais également ancrée dans le passé et projetée dans l’avenir. La notion de « résilience prospective » que je développe dans la dernière partie permet en effet de penser la sécurité comme le projet de préparer la technologie à pouvoir être réparée quand le prochain bogue apparaîtra.

Le cas d’OpenPGP s’est révélé particulièrement stimulant pour questionner la sécurité informatique. Durant cinq années de recherche, j’ai rencontré et discuté avec de nombreux·se·s ingénieur·e·s, codeuses, cryptographes, universitaires et journalistes. Certaines personnes ont déclaré cette technologie désuète et morte (Green 2014; Katwala 2018; Latacora 2019) alors que d’autres affirmaient que c’était une technologie tout à fait fiable et sûre (Yen 2018; Walfield 2019). Par ailleurs, lors d’une discussion informelle, Philip R. Zimmermann m’a assuré que sa plus grande satisfaction avec PGP était d’avoir sauvé des vies humaines, tout en confessant qu’il n’utilisait plus ce système cryptographique⁹. De ces discussions, j’ai retenu qu’aucun consensus ne se dégagait sur la définition même de ce qu’était la sécurité, et OpenPGP m’a ainsi permis de questionner comment la sécurité évoluait dans le temps, comment elle était produite, compromise et négociée dans la durée, et quel était le rythme de ses transformations.

Ce qui m’intéresse dans cette thèse, ce sont les gens, les activités, les processus et les objets qui font et défont au quotidien la sécurité informatique. Je n’ai pas étudié

9 Sur son site Internet, il a publié quelques lettres reçues qui expliquent en quoi cette technologie a changé le cours de la vie de milliers de personnes : <https://philzimmermann.com/EN/letters/index.html>. Un article de 2015 intitulé « *Even the Inventor of PGP Doesn’t Use PGP* » du magazine en ligne Motherboard revient sur le fait que Zimmermann semble avoir abandonné PGP (Franceschi-Bicchieri 2015).

les discours et pratiques des utilisatrices mais celles des personnes en charge de la conception et de la maintenance de ces technologies. J'ai notamment analysé les pratiques et discours de nombreuses personnes qui travaillent sur OpenPGP en négociant ses spécifications, en le codant, en étudiant ses propriétés pour comprendre ses défauts, et en tentant de les corriger. Cette thèse propose une contribution à l'anthropologie de la sécurité en suggérant l'idée de penser la sécurité non pas comme une propriété, un discours ou un projet, mais comme un projet en constante négociation. Plutôt que de concevoir la sécurité comme une série de paramètres, ou comme des discours de sécurisation, je l'envisage comme une source d'inquiétude qui engage des pratiques de soin, de maintenance et d'anticipation des menaces à venir.

Cette thèse apporte aussi une contribution aux études sur le soin des technologies numériques. En effet, mon analyse du vieillissement de la technologie OpenPGP me permet d'affirmer que ce sont les pratiques de soin qui font vieillir les technologies numériques. Ce soin implique non seulement des pratiques de maintenance, mais aussi d'innovation, deux dynamiques qui sont fondamentalement entremêlées. Dans la troisième partie de la thèse, je développe notamment l'idée d'une (p)réparation de l'avenir à travers la recherche d'une résilience prospective. Cette notion permet de parler de la volonté de non seulement rendre la technologie aussi sûre que possible pour le présent, mais également de la rendre aussi résiliente que possible pour l'avenir en tâchant de faciliter le travail de maintenance à venir.

Un de mes objectifs à travers cette recherche s'est construit autour d'une volonté de rendre compte du travail ordinaire, voire routinier, de la sécurité informatique en déconstruisant son aspect spectaculaire et exceptionnel. La spectaculaire effervescence de la sécurité informatique est en effet surmédiatisée par rapport au travail quotidien des personnes qui s'occupent de cette sécurité. Je ne me suis ainsi guère attardé sur les nouvelles attaques qui ont fait les grands titres de la presse, sur de nouveaux virus probablement mis au point par des gouvernements dans une logique de guerre numérique, ou sur le développement d'innovations en cryptographie quantique par exemple. Au contraire, cette recherche contribue à penser la sécurité

dans sa banalité quotidienne : comment prend-on soin d'elle ? Pourquoi et comment de vieux protocoles sont-ils maintenus ? Qu'est-ce que cela signifie, concrètement ?

De manière transversale, ce dialogue entre ces deux champs de recherche, sécurité et maintenance, apporte également une occasion de repenser la manière de parler de la sécurité informatique. Je propose de déconstruire le vocabulaire stéréotypé et guerrier de la sécurité informatique pour la considérer comme une chose fragile, soumise à d'incessants bricolages et de nombreux compromis. La section suivante permet de préciser cet aspect en partant d'une réflexion sur la prolifération des symboles masculins dans les représentations de la sécurité informatique.

1.3. OpenPGP, une chose fragile et frêle plutôt qu'un solide cadenas

Ma recherche au sein de l'écosystème OpenPGP a montré une surabondance de symboles masculins et divins qui ne peut pas être qu'anecdotique et qui doit, au contraire, être questionnée. En effet, la sécurité informatique en général, et la cryptographie en particulier, sont très souvent représentées comme des domaines hautement complexes et peu compréhensibles pour les néophytes, accessibles à une petite élite seulement, économiquement très profitables – et surtout encore largement réservées aux garçons (Collet et Mosconi 2010). Dans ce qui suit, je propose d'interroger cette conjonction de symboles masculins en suivant plusieurs pistes. Il s'agit dans un premier temps de repérer et questionner ces symboles utilisés pour représenter la sécurité informatique, d'analyser le vocabulaire utilisé pour en parler, et finalement de déterminer la perspective heuristique adéquate pour examiner la sécurité informatique en étant sensible à cette conjonction de symboles masculins. Commençons donc par revenir à la métaphore du cadenas.

Introduction

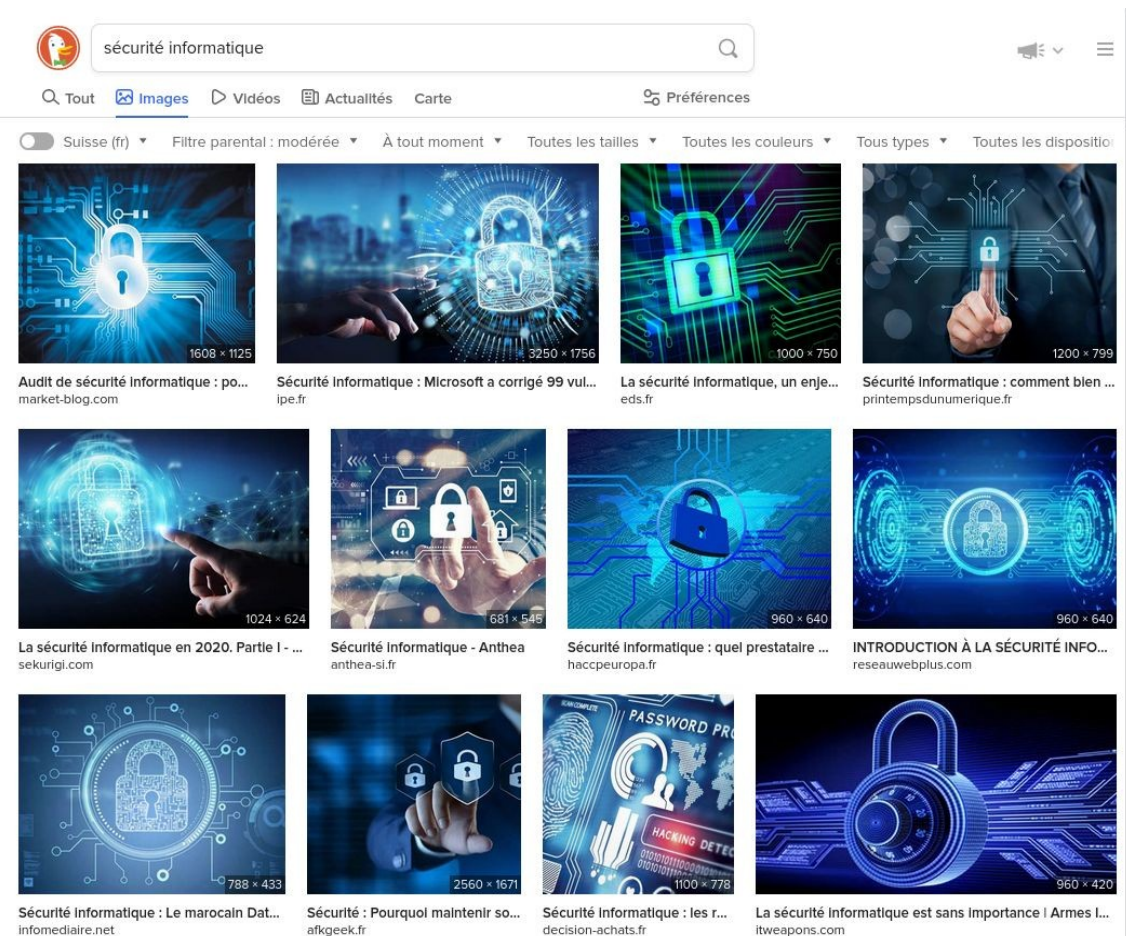


Illustration 3 : Copie d'écran des résultats de recherche pour "sécurité informatique" dans duckduckgo (recherche effectuée le 28 novembre 2022). Source : <https://duckduckgo.com/>.

La capture d'écran ci-dessus montre à quel point la métaphore du cadenas pour évoquer la sécurité informatique s'est généralisée. Une recherche avec les termes « sécurité informatique » dans un moteur de recherche donne presque exclusivement des images bleues particulièrement stéréotypées sur lesquelles un cadenas est pratiquement systématiquement présent. On remarque néanmoins bien que la métaphore du cadenas IRL¹⁰ semble avoir pris le virage numérique. Non seulement la nouvelle iconographie se pare d'un code couleur composé principalement de bleu, avec une touche de vert et parfois une lueur d'un rouge alarmiste, mais un autre changement s'est opéré dans la position de la serrure qui devient visible sur la face des nouvelles « icônes » de la sécurité informatique. Très paradoxalement, ce changement semble être une régression car il évoque l'usage de vieilles clés allongées que

10 IRL est l'acronyme d'« *in real life*, « dans la vraie vie » en français.

l'on utilise encore parfois pour ouvrir les portes en bois des caves d'immeubles. Or, à mesure que la métaphore du cadenas pour figurer la sécurité informatique devient commune, on constate en même temps une raréfaction progressive des clés physiques : ces objets ordinaires de notre quotidien sont petit à petit remplacés par des dispositifs numériques, comme des lecteurs de cartes, des scanners d'empreintes digitales ou d'iris, ou des applications sur les téléphones dits intelligents. Alors que le cadenas IRL perd petit à petit son emprise sur les portes physiques, son double numérique semble proliférer.

De plus, alors que le cadenas IRL sur la couverture du livre de Garfinkel (1995) était simplement posé sur un fond blanc et semblait en ce sens n'évoquer qu'une métaphore, les nouveaux cyber-cadenas bleus paraissent incarner une réalité certes dématérialisée mais connectée et ancrée dans le prolongement de la sécurité du monde physique. Des lignes et des nœuds évoquent les circuits imprimés des cartes mères et les arrière-plans, souvent floutés comme s'il s'agissait d'une photographie avec une faible profondeur de champ, laissent deviner ce qui est protégé : une ville, un humain dans un costume-cravate ou encore une maison. Ces nouveaux cyber-cadenas, à la fois vintage et ultramodernes, paraissent paradoxalement presque plus réels que le « cadenas-IRL » de la couverture du livre tant ils sont omniprésents dans cette iconographie : même s'il semble difficile de s'en rapprocher, on peut se demander s'ils n'existeraient pas réellement.

D'ailleurs, sur plusieurs de ces images, on voit un doigt tendu vers ces cadenas, comme si on pouvait les toucher, voire cliquer dessus. Ces représentations ne sont pas sans évoquer la peinture du plafond de la chapelle Sixtine de Michel-Ange, où les doigts d'Adam et de Dieu se tendent l'un vers l'autre sans pour autant se toucher. Au-delà de l'anecdote, ces représentations de la sécurité informatique semblent s'ancrer dans une tradition établie de « déification » du numérique. Dans son livre « Les oubliées du numérique », Isabelle Collet (2019) explore l'hypothèse qu'en créant des ordinateurs à leur image, les « pères fondateurs »¹¹ de l'informatique nourrissaient un

11 L'histoire a surtout retenu le nom d'hommes, alors que de nombreuses femmes ont participé au développement des ordinateurs et qu'il n'y a pas si longtemps, avant que l'informatique

fantasme d'auto-engendrement empreint d'une vision machiste du monde. Les inventions de ces « pères fondateurs », qu'il s'agisse de machines ou de théories, étaient tout entières dédiées à l'imitation de Dieu. Collet précise même que le mot « ordinateur » a été choisi comme traduction du mot anglais « *computer* » justement à cause de la référence divine – l'« Ordinateur » étant « Celui qui met de l'ordre dans le monde ».

Prolongeant la métaphore divine, la couleur bleue présente dans la plupart des représentations du cyber-cadenas n'est pas anodine non plus. En effet, le bleu est l'image du ciel, mais aussi de la puissance électrique et d'une certaine idée d'un corps dématérialisé, des « avatars bleus » comme le dit Antonio Casilli (2005). Toutefois, et contrairement à ce qu'affirme Casilli, je ne crois pas qu'il s'agisse de corps sans genre, car le bleu est aussi encore la couleur associée aux garçons. Cela s'observe d'ailleurs sur les résultats de recherche de l'illustration 3 (p. 35) : de nombreux objets associés au genre masculin sont présents : costumes, cravates, mains qui semblent être celles d'hommes.

Le solide et high-tech cyber-cadenas bleu représente donc une nouvelle icône de la sécurité informatique – « icône » dans tous les sens du terme. Il s'agit aussi d'un symbole bien pratique pour évoquer la sécurité de manière générique et en faire une « boîte noire » que l'on peut montrer, critiquer, vendre. Symboles évidents de fermeture, de cloisonnement, de protection physique, ces cyber-cadenas font en effet l'impasse sur les processus complexes qui font et défont la sécurité au quotidien, et mettent l'accent sur la sécurité en tant que produit fini.

Cela concorde d'ailleurs avec le champ lexical utilisé pour parler de sécurité informatique. En effet, les mots eux-mêmes renvoient à un imaginaire viril, violent et guerrier : on parle d'attaque et de défense, d'exploitation de vulnérabilité [*vulnerability exploits*], de risques, de modèles de menace [*threat models*], d'actes (et de *hacks*)

connaître une croissance fulgurante dans les années 70-80, la plupart des programmatrices étaient des femmes (Abbate 2012; Collet 2019).

criminels, de *firewall*, de *redteam mission*¹², de concours « *capture the flag* »¹³, et bien entendu, de (cyber)guerre. Ces termes renvoient à des valeurs telles que la puissance, la solidité et l'héroïsme.

Or, ce vocabulaire guerrier et agressif ne permet pas de décrire ce que j'ai observé lors de mon ethnographie au sein d'équipes de développement ou lors de mes entretiens. En effet, inspiré par les travaux féministes en philosophie morale qui m'ont enjoint à prêter attention aux interprétations alternatives possibles de la sécurité informatique et aux « voix différentes » (Gilligan 2017), je me suis concentré sur la caractéristique d'OpenPGP et ses dérivés qui était la plus flagrante : sa fragilité. Lors de mes observations, j'ai effectivement constaté que la sécurité était largement abordée sous l'angle des vulnérabilités et de la faillibilité des systèmes informatiques (Denis 2009; 2012). C'est lors d'une visite de la cave d'un-e des membres de l'équipe de développement de Sequoia, une implémentation d'OpenPGP, que je me suis le mieux figuré ce mélange de fragilité et de bricolage qui pouvait, malgré tout, incarner la sécurité informatique. C'est en effet dans cette cave que j'ai rencontré Cookie Jar, l'ordinateur continuellement allumé et connecté à Internet sur lequel se passent tous les tests automatiques du code de l'implémentation (voir illustration 4 ci-dessous).

12 Les *redteams missions* sont un exercice militaire consistant à simuler grandeur nature une attaque sur des infrastructures pour en tester les défenses.

13 Cela vient des jeux vidéos de guerre où le but est de s'introduire dans la base ennemie pour en capturer un drapeau et le ramener chez soi. Dans le monde du hacking, il s'agit de découvrir une vulnérabilité intentionnelle cachée dans un système informatique.



Illustration 4 : Cookie Jar, en équilibre sur un meuble de la cave d'une des codeuses de Sequoia. En tant que « membre » de l'équipe de développement, Cookie Jar faisait un immense travail... et donnait aussi beaucoup de travail à ses collègues humains qui devaient régulièrement s'en occuper. Source : photographie personnelle, juin 2021.

Dans une perspective centrée sur le soin de la sécurité, mon travail analyse différentes manières de prendre soin. Le « *care* », en anglais, renvoie en effet à l'idée de soin [*to take care of*], mais également à l'idée de porter de l'attention sur quelque chose [*to care about*] et d'être sentimentalement attaché à quelque chose [*to care for*]¹⁴. Le concept de *care* est attaché à un champ de la philosophie morale avec les éthiques du *care* (voir notamment Gilligan 2017; Tronto 2009; et la très bonne synthèse de Brugère 2017) et à un champ des études socio-anthropologiques des sciences et des techniques (Mol, Moser, et Pols 2010; Puig de la Bellacasa 2010; Kocksch et al. 2018). L'un des apports de ces recherches récentes en socio-anthropologie des sciences et des techniques est le renversement de perspective que ces études proposent : ce n'est pas la sécurité informatique qui prend soin de nous, humains, mais ce sont nous, humains, qui prenons soin de la sécurité informatique.

¹⁴ Cette polysémie du terme existe en partie aussi en vieux français où « soin » pouvait aussi signifier « s'inquiéter » pour quelque chose ou quelqu'un (Denis et Pontille 2022).

C'est un processus actif qui met l'accent sur les improvisations constantes qu'implique le travail de soin en général, et le travail de soin des technologies en particulier : que ce soit pour bidouiller les privilèges d'accès dans un cloud (Kocksch et al. 2018) ou faire face à des comportements imprévus du code (Alcaras et Larribeau 2022), il semble que le bricolage soit au centre de ce travail.

J'ai ainsi porté une attention sensible à ce travail humble et routinier de « bricolage persistant dans un monde plein d'ambivalences complexes et de tensions mouvantes » (Mol, Moser, et Pols 2010, 14, ma traduction). Ce travail est entrepris par des ingénieur·e·s, des codeuses et des cryptographes qui s'occupent d'OpenPGP. Le travail d'écriture est un des dispositifs centraux de leurs activités en considérant qu'« [é]crire un logiciel ne se résume pas à produire du nouveau code » (Alcaras et Larribeau 2022, 4). Il s'agit également de comprendre le code déjà existant et de s'en occuper de sorte qu'il continue d'être fonctionnel (Couture 2012). Il s'agit aussi d'écrire des éléments qui entourent le code : commenter une ligne de code et commenter un commit, ou coder un autre programme qui teste ce code, écrire, relire et corriger le manuel d'utilisation, décrire le fonctionnement du protocole, écrire et mettre à jour le site Internet du logiciel. De même, le travail des informaticien·ne·s implique de lire énormément : lire et comprendre les résultats des tests, les rapports de bogues, les propositions de modifications, le code des autres codeuses, les questions, critiques ou suggestions des utilisateurices, le « putain de manuel d'utilisation »¹⁵, les dernières recherches scientifiques sur le sujet, les emails des listes de diffusion, etc.

Ainsi, le travail de sécurité ne consiste pas seulement dans la construction d'un solide cadenas, mais dans sa maintenance dans le temps. Comme je le montre dans cette thèse, la notion de durée est centrale dans la compréhension de la sécurité informatique : la sécurité informatique prodiguée par un système de chiffrement n'a en effet de sens que si l'on considère la durée de cette sécurité. De plus, et malheureusement, la construction d'un solide cyber-cadenas résistant au temps ne peut jamais

15 « *Read The Fucking Manual* » (RTFM) est une injonction omniprésente dans les forums des communautés Linux : https://fr.wikipedia.org/wiki/RTFM_%28expression%29.

aboutir et OpenPGP en est un très bon exemple : cette technologie n'a jamais arrêté d'évoluer car elle a toujours été cassée – ou plutôt, elle n'a jamais cessé d'être réparée et préparée à ce qui allait se présenter à l'avenir. Ceci est très éloigné de l'existence d'un cadenas IRL qui certes évolue, mais à sens unique vers une dégradation inéluctable : il rouille, se grippe et finalement casse – j'ai même réussi un jour à casser une clé à l'intérieur de la serrure du cadenas de mon vélo et ce n'était pas rigolo. Certaines activités d'entretien permettent parfois de retarder ce processus et il est probable que si j'avais mieux et régulièrement lubrifié la serrure de mon cadenas de vélo, ma clé n'aurait pas connu un tel sort. Cependant, il y a là une différence fondamentale bien que triviale entre le cadenas IRL et le « cyber cadenas » : il n'est pas possible d'appliquer une mise à jour sur un cadenas IRL. De plus, une fois cassé, par exemple comme mon cadenas scié, l'objet est détruit et n'est plus fonctionnel. En m'inspirant de Steven Jackson (2014) et de son appel à développer « une pensée du monde cassé », je dirais qu'un cyber-cadenas, s'il devait exister, serait non seulement nécessairement incomplet et toujours plus ou moins cassé, mais surtout constamment en train d'être réparé. Ainsi, contrairement au cadenas IRL, celui de la sécurité informatique est sans cesse simultanément sur mon vélo, dans l'atelier de réparation, sur la table de travail des designers et bien sûr aussi chez la malicieuse personne qui l'étudie pour pouvoir voler mon vélo.

Par ailleurs, il n'y a pas *un seul* cadenas PGP car il s'agit d'une technologie plurielle. Contrairement à un cadenas que je peux tenir entre mes mains, que je peux manipuler, que je peux objectiver, le cadenas PGP peut représenter de multiples choses, à différents niveaux d'ailleurs. Le programme de Zimmermann Pretty Good Privacy – qui a cessé d'être utilisé – a donné son nom à une norme de l'IETF : OpenPGP. De cette norme sont nées différentes implémentations, dont the Gnu Privacy Guard (GnuPG, ou simplement GPG), OpenPGP.js et Sequoia PGP. Le « cadenas PGP » n'est donc pas un objet unique et isolé¹⁶.

16 L'analogie me fait penser à une situation cocasse : même s'il est vrai que la cryptographie a notamment été utilisée par un nombre important d'amants et d'amantes (Kahn 1996), je ne crois pas que beaucoup de couples accrochent un cyber-cadenas sur les ponts pour espérer immortaliser leur amour.

Les contradictions de la métaphore du cadenas se révèlent aussi d'une grande aide pour toucher de manière intuitive une particularité importante du numérique : contrairement à un cadenas IRL, une technologie de sécurité informatique n'est pas un objet fixe mais est constituée d'éléments qui évoluent sans cesse. Chaque jour, des codeurs et codeuses travaillent des lignes de code. Ces changements sont sauvés localement, mais sont aussi sur des plateformes de partage de code GIT comme Git-Lab ou GitHub. Ces fragments de code comprennent des ajouts de fonctionnalités et des corrections du code déjà écrit, ainsi que des commentaires. De temps en temps, plusieurs de ces changements sont agrégés pour donner naissance à une nouvelle version du code – ou devrais-je dire des nouvelles versions du code (car le code doit s'adapter à différents contextes, par exemple en fonction du système d'exploitation de la machine sur laquelle il doit fonctionner). Parmi les fragments de code que j'ai croisés lors de ma recherche, certains n'ont jamais quitté l'ordinateur de la personne qui les avait écrits, d'autres ont pris place dans le code de logiciels que j'utilise et d'autres encore ont déjà été remplacés. Autrement dit, ça bouge tout le temps, ça ne se laisse pas saisir, et surtout ça n'est jamais aussi simple que ça. Ces versions sont ensuite généralement installées sur les machines des utilisatrices, même si ce n'est pas toujours le cas et certaines personnes continuent d'utiliser des versions anciennes pour différentes raisons.

Il n'y a donc pas un seul objet et une seule version, mais une multitude d'états d'un même objet. David Bozzini et moi-même avons appelé cette bizarrerie du numérique « instance »¹⁷ (Besençon et Bozzini 2020). L'instance de GnuPG que je fais marcher sur mon ordinateur privé n'est pas identique à celle que j'utilise sur l'ordinateur prêté par l'Université de Fribourg, ni à celle utilisée par les codeurs et codeuses qui travaillent déjà à la prochaine version. Ce que j'arrive à faire avec l'une de ces instances n'est pas forcément possible avec l'autre ; de même, une vulnérabilité peut les affecter toutes, ou seulement une seule.

17 Nous nous sommes inspirés de la terminologie des sciences informatiques, mais avons détourné ce mot ayant une signification technique précise pour l'adapter à notre besoin de chercheurs en sciences sociales. Dans cette thèse, le mot « instance » aura la signification que Bozzini et moi-même lui avons donné, et non la signification émiqque issue des sciences informatiques.

La sécurité, dans cette perspective, n'est pas une chose, ni un produit, ni même un processus identifié et unique. Il s'agit plutôt du fruit d'un agencement bricolé de pratiques ordinaires et banales de réparation, de maintenance et d'anticipation. L'idée du mouvement, dans le sens de transformation, évolution, est également centrale pour appréhender la sécurité informatique : d'où vient la technologie, où va-t-elle ? Par ailleurs, ces pratiques de soin sont régulièrement interrompues par des événements singuliers, comme l'implémentation d'une nouvelle fonctionnalité, la découverte d'un bogue ou la divulgation d'une vulnérabilité. En tant que pratique de soin, la sécurité n'a rien d'extraordinaire ; elle doit pourtant régulièrement faire face à l'extraordinaire. S'enchevêtrent alors plusieurs temporalités, plusieurs rythmes qui façonnent le produit finalisé – qui n'est d'ailleurs jamais « final » car exigeant sans cesse de nouvelles opérations de maintenance. Interroger le mouvement, la durée et le rythme est donc nécessaire pour comprendre comment s'articulent les différentes logiques de la sécurité informatique, entre l'ordinaire et l'extraordinaire, la route et l'événement, la lenteur et l'urgence, les contraintes du passé et les menaces de l'avenir.

1.4. Prendre soin de la sécurité informatique

En suivant la piste ouverte par l'école de Copenhague et sa théorie de la sécurisation qui considère la sécurité comme un processus discursif (Buzan, De Wilde, et Wae-ver 1998; Balzacq 2011; 2015; Balzacq et Dunn Cavelty 2016), j'ai examiné comment la sécurité était tour à tour faite et dé faite, proclamée et contredite, codée et décodée. J'ai notamment analysé des périodes de crises, par exemple lorsque des vulnérabilités étaient découvertes et divulguées, ou lorsque des bogues interrompaient le travail des développeuses et celui des utilisatrices, ou encore lorsque certaines décisions étaient prises qui contrariaient les codeuses et ingénieur·e-s de la sécurité informatique. L'attention que j'ai portée sur les temporalités de la sécurité m'a fait apercevoir deux perceptions différentes de la durée : il y a d'une part l'horizon temporel de la sécurité, que les développeuses et les utilisatrices voudraient

éternel, ou du moins existant le plus longtemps possible et avec le moins d'interruptions possible ; et d'autre part les inévitables césures de l'insécurité, qui interrompent ce temps et qui bousculent l'agenda des codeuses. Dans cette thèse, je donne plusieurs exemples de rupture dans la temporalité de la sécurité. Ces ruptures sont notamment mises en scène par une rhétorique de crise typique de la sécurité informatique : « vous vous croyiez en sécurité en utilisant tel et tel logiciels ; mais nous les avons étudiés et y avons trouvé plein de bogues. » Ce genre de formulations sont à considérer avec sérieux non seulement pour ce qu'elles affirment – des failles existent –, mais aussi pour ce qu'elles font : rythmer le développement des technologies. Dans cette thèse, je montre comment l'insécurité est centrale dans les processus de sécurisation et comment elle est un moteur essentiel pour l'évolution et l'innovation technologique.

Cependant, la crise n'explique pas tout. Malgré le grand intérêt des sociologues et anthropologues pour la figure de la panne et de la réparation (Graham et Thrift 2007; Bourrier et Nova 2019), une autre temporalité qui n'était pas basée sur la rupture rythmait le quotidien des personnes avec qui j'ai collaboré pour cette recherche : celle du soin ordinaire dédié aux technologies. En effet, malgré les discours *mainstream* sur l'innovation et les promesses d'avenir radieux qui les accompagnent, les technologies s'usent et ont elles aussi besoin d'assistance et de soin (Puig de la Bellacasa 2010; Mol, Moser, et Pols 2010; Kocksch et al. 2018; Denis et Pontille 2022). Prendre soin d'OpenPGP n'a rien d'évident car cette technologie est liée à une infrastructure elle-même vieillissante. Au contact des personnes intéressées, j'ai observé une résistance à cette obsolescence désirée et un acharnement à prolonger la durée de vie d'OpenPGP. Ainsi, ni les critiques récurrentes à l'égard d'OpenPGP, ni la divulgation de vulnérabilités informatiques, ni les recommandations d'expertes et experts ne sont venues à bout de l'attachement et l'entêtement de ces codeuses, bien décidées à faire durer cette technologie.

« Faire durer les choses » : c'est justement ainsi que Jérôme Denis et David Pontille définissent la maintenance dans leur livre « Le soin des choses » (2022). Ils ont identifié quatre horizons temporels de la maintenance :

- La prolongation, qui est l'une des formes les plus courantes de la maintenance, souvent appelée « entretien » et visant à prolonger la durée de vie des choses, par exemple en prenant soin d'un vélo. Dans de nombreux cas, l'intégrité physique de la chose maintenue importe bien moins que la préservation de sa fonction ;
- La permanence, qui est la forme la plus ambitieuse de la maintenance puisqu'elle vise à rendre éternelle la chose maintenue ; les auteurs donnent notamment l'exemple de la conservation du corps de Lénine ;
- Le ralentissement qui est typique du soin porté aux œuvres d'art qu'on ne peut rendre permanentes mais dont on tente de retarder la disparition que l'on sait pourtant inéluctable ;
- et finalement l'obstination, où la chose en question s'obstine à résister au temps et dure plus longtemps que prévu, tenant ainsi engagées les équipes en charge de sa maintenance. L'exemple d'une sonde spatiale qui a continué à transmettre des informations des années après la fin de sa mission première est une illustration de cette obstination à durer.

Ces quatre approches semblent toutes apporter des pistes intéressantes pour la compréhension de la maintenance d'OpenPGP. La prolongation met l'accent sur « l'expérience quotidienne, presque insignifiante » (Denis et Pontille 2022, 185) du soin apporté au code et aux spécifications de la norme. Cette pratique est particulièrement visible sur les plateformes de partage de code, comme GitHub ou GitLab, ainsi que bien entendu auprès des personnes qui codent au quotidien. La permanence met l'accent sur la volonté de rendre les algorithmes de chiffrement indéchiffrables de manière permanente. La permanence du secret est en cela un horizon vers lequel tend la sécurité informatique – et c'est cette permanence que viennent régulièrement interrompre les nouvelles vulnérabilités et innombrables bogues qui ne cessent

d'émerger. Face à ce déclin du code à cause de ses propres failles d'une part, et des nouvelles techniques en cryptanalyse d'autre part, le ralentissement est un objectif pragmatique qui s'observe aussi au quotidien. Même le dernier cas est intéressant à étudier : en effet, malgré leur obsolescence, certaines choses s'obstinaient à rester. Comme je l'analyse dans la première partie de la thèse, « s'accommoder du passé », certains paramètres, certains algorithmes et certains formats persistaient à survivre. C'est par exemple le cas de l'usage d'OpenPGP pour signer et authentifier le code des logiciels open source. Il est d'ailleurs probable que sans cet usage intensif du protocole à des fins d'authentification, OpenPGP aurait été remplacé par un autre protocole de chiffrement pour les communications.

Mais si les quatre horizons temporels identifiés par Denis et Pontille permettent effectivement de décrire bien des aspects du travail de soin d'OpenPGP, un aspect me semble absent : la manière dont l'avenir est anticipé afin de produire ce que j'appelle la « résilience prospective » (voir notamment la dernière partie de cette thèse « la durée de l'avenir », p. 296). Le souci de l'avenir est continuellement présent, mais dans une approche très éloignée de la rhétorique classique de l'innovation. Ici, il s'agit non pas de promettre l'avenir, mais de le soigner – ou du moins de faire en sorte qu'il soit le plus résilient possible quand de nouvelles vulnérabilités apparaîtront. Par exemple, les développeuses que j'ai interviewées ont régulièrement insisté sur le fait qu'elles codaient en ayant conscience que leur code devait protéger les données des utilisatrices durant les dix ou les vingt prochaines années. Leur objectif est de produire une technologie qui puisse faire face à toutes les incertitudes de l'avenir. Cette dimension est particulièrement importante dans ma recherche et la notion de projet-sécurité que je définis dans le troisième chapitre (« Alice et Bob aux pays des bogues », p. 104 » permet de parler de la sécurité comme un travail de soin dirigé conjointement vers le présent et l'avenir.

Je définis le projet-sécurité comme l'ensemble de représentations induites par un souci de sécurité. Cette notion permet de comprendre ce qui est fait au nom de la sécurité sans avoir à définir de manière normative cette sécurité. Il permet aussi d'ana-

lyser la production des technologies de sécurité informatique en se focalisant non pas sur les caractéristiques techniques mais sur les attentes que ces technologies nourrissent. Autrement dit, cette notion permet de saisir la sécurité non pas comme un état de fait mais comme un projet pour le présent et pour l'avenir que les actrices tentent de mettre en place. En ce sens, l'avenir est sans cesse présent dans le développement et la maintenance des technologies de sécurité, car il en constitue le projet. J'insiste ainsi sur la dimension à la fois propositionnelle et prospective de la sécurité : prospective, car il s'agit d'anticiper ce qui pourrait advenir et propositionnelle, car ce ne sont jamais que des propositions qui sont faites.

La prégnance de l'avenir dans les pratiques des personnes qui maintiennent des technologies de sécurité informatique est particulièrement saillante lorsque l'on s'attarde sur la manière dont l'avenir est préparé. En effet, face à la fondamentale incertitude quant à ce que l'avenir prépare, et à la certitude non moins unanime que des vulnérabilités finiront bien par émerger, les codeuses d'OpenPGP ont mis en place ce que j'ai appelé des stratégies afin d'implémenter la résilience prospective de leur technologie. Autrement dit, leur travail ne consiste pas seulement à implémenter un logiciel de sécurité informatique et à continuellement le corriger à mesure que des bogues apparaissent, mais également à faciliter les futures corrections qui devront être implémentées. Concrètement, il s'agit d'avoir une technologie aussi compréhensible que possible, et donc de faire en sorte que les futures codeuses puissent disposer d'une documentation aussi complète que possible. Il s'agit aussi de laisser suffisamment de « place » dans le code afin de pouvoir ajouter en temps voulu de nouveaux algorithmes de chiffrement, de nouveaux paramètres, de nouvelles capacités. Autrement dit, il s'agit de préparer les futures réparations et ajustements de la technologie. Cette notion de « résilience prospective » est développée dans la troisième et dernière partie de cette thèse.

1.5. Interroger le mouvement, le rythme et la durée de la sécurité

Revenons encore une fois au cadenas PGP présenté sur la couverture du livre de Simson Garfinkel (1995). Comme je l'ai déjà suggéré, cette image permet de mettre facilement en évidence une différence entre le cadenas IRL et le cyber-cadenas : ce dernier, contrairement à un cadenas IRL qui s'use, rouille, se grippe, le cyber-cadenas ne s'altère pas à cause de sa matérialité qui s'érode avec le temps, mais à cause du fait que le monde change autour de lui. Autrement dit, si on ne peut plus utiliser le programme PGP original de Zimmermann, ce n'est pas parce que ce dernier est brisé, rouillé ou perdu, mais parce qu'il n'est plus adapté à l'environnement actuel. Il fonctionne sur des systèmes d'exploitation qui ne sont plus utilisés, il n'est pas compatible avec les logiciels actuels et il utilise un algorithme de chiffrement qui comporte de nombreuses vulnérabilités connues. En revanche, la technologie qui est issue de ce logiciel originel demeure utilisée trente ans plus tard. Elle partage avec le programme de base certaines idées, comme les fonctions de chiffrement et d'authentification de différents fichiers numériques ou l'usage d'une architecture décentralisée ; mais il ne reste probablement aucune ligne de code du programme original dans les implémentations actuelles. Ce constat pose la question du vieillissement des technologies numériques et de la manière dont elles s'usent et sont entretenues dans le temps.

Focalisée sur les différents temps de la sécurité informatique, cette thèse explore les modalités du soin qui est porté à une technologie numérique. Mon analyse identifie trois dimensions de ces temps de la sécurité : celle du **mouvement**, celle du **rythme** et celle de la **durée**. Une attention portée sur ces trois dimensions permet de comprendre ce que signifie prendre soin d'une technologie numérique : cela implique de faire face à la fois à des contraintes issues du passé de cette technologie et des bogues qui ne cessent jamais d'émerger, tout en faisant en sorte que cette technologie reste pertinente, compatible et la plus sûre possible dans le contexte à venir. Ce thèse contribue à appréhender la sécurité informatique en tant qu'agencement de

pratiques et de discours cherchant à répondre aux besoins actuels et futurs des utilisatrices des technologies numériques.

Ma problématique chemine donc par l'analyse du passé, du présent et de l'avenir, en concédant une grande attention à l'entrelacement de ces différents horizons temporels, ainsi qu'aux différents tempos qui rythment ce travail de soin et aux durées que la sécurité met en perspective. Après cette introduction et un chapitre méthodologique (p. 57), ma démonstration suit un raisonnement en quatre étapes, orientées respectivement vers les questions suivantes :

- **Comment des ingénieur·e·s et des codereuses pensent et traduisent la sécurité informatique en protocoles et en codes informatiques ?** Cette question est abordée dans le troisième chapitre intitulé « Alice et Bob aux pays des bogues » (p. 104). Ce chapitre décrit la vision émique des participant·e·s à ma recherche, autour des figures d'Alice, Bob et Ève, personnages fictifs et néanmoins emblématiques de la cryptographie numérique. Je propose ensuite une analyse anthropologique des processus et pratiques de sécurité informatique, en mobilisant d'une part les théories de la sécurisation et d'autre part les études sur le soin des choses. Je montre comment la sécurité est conçue par les participant·e·s à ma recherche à travers différentes projections qu'ils et elles cherchent à réaliser. J'ai nommé ces projections les « projets-sécurité » des unes et des autres. Cette notion permet de penser l'articulation entre héritages et projections, scénarios idéaux et décisions pragmatiques, contraintes et opportunités.
- **Comment le passé de ces technologies et de l'écosystème dans lequel elles s'insèrent contraint-il le travail de développement et de maintenance ?** Cette question cherche à mieux cerner le mouvement de la sécurité informatique, pris en étau entre l'inertie de l'infrastructure et la complexité croissante du code informatique. Elle contribue à comprendre comment le passé des technologies numérique et l'écosystème dans lequel elles s'insèrent s'enlisent réciproquement, et comment le travail de développement et de

maintenance consiste à lutter contre l'immobilité. La première partie de cette thèse (« L'héritage du passé en mouvement », p. 146) s'interroge sur ce que vieillir veut dire pour une technologie numérique. Je commence par questionner comment OpenPGP est une technologie « enlisée » dans une vaste infrastructure que beaucoup considèrent comme obsolète (chapitre 4, « Enlissement et ancrage infrastructurels », p. 150). Je montre aussi comment cet écosystème est aussi une source d'opportunités pour les technologies, malgré les nombreux blocages et contraintes. Le chapitre 5 (« Hériter, faire vieillir et transmettre le code », p. 186) examine quant à lui comment le code informatique sert de support à un dialogue entre différentes générations de codeureuses. À la fois hérité du passé et transmis aux générations qui suivent, le code informatique est l'élément central des technologies numériques. Je commence par présenter à quoi ressemble un code source en prenant en exemple un module existant du code de Sequoia. Je montre ensuite pourquoi ce code vieillit et pourquoi certaines personnes trouvent qu'il vieillit mal. Ainsi, cette partie expose en quoi le passé est une source importante d'inconfort pour les développeuses.

- **Comment les pratiques ordinaires de développement et de maintenance des technologies numériques sont-elles sans cesse interrompues par des événements extraordinaires ?** La deuxième partie « Les rythmes du présent » (p. 216) est dédiée aux différents rythmes de la vie des technologies numériques. À travers l'exemple de la série de vulnérabilité EFAIL divulguée publiquement en mai 2018, je décris comment l'urgence et la lenteur cohabitent, comment l'ordinaire et l'extraordinaire s'entremêlent. Le chapitre 6 (« EFAIL et le temps de l'urgence », p. 226) présente la controverse suscitée par la divulgation publique d'EFAIL. Je montre comment est née cette controverse et quels sont les événements qui ont été les plus contestés. J'insiste sur les différents niveaux d'incertitudes et le sentiment d'urgence qui en découlait. Le chapitre 7 (« Lenteur et consensus à l'IETF », p. 256) étudie quant à lui un autre versant

des vulnérabilités d'EFAIL : la manière dont le groupe de travail de l'IETF s'en est saisi pour tâcher de corriger ces vulnérabilités sur une échelle de temps beaucoup plus longue que le temps de l'urgence étudié dans le chapitre précédent. Durant des années, le groupe de travail OpenPGP de l'IETF a en effet négocié différents aspects des spécifications du protocole afin de pallier ces vulnérabilités sur le long terme. Cette deuxième partie permet donc de comprendre de quoi est fait le quotidien des codeuses et ingénieurs qui prennent soin d'OpenPGP. C'est une plongée dans le présent de la technologie qui permet de comprendre ce travail de soin qui devient très concret et spécifique.

- **Que font les développeuses et ingénieurs pour préparer leur technologie à survivre aux menaces à venir ?** Cette question permet d'aborder la sécurité sous l'angle de la durée en présentant les pratiques d'anticipation qui sont au cœur de la sécurité informatique. La troisième partie, « La durée de l'avenir » (p. 296) décrit la manière dont l'avenir contraint et stimule le développement d'OpenPGP. J'y propose les notions de « (p)réparation de l'avenir » et de « résilience prospective » pour décrire la manière dont l'anticipation de l'avenir est omniprésente dans le travail de soin d'une technologie vieillissante (chapitre 8 « Penser les fragilités à venir pour panser l'avenir », p. 300). Je propose une critique des discours stéréotypés à propos de l'innovation et suggère une vision plus nuancée de ces pratiques de création. Je soutiens d'une part que l'innovation est un moyen de maintenir la technologie, et d'autre part que pour perdurer, une invention a nécessairement besoin de maintenance. Ainsi, innovation et maintenance sont les deux faces d'une même médaille. Dans le chapitre 9 (« Équiper OpenPGP face aux ordinateurs quantiques », p. 326), j'étudie la menace, considérée par beaucoup comme existentielle, de l'arrivée des ordinateurs quantiques. Décrire comment la communauté OpenPGP se saisit de cet enjeu me permet de montrer comment

l'avenir est un domaine très incertain auquel il faut toutefois se préparer activement.

Finalement, j'aimerais clore cette introduction en brossant un panorama des différentes actrices de la communauté. Ce panorama, qui est affiné au cours des chapitres qui suivent, a pour but de situer les actrices principales dans un champ complexe et ramifié, et d'apercevoir d'ores et déjà quelques lignes de tension dues à des relations asymétriques de pouvoir, des divergences d'opinion ou des conflits personnels. Ce panorama présente les implémentations principales d'OpenPGP, ainsi que certains logiciels ou matériels liés à OpenPGP. La plupart des points présentés seront bien entendu détaillés, affinés et approfondis par la suite. Ce panorama n'est donc qu'une première étape pour apercevoir les différents axes autour desquels gravite la « communauté OpenPGP ».

Panorama de la communauté OpenPGP

L'écosystème OpenPGP est composé de différentes technologies. Il y a tout d'abord les logiciels qui sont destinés à être utilisés par les utilisatrices finales. Ce sont les logiciels qu'il est possible d'installer sur son ordinateur ou son téléphone et d'utiliser directement. Ces logiciels implémentent la norme OpenPGP pour signer des paquets ou chiffrer des emails : on les appelle **des implémentations**. Parmi elles, j'ai choisi d'en présenter six principales :

- **Pretty Good Privacy** (PGP), le programme de Zimmermann, n'est plus utilisé aujourd'hui (en 2023). La première version a été publiée en juin 1991 et il a connu de multiples versions depuis. Certaines étaient open source, d'autres étaient payantes. Le programme a d'ailleurs été racheté plusieurs fois, avant d'être petit à petit remplacé par GnuPG, un logiciel libre et gratuit développé dès 1998.
- **Gnu Privacy Guard**, ou GnuPG (ou encore GPG), est un programme en ligne de commande codé en C. GnuPG sert également de base à la librairie GnuPG Made Easy (GPGME), utilisée par de nombreux clients email. Son histoire remonte à 1997 lorsque Werner Koch a codé une version open

source du programme PGP de Zimmermann. GnuPG a son siège en Allemagne où est employé le noyau dur de l'équipe de développement du projet, même si des contributeurices se trouvent dispersé-e-s sur tout le globe. Le code source se trouve sur le site de GnuPG (*GnuPG, GIT). GnuPG est certainement l'implémentation la plus utilisée de l'écosystème, pour au moins trois raisons. Premièrement, il s'agit d'une implémentation historique qui a été la première et, pendant longtemps, la seule, à être complètement libre et gratuite (certaines versions de PGP étaient en effet payantes). La deuxième raison est une conséquence directe : sa librairie GPGME est utilisée dans de nombreux logiciels, dont Enigmail qui a été le programme pour les utilisateurices finales le plus utilisé jusqu'à récemment (voir ci-dessous). La troisième raison est due à l'usage d'OpenPGP pour authentifier des logiciels open source. Comme l'annonce le site Internet du projet Debian, « le logiciel le plus souvent utilisé pour travailler avec OpenPGP dans Debian est GnuPG »¹⁸. GnuPG est si populaire que pour de nombreuses personnes, l'implémentation correspond à la norme de l'IETF. Par exemple, lors des *cryptoparties* auxquelles j'ai participé, OpenPGP était systématiquement présenté à travers GnuPG et Enigmail (qui utilisent ce code).

• **ProtonMail** est le service email de l'entreprise Proton, qui a des bureaux en Suisse et aux États-Unis. ProtonMail est un service d'emails sécurisés qui utilise la norme OpenPGP pour chiffrer et signer les messages. Proton est probablement l'entreprise la plus importante de l'écosystème OpenPGP en termes de nombre d'employé-e-s (plus d'une centaine à Genève) et de moyens financiers, même si OpenPGP n'est qu'un projet parmi de nombreux autres. Proton est la mainteneuse officielle de la librairie OpenPGP.js, qui implémente OpenPGP en JavaScript, un langage informatique destiné à être exécuté au sein de navigateurs web. Cela rend possible l'utilisation d'OpenPGP sans avoir à installer un programme spécifique sur l'ordinateur car le code est interprété directement par le navigateur au sein d'une application web. Le code de cette librairie est open

18 <https://wiki.debian.org/OpenPGP>, ma traduction.

source et disponible sur GitHub ; il est utilisé par de nombreux autres projets, dont Mailvelope, GlobalLeaks et FlowCrypt.

- **Sequoia PGP** est la dernière venue de ces implémentations : il s'agit d'une librairie open source débutée en 2017 qui implémente OpenPGP en RUST, un nouveau langage informatique axé sur la performance et la sécurité. Sequoia PGP vise explicitement à créer une implémentation plus performante, légère et simple à utiliser que GnuPG. Sequoia PGP emploie une demi-douzaine de personnes en Europe mais reçoit également des contributions de codeuses d'Amérique du Nord et d'Asie. Son code source se trouve sur sa plateforme GitLab (*GitLab, Sequoia). Il est important de noter que Sequoia a été fondé par trois anciens employés de GnuPG qui avaient démissionné parce qu'ils ne s'identifiaient plus avec la direction du projet. Il en résulte une certaine tension entre Sequoia et GnuPG, non seulement pour des raisons techniques, mais également interpersonnelles.

- **Thunderbird** est un client de messagerie email, libre et gratuit, issu du projet Mozilla. C'est l'un des clients email libres et gratuits les plus populaires et il est installé par défaut sur de nombreuses distributions Linux (notamment Ubuntu). C'est un logiciel extensible qui peut recevoir de nouvelles fonctionnalités à l'ajout d'extensions. Enigmail (voir ci-dessous) était l'extension préconisée pour utiliser OpenPGP avec Thunderbird, mais depuis la version 78 de Thunderbird, parue en 2020, le chiffrement de bout en bout et l'authentification des emails avec le protocole OpenPGP est implémenté nativement dans Thunderbird¹⁹, et Enigmail n'est plus nécessaire. Thunderbird utilise la librairie RNP, écrite en C++, qui est développée depuis 2017²⁰.

- **Enigmail** est une extension libre et gratuite de Thunderbird qui, depuis la publication de la version 78 du client email, n'a plus de raison d'être car Thunderbird n'utilise plus que l'implémentation native d'OpenPGP. Durant des années, Enigmail était l'une des implémentations les plus utili-

19 <https://www.thunderbird.net/fr/thunderbird/78.0/whatsnew/>.

20 <https://www.rnpgp.org/>.

sées pour chiffrer et authentifier des emails, et certainement l'une que l'on recommandait le plus d'installer lors des *cryptoparties* auxquelles j'ai participé. Enigmail utilise la librairie GPGME pour implémenter les fonctions cryptographiques utilisées.

De nombreuses autres implémentations existent et une liste plus complète peut être consultée sur le site dédié du protocole²¹. Des applications, libres ou propriétaires, gratuites ou payantes, sont disponibles en de nombreux langages informatiques, et sur les principaux systèmes d'exploitations pour ordinateurs et *smartphones*. Si j'ai choisi de présenter les six implémentations ci-dessus, ce n'est en aucun cas parce que je les trouve meilleures ou plus sûres (je ne suis pas capable d'en juger), mais parce qu'elles sont les plus influentes dans l'écosystème OpenPGP.

GnuPG a un poids historique très important et, jusque vers 2020, l'orientation de la norme de l'IETF suivait presque systématiquement les choix de GnuPG. Pourtant, plusieurs changements ont eu lieu : l'implémentation de Proton-Mail, dont le projet a débuté en 2013, est devenue de plus en plus influente à mesure que l'entreprise mère Proton grandissait. Selon plusieurs personnes interviewées, c'était, en 2022, l'implémentation la plus puissante et elle avait de plus en plus de poids dans les prises de décisions. Depuis 2017, une nouvelle implémentation a aussi été très active dans l'écosystème : Sequoia. Bien qu'elle ne soit pas encore utilisée par beaucoup de services, son équipe de développement s'est fixée comme objectif non seulement de coder une implémentation qui puisse remplacer à terme GnuPG, mais également d'améliorer l'écosystème OpenPGP dans son ensemble, notamment en proposant de nombreuses initiatives au niveau de la norme de l'IETF.

Pour compléter ce panorama de l'écosystème OpenPGP, il me faut encore mentionner d'autres éléments clés de l'infrastructure. Parmi ces éléments, les **serveurs de clés** jouent un rôle fondamental : en effet, pour pouvoir chiffrer un email à une destinataire, il est nécessaire de posséder sa clé publique. Dans de nombreux cas, celle-ci peut être trouvée sur des serveurs de clés que l'on peut concevoir comme des annuaires : les recherches s'effectuent, selon

21 <https://www.openpgp.org/software/>.

les modalités, avec le nom de la personne, avec son adresse email ou avec l'identifiant de sa clé publique. Il existe différents types de serveurs, qui sont hébergés et maintenus par différentes organisations (notamment MIT, Debian, Ubuntu, OpenPGP.org).

OpenPGP a aussi un *hardware* dédié. Il s'agit par exemple des **smartcards**, qui ont la forme soit d'une carte à puce standard (comme une carte bancaire par exemple), soit d'une clé USB. Une smartcard est une carte à puce sécurisée qui permet de stocker des clés cryptographiques privées de manière sécurisée et protégée par un code PIN. Le principe est que les opérations cryptographiques se passent sur la carte et non dans l'ordinateur. Ainsi, la clé secrète n'est jamais lue par l'ordinateur et encore moins stockée dans sa mémoire ; l'ordinateur peut seulement demander à la clé d'exécuter des opérations de déchiffrement ou de signature. C'est pourquoi les clés publiques ne sont pas accessibles via la smartcard et doivent être récupérées ailleurs, par exemple sur un serveur de clé. L'accès aux données de la carte est dit « write-only ». C'est une protection supplémentaire importante car il rend impossible la copie de la clé privée. Plusieurs entreprises proposent de telles cartes, notamment Yubikey et Nitrokey. Elles n'implémentent pas directement OpenPGP mais offrent un support sur lequel les implémentations peuvent fonctionner. Pour clore ce panorama, j'aimerais également mentionner certains projets annexes de normes associées à OpenPGP, comme Autocrypt, dont il sera question plus loin. Ici encore, il ne s'agit pas d'implémenter directement OpenPGP mais d'offrir des fonctionnalités supplémentaires, ou d'adapter l'usage du protocole à d'autres cas d'utilisation.

2. Ethnographie de l'écosystème OpenPGP

23 mars 2018, à Genève. Je n'avais commencé mon doctorat qu'un mois auparavant et je visitais ma première conférence de sécurité informatique, Insomni'hack, à Genève²². Cette conférence n'est pas qu'une conférence, c'est aussi une compétition de hacking mondialement connue et organisée chaque année depuis 2008. J'étais à la fois très excité par cette nouvelle ethnographie qui commençait et intimidé par l'étendue et la diversité de ce champ auquel je ne comprenais pas grand-chose. Une connaissance m'avait toutefois fortement recommandé d'assister à la présentation de Jean-Philippe Aumasson, un cryptographe réputé. Sa présentation s'intitulait « Quantum Cyber Blockchain IoT » et son résumé tenait en une ligne sur la page du site internet :

*JP will show attacks on blockchains and their crypto, then he'll discuss quantum FUD [fear, uncertainty, and doubt] and theoretical quantum attacks (on blockchains). Also, IoT.*²³

Après avoir mangé quelques viennoiseries car j'étais arrivé bien en avance, je suis entré dans la salle de conférence, j'ai trouvé une place assez à l'avant – mais pas

22 <https://insomnihack.ch/conference-2018/>

23 J'ai laissé la citation en anglais afin de rendre au mieux le style informel de ce résumé. Voici une proposition de traduction: « Jean-Philippe expose des attaques sur les blockchains et leur cryptographie, puis il discutera du PID quantique [peur, incertitude et doute] et des attaques quantiques théoriques (sur les blockchains). Aussi l'Internet des Objets. » <https://insomnihack.ch/conference-2018/#jp>.

trop –, j’ai sorti mon calepin et j’ai commencé à prendre des notes sur ce qui m’entourait :

Beaucoup d’hommes, à peine quelques femmes, la plupart faisant partie du personnel d’accueil. Des T-shirts et hoodies avec des logos de conférences passées, dont plusieurs d’Insomni’Hack avec cette même tête de mort numérique déclinée dans différents styles. Des pirates modernes, effrayants et libres ? Aussi des logos du CCC, Black Hat, ... Je suis frappé par la dimension spectaculaire, grandiose, des « décors », et par la profusion de nourriture et de café. [...] Ah ça commence.

J’ai ensuite écouté avec un grand sérieux cette présentation. J’ai certes relevé qu’au début de son intervention, Aumasson a suggéré qu’il s’agissait juste d’une grosse blague, mais n’ayant pas compris où il voulait en venir, je me suis convaincu que j’avais mal entendu. Ce n’était d’ailleurs pas l’unique élément que je n’avais pas compris et beaucoup de choses m’échappaient. Aumasson parlait une langue que je connaissais bien, l’anglais, mais les mots qu’il utilisait ne semblaient pas toujours avoir le même sens que celui que j’avais appris. Heureusement pour moi, cette conférence a été filmée et publiée sur Youtube²⁴. Cinq ans plus tard, je la réécoute et je comprends que je n’avais vraiment pas compris certaines choses. Je comprends aussi que je n’en comprends toujours pas d’autres aujourd’hui même. Je n’avais par exemple pas compris pourquoi c’était amusant d’intituler sa présentation ainsi et en quoi consistait la blague : les mots qui composaient le titre étaient chargés de sous-entendus et en les utilisant, Aumasson se moquait de ces *buzzwords* de la sécurité informatique.

Bourdieu (2001) aurait sans doute proposé une analyse en termes de distinction et de lutte au sein d’un champ où la meilleure part du gâteau était disputée : des acteurs légitimes, cryptographes en tête, voyaient leur champ se faire envahir par des nouveaux et nouvelles venues qui détournaient la lumière des projecteurs et leur volaient la place d’honneur. Personnellement, peut-être parce que je pressentais que le vocabulaire de la lutte ne me conviendrait pas longtemps, j’ai été attiré par d’autres approches, notamment celles de l’humour et de l’attachement affectif. Ce qui m’a at-

24 « Quantum Cyber Blochain IoT - Jean-Philippe Aumasson », publiée le 28 mars 2018 par scrt.insomnihack : <https://www.youtube.com/watch?v=mrLLWW7rrls>.

tiré dans le domaine de la cryptographie n'est pas uniquement mon amour des chiffres et des problèmes, mais aussi le grand attachement que je pouvais observer chez les personnes que je fréquentais durant cette recherche. Attachement à un sujet qui les passionnait ; attachement à certaines valeurs ; attachement à certains termes²⁵ ; attachement à certains types de pratiques humoristiques ; attachement à de vieux t-shirts de conférences ayant eu lieu des années, voire des décennies en arrière ; et encore attachement à certains programmes et protocoles.

C'est peut-être d'ailleurs pour cela que j'ai décidé de m'attacher à mon tour à OpenPGP : en tant qu'objet socio-technique qui déchaîne les passions, OpenPGP présente tous les arguments clés pour enquête ethnographique dans le champ de recherche des *Science and Technology Studies*. Ce domaine de recherche aime s'intéresser aux objets sociotechniques, aux controverses qu'ils suscitent, aux laboratoires qui les fabriquent, aux connaissances qu'ils génèrent, aux gens qu'ils font travailler, aux boîtes noires qu'ils cachent et dévoilent, aux interactions qu'ils génèrent ou favorisent, aux solutions qu'ils veulent offrir et aux problèmes qu'ils créent. Pour tout cela, OpenPGP était un candidat idéal. Ce protocole présente une longue histoire chargée d'un nombre incalculable de controverses. Il reste pourtant utilisé par de nombreuses personnes. Il s'est d'ailleurs répandu au sein de certaines communautés d'utilisatrices grâce à des pratiques qui n'étaient pas prévues par son concepteur, échappant ainsi au script de sa conception. Et surtout, des hommes et des femmes l'aiment et le détestent, y consacrent beaucoup de temps et d'énergie, et nourrissent des projets avec ou contre lui.

Ainsi, durant ces cinq ans de recherche, je me suis à mon tour attaché à ce protocole et aux communautés qui gravitaient autour de lui. En renonçant à utiliser le terme « crypto », je suis sorti d'une rhétorique tendance, populaire et racoleuse pour m'intéresser à une norme de chiffrement et d'authentification austère et vertement critiquée par beaucoup, et néanmoins centrale pour certaines communautés, notam-

25 J'ai compris cela quand un expert en informatique, qui a aussi été mon voisin, m'a dit : « Non, Sylvain, en français, on dit "chiffrer", pas "encrypter". Et on ne dit pas "crypto", on dit "chiffrement", à la limite "cryptographie". »

ment les communautés open source. J'ai alors rencontré un regroupement hétéro-clyte de personnes qui s'affairaient à maintenir une technologie vétuste et de niche. Il y avait d'anciens professeurs universitaires à la retraite et des jeunes codeuses prometteuses, des employés d'une grande entreprise ou des contributeurices bénévoles d'un logiciel open source, des juristes, des journalistes, des passionné-es. La plupart de ces gens aimaient ce protocole et ses implémentations, y consacraient du temps et de l'énergie, et c'est avec ces personnes que j'ai mené ma recherche. Ensemble, ces personnes formaient une communauté hétérogène et relativement discrète bien qu'active et très bruyante dès que je m'en suis approché. Il me semble important de souligner dès à présent une particularité des communautés de logiciels libres, que Christophe Lazaro (2008) avait déjà noté à propos de son enquête auprès de la communauté Debian. En interrogeant « les mécanismes de production de l'action collective par le biais d'individualités multiples », Lazaro a mis en lumière « la négociation entre la logique collective du projet et les idéaux individuels » (2008, 254).

Christopher Kelty (2008) a quant à lui identifié une autre particularité de ces communautés : leur récursivité. Il a établi la notion de « public récursif » qu'il définit comme « un public qui se préoccupe essentiellement de l'entretien matériel et pratique et de la modification des moyens techniques, juridiques, pratiques et conceptuels de sa propre existence en tant que public » (2008, 3). Cette capacité de ces communautés à penser à elles-mêmes en tant que communautés et à s'adresser à elles-mêmes via de nombreux canaux en fait des sujets d'étude particulièrement riches pour les sciences sociales. Ainsi, dans ce chapitre, je commence par décrire cette « communauté OpenPGP », aussi appelé « écosystème OpenPGP ». Je décris ensuite les méthodes d'enquêtes qui m'ont permis de comprendre les pratiques, discours et préoccupations au sein de cet écosystème : observation participante en ligne et hors ligne, ainsi que des entretiens semi-directifs conduits avec différent-e-s membres de la communauté. Puis, je termine par une réflexion sur l'analyse et l'écriture, en mobilisant la notion de crise. Avant cela toutefois, j'aimerais préciser quelques rapides éléments de vocabulaire.

Les mots pour le dire : norme, protocole, code, programme

Afin de rendre la lecture de cette thèse la plus commode possible, il me semble nécessaire de s'entendre dès à présent sur l'usage de certains mots récurrents. Il ne s'agit pas d'une explication technique ou anthropologique de ces termes, mais une simple précision du sens qui leur est donné ici.

- **Un protocole** est un ensemble de spécifications et de règles qui précisent la marche à suivre pour effectuer une action spécifique. De très nombreux protocoles sont mobilisés pour utiliser Internet, parmi lesquels on compte notamment la suite de protocole TCP/IP qui permet à des serveurs de communiquer entre eux et d'accéder ainsi à Internet, le protocole TLS qui chiffre la connexion d'un serveur à un autre (le fameux « S » du HTTPS de la barre de navigation), les protocoles IMAP et SMTP qui permettent d'envoyer et recevoir des emails, sans oublier bien entendu le protocole OpenPGP, qui permet de chiffrer et signer des informations numériques. Les protocoles prennent généralement la forme d'un texte écrit. Certains protocoles, mais pas tous, sont des normes.
- **Une norme** est un protocole normalisé par un organisme de normalisation tel l'Organisation internationale de normalisation (ISO), le World Wide Web Consortium (W3C) et l'Internet Engineering Task Force (IETF). L'IETF publie par exemple ses normes sous forme de *Requests for Comment*, abrégées RFC. Les normes sont des outils centraux de la gouvernance d'Internet car elles permettent l'interopérabilité de différents systèmes. Certaines normes, comme les RFC publiées par l'IETF, sont dites **fédérées** : cela signifie que ce sont des normes qui ont été discutées de manière collective et ouverte. Ainsi, les spécifications ne sont pas imposées de manière autoritaire par l'organisme de normalisation, mais ce sont ses membres qui ont fait aboutir cette norme par la voie du consensus approximatif.

Dans quelques cas, on parle de **normes de facto**. Ces normes sont en réalité des protocoles souvent propriétaires (donc pas fédérés) qui deviennent si populaires qu'il est difficile d'y échapper. Un exemple de

normes *de facto* est le protocole Signal qui sert de modèle pour de nombreux systèmes de chiffrement (Ermoshina et Musiani 2019).

N.B. Souvent, en français, le terme « standard » est utilisé comme synonyme de « norme », mais il s'agit d'un anglicisme. Dans cette thèse, j'utilise le mot « norme » comme traduction du terme anglais *standard*.

- **Le code informatique** est ce qui constitue les **programmes informatiques** (ou **logiciels**). Il a y tout d'abord le **code source**, qui est un document texte que les codeuses écrivent dans un langage informatique donné (C, C++, Rust, JavaScript, Python, etc.). Les codes sources peuvent être propriétaires (fermés) ou open source (libres).

Le code source est compilé en **code binaire** que les ordinateurs peuvent interpréter. Le code binaire est un enchaînement de *bits* souvent représentés sous la forme de 0 et de 1.

Le chapitre 5 (« Hériter, faire vieillir et transmettre le code », p. 186) présente plusieurs exemples de code et décrit de manière plus approfondie leur architecture.

N.B. Dans le contexte informatique – et même en parlant de cryptographie – le terme « code » ne fait jamais allusion à un « code secret ». Un code source propriétaire peut être tenu secret, mais il ne fait pas office de mot de passe.

- Lorsqu'un code met en œuvre un protocole, on dit de lui qu'il est une **implémentation** de ce protocole. Ainsi, GnuPG, Sequoia et OpenPGP.JS sont trois implémentations d'OpenPGP ; elles implémentent cette norme. Il est important de comprendre qu'un même protocole peut être implémenté par différents programmes.
- **Internet** est un ensemble de réseaux interconnectés, le « réseaux des réseaux ». Ces réseaux, qui appartiennent à des états, des entreprises et des personnes privées, sont utilisés pour faire fonctionner des applications et des protocoles, dont le Web. Internet n'est donc pas une technologie mais une infrastructure réseau qui supporte de nombreuses technologies.
- **Le Web** (ou le **World Wide Web** dans sa version longue) permet quant à lui de se connecter à des sites Internet et de les relier entre eux grâce à des

liens hypertextes (le « www »). Le Web est donc une des applications qui utilisent Internet, distinct des emails, des visio-conférences ou du partage de fichier pair-à-pair. Les technologies du Web sont pour la plupart normalisées par le World Wide Web Consortium (W3C).

N.B. Le **Web of Trust (WoT)**, qui est une notion issue de l'écosystème OpenPGP, est distinct du World Wide Web. Plus d'informations à ce propos seront données au chapitre 4 p. 150.

2.1. Du village à l'écosystème OpenPGP

Il est coutume d'appeler « terrain » le lieu où l'ethnologue mène sa recherche ethnographique. Le « terrain » a longtemps été compris comme la délimitation géographique de la recherche en anthropologie et permet de rattacher les ethnographes à des régions d'expertise : les îles Trobriand pour Bronisław Malinowski, l'Aurès pour Germaine Tillon et les laboratoires pour Bruno Latour. Ces espaces étaient supposés être clos, totalement différent des autres espaces « physiques » et surtout « donnés » aux ethnographes qui n'avaient qu'à y planter leur tente pour pouvoir observer ce qui s'y passait. Toutefois, la conception du terrain a beaucoup évolué. Par exemple, les limites géographiques ne suffisent plus à délimiter les terrains et il est désormais reconnu que les terrains sont des constructions épistémiques complexes (Geertz 1973; Fabian 2017; Marcus 1995; Faubion et Marcus 2009).

Le "cyberespace" ne fait pas exception : il s'agit d'un espace complexe qui est souvent simplifié et mystifié. Dans un appel provocateur à « renoncer à l'étude *du* cyberespace », Madeleine Pastinelli (2011) a dénoncé la manière très stéréotypée d'imaginer cet ensemble de lieux comme s'il s'agissait d'un site distinct, radicalement différent de tous les autres contextes d'enquête ancrés dans le « réel ». En s'appuyant notamment sur dannah boyd (2008) et en critiquant certains aspects des travaux de Christine Hine (2000), Pastinelli a insisté sur la diversité des espaces d'In-

ternet et sur l'importance de les considérer dans leur spécificité et non de manière générale.

Renoncer à l'étude *du* cyberspace, c'est aussi reconnaître qu'on n'a pas simplement affaire à un espace qu'on pourrait circonscrire et caractériser, mais bien à une technologie qui a pleinement pénétré la vie quotidienne d'une majorité d'Occidentaux et qui est au cœur d'un nombre impressionnant de pratiques des plus variées – certaines donnant corps à des espaces –, lesquelles sont loin de présenter des caractéristiques homogènes. (2011, 40 souligné dans l'original)

En lançant cet appel, elle souligne que c'est de l'objet de recherche que doit dépendre la méthode et le lieu d'investigation, et non du site où elle est menée. Sans le citer, elle reformule l'affirmation de Clifford Geertz pour lequel le « lieu de la recherche n'est pas l'objet de la recherche. Les anthropologues n'étudient pas des villages (des tribus, des villes, des quartiers) ; ils et elles étudient dans des villages » (Geertz 1973, 22 ma traduction). Dans mon cas, le lieu de ma recherche est difficile à circonscrire car il n'y a pas de « village OpenPGP » en bordure duquel j'ai pu planter ma tente d'ethnographe. Il n'y a d'ailleurs pas non plus *un* seul site Internet, *un* seul forum, *une* seule équipe de développement, mais une multitude de lieux et de groupes de personnes engagées dans ses lieux. Didier Demazière et ses collègues (2011) ont d'ailleurs décrit la désorientation des ethnographes comme l'une des caractéristiques typiques de ces nouveaux terrains aux contours flous :

Les contraintes de l'enquête [...] découlent des propriétés distinctives : le caractère informel des contours du groupe et des appartenances rend particulièrement difficile l'orientation sur un terrain qui se dérobe en s'offrant. Cela rend, à nos yeux, nécessaires la multiplication des points d'entrée, la variation des données et l'insertion de longue durée dans le groupe, de manière à « armer » (Weber, 2009) l'ethnographie quand l'ethnographe est confronté à la répétition d'événements et de contacts qui précisent le désarment. (2011, 179)

Selon Demazière et ses collègues, une simple posture d'observation externe ne permet pas de comprendre l'entier des traces en ligne et leur article propose trois leviers méthodologiques pour de telles ethnographies : la collecte en ligne des traces des activités, puis des entretiens avec les personnes concernées et finalement de l'observation directe des activités en cours d'action. Je me suis largement inspiré de

leur article et même si, pour des raisons pratiques, je n'ai pas respecté l'ordre et la systématique suggérées par les auteurs, ces trois leviers méthodologiques se trouvent au centre de mon dispositif de recherche.

Cela n'allait toutefois pas sans poser de question, et notamment la question de la circulation et des connexions que je pouvais observer entre différentes communautés de pratiques, différents lieux et différents objets. Dans son article *Ethnography in/of the world system*, George Marcus (1995) décrit l'émergence de « terrains multi-situés » en anthropologie. Les flux de plus en plus importants de personnes, de biens et d'idées remettaient en question la dichotomie entre le local et le global, ce qui a poussé Marcus à proposer d'analyser le déplacement de six éléments : (1) les gens, (2) les choses, (3) les signes, symboles et métaphores des discours et modes de pensées, (4) les intrigues, histoires et allégories, (5) les biographies des personnes et (6) les conflits. Mis à part le cinquième point, tous ces éléments ont été très présents lors de ma recherche et ce chapitre permet aux lecteurices de mieux saisir qui ont été les gens (codeureuses, développeureuses, cryptographes, chercheureuses, activistes, journalistes, scientifiques), les choses (codes, normes, algorithmes, *bits*, et ordinateurs), les symboles (le cadenas bleu), les intrigues (Alice, Bob et Ève) et les conflits (au sein d'une équipe de développement ou du groupe de travail de l'IETF, entre différentes implémentations) qui m'ont guidé lors de ma recherche. Il ne s'agissait pas d'étudier l'un ou l'autre de ces points de manière isolée, mais de saisir « les connexions, les associations et les relations à travers l'espace » (Falzon 2016, 1s, ma traduction). Le travail de l'ethnographe est donc de comprendre et rendre compte de ces connexions qui se prolongent sur des sites qui ne sont pas spatialement contigus.

Mon « terrain » pour cette recherche n'a donc pas été un terrain délimité par un ou plusieurs espaces physiques et sa cohérence tient plutôt à la centralité de son objet autour duquel gravitaient plusieurs communautés. Mon terrain s'est donc d'abord construit autour de mon intérêt pour ce que Bowker et Star (2000) ont appelé « les choses ennuyeuses » : des normes, des registres, des classifications, et d'autres objets qui paraissent souvent monotones et insipides mais qui le sont rarement. Je n'ai

pas étudié dans un seul lieu spécifique mais dans « un territoire tenu par les liens d'individus dispersés entre des lieux discontinus » (Lima 2013, 341). Une notion m'a particulièrement aidé à saisir les interaction entre ces différents groupes dispersés dans l'espace : la notion d'« objet-frontière », qui permet de saisir comment OpenPGP agit comme un liant entre différents groupes de personnes :

Les objets-frontière sont des objets suffisamment plastiques pour s'adapter aux besoins locaux et aux contraintes des divers groupes qui les utilisent, tout en étant suffisamment robustes pour maintenir une identité commune d'un site à l'autre. [...] Ils ont différentes significations dans différents mondes sociaux mais leur structure est suffisamment commune à plusieurs mondes pour les rendre reconnaissables, un moyen de traduction. La création et la gestion d'objets-frontières est un processus clé dans le développement et l'entretien de la cohérence à l'intersection de plusieurs mondes sociaux. (Star et Griesemer 1989, 393, ma traduction)

Autrement dit, « [l]es objets-frontières facilitent la coopération et les interactions entre des acteurs hétérogènes, sans imposer une signification unique de l'objet » (Latzko-Toth et Millerand 2015, 163s). La notion d'objet-frontière est particulièrement intéressante dans le contexte de mon étude car en plus de souligner la flexibilité interprétative d'OpenPGP qui est tantôt un objet de soin et d'attachement, tantôt une dangereuse relique du passé, la notion engage à étudier les différentes communautés de pratiques qui gravitent autour d'OpenPGP. Les pratiques de coopération et ces interactions entre différentes communautés sont au centre de mon questionnement sur les temps de la sécurité, et la notion d'objet-frontière permet de saisir l'« écologie » de l'infrastructure d'OpenPGP (selon les termes de Star et Griesemer 1989), ou le fonctionnement de l'« écosystème OpenPGP » (selon le langage émique utilisé par les participant-es à ma recherche). L'« écosystème OpenPGP » est en effet une expression que j'ai régulièrement entendue lors de ma recherche. Elle fait référence au réseau sociotechnique composé principalement d'objets, de normes, d'arrangements infrastructurels et des communautés qui utilisent ou contribuent d'une manière ou d'une autre à OpenPGP.

Étudier d'un point de vue anthropologique le cas de la norme OpenPGP implique d'entrer dans le quotidien de la fabrique de la sécurité informatique et de se plonger

dans des pratiques de lecture, d'écriture, de relecture et de réécriture du code informatique. Cela nécessite aussi de déchiffrer les multiples traductions qui relient le travail des cryptographes qui développent des algorithmes cryptographiques aux utilisatrices finales des programmes de chiffrement. Ma démarche ethnographique a été largement inspirée de la sociologie de la traduction (Akrich, Callon, et Latour 2006; Latour 2007), aussi connue sous le nom de la théorie de l'acteur-réseau (Law et Hassard 1999; Michael 2017). Cela a consisté à « suivre » différentes choses dans cet écosystème, et notamment des vulnérabilités, des bogues, des fragments de code et des controverses.

Le verbe « suivre » est souvent utilisé dans ce contexte : suivre les gens, les objets, les conflits. Il ne doit toutefois pas laisser croire que le terrain préexiste au travail de l'anthropologue, qu'il s'impose en tant que réalité objective et que le travail de terrain se résume à un « dévoilement de la vérité » (Dakhli 1995). Dans l'article déjà cité, Marcus (1995) a d'ailleurs affirmé que la question de la production sociale de l'espace (voir aussi Massey 2005) impliquait de repenser la responsabilité de l'ethnologue à construire son propre terrain. Le terrain est aussi une construction de l'ethnologue qui, selon les possibilités et contraintes sociales, géographiques et temporelles de son enquête, fait des choix. C'est pourquoi il me semble primordial d'explicitier ici les choix que j'ai faits au cours de ma recherche, afin de rendre mon raisonnement le plus transparent possible.

Dans la suite de ce chapitre, je décris en détail les différentes phases de ma recherche, de la définition de la problématique à l'analyse des données. Durant toute cette période, je n'ai pas arrêté de penser à mon « terrain », tâchant d'en comprendre à la fois l'étendue et ses limites – tout en reconnaissant que cette étendue et ces limites ne dépendaient que de moi, de mes possibilités et de mes affinités dans la recherche. Avant cela, il me semble toutefois important de préciser quelles sont les différentes communautés de pratiques avec lesquelles j'ai mené ma recherche.

2.2. Les différentes communautés d'OpenPGP

Étudier les pratiques de soin vis-à-vis de la norme technique OpenPGP nécessite de rendre compte des activités de plusieurs communautés de pratiques qui accomplissent ces tâches variées dans différents lieux, à la fois en ligne et hors ligne. La définition des communautés de pratiques remonte à un livre de Jean Lave et Etienne Wenger (1991) sur le savoir situé. Ils définissent les communautés de pratiques selon trois caractéristiques : une définition partagée de ce qui constitue la base de ce groupe, une passion, une activité ou une pratique ; le type de relations au sein de la communauté consistant généralement en des échanges intenses et simples ; et finalement l'existence d'un objet commun autour duquel échanger et construire ou approfondir des connaissances (Lave et Wenger 1991). Inspiré par cette définition et différents commentaires qu'elle a suscités (Akrich 2010; Cross 2013), j'ai distingué quatre principales communautés de pratiques.

Quatre communautés de pratique de l'écosystème OpenPGP

- **Le groupe de travail OpenPGP de l'IETF.** Leur pratique est centrée autour de la négociation des spécifications du protocole sous la forme d'un RFC. Les membres échangent via une liste de diffusion²⁶ ainsi que lors des réunions de l'IETF, trois fois par année. Les interactions ne sont pas régulières : il arrive que plusieurs semaines se passent sans qu'aucun message ne soit envoyé, ou au contraire, que des dizaines d'emails soient échangés par jour. Le contenu de ces messages est avant tout technique, centré sur les spécifications mêmes d'OpenPGP.
- **Les communautés de développeuses.** Elles sont regroupées autour de différents programmes qui implémentent OpenPGP : GnuPG, OpenPGP.js, Sequoia PGP, PGPainless, etc. La manière d'interagir au sein de ces différentes communautés est très variable, d'une entreprise très hiérarchique à des gouvernances très horizontales. Bien qu'ils et elles ne travaillent pas forcément dans un même lieu et qu'elles ne se soient pas

26 <https://mailarchive.ietf.org/arch/browse/openpgp/>.

forcément rencontrées en personne, les membres se connaissent généralement bien, ont l'habitude de travailler ensemble et échangent régulièrement. Ces échanges se déroulent sur plusieurs sites : les bureaux d'une entreprise, mais aussi les salons IRC, les listes de diffusion, les plateformes de partage de code (GitHub, GitLab) et, parfois, certains événements festifs (Fosdem, Chaos Communication Congress, etc.). L'objet central est le code informatique. Dans ma recherche, j'ai particulièrement suivi le travail de trois équipes de développement qui implémentent GnuPG, Sequoia, et OpenPGP.js.

- **La communauté scientifique.** Les communautés de chercheuses et chercheurs sont très importantes dans l'écosystème d'OpenPGP car ce sont souvent elles qui identifient et révèlent des vulnérabilités importantes présentes dans le code des implémentations ou plus largement dans le protocole lui-même. Lors de mon enquête, plusieurs recherches scientifiques ont révélé des vulnérabilités dans OpenPGP (Poddebniak et al. 2018; Müller, Brinkmann, Poddebniak, Böck, et al. 2019; Müller, Brinkmann, Poddebniak, Schinzel, et al. 2019; Schwenk et al. 2020; Müller et al. 2020; Leurent et Peyrin 2020; Halpin 2020). Je me suis particulièrement focalisé sur la première équipe de recherche qui a révélé les failles d'EFAIL.
- **La communauté des utilisatrices.** Cette communauté est très hétérogène et il est difficile de la définir précisément. Tout d'abord, un grand nombre de personnes appartenant aux précédentes communautés sont également des utilisatrices d'OpenPGP. De plus, je suis certain que la grande majorité de cette communauté est silencieuse. Toutefois, quelques « pans » de cette communauté sont particulièrement visibles sur des sites en ligne. La liste de diffusion des utilisatrices de GnuPG²⁷ est par exemple particulièrement active et il peut arriver que des dizaines d'emails soient échangés par jour. J'ai aussi rencontré de nombreux « fans » d'OpenPGP lors de *Cryptoparties* ou de *Signing Parties*, événements lors desquels les utilisatrices signent réciproquement leurs clés publiques pour accroître leur légitimité en ligne. C'est bien entendu

27 <https://lists.gnupg.org/mailman/listinfo/gnupg-users>.

l'usage d'OpenPGP qui crée une cohésion entre les différentes personnes qui échangent sur les différents « trucs et astuces » des programmes en question. Dans certains cas, c'est l'appartenance à une autre communauté (Linux, Debian, Wikipédia) qui crée un lien entre ces personnes. Mes questions de recherche ne la concernant pas directement, je n'ai pas particulièrement enquêté sur cette communauté de pratique.

Je précise d'emblée que ces communautés ne sont pas étanches et que beaucoup de personnes se retrouvaient dans différentes communautés en même temps. C'était notamment le cas de nombreuses personnes qui participaient au groupe de travail de l'IETF tout en étant soit utilisatrices, soit codeuses. Prises dans leur ensemble, ces différentes communautés de pratique formaient « la communauté OpenPGP ». Il s'agit d'une expression émique qui permet de désigner de manière bien pratique un ensemble flou et hétérogène de personnes plus ou moins rattachées à OpenPGP. Comme me l'a écrit un·e participant·e à la recherche, ce terme est pratique pour désigner un groupe qui peut être aussi lâche et mouvant que la famille éloignée :

La communauté est composée de toutes les personnes intéressées par OpenPGP. Cela inclut principalement les personnes travaillant sur des implémentations d'OpenPGP, par exemple Sequoia, OpenPGP.js/GopenPGP maintenu par ProtonMail, GnuPG, RNP (utilisé par Thunderbird). [...]

À mon avis, la communauté comprend également des personnes qui ne pensent pas être proches du processus, comme des sociétés commerciales (FlowCrypt) et des librairies qui n'interviennent pas dans la liste de diffusion (BouncyCastle). C'est juste comme une famille éloignée :)

(Codeuseuse d'une implémentation d'OpenPGP, email personnel, 22 octobre 2020)

Un des points qui me plaît beaucoup dans cette conception émique est que cette communauté paraissait beaucoup plus inclusive qu'exclusive : il suffisait de s'y intéresser pour en faire partie. Autrement dit, il ne s'agissait pas d'un groupe restreint et élitiste mais au contraire d'un groupe ouvert dont la composition dépend souvent du

contexte et des opinions personnelles. Selon les opinions et les sensibilités de chacune et chacun, cette « communauté OpenPGP » incluait non seulement les codeuses des différentes implémentations et les membres du groupe de travail de l'IETF, mais également de nombreuses utilisatrices qui forment un public récuratif particulièrement actif (Kelty 2008)²⁸. Par ailleurs, certaines personnes n'hésitaient pas à inclure également d'autres actrices qui interviennent de près ou de loin dans les affaires d'OpenPGP. Il s'agit par exemple des chercheuses universitaires qui travaillent sur ce protocole notamment pour y trouver des failles ou – dans mon cas – en comprendre les us et coutumes. Étaient parfois aussi compris·es des cryptographes et autres expert·e·s qui défendent ou critiquent OpenPGP dans des articles, des billets de blog, des tweets ou lors de conférences. Certaines personnes nourrissaient un sentiment d'appartenance à cette communauté, et d'autres cherchaient même à consolider ce sentiment d'appartenance en organisant des événements²⁹. C'est dans cette compréhension inclusive que j'utilise le terme « communauté OpenPGP » dans cette thèse.

2.3. La première phase de ma recherche : *lightshows*, ateliers de crochetage et discussions assourdissantes

Maintenant que sont faites les présentations avec les différentes communautés de mon terrain, j'aimerais présenter avec plus de détails comment j'ai mené ma recherche. J'ai commencé à m'intéresser à OpenPGP en mai 2018 lors de la révélation des vulnérabilités d'EFAIL³⁰. C'était quelques mois à peine après le début de mon doctorat, alors même que je n'avais pas encore défini ma problématique. David Bozzini et moi venions de soumettre le projet « *Windows of Exposure: toward an anthropology of computer (in)security* »³¹, centré sur un type particulier d'objets numériques : les

28 Voir par exemple la liste de diffusion des utilisatrices de GnuPG : <https://lists.gnupg.org/mailman/listinfo/gnupg-users>.

29 Par exemple, une codeuseuse m'a un jour demandé comment, en tant que supposé expert des communautés, je pensais qu'il serait possible de renforcer ce sentiment d'appartenance.

30 <https://efail.de/>. Voir aussi chapitre 6 (« EFAIL et le temps de l'urgence, p. 226 »).

31 Ce projet de recherche de 18 mois dirigé par David Bozzini a été financé par le Fonds national suisse de la recherche scientifique. La description du projet peut être consultée dans la base de

vulnérabilités informatiques. Ces vulnérabilités sont des objets complexes, qui créent souvent des situations de crises et parfois de controverses importantes au-delà des milieux spécialisés. Ces vulnérabilités informatiques sont au centre des préoccupations des ingénieur·e·s de sécurité informatiques (Arbaugh, Fithen, et McHugh 2000), mais demeurent encore peu abordées en sciences sociales, avec quelques notables exceptions (Chateauraynaud et Trabal 2007; Denis 2009; 2012; Coleman 2009; 2010; 2013; 2015; Balzacq et Dunn Cavelty 2016; Korn et Wagenknecht 2017; Besençon et Bozzini 2020; Besençon 2022; Bozzini 2023).

2.3.1. Les conférences de sécurité informatique

Pour ce projet, David Bozzini et moi nous sommes focalisés sur les vulnérabilités informatiques et leur « vie sociale », c'est-à-dire la manière dont elles créaient des ponts entre différentes actrices de différents milieux. Nous avons décidé d'explorer ce domaine encore méconnu en anthropologie par le biais de l'événement, en particulier les divulgations de vulnérabilités et la « rupture d'intelligibilité » qu'elles impliquent (Bensa et Fassin 2002). En nous inspirant d'un article de l'anthropologue Gabriella Coleman (2010) sur les conférences hacker, nous avons placé les conférences de sécurité informatique au centre du dispositif de recherche. En effet, selon Coleman, ces conférences permettent de nuancer l'affirmation de Manuel Castells (2004, 50) quant aux « vrais hackers » qui « existent principalement en tant qu'hacker en ligne » (cité par Coleman 2010, 48, ma traduction).

Dans le cadre du projet *Windows of Exposure*, j'ai ainsi eu l'occasion de participer à une quinzaine d'événements sur environ deux ans (voir tableau 1 p. 74). Je ne suis pas sûr que le terme « conférence » soit le plus approprié. Certaines personnes n'hésitaient pas à parler de « festivals du hacking » pour évoquer le Chaos Communication Congress ou de « grande messe de la sécurité informatique » pour qualifier Black Hat. Ce type d'événements présente un côté spectaculaire, avec d'immenses salles de conférences plongées dans le noir où des milliers de personnes regardent la grande scène illuminée par de nombreux projecteurs au moment de la conférence

données du FNS: <http://p3.snf.ch/project-183223>.

d'ouverture. Ces *shows* contrastent avec d'autres conférences universitaires plus discrètes, parfois même organisées avec une certaine sobriété.

En participant à ces événements variés tant sur la forme que sur le type de présentations et de publics, j'ai pu cumuler une certaine expérience de ces milieux. Ces événements ont notamment été d'une grande aide pour m'aider à acquérir des connaissances basiques et néanmoins utiles à propos de différents objets (code, vulnérabilité, preuves de concepts, etc.) et pour me familiariser d'une manière générale avec le champ de la sécurité informatique.

Tableau 1: Liste des conférences et autres événements de hacking auxquels j'ai participé lors de la phase exploratoire de mon ethnographie.

Conférence	Date et lieu	Lien vers la conférence
Insomni'hack	23.03.2018, Genève, CH	https://insomnihack.ch/conference-2018/
Black Alps	8-9.11.2018, Yverdons-les-Bains, CH	https://www.blackalps.ch/ba-18/summary.php
Black Hat Europe	3-6.12.2018, Londres, UK	https://www.blackhat.com/eu-18/
Chaos Communication Congress (35c3)	27-30.12.2018, Leipzig, DE	https://events.ccc.de/congress/2018/wiki/index.php/Main_Page
Winterkongress (Digitale Gesellschaft)	23.02.2019, Zürich, CH	https://www.winterkongress.ch/2019/
Swiss Cyber Security Days	27-28.02.2019, Fribourg, CH	https://web.archive.org/web/20190120173851/https://swisscybersecuritydays.ch/en/
Troopers	18-22.03.2019, Heidelberg, DE	https://troopers.de/troopers19/
European Symposium on Security and Privacy (IEEE)	17-19.06.2019, Stockholm, SE	https://www.ieee-security.org/TC/EuroSP2019/index.php
Privacy Enhancing Technologies Symposium	15-20.07.2019, Stockholm, SE	https://petsymposium.org/2019/
Swiss Cyber Storm	15.10.2019, Bern, CH	https://2019.swisscyberstorm.com/
Black Alps	7-8.11.2019, Yverdons-les-Bains, CH	https://www.blackalps.ch/ba-19/summary.php
Chaos Communication Congress (36c3)	27-30.12.2019, Leipzig, DE	https://events.ccc.de/congress/2019/wiki/index.php/Main_Page
Winterkongress (Digitale Gesellschaft)	22.02.2020 Zürich, CH	https://www.winterkongress.ch/2020/
Du fait de la pandémie de coronavirus de 2020, j'ai arrêté de participer à ces conférences en présentiel dès février 2020. J'ai encore suivi quelques événements en ligne, parfois en direct (par exemple la conférence HOPE en 2020, et plusieurs sessions de travail de l'IETF, dès l'été 2019), le plus souvent en différé (par exemple sur des plateformes comme media.ccc.de ou Youtube). Comme ces événements en ligne ne permettent pas le même niveau d'interaction, il ne me semble pas nécessaire de les mentionner sur cette liste.		

Les conférences de sécurité sont des espaces privilégiés pour observer la manière dont la sécurité est présentée, déconstruite et négociée entre différents acteurices de l'industrie. Ce sont par exemple des événements intéressants où observer certaines pratiques clés de la sécurité informatique, comme la révélation publique de vulnérabilité.

bilités (Bozzini 2023). Ces révélations de failles sont des événements majeurs qui sèment le trouble et l'incertitude même si, la plupart du temps, les vulnérabilités sont souvent déjà corrigées dans les nouvelles versions du code. Dans une rhétorique assez stéréotypée basée sur des discours de sécurisation très explicite (Korn et Wagenknecht 2017), ces présentations expliquent les failles découvertes, mais aussi parfois les difficultés d'interaction entre les chercheuses de vulnérabilités et les développeuses. Ces discours impliquent typiquement la mise en exergue d'une illusion (« vous vous croyiez en sécurité grâce à ce logiciel »), la révélation du problème (« mais nous avons trouvé toutes ces vulnérabilités »), et les mesures de protection (souvent un simple « mettez à jour ! », mais parfois aussi : « mettez cette technologie à la poubelle »).

Par ailleurs, les conférences de sécurité informatique sont souvent des moments festifs et rassembleurs, durant lesquels différentes communautés de pratiques se retrouvent, font corps, et consolident leurs liens dans une ambiance de « concentration rituelle et célébration émotionnelle » (Coleman 2010). Elles offrent aussi des lieux et des moments informels et ludiques, comme des soupers de gala pour *Black Alps* ou *Troopers*, des lieux de danse et de concerts pour le *Chaos Communication Congress*, et des randonnées en pleine nature pour le *PET symposium*³². Ce sont aussi des lieux de promotion et de démonstration de certaines valeurs. Ainsi, il n'était pas rare de trouver dans les couloirs du *Chaos Communication Congress* des slogans prônant non seulement l'excellence, mais aussi l'entraide, le respect, la curiosité et l'ouverture. D'un point de vue de l'industrie, ce sont aussi des événements qui permettent de faire la démonstration et la promotion d'un produit ou d'un service. D'ailleurs, on repart rarement les mains vides de ces événements et un nombre impressionnant de brochures, d'accessoires et d'autocollants sont distribués.

Ces conférences ont aussi contribué à me façonner une impression beaucoup plus incarnée de ce qu'était la sécurité informatique. En me faisant vivre avec mes propres

32 Le nom de cette conférence ne fait pas référence aux animaux domestiques ou aux bouteilles en plastique. PET est l'acronyme de *Privacy Enhancing Technology*, technologie favorisant le respect de la vie privée.

sens des sensations qui se rapprochent peut-être de celles ressenties par les participant·e·s à ma recherche, j'ai pu ressentir des impressions qui peuvent me donner un aperçu de ce que peut faire la sécurité sur le corps dans une approche d'ethnographie sensorielle (Pink 2015). J'ai par exemple ressenti une excitation au moment où les immenses salles de conférences devenaient noires alors que les projecteurs s'allumaient sur scène pour l'entrée des présentateurices qui ouvraient cette X^e édition sous un tonnerre d'applaudissements ; ou un certain sentiment d'inquiétude mêlée d'impuissance en entendant des présentations démontrer combien les technologies que nous utilisons étaient fragiles voire dangereuses ; ou encore un frisson certain et une pointe de fierté au moment d'ouvrir mon premier cadenas à l'aide d'outil de crochitage lors d'un atelier du *Chaos Communication Congress*, frisson qui m'est souvent revenu en tête quand mes interlocuteurices m'expliquaient leur excitation au moment clé de leur recherche, où la boîte noire qu'elles essayaient d'ouvrir finissait pas céder. J'ai aussi ressenti la joie de retrouver, d'une conférence à l'autre, quelques personnes avec qui j'ai tissé un lien d'amitié ; j'ai finalement ressenti une espèce de surexcitation, d'ivresse intellectuelle et d'épuisement physique et cérébral à la fin de ces conférences après avoir passé un temps *hors du temps*.

Ces impressions sur mon propre corps de ce que fait la sécurité m'ont non seulement permis d'être plus empathique avec les personnes que j'interviewais, mais elles m'ont également fait prendre conscience d'un aspect fondamental de la sécurité : la manière fluctuante de percevoir la sécurité et l'ambivalence de ce ressenti. J'ai en effet été intellectuellement, mais aussi corporellement sensibilisé aux différentes impressions subjectives du travail de la sécurité, où l'urgence et la lenteur sont avant tout des sensations physiques de stress, de nervosité, d'empressement, d'impatience ou au contraire de discussions calmes et posées.

Ces conférences ont également grandement facilité ma rencontre avec différentes personnes clés de ma recherche. Le cadre de ces conférences était très propice à rencontrer différentes personnes et à leur poser des questions avec toute ma naïveté de néophyte. C'est ainsi qu'il m'est arrivé de boire le thé assis en tailleur, pieds nus et en

compagnie d'une des scientifiques les plus réputées dans le champ de la cryptographie résistante aux ordinateurs quantiques qui m'exposait pourquoi il ne fallait pas m'intéresser à OpenPGP ; ou de manger sur une table beaucoup trop petite avec deux des fondateurs du protocole Off-The-Record qui me racontaient pourquoi, selon eux, OpenPGP n'a toujours pas été patché des failles qu'ils mentionnaient dans leur article de 2004 (Borisov, Goldberg, et Brewer 2004). De ces rencontres sont parfois nés des liens qui, pour certains, se sont prolongés durant toute la période de ma recherche. C'est le cas de mon lien avec Ben, un cryptographe qui m'a grandement aidé dans cette première phase de recherche. Sans qu'il ne le sache d'ailleurs, je le considère comme une espèce de mentor qui m'a guidé dans les couloirs du CCC, d'EuroS&P et PET et qui m'a donné certaines clés pour déchiffrer quelques boîtes noires. Il m'a par exemple permis de comprendre comment fonctionnaient certains algorithmes et comment il était possible de vérifier mathématiquement leur fiabilité grâce à des preuves formelles. Je préciserais toutefois une certaine frustration face aux conditions souvent insatisfaisantes de ces rencontres : fond sonore assourdissant, niveau de préparation souvent réduit à néant par la spontanéité de la discussion, pas ou peu de moyens de prendre des notes et encore moins d'enregistrer ces conversations.



Illustration 5 : Moi, en train d'essayer de « faire un truc » sous le regard patient et bienveillant d'un-e participant-e à ma recherche. Interaction au Chaos Communication Congress de décembre 2019. Dessin de Léa Kissling (@lea_kissling_officielle).

2.3.2. Une exploration par tâtonnements

La participation à ces conférences m'a également fait vivre trois conditions communes à la fois au travail ethnographique et au travail de la sécurité informatique, et qui ne sont toutefois que rarement mentionnées en tant que telles : le hasard, la frustration de ne pas avoir des conditions idéales pour travailler et le bricolage improvisé malgré ces médiocres conditions.

En effet, il arrive que des vulnérabilités soient découvertes par hasard et des co-deureuses doivent alors bricoler des solutions sur le pouce en improvisant en fonction de la situation, sans aucune certitude que cela fonctionnera. Cette contingence, cette improvisation et cette nécessité d'agir malgré une situation peu idéale, je les ai moi-même vécues à ma manière tout au long de mon travail ethnographique et cela a certainement participé à constituer de manière incarnée ma compréhension de ce qu'est la sécurité informatique. En somme, chaque situation portait en elle des

« bogues » que je devais traiter du mieux que je pouvais dans l'instant, dans l'improvisation et le bricolage. Un de ces bogues était par exemple les conditions d'entretiens pénibles à cause d'un bruit de fond considérable lors des conférences.

D'ailleurs, de la même façon que certains bricolages ne fonctionnent simplement pas sur le long terme, certaines pistes qui m'avaient semblé séduisantes dans un premier temps se sont finalement révélées moins pertinentes qu'escompté. J'ai, par exemple, commencé à suivre certaines communautés et certains protocoles pour finalement me rendre compte que ce n'était pas la voie à suivre. J'ai souvent vécu ces égarements, ces hésitations et ces retours en arrière comme des erreurs. En tant qu'étudiant et chercheur, je n'ai jamais aimé me tromper. En même temps, en tant qu'enseignant, je répète souvent à mes étudiant·e·s que se tromper fait non seulement partie du processus d'apprentissage et de recherche, mais que c'est quelque chose de tout à fait souhaitable. Peut-être me trompè-je en affirmant ceci (et ce serait ainsi une nouvelle opportunité pour apprendre), mais j'aimerais proposer de voir ces égarements et ces hésitations de la même façon que les codeuses expérimentent avec le code qu'il ou elle est en train de produire ou corriger : des tentatives, beaucoup d'erreurs et, petit à petit, à force d'insistance, d'obstination et de créativité, la production de quelque chose qui fonctionnera ... jusqu'au prochain bogue.

2.3.3. Les données produites lors de la phase exploratoire

Lors de cette première phase, j'ai produit différents types de données. Le premier ensemble concerne des notes prises lors de ces conférences. Ces notes, sous forme manuscrite et électronique, et parfois accompagnées de photographies ou captures d'écran, sont souvent des artefacts hybrides mêlant informations techniques et commentaires anthropologiques sur la situation d'enquête. En tant que document ethnographique, la plupart d'entre elles se sont révélées peu utiles sur le long terme du fait de la portée souvent très générales de ces premières observations ; elles ont toutefois été un premier terrain de jeu pour forger mes hypothèses de travail et définir la manière dont je voulais approfondir ma recherche.

J'ai également mené plusieurs entretiens exploratoires, notamment avec des scientifiques, des codeuses de logiciels cryptographiques, des expert-es en cryptographie, des journalistes et des utilisatrices. Lors de mes entretiens avec des scientifiques qui cherchaient des vulnérabilités, j'ai surtout tenté de comprendre quels avaient été les tenants et aboutissants de certains choix thématiques (choix de l'objet de recherche) ou stratégiques (choix du mode de divulgation des vulnérabilités trouvées). Dans le cas d'EFAIL, je me suis également intéressé à la manière dont les scientifiques avaient perçu la controverse à ce sujet (voir chapitre 6 « EFAIL et le temps de l'urgence », p. 226). En interviewant des codeuses de programmes cryptographiques (qui n'étaient pas forcément liés à OpenPGP), j'ai posé de nombreuses questions sur le travail quotidien d'écriture et de réécriture du code informatique – et j'ai appris qu'il s'agissait surtout d'un travail de lecture et de relecture (voir chapitre 5 « Hériter, faire vieillir et transmettre le code », p. 186). Je me suis également intéressé à certaines connaissances techniques que je me suis appliqué à assimiler, que ce soit en compulsant des pages souvent très longues de Wikipédia ou en suivant des ateliers en ligne ou lors de conférences, ainsi qu'en suivant des MOOCs sur la cryptographie ou la sécurité informatique.

Puis, subitement, l'hiver 2020 a amené un autre événement imprévu dans ma recherche : la pandémie de coronavirus. Les confinements et les appels à la « distanciation sociale » ont stoppé net cette phase d'exploration en causant l'annulation de toutes les réunions « en présentiel »³³.

33 Le vocabulaire que la pandémie a contribué à répandre est parfois malheureux, particulièrement quand il s'agit d'étudier les comportements en ligne. En effet, la fameuse « distanciation sociale » était plutôt une distanciation physique car les personnes continuaient de socialiser, mais différemment et en ligne. De même, la terminologie « en présentiel » qui s'oppose à « en ligne » laisse supposer qu'on ne peut pas être « présent-e » en ligne. Je renvoie ici au texte de Gabriele De Seta (2020) qui a notamment écrit sur la « co-présence » en ligne en proposant une stimulante réflexion sur la question récurrente dans les *chats* en ligne : « t'es là ? »

2.4. La deuxième phase : visioconférences et après-midi « piscine »

À cette première période d'exploration a succédé une période marquée par une intensification de la relation d'enquête avec certaines personnes – et ce, malgré les restrictions liées à la pandémie. Durant cette période, qui s'étend du début de l'année 2020 à la fin de 2021, je me suis rapproché en particulier de deux communautés de pratiques distinctes et néanmoins interconnectées : d'une part le groupe de travail de l'IETF que je n'ai jamais eu l'occasion de rencontrer en personne mais dont j'ai suivi plusieurs séances de travail à distance ; et, d'autre part, l'équipe de développement de Sequoia dont j'ai rencontré et interviewé toutes les membres salarié·e·s et d'autres contributeurices bénévoles. D'une manière générale, me rapprocher de la première communauté permettait d'étudier la gouvernance à long terme des spécifications d'OpenPGP alors que fréquenter la deuxième communauté permettait d'avoir un cas d'étude du quotidien d'une équipe de développement.

Le groupe de travail de l'IETF rassemble des personnes d'horizons très variés, avec des différentes attentes. Il y a notamment des codeuses de différentes implémentations, des personnes qui avaient participé aux débuts d'OpenPGP dans les années 1990, des utilisateurices d'OpenPGP, des scientifiques ayant trouvé des vulnérabilités, ou simplement des personnes curieuses de l'avenir d'OpenPGP. Le rôle de ce groupe de travail est d'aboutir à un consensus quant aux spécifications de la norme OpenPGP. Suivre ses activités et apprendre à connaître son fonctionnement était dès lors particulièrement important pour comprendre les transformations d'OpenPGP sur le long terme et les négociations autour d'un projet-sécurité commun au groupe de travail.

D'un autre côté, suivre les activités de l'équipe de développement de Sequoia m'a permis d'observer les pratiques de conception et de correction du code informatique, ouvrant une fenêtre sur les problèmes ordinaires que pose le développement d'un programme de sécurité informatique au quotidien. J'ai notamment pu observer comment la plateforme de partage de code GitLab est utilisée, comment le travail est réparti entre les différents individus et comment l'équipe est en lien avec d'autres

équipes de développement. La fréquentation de cette équipe de développement m'a en outre fait voir comment une même implémentation peut regrouper des individus qui ne partagent pas la même vision de la sécurité, et comment des décisions sont prises entre l'équipe salariée et les contributeurices bénévoles engagées.

Finalement, la question de l'accès à ces communautés a aussi été un critère de sélection. Le groupe de travail de l'IETF regroupe des personnes du monde entier (principalement d'Europe et d'Amérique du Nord) interagissant principalement en ligne, ce qui facilitait grandement mon accès aux négociations. L'équipe de Sequoia était quant à elle formée d'une demi douzaine de personnes, principalement basées en Allemagne ou du moins en Europe. Cela m'a permis de rencontrer la plupart de ses membres en chair et en os et de faire des séjours de terrain au sein de leur équipe. Dans ce qui suit, je spécifie les conditions d'enquête au sein de ces deux communautés. Puis je présente aussi comment j'ai mené une quarantaine d'entretiens semi-directifs enregistrés avec différent·e·s membres de l'écosystème OpenPGP au-delà de ces deux communautés.

2.4.1. Du côté de l'IETF

Je ne saurais pas dire avec précision quand a commencé mon travail ethnographique auprès du groupe de travail OpenPGP de l'IETF. Depuis la révélation d'EFAIL en mai 2018, je consultais de temps à autre la liste de diffusion dédiée³⁴, sans comprendre vraiment le contenu hautement technique des discussions, ni ce que représentait exactement ce groupe de travail. Il faut dire que son statut était ambigu : officiellement dissout, ce groupe continuait d'échanger au travers de la liste de diffusion, et un brouillon d'une nouvelle version du RFC était régulièrement mis à jour sur le site de l'IETF³⁵. Avec le temps, mon rythme de consultation de ces emails est passé

34 Les archives publiques de cette liste de diffusion sont accessibles sur le site de l'IETF : <https://mailarchive.ietf.org/arch/browse/openpgp/>. Pour s'inscrire à cette liste de diffusion, il suffit de se créer un profil sur la page d'inscription (<https://www.ietf.org/mailman/listinfo/openpgp>). Aucun prérequis ni aucune cotisation ne sont exigées. Une fois enregistré dans le système, il est possible d'envoyer des emails à la liste. Il en va de même pour la plupart des listes de diffusion que j'ai consultées lors de ma recherche.

35 Il s'agit du document RFC4880bis.

de « de temps à autre » à hebdomadaire puis quotidien. Au final, je me suis trouvé de plus en plus investi dans ce groupe de travail, qui, à un certain moment, a commencé à dangereusement envahir ma boîte mail (jusqu'à ce que je mette en place des filtres pour classer les différentes listes de diffusion auxquelles j'étais abonné). Au début de ma recherche, le groupe n'était pas officiellement ouvert et ne se réunissait pas lors des réunions de l'IETF. Après sa refonte en 2021, les réunions qui ont suivi ont continué de se dérouler en ligne.

Mon plus gros défi pour suivre cette communauté de pratique a été de comprendre les enjeux de ces emails dont le contenu était et demeure encore aujourd'hui relativement cryptique pour moi. Comme lors des conférences, je comprenais la plupart des mots, mais ne comprenais pas le sens qu'ils avaient dans ce contexte. Dès que j'en avais l'occasion, je posais donc de nombreuses questions techniques à mes interlocuteurices ; ces questions ne m'aidaient pas forcément à avancer dans ma compréhension anthropologique de ce qui était en train de se passer, mais étaient néanmoins fondamentales pour cerner les enjeux des discussions. Grâce à ces discussions, souvent en visioconférence, mais parfois par email ou conversation IRC, j'ai pu « humaniser » ces emails de la liste de diffusion. « Humaniser », dans le sens de donner un visage humain à ces propos, mais aussi de comprendre ce qu'il y avait derrière ces considérations techniques et pourquoi telle personne affirmait ceci ou cela à ce moment-ci. Avant de participer (en ligne) à l'une des sessions de travail du groupe à l'IETF, j'ai également envoyé un email à la liste de diffusion pour informer les personnes que je faisais cette recherche et me dire ouvert à toutes questions ou remarques³⁶. Plusieurs personnes ont répondu à cet email, parfois simplement pour me dire « bonjour et bienvenu », parfois pour préciser que ces réunions étaient de toute façon enregistrées et publiées sur Youtube et que je n'avais pas à demander une autorisation, et d'autres fois, plus rares, pour engager une discussion à propos d'anthropologie, d'anarchie ou d'OpenPGP.

36 <https://mailarchive.ietf.org/arch/msg/openpgp/4djhaizzVsB7gnQjF1T8nc-UbE/>.

Par deux fois, un problème d’agendas m’a fait manquer une session en présentiel à laquelle j’aurais pourtant vraiment voulu participer. Dans les deux cas, il s’agissait de l’*OpenPGP Email Summit*, organisé en octobre 2019 et en mai 2022. Ces deux événements ont rassemblé les développeuses des principales d’implémentations d’OpenPGP pour les emails. Si le contenu des échanges peut être consulté sur le site dédié³⁷, je suis quand même frustré de ne pas avoir pu rencontrer en chair et en os ces personnes dont je ne connais pour certaines d’entre elles que l’apparence bi-dimensionnelle de leur visage encadré dans des rectangles de mon ordinateur.

2.4.2. Du côté de chez Sequoia

Cette frustration est pourtant en partie comblée par la bonne relation de travail et d’amitié que j’ai pu construire avec les membres de l’équipe de Sequoia, la deuxième communauté de pratique auprès de laquelle j’ai pu mener mon enquête dès l’été 2020. J’ai conduit quelques entretiens préliminaires dès le mois de juin mais la plus grande partie des observations a commencé en septembre 2020, période à laquelle j’ai commencé à assister à la plupart de leurs deux séances hebdomadaires qui avaient lieu en ligne. Travailler en ligne n’était pas uniquement dû à la pandémie mais au fait que les membres salarié·e·s de cette équipe habitaient dans différents lieux en Europe (principalement en Allemagne, mais aussi en Pologne et en Finlande). D’autres contributeurices bénévoles résidant en Amérique du Nord et en Asie se joignaient parfois à ces séances.

Comment, concrètement, se déroulaient ces séances et que s’y passait-il ? Elles avaient lieu deux fois par semaine en ligne par visioconférence sur une instance de Big Blue Button hébergé par la fondation Pretty Easy Privacy (PEP), la fondation qui soutient Sequoia³⁸. Elles duraient généralement entre deux et trois heures, l’anglais était la langue utilisée et nous étions d’habitude entre cinq et huit personnes. À tour de rôle, un·e des membres modérait la séance et donnait la parole à chacune et chacun, moi y compris, et nous racontions alors ce que nous avions appris, ce qui nous

37 <https://wiki.gnupg.org/OpenPGPEmailSummits>.

38 <https://bigbluebutton.org/> ; <https://pep.foundation/>.

avait occupé les derniers jours et ce que nous comptions faire les jours suivants. S'il y avait des problèmes à régler, comme un bogue récalcitrant, un manque de matériel, ou des questions en suspens, c'était généralement à ces occasions que cela se réglait. Ces séances se déroulaient généralement dans une atmosphère très détendue et agréable, bien qu'évidemment, il y eût quelques tensions inévitables qui éclataient de temps en temps.

Durant plusieurs mois, lors de la deuxième séance de travail hebdomadaire, l'équipe a organisé ce qu'elle appelait un « *community meeting* » qui consistait à inviter une personne externe à leur équipe de travail mais travaillant également sur OpenPGP. L'objectif était simplement d'apprendre à mieux se connaître et ainsi renforcer les liens de la communauté. Lors de ces réunions, l'équipe et la personne invitée se présentaient et discutaient d'un sujet – ou d'un problème – qui les occupait. C'était alors pour moi l'occasion de rencontrer de nombreuses personnes différentes, mais aussi d'avoir un large panorama des questions générales qui occupaient la communauté. Bien qu'intermittente selon les périodes, et notamment en fonction de mes enseignements, ma participation aux séances de Sequoia s'est étendue sur une année et demie, jusqu'à la fin de l'année 2021.

Participer à ces séances de travail en ligne m'a permis de produire une grande quantité de données qualitatives sur le travail quotidien de ces personnes. J'ai tenu un journal en notant comment se déroulaient ces séances, qui étaient présent·e·s, l'ambiance générale, les sujets abordés, la façon de résoudre certaines difficultés, et l'organisation générale du travail. J'ai pu assister à des séances de débriefing après la réalisation d'un objectif important. J'ai assisté à des séances où des membres de l'équipe annonçaient leur démission ainsi qu'à des discussions sur des candidat·e·s prêt·e·s à rejoindre l'équipe. J'ai assisté à des séances très techniques durant lesquelles je ne comprenais rien, et d'autres où l'on me demandait parfois mon avis. Ces données sont complétées par des entretiens que j'ai réalisés parallèlement à cette observation (voir section suivante).

Par deux fois, un des membres fondateurs de Sequoia a invité toute l'équipe chez lui pour passer du temps ensemble en dépit des contraintes liées à la pandémie de coronavirus qui avaient largement réduit le nombre d'occasions de se rencontrer. J'ai été très aimablement invité à ces séjours et j'en suis très reconnaissant, d'autant plus que nous profitons avidement du soleil sur la terrasse, et qu'entre deux sessions de codage, nous nous sommes même payé le luxe d'une partie de volley dans la piscine du jardin. Ces deux séjours ont été pour moi une occasion de vivre au rythme de cette équipe, d'apprendre à mieux connaître les membres présent·e·s et de *voir* leur travail. J'ai assisté à des séances de codage, de décodage, de *brainstorming* et de stratégie. Lors de ces séjours, je n'étais pas le seul invité et j'ai rencontré quelques ami·e·s des membres de l'équipe, parmi lesquels Philip Zimmermann, l'inventeur de Pretty Good Privacy, qui est venu des Pays-Bas pour passer quelques jours avec nous.

Ces sessions de travail avec l'équipe de Sequoia ont été très importantes pour moi. Elles m'ont donné l'occasion de voir le travail de codage, ainsi que tout le dispositif qu'on ne voit pas en visioconférence : les différents ordinateurs, les claviers, les écrans, et bien plus encore. N'étant plus restreint à deux fois deux heures par semaine, j'ai pu poser autant de questions que je voulais, de manière informelle et sans avoir le stress du temps, puisque nous avions beaucoup de temps à partager ensemble. Ces deux sessions m'ont également donné l'occasion de « traîner » avec ces personnes, les observant en train de coder, mais aussi en train d'incarner une oie dans un jeu vidéo ou de regarder des vidéos d'archives sur les péripéties de Philip Zimmermann à l'époque où il était encore poursuivi par le FBI.



Illustration 6: Interactions lors d'un terrain dans la maison d'une des codeuses de l'équipe de Sequoia. Source : photo personnelle.

2.4.3. Les entretiens

En plus de ces deux communautés de pratiques dont j'ai particulièrement suivi le travail durant cette deuxième phase de terrain, j'ai également rencontré en ligne et hors ligne de nombreuses autres personnes concernées par OpenPGP. Ces rencontres étaient parfois de simples *chats* ou de quelques échanges d'emails, et d'autres fois elles ont pris la forme de véritables conversations de plusieurs heures en présentiel. La plupart des personnes consultées étaient des codeuses qui travaillaient ou avaient travaillé avec ou pour OpenPGP, des participant·e·s au groupe de travail OpenPGP de l'IETF, des scientifiques qui avaient étudié OpenPGP, des journalistes qui avaient écrit sur OpenPGP, et quelques utilisatrices qui ont manifesté de l'intérêt pour ma recherche.

Ainsi, en plus des observations faites dans différents lieux, j'ai conduit une quarantaine d'entretiens d'au moins une heure avec différentes personnes de la communauté OpenPGP. Pour ces entretiens semi-directifs (Beaud 1996), j'ai utilisé un guide d'entretien contenant des questions ouvertes en planifiant assez de temps pour

aborder les sujets en profondeur, tout en gardant suffisamment de flexibilité pour explorer des pistes que je n'avais pas prévues. Je n'ai pas utilisé un seul et même guide d'entretien pour tous ces entretiens : j'adaptais mes questions au cas par cas en fonction de la personne interviewée. Les réponses sont donc circonstanciées et contextuelles.

En pratique, toutes les conditions n'étaient pas toujours réunies pour mener ces entretiens de manière systématique et idéale. Que ce soit les entretiens que j'ai menés dans les couloirs bruyants d'une conférence ou que j'ai dû improviser sur le pouce, la situation ethnographique demande parfois flexibilité et spontanéité, et offre en retour des surprises importantes, comme de nouveaux récits, de nouvelles hypothèses et interprétations, ou de nouvelles personnes à rencontrer.

Parmi les personnes avec qui j'ai discuté se trouvent des développeuses de diverses implémentations d'OpenPGP (principalement Sequoia, GnuPG, d'OpenPGP.js, rPGP, DeltaChat, PGPainless), des membres du groupe de travail (souvent aussi codeuses), ainsi qu'une mainteneuse de serveur de clés publiques et une codeuse pour une entreprise proposant des SmartCard. J'ai aussi interviewé six scientifiques, dont trois faisant partie de l'équipe d'EFAIL. Dans de rares cas, j'ai utilisé des entretiens conduits lors de la recherche exploratoire avec des cryptographes et codeuses engagées dans d'autres projets, qui sont néanmoins en lien avec mes questions de recherche. C'est par exemple le cas de deux développeuses de bibliothèques cryptographiques.

La plupart de ces entretiens ont eu lieu en ligne, à cause de la pandémie de coronavirus d'une part, et de la distance qui me séparait de ces personnes résidant souvent dans d'autres pays (notamment Allemagne, Pologne, Italie, Pays-Bas, Finlande, États-Unis, Canada, Inde, Japon). Cela a été l'occasion de développer de bonnes compétences dans la prise de rendez-vous en dépit des décalages horaires. Mais cela m'a surtout permis de discuter avec un nombre beaucoup plus important de personnes dans des conditions logistiques très agréables : installé·e·s derrière nos ordinateurs respectifs, nous pouvions faire ces entretiens au calme, avec tout le confort

du « chez soi » et sans perdre un temps considérable en déplacement. Par ailleurs, cette façon de faire s'est montrée très agréable pour partager des liens et autres informations, puisque tout pouvait être transmis directement dans la boîte de dialogue des différents logiciels que nous utilisions. Bien entendu, il existe aussi des inconvénients à cette démarche en ligne et le plus important d'entre eux est certainement un manque de fluidité dans les interactions, notamment à cause de toutes sortes de problèmes techniques (vidéo qui ne fonctionne pas, écho dans le son), ainsi que de la récurrente question : « est-ce que vous m'entendez ? » Parmi les autres inconvénients, il y avait aussi le manque de possibilité de lire le langage non verbal, le fait d'être « figé-e » devant l'écran plutôt que plus naturel·le, ou encore l'impossibilité d'avoir accès à l'environnement autour des personnes, ne serait-ce que pour partager une tasse de thé ou un bout de chocolat.

D'autres entretiens ont eu lieu en présence, dans des lieux les plus divers : une accueillante terrasse un matin ensoleillé dans la Fribourg allemande (Freiburg im Breisgau), les halles sombres et bruyantes du Chaos Communication Congress, ou une cage d'escalier d'une université suédoise. Cela m'a non seulement permis de tester les capacités de mon petit enregistreur à capter toutes sortes de sons, y compris la voix de mes interlocuteurices, mais également de confirmer la diversité des milieux dans lesquels OpenPGP évolue. Contrairement aux entretiens en ligne, ces entretiens étaient souvent moins confortables d'un point de vue logistique, mais beaucoup plus riches humainement. Nous pouvions partager à boire et à manger ; nous pouvions tester des choses ensemble sur le même ordinateur ; et surtout l'interaction était plus naturelle et sans écho – bien que parfois avec un bruit de fond assourdissant. Grâce à ces entretiens, j'ai lié une relation plus profonde et, peut-être, plus sincère – ou du moins moins superficielle – avec certaines personnes que j'ai revues plusieurs fois.

2.4.4. L'éthique de la recherche

J'aimerais préciser quelques éléments liés à l'éthique de la recherche. Mes interlocuteurices savaient que je faisais une recherche anthropologique et que les entre-

tiens étaient enregistrés. Elles ont toutes consenti oralement à cet enregistrement³⁹. D'une manière générale, je prenais toujours un moment avant les entretiens pour expliquer les buts de ma recherche, les méthodes de l'anthropologie et ce que j'espérais faire de ces données que nous produirions ensemble durant ces entretiens. J'ai notamment insisté sur le fait qu'aucune citation ne serait sortie de son contexte de manière à détourner le propos.

J'ai opté pour une approche processuelle de l'éthique (Perrin et al. 2020), revenant fréquemment vers les participant·e·s de ma recherche pour leur demander si ils et elles étaient toujours à l'aise avec ma présence et comprenaient toujours la raison de ma recherche. Quand ce n'était pas le cas, ce qui était finalement assez fréquent, je tâchais de clarifier ce que j'espérais réussir. Parfois, j'usais aussi de l'humour pour leur dire qu'il ne faut jamais faire confiance aux anthropologues qui tâchent souvent d'enfermer « leurs enquêté·e·s » dans des clichés, ce qui profiterait à leur propre carrière professionnelle sans pour autant nuire aux enquêté·e·s car peu de monde écoutent vraiment les anthropologues, à part d'autres anthropologues. Je leur montrais parfois la caricature de Gary Larson (illustration 7 ci-dessous), avant de reprendre un ton plus sérieux en expliquant ce que je faisais, comment je le faisais et pourquoi je le faisais.

39 J'ai renoncé à faire signer un formulaire de consentement car ce dernier a soulevé de nombreux soupçons lors de mes premiers entretiens.

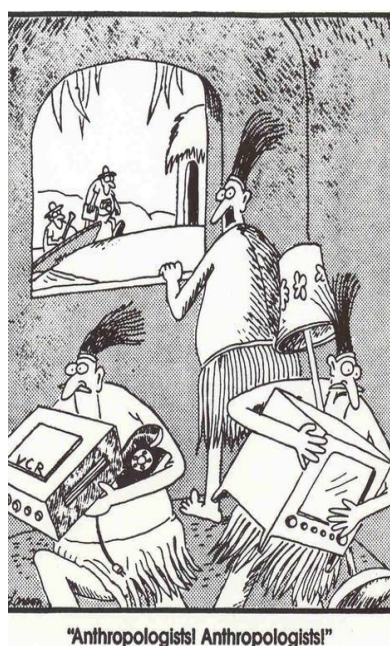


Illustration 7 : Caricature de Gary Larson sur les anthropologues.

J'expliquais par exemple en quoi consistait la « rigueur » de la recherche qualitative. Je faisais référence à ma volonté de me rapprocher au plus près de la perception de mes interlocutrices en adoptant les outils de recherche de l'anthropologie et de la sociologie qualitative (Olivier de Sardan 1996; Olivier de Sardan 2008). Dans cette quête de m'approcher au plus près des actrices de ma recherche, j'ai largement recherché les contre-exemples qui contredisaient mes hypothèses afin de pouvoir affiner celles-ci (Olivier de Sardan 1996). J'ai ainsi tâché de faire preuve de nuance et de finesse dans mes analyses sans vouloir imposer un modèle théorique à mes données ethnographiques, mais au contraire de faire en sorte que mes analyses soient « ancrées » dans mes données (Glaser et Strauss 2010). J'expliquais ainsi à mes interlocutrices que ce que j'essayais de faire était une « photographie » fidèle des pratiques de sécurité autour du protocole OpenPGP. Cette photographie aurait un temps d'exposition de plusieurs années, la rendant inévitablement un peu floue ; mais j'ai tenté de proposer un cadrage, une profondeur de champ et une mise au point aussi rigoureuse que ce que mes observations et mes entretiens m'ont permis de faire.

Un autre principe important qui a guidé ma démarche ethnographique est le principe de « ne blesser personne » [*no-harm principle*]. Un premier élément a été d'éta-

blir quelques règles de conduite et de rester vigilant sur ce qui pouvait éventuellement mal tourner. Il s'agissait premièrement de respecter le temps et le travail des personnes avec qui je voulais faire la recherche, par exemple en n'insistant pas si je n'avais pas de réponse de ces personnes⁴⁰. Deuxièmement, j'ai fait mon possible pour protéger les informations qu'elles me confiaient, notamment en chiffrant toutes mes données. Ces données sont en outre soigneusement stockées dans une archives chiffrées sur mon ordinateur portable et sur le serveur de l'université, ainsi que sur une série de sauvegardes incluant une partition permanente de TAILS par exemple. Troisièmement, lors des mois où je participais régulièrement aux séances de Sequoia, j'ai souvent demandé si cela était toujours bon pour l'équipe – en étant prêt à recevoir un « non » comme réponse.

Finalement, j'ai aussi tenté de ne pas prendre parti dans les comptes rendus de ma recherche. Par exemple, j'ai tâché de décrire les désaccords sans vouloir servir d'arbitre mais en tâchant de représenter aussi fidèlement que possible le point de vue des personnes concernées. J'ai notamment suivi le principe de symétrie qui m'incitait à rendre compte aussi bien des échecs que des succès de certaines idées (Bijker, Hughes, et Pinch 1987), ainsi que des éléments et arguments qui étaient considérés comme illégitimes ou qui étaient simplement exclus des controverses (Callon 2006).

2.4.5. Les données produites lors de la deuxième phase

Au final, les données que j'ai produites durant cette deuxième phase sont relativement hétérogènes dans leur forme et leur contenu. Elles consistent en des notes d'observations, des enregistrements et transcriptions d'entretiens, plusieurs extraits de discussions écrites (mail, *chat*, forum, discussion sur les plateformes GIT), ainsi que la littérature grise trouvée en ligne ou hors ligne récoltée durant ces années de recherche.

Les notes d'observations, qui s'étalent sur plusieurs années, m'ont permis de voir le temps long du travail de maintenance. Typiquement, j'ai pu observer comment

40 Il y a une personne en particulier avec laquelle j'aurais vraiment voulu discuter. Il s'avère que je l'ai contactée plusieurs fois, mais cela ne tombait jamais à un bon moment pour elle et je n'ai donc plus insisté.

certaines sujets revenaient régulièrement durant des semaines, voire des mois ou des années, et comment ils étaient ensuite clos de manière explicite ou implicite (parce qu'on n'en parlait plus). Dans ces notes, j'ai également pris du recul sur lesdits sujets techniques (que je ne comprenais pas toujours) pour me focaliser justement sur les relations entre les gens, et entre les gens et les objets de leur quotidien. Ces données ont été essentielles pour me forger une compréhension générale de la manière dont les projets-sécurité des un·e·s et des autres sont négociés au jour le jour, ainsi que pour me forger une vision à long terme de ces processus.

Dans une approche d'anthropologie sensorielle (Pink 2015), j'ai aussi tenté d'inclure mes impressions et sensations physiques dans mon analyse, notamment en écrivant des « mémos » sur les moments marquants de ma recherche. Ces données sont reproduites tout au long de ces pages et permettent de donner aux lecteurices une idée des conditions de recherche, du type de données que j'ai produites et analysées, et de ce que j'ai pu en déduire.

Les entretiens et les discussions écrites (emails, *chat*) que j'ai pu menées ont été d'une grande importance pour ajouter de l'épaisseur aux observations que je faisais, notamment en précisant certains points et en tâchant de comprendre différents points de vue. Ce sont les données les plus précises que j'ai produites, et celles que j'ai le plus mobilisées dans mon analyse. Elles sont très complémentaires aux notes d'observations, ainsi qu'aux lectures des nombreux documents que j'ai consultés lors de ma recherche (par exemple la documentation sur les implémentations ou des discussions sur des forums).

Parmi tous ces entretiens, seuls six ont été réalisés avec des personnes ne s'identifiant pas au genre masculin. Cela est en deçà du ratio déjà peu diversifié en sciences informatiques ⁴¹. Même si j'ai activement cherché à rencontrer des personnes qui

41 Après avoir discuté de cela avec Isabelle Collet, nous avons émis l'hypothèse que ce manque de diversité de genre n'était peut-être pas dû au manque de femmes, mais à la venue beaucoup plus massive d'hommes dans ce domaine d'élite et de prestige que représente la cryptographie. Lors de mes entretiens, et notamment ceux avec ces personnes qui n'étaient pas des hommes cisgenres, j'ai posé au moins une question concernant la surreprésentation des hommes cisgenres dans le milieu. Certaines personnes m'ont affirmé en avoir souffert et avoir eu le sentiment d'être souvent discriminées. Toutefois, je n'ai pas voulu creuser trop ce sujet car je savais que ma re-

s'identifient en tant que femmes, je n'ai rencontré que deux développeuses directement engagées dans une équipe de développement d'OpenPGP et à ma connaissance une seule membre féminine a directement contribué au groupe de travail OpenPGP lors des cinq ans de ma recherche. J'ai rencontré d'autres femmes, notamment des cryptographes et des scientifiques, mais elles n'étaient pas directement actives dans OpenPGP. Plusieurs étaient engagées dans des projets connexes ou dans l'activisme pour les droits numériques.

Pour compléter ces sources, j'ai également utilisé des supports visuels, qu'il s'agisse de photographies que j'avais moi-même prises, d'illustrations que je trouvais en ligne, des copies d'écran que je faisais ou encore d'images que j'avais moi-même dans la tête. L'iconographie me semble particulièrement importante car elle permet de montrer certains aspects cruciaux de ma recherche qui pourraient cependant facilement être occultés dans un document purement textuel. Lorsque je n'avais pas d'image sous la main pour exprimer ce que je désirais – et comme je suis un piètre dessinateur – j'ai parfois eu recours aux talents d'une amie, Léa Kissling⁴², ainsi qu'à une intelligence artificielle, *craiyon.com*, pour créer des images qui me permettaient d'exprimer ce que je ne savais pas dessiner. Ce sont des dessins fictifs qui ne servent qu'à illustrer mon propos.

Finalement, j'aimerais aussi indiquer que cette deuxième phase a aussi été marquée par l'arrivée de Francesca Musiani dans la direction de ma thèse. En effet, après avoir lu plusieurs de ses travaux sur les messageries sécurisées et la gouvernance d'Internet (Musiani 2012; 2015; 2018; Musiani et Ermoshina 2017; Ermoshina et Musiani 2022; Ermoshina, Musiani, et Halpin 2016; Musiani et al. 2016), je lui ai demandé si elle désirait codiriger ma thèse avec David Bozzini. Son encadrement m'a aidé à penser plus finement les enjeux de standardisation et de gouvernance d'OpenPGP.

cherche ne pourrait pas intégrer tous ces aspects et je ne voulais pas les discriminer à mon tour en leur posant moins de questions sur la fabrique de la sécurité, contribuant ainsi à les invisibiliser à mon tour dans ma recherche. Il me semble pourtant important de relever ici qu'un important travail mériterait d'être fait pour prolonger les études sur cette discrimination à l'encontre des personnes qui ne s'identifient et qui ne sont pas identifiées en tant qu'hommes cisgenres (Lagesen 2008; Abbate 2012; Collet 2019).

42 Elle peut être trouvée sur Instagram : @lea_kissling_officielle.

J'ai également profité de ses expériences de recherche au sein de communautés engagées dans la fabrique sociale des technologies de sécurité informatique, et notamment les messageries sécurisées. Finalement, son expertise dans le domaine m'a également permis d'affiner ma compréhension des modes de gouvernance de la sécurité informatique et des infrastructures numériques.

2.5. La troisième phase : désécriture et air iodé

Finalement, la troisième phase de ma recherche a consisté en l'analyse de ces données et la rédaction des pages (ou des pixels) que vous tenez devant vos yeux. Les processus d'analyse et d'écriture sont pour moi indissociables, bien que les manuels d'ethnographie ne soient pas forcément de cet avis. Ils sont indissociables pour moi parce qu'ils se renforcent l'un l'autre : poser mes idées dans une forme écrite m'a permis d'affiner mes analyses, et mes analyses se sont poursuivies en cours d'écriture. Je devrais d'ailleurs peut-être plutôt parler de désécriture. L'ajout du préfixe « dé » est inspiré d'une réflexion de François Laplantine dans son livre « Chronotopie » :

[...] l'usage de la préposition "dé" (comme dans désordre, déséquilibre, défection, démesure) [peut] présupposer un état préalable et même initial de stabilité considéré comme normalité. Si nous examinons avec attention le processus en apparence très simple de la marche, c'est le déséquilibre qui est premier et permanent alors que l'équilibre obtenu par un mouvement d'ajustement est un déséquilibre compensé. (Laplantine 2023, 39)

Inspiré de cette réflexion, j'ai commencé à percevoir l'écriture comme un mouvement d'ajustement perpétuel entre ce que je pense, ce que j'ai envie de dire et ce que j'écris effectivement. La désécriture est ce processus d'écriture, de relecture et de réécriture que j'ai observé tant chez les codeuses que chez moi afin d'obtenir un état provisoire de déséquilibre compensé. Alors que l'écriture fixe les choses, la relecture en relève les incohérences et les bogues. Cette dialectique m'a permis d'affiner mes conceptions et ainsi de proposer certaines idées plus abouties.

L'ethnographie suppose souvent de prendre du recul avec ce que l'on croit connaître pour aborder le sujet de notre étude sans préconception ; je pense que

l'écriture mérite le même travail d'introspection et de *désapprentissage*. J'ai par exemple désappris certaines règles de grammaire que je trouvais sexistes, notamment la règle de primauté du masculin et le soit-disant « masculin neutre » : il n'y a rien de neutre dans la volonté d'invisibiliser⁴³ une partie de la population, d'autant plus quand celle-ci est déjà minoritaire comme dans le domaine de la cryptographie qui, rappelons-le, compte encore très peu de femmes. J'ai ainsi préféré la règle de proximité pour l'accord. Cette règle consiste à accorder le genre et éventuellement le nombre de l'adjectif avec le plus proche des noms qu'il qualifie, et le verbe avec le sujet le plus proche⁴⁴. Quitte à agacer certaines lecteurices, j'ai aussi utilisé des mots épiciens comme « lecteurice » et « toustes », que j'ai accordés au féminin⁴⁵. J'ai également eu largement recours au point médian quand je ne trouvais pas de néologisme adapté.

Lors de l'écriture de cette thèse sur le vieillissement des technologies, j'ai été confronté à un autre problème : quels temps verbaux devais-je utiliser ? L'anthropologie a construit son objet d'étude, « les autres », en l'enfermant dans un présent ethnographique problématique (Fabian 2017). J'étais dès lors récalcitrant à utiliser le présent pour parler de mes observations, car mes observations ont été faites plusieurs années avant l'écriture. Toutefois, le résultat de cette écriture au passé n'était pas très convaincant et c'est pourquoi je me suis autorisé à utiliser le présent pour parler des éléments qui sont encre d'actualité à l'heure où je corrige une dernière fois ces lignes, en été 2023. Par exemple, quand je dis que « le cadenas bleu est un symbole de la sécurité informatique », cela ne signifie pas que ce sera toujours le cas, mais que ça l'est en 2023. Cela peut sembler aller de soit, mais cette précision me

43 J'ai visiblement aussi recours à des mots qui n'existent pas dans le dictionnaire, comme « invisibiliser ».

44 Par exemple, dans la phrase « le bogue et la vulnérabilité sont dangereuses », l'adjectif « dangereux » est accordé au féminin car « la vulnérabilité » est le nom le plus proche. En revanche, l'accord est au masculin si c'est « bogue » le sujet le plus proche : « la vulnérabilité et le bogue sont dangereux. » Voir https://fr.wikipedia.org/wiki/R%C3%A8gle_de_proximit%C3%A9.

45 J'ai été inspiré par l'excellent podcast de Victoire Tuaillon « Les couilles sur la table » (<https://www.binge.audio/podcast/les-couilles-sur-la-table>).

semble particulièrement indiquée dans le cas de cette thèse sur le vieillissement des technologies du numérique.

François Laplantine (2023) suggère également que les catégories de langue se retrouvent dans nos manières de catégoriser ce qui nous arrive et ce qui nous entoure. En écrivant, j'ai réalisé que mes pensées – qui ne s'expriment pas en moi à travers des mots mais des formes, des textures et des sons – se retrouvaient parfois trahies par leur traduction en mots qui mobilisaient des catégories trop rigides ou trop orientées : « être dangereux », « être sûr », « être bien ou mal écrit », etc. Ayant appris à bredouiller en espagnol, j'ai par exemple été sensibilisé aux différences que j'ai trouvées subtiles entre les verbes « *ser* » et « *estar* », tous deux traduits « être » en français. L'usage de l'un ou l'autre de ces verbes n'a pourtant rien de subtil pour des hispanophones pour qui il s'agit de deux verbes bien distincts. Le premier est utilisé pour décrire l'essence ou l'identité permanente d'un sujet, tandis que le deuxième est utilisé pour décrire l'état ou la condition temporaire d'un sujet. En écrivant, « cet algorithme cryptographique est sûr », lequel de ces deux verbes devrais-je utiliser en espagnol ? Dans cette même phrase, un autre mot mérite qu'on s'y attarde : l'adjectif « sûr ». En français, « sûr » signifie à la fois que la chose ne présente pas de danger (sécurité) et qu'elle est fiable (sûreté). Désapprendre ce qui va de soi, y compris des catégories de langages, m'a ainsi permis de repenser en profondeur ce qu'est la sécurité, où plutôt comment je pourrais me la représenter et décrire la manière dont mes interlocuteurs se la représentent.

2.5.1. Les processus d'analyse

Pour mener mon analyse, j'ai premièrement rassemblé le corpus hétérogène de données que je voulais utiliser. Il consistait d'une part en des documents écrits (transcriptions d'entretiens, extraits de conversations par emails ou par *chats*, PV d'observations, extraits de sites Internet, blogs, tweets), et des images (mes propres photographies ou des images récoltées lors de ma recherche, comme des visuels de site Internet ou l'iconographie du cadenas bleu). Selon le modèle de la théorie ancrée (Glaser et Strauss 2010), j'ai ensuite codé (manuellement) ces données pour créer des

catégories générales autour desquelles s'est construite mon analyse. Parmi les codes les plus récurrents se trouvent, dans l'ordre alphabétique, « art/artisanat », « bogue/failles/vulnérabilité », « bricolage », « codage », « collaboration », « conflits », « EFAIL », « expertise/savoirs », « incertitude », « infrastructure/réseau », « gouvernance/prise de décision », « négociations », « sécurisation », « temps/*tempo* », « vision à long terme ». C'est ainsi que je me suis laissé surprendre par certaines thématiques, et notamment celle du temps et de ses trois dimensions étudiées dans cette thèse : le mouvement, le rythme et la durée. En commençant ma recherche, la thématique du temps m'était totalement étrangère et je ne pensais pas l'aborder. Or, à force d'être au contact de durées extrêmes (de quelques centièmes de secondes à plusieurs milliards d'années) et de rythmes tout aussi divers (des centaines de tweets par jour à quelques emails par mois, en passant par le quotidien routinier), le temps s'est imposé comme un élément central de la sécurité informatique.

Parmi les autres surprises qu'une telle approche m'a procurées se trouvent les thématiques liées à la collaboration, l'entraide et l'inquiétude altruiste. Alors que ces thématiques sont récurrentes dans les communautés open source (Raymond 2001; Levy 2013; Lallement 2015; Broca 2018), elles sont beaucoup moins mises en avant dans l'imaginaire de la sécurité informatique où une pensée binaire domine (attaque/défense ; sécurité/danger ; allié/ennemi). Finalement, l'incertitude est aussi ressortie comme un thème central après l'analyse de mes données, ce qui n'est certes pas étonnant du point de vue des STS, mais qui m'a tout de même étonné de par l'ampleur du phénomène. Alors qu'au début de ma recherche, je pensais que la cryptographie était un champ relativement systématique et scientifique, j'ai été étonné de constater la récurrence de données soulignant son côté improvisé et bricolé.

Cette troisième phase de ma recherche a par ailleurs impliqué de me détourner des actualités de la communauté OpenPGP pour ne me concentrer que sur les données que j'avais déjà produites. Je ne suis toutefois pas parvenu à totalement « tourner le dos » aux actualités relatives à OpenPGP. J'ai notamment gardé un contact régulier avec plusieurs participant·e·s à la recherche, notamment des codeuses de

Sequoia et ProtonMail, ce qui m'a permis d'avoir un suivi sur certaines thématiques. Ce contact, souvent informel, avait non seulement pour but de maintenir la relation d'amitié et de travail avec ces personnes, mais également de prendre des nouvelles de l'état d'OpenPGP et de leurs projets respectifs. Et c'est donc avec un regard à la fois médusé et hagard que j'ai pris connaissance en automne 2022 d'un nouveau coup dur pour la communauté, et spécialement pour plusieurs codeureuses qui avaient investi de nombreuses heures de travail dans ce dossier, de même que leur énergie et leur patience. Une partie du groupe de travail se désolidarisait des consensus trouvés durant la dernière année de travail, refusait de continuer les négociations et quittait le processus⁴⁶. Tout espoir de voir une nouvelle norme publiée n'était pas encore perdu mais une fois de plus repoussé.

2.5.2. De crises en crises : la banalité du quotidien

Cette « crise » me permet de revenir ici sur un point important à souligner : je suis « entré » dans la communauté OpenPGP en même temps que la « crise » d'EFAIL l'ébranlait en mai 2018, et je me suis distancié de ses activités quotidiennes pour me concentrer sur le processus de désécriture au moment d'une autre « crise » de scission du groupe de travail. Hasard du calendrier ? Sérendipité ? Ou au contraire, construction *a posteriori* ? En réalité, je pense qu'il s'agit de quelque chose qui dépasse largement l'anecdote. Ces deux « crises » sont certes importantes, mais elles ne sont pas les seules à avoir ponctué ma recherche, et encore moins à avoir ébranlé la communauté OpenPGP ces dernières décennies. Rien que durant les quelques années de ma recherche, j'ai pu être témoin de nombreuses « crises » mineures et majeures, et il y en aura encore certainement toujours autant dans l'avenir. Certaines

46 Résumer cette affaire n'est pas simple car certaines tensions remontent à plusieurs années. Toutefois, la « crise » a éclaté après qu'une nouvelle version du brouillon de la norme OpenPGP ait été publiée par Werner Koch, fondateur de GnuPG et co-éditeur du nouveau groupe de travail. Ce nouveau brouillon n'était pas compatible avec plusieurs décisions prises depuis la refonte du groupe de travail. De nombreux emails ont été envoyés sur la liste de diffusion, et comme Werner Koch ne répondait pas aux questions qui lui étaient adressées, plusieurs interprétations ont été faites quant à ses motivations à publier un tel brouillon. Voir le fil de discussion « *a new draft overlapping the WG draft* » sur la liste de diffusion du groupe de travail : https://mailarchive.ietf.org/arch/msg/openpgp/PWp3ZcZ_qnDNLhuT-zR7gA2ddeg/.

étaient provoquées volontairement avec une nette intention de nuire⁴⁷ alors que d'autres étaient liées à la divulgation de failles⁴⁸. D'autres étaient dues à des dissensions internes à la communauté OpenPGP ou aux équipes de développement⁴⁹. Bien entendu, il y a aussi eu des moments de déblocage qui peuvent être vus comme des « décrises » : la refonte du groupe de travail en hiver 2020 en est un exemple, ou l'implémentation de solutions face à des bogues et autres vulnérabilités.

Par ailleurs, les « crises » sont également une banalité dans mon propre travail quotidien, spécialement en ce qui concerne cette dernière phase d'analyse et d'écriture. Les lecteurices n'ont sans doute que peu d'intérêt à découvrir les incroyables talents de mon moral à jouer au yo-yo et les bienfaits de la méditation, du yoga et surtout des côtes bretonnes en la matière. Pourtant, un point me semble important car il concerne directement le produit que vous, lecteurices, tenez devant vos yeux : si l'état de « crise » était à ce point présent à la fois dans les communautés concernées et dans la cervelle de celui qui écrit ces lignes, c'est peut-être qu'elles ne sont pas si exceptionnelles que cela. En effet, il semble que les moments de crise constituent une certaine normalité. À l'instar de la panne et de la casse qui constituent le quotidien de la vie matérielle et logicielle des infrastructures (Vyas 2013; Jackson 2014; Nova et Bloch 2020; Denis et Pontille 2022), les crises semblent être un état de normalité dans les communautés open source avec lesquelles j'ai effectué ma recherche.

D'un point de vue de la méthodologie, décrire ce rapport aux crises me semble crucial pour comprendre ma démarche. En effet, ces crises, de même que les conflits (Olivier de Sardan 2008) ou les controverses (Callon 2006; Lemieux 2007; Venturini 2010; Raynaud 2018), présentent un avantage heuristique majeur : ce sont des moments d'explicitations des enjeux et d'élicitation de la situation (Dousset 2019). En tâ-

47 Par exemple la crise de la pollution des clés publiques de certaines personnalités de la communauté OpenPGP sur les serveurs de clés en été 2019.

48 EFAIL (Poddebniak et al. 2018), mais aussi « SHA-1 is a Shambles » (Leurent et Peyrin 2020).

49 Le groupe de travail de l'IETF est par exemple régulièrement ébranlé par des controverses internes, et les équipes de travail ont bien entendu aussi des inévitables problèmes : démissions, pertes de financement, désaccord interne.

chant d'observer, de décrire et de donner du sens à ces moments, les anthropologues et sociologues peuvent alors étudier des enjeux plus grands, des enjeux de gouvernance notamment, mais aussi de rapport au monde. Une question m'a particulièrement occupé : que peuvent nous apprendre ces « crises », et leur répétition perpétuelle, sur la nature de ce qu'est la sécurité ?

L'ouvrage de Janet Lee Roitman « *Anti-Crisis* » (2014), centré sur le contexte de la crise économique de 2008, apporte quelques réponses méthodologiques intéressantes. En considérant le pouvoir performatif de certains termes et la manière dont ceux-ci orientent nos pensées et analyses, l'anthropologue s'est interrogée sur le statut de la notion de crise dans les sciences sociales et sur les conséquences d'un tel diagnostic de crise. Dans un entretien à propos de son ouvrage, elle affirme que « qualifier une situation de “crise” implique que ce qui était à un moment donné parfaitement intelligible et présenté comme productif [...] est maintenant sans fondement, et interprété comme porteur d'une valeur négative [...] » (Angeli Aguiton, Cabane, et Cornilleau 2019, 114). Dans cette perspective, les « crises » sont perçues comme des échecs, des erreurs, dont il s'agit de trouver les coupables. En considérant les « crises » comme des « moments de vérité » qui permettent de rendre certaines choses visibles et d'autres invisibles, elle ne nie pas les « crises » ni le statut de victimes de celles et ceux qui en souffrent, mais examine comment ces « crises » (ré)affirment des principes moraux, économiques et normatifs.

Méthodologiquement, c'est un argument intéressant pour examiner la manière dont la sécurité informatique se fait et se défait. Le diagnostic de « crise » est généralement rapidement prononcé, mais la sortie de « crise » n'est que très rarement décrétee. Est-ce parce qu'il est plus facile de casser la sécurité que de la réparer ? Parce qu'une vulnérabilité peut être prouvée par une preuve de concept, alors qu'un patch ne peut tenir que tant qu'un autre bogue n'apparaît pas ? Ou parce que la « crise » fait intrinsèquement partie de la manière de penser et d'organiser la sécurité, les pratiques d'écriture et de réécriture du code, et la recherche scientifique à ce sujet⁵⁰ ? Et

50 La tendance à utiliser des titres de plus en plus courts et accrocheurs pour des publications académiques serait en ce sens un bon point de départ pour voir comment le simple titre fait déjà passer

comment identifier quelles sont les temporalités qui constituent ces diagnostics de « crises » et qui, en même temps, en émergent ?

Il me semble en outre important de relativiser les crises qui ont souvent tendance à être proclamées haut et fort, en particulier dans un contexte de processus de sécurisation. Toutefois, comme l'affirme Michel Serres (2012, 18), « [la nouveauté d'un événement] est proportionnelle à la longueur de l'ère précédente, que cet événement clôt ». La plupart de ces crises n'ont pas clôturé des périodes très longues et même EFAIL, qui a été un événement marquant des dernières années, n'a pas marqué un tournant définitif. Fait marquant : aucun événement n'a par exemple « tué » OpenPGP. Cela permet donc de nuancer l'importance de certaines « crises », sans pour autant nier l'impact émotionnel que celles-ci peuvent avoir sur le moment auprès des personnes concernées.

Finalement, la manière dont j'ai analysé mes données est marquée par la conviction que ma recherche ethnographique aurait pu ne jamais s'arrêter tant les situations de crises s'enchaînaient avec à chaque fois de nouveaux éléments. Une approche centrée sur l'événement (Bensa et Fassin 2002) permet de comprendre l'ordinaire grâce à l'extraordinaire. L'événement offre une remise en question et des moments d'élicitation de l'incertitude (Dousset 2019) qui permettent de comprendre le « normal ». Toutefois, une focalisation uniquement sur l'extraordinaire ne saurait rendre compte des pratiques quotidiennes de soin qui rythment le travail des participant·e·s à ma recherche. Cette conviction que la « photographie » que j'ai prise durant ces cinq années de recherche n'a rien de définitif et ne représente qu'un petit instant dans le cours des choses liées à OpenPGP m'a donné un angle de vue et un cadrage particuliers pour décrire ce qui me semblait compter et que je me suis appliqué à décortiquer tout au long des chapitres qui suivent : l'intrication des différents

les lecteurices d'un état stable (avant sa lecture) à un état de doute et d'incertitude : « *Johnny you are fired!* » (Müller, Brinkmann, Poddebniak, Böck, et al. 2019), « *SHA-1 is a Shambles* » (Leurent et Peyrin 2020) ou « *In Encryption We Don't trust* » (Dechand et al. 2019) sont trois exemples de titres où l'état de crise est latent. Cela est parfois accentué par le nom donné aux vulnérabilités : EFAIL (Poddebniak et al. 2018), « Drown » (Aviram et al. 2016), « Spectre » (Kocher et al. 2019) et « Melt-down » (Lipp et al. 2018), etc.

« temps » dans le façonnage, la fabrication ou la détérioration de la sécurité informatique.

3. Alice et Bob aux pays des bogues

Yes, I'm Paranoid – But Am I Paranoid Enough?
*David Foster Wallace, Infinite Jest*⁵¹

Elle s'appelle Alice et il s'appelle Bob. Nous ne connaissons jamais tous les secrets de leur histoire, mais ce sont les protagonistes principaux de toutes les intrigues cryptographiques depuis 1978⁵². Alice et Bob sont les deux acolytes les plus célèbres de l'univers « crypto », prototypes de la paranoïa cryptographique. Alice et Bob font toutes sortes de plans pour communiquer le plus discrètement possible, concevant au passage de brillants protocoles cryptographiques. La cause de leur prudence ? Une troisième personne, très curieuse, probablement malveillante et certainement tout aussi ingénieuse, tente d'intercepter leur communication : il s'agit d'Ève. Son nom provient du fait qu'elle désire espionner la conversation d'Alice et Bob : « *Eve, the eavesdropper* »⁵³. Ève est capable de presque tout : très intelligente, elle a accès à

51 1996. Little, Brown and Company Inc.

52 Soit depuis que les créateurs de RSA, Ronald L. Rivest, Adi Shamir et Leonard Adleman ont publié leur article « A method for obtaining digital signatures and public-key cryptosystems » (Rivest, Shamir, et Adleman 1978).

53 On peut se demander si c'est un hasard que la personne (imaginaire) la plus agressive et malveillante des histoires de cryptographie porte le même nom que la femme d'Adam dans la Bible, celle qui a été tentée par le fruit défendu et qui a banni l'humanité (qui se réduisait à Adam et elle soit dit en passant) du paradis. Les personnes à qui j'ai posé la question, très souvent des hommes car les hommes sont encore beaucoup plus nombreux que les femmes dans le milieu, s'amusaient de ma remarque et me répondaient que c'était surtout parce que son nom était un diminutif pratique de « *the Eavesdropper* », celle qui écoute aux portes. Mais pourquoi est-ce une femme, et pourquoi s'appelle-t-elle ainsi ? L'imaginaire sexiste et patriarcal du numérique, ainsi que la très

d'infinies ressources matérielles, logistiques et computationnelles. Alice et Bob peuvent donc avoir la certitude qu'elle saura trouver la moindre faiblesse dans leur protocole s'il en existe une.



Illustration 8 : Alice, Bob et Ève sont souvent l'objet de plaisanteries dans le milieu. Parfois, on se moque aussi des cryptographes.
Source : <https://xkcd.com/1323/>.

Ensemble, Alice, Bob et Ève forment le triangle inséparable de la cryptographie numérique⁵⁴. Il et elles étaient présentes dans toutes les présentations de protocole de chiffrement que j'ai pu voir – même les plus techniques et scientifiques – et ensemble, ce trio sert à penser la sécurité des protocoles et des chiffrements. Parfois,

forte propension des informaticiens (et, plus rarement, des informaticiennes) à se prendre pour Dieu qui crée un monde nouveau (Collet 2019) ne me semblent pas négligeables dans cette affaire. De plus, il n'y a pas besoin de beaucoup de créativité pour imaginer d'autres noms, comme Spike (pour « *the spy* », l'espion) et qui pourrait renvoyer à l'idée de pic. Tout cela mériterait une enquête plus poussée.

54 D'ailleurs, plusieurs sites dédiés à la cryptographie portent leur nom, notamment une implémentation d'OpenPGP (<https://aliceandbob.io/>) et un site de vulgarisation du fonctionnement de la cryptographie (<https://alicebobstory.com/>). Un site Internet dédié raconte leur histoire et présente une chronologie interactive (<http://cryptocouple.com/>). Il faut également mentionner la page Wikipédia qui leur est dédiée en de nombreuses langues (https://fr.wikipedia.org/wiki/Alice_et_Bob).

d'autres personnes se joignent à la fête : Carol et Dave (ou parfois Charlie et David) sont deux autres personnes qui connaissent soit Alice, soit Bob, mais généralement pas les deux à la fois, et qui aimeraient participer à la conversation. Mallory est quant à elle une personne assez détestable puisqu'elle contrôle les serveurs et peut à tout moment récupérer toutes les données qui s'y trouvent pour en faire ce que bon lui semble, et même modifier les messages. Mallory est un fléau encore plus embarrassant qu'Ève, car elle est une attaquante active alors qu'Ève se restreint généralement à un rôle passif d'écoute.

Il y a encore Johnny, un pauvre type qui se donne beaucoup de peine pour utiliser correctement des outils cryptographiques, qui a beaucoup de peine dans cette tâche, et qui fait également beaucoup de peine⁵⁵. Alors qu'Alice, Bob et Ève sont généralement mobilisées pour parler des spécifications des protocoles ou logiciels en question, Johnny est quant à lui régulièrement utilisé pour représenter les utilisateurices qui ne s'en sortent pas aussi bien. Toute une série d'articles sur la facilité d'usage [*usability*] d'OpenPGP lui est d'ailleurs consacrée⁵⁶. Ce ne sont là que quelques exemples et d'autres personnes viennent parfois encore s'ajouter⁵⁷.

55 Comme on le verra par la suite, Johnny est un surnom récurrent dans les études sur l'usage des technologies de sécurité informatique. Il s'agit sans doute du surnom de John Doe, une expression pour qualifier un homme lambda : https://fr.wikipedia.org/wiki/John_Doe.

56 La section 3.1.3 (p. 119) du présent chapitre donne un aperçu de ces études qui s'étalent sur des décennies.

57 Une liste de ces personnages peut être consultée sur la page de Wikipédia : https://fr.wikipedia.org/wiki/Alice_et_Bob.

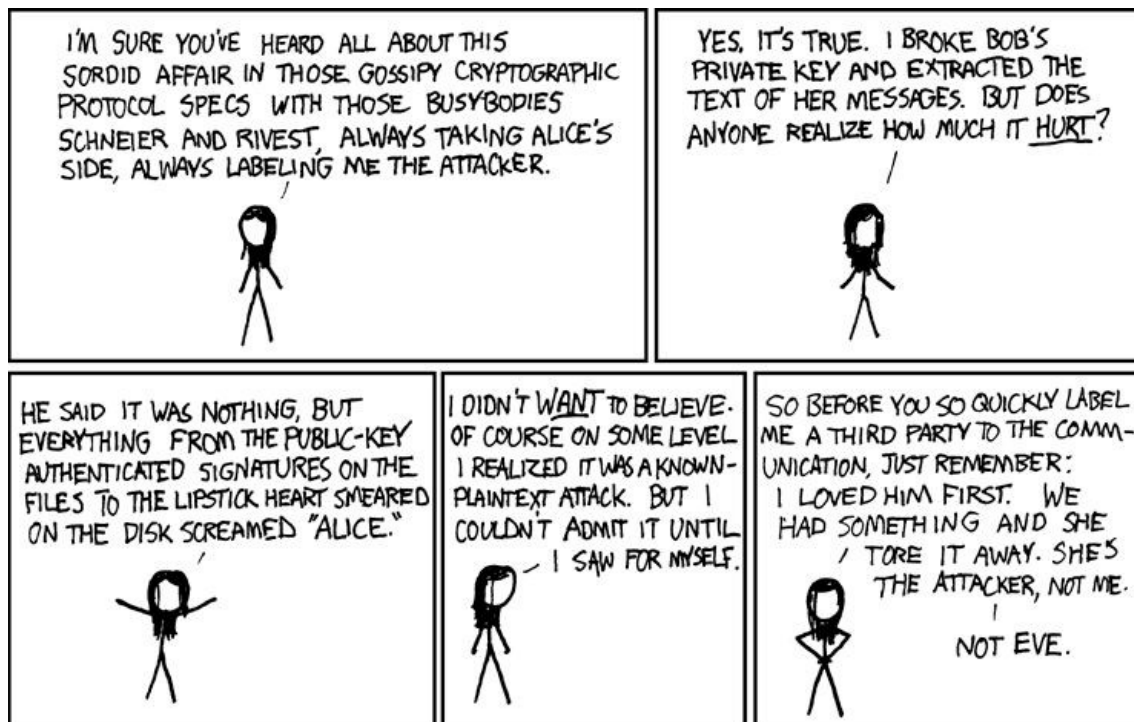


Illustration 9 : Les représentations misogynes ne sont pas absentes de ce triangle entre Alice, Bob et Ève. Les caricatures présentes sur le site [xkcd.com](https://xkcd.com/177/) permettent de rendre compte de certaines représentations qui font rire (ou sont supposées le faire) quelques ingénieur·e·s en informatique. Source : <https://xkcd.com/177/>.

Maintenant que sont faites les présentations avec Alice, Bob et Ève (et les autres), intéressons-nous à ce que leurs aventures nous apprennent sur le fonctionnement de la cryptographie. Ce chapitre a pour but de comprendre comment les participant·e·s à ma recherche se représentaient la sécurité informatique et comment ils et elles la traduisaient en différents produits. Dans un premier temps, je décris comment la sécurité informatique est pensée, conçue et implémentée dans des protocoles et des programmes. J'analyse les différentes traductions dont font l'objet les histoires d'Alice, Bob et Ève, de l'élaboration de scénarios au codage de programmes cryptographiques. Ces scénarios permettent de comprendre comment ingénieur·e·s, cryptographes et codeureuses tâchent de concevoir des systèmes de communication qui répondent à trois critères fondamentaux : la confidentialité, l'intégrité et l'accessibilité. Cette vision de la sécurité centrée sur des propriétés techniques est nuancée dans la deuxième section de ce chapitre, qui mobilise les théories de la sécurisation d'une part et les études sur le soin [*care*] d'autre part. Cette deuxième section apporte un

éclairage plus sociologique sur les processus qui créent des situations d’(in)sécurité, ainsi que sur les pratiques de maintenance des technologies numériques. Finalement, dans la troisième section, je définis la notion de « projet-sécurité » qui vise à compléter ces différentes approches de la sécurité par un outil qui se veut à la fois heuristique et épistémologique.

3.1. De la CIA aux multiples traductions successives

sylvi : Pourrais-tu peut-être m'expliquer comment on s'y prend pour évaluer la sécurité d'un système cryptographique ? [...]

Professeur·e de cryptographie : Okay, donc... tu veux certaines propriétés, n'est-ce pas ? J'aurais besoin de tout un semestre de cours pour t'expliquer tout en détail, mais à la base, tu as la confidentialité, l'intégrité et l'accessibilité.

(Professeur·e de cryptographie, entretien du 27 décembre 2018)

Je n'ai pas eu l'occasion de suivre les cours de ce·tte professeur·e pour avoir la réponse à ma question. J'ai toutefois suivi d'autres cours et ateliers en ligne et hors ligne sur la sécurité, le hacking et la cryptographie. J'ai aussi assisté à de nombreuses conférences où il était question de sécurité, d'insécurité, de *hacks* et de divulgations fulgurantes de vulnérabilités critiques. À chaque fois, la triade CIA – Confidentialité, Intégrité, Accessibilité⁵⁸ – s'immisçait dans la définition de ce qu'était la sécurité. Il s'agit d'une vision émique centrale et largement répandue au sein du champ de la sécurité informatique. Commençons par voir à quoi font référence ces trois termes, avant d'analyser comment ils sont traduits en protocoles, en programmes informatiques et en usages.

La triade CIA

La **confidentialité** est la propriété qui empêche un accès illégitime aux informations privées. Elle est mise en œuvre à travers le chiffrement des dites don-

58 Parfois, cette triade est appelée « les trois A de la sécurité », pour Anonymat, Authenticité et Accessibilité.

nées grâce à une clé (un mot de passe ou la clé publique des destinataires). Bien que parfois utilisés comme synonymes, les termes « confidentialité » et « anonymat » présentent certaines différences. L'anonymat signifie qu'on ne peut pas avoir accès à l'identité de la personne qui émet un message, alors que la confidentialité ne concerne généralement que le secret du message lui-même. Ainsi, l'anonymat concernerait plutôt les méta-données (Alice a écrit à Bob hier soir depuis son téléphone portable) alors que la confidentialité concernerait quant à elle les données elles-mêmes (l'heure et le lieu du rendez-vous). Toutefois, il ne faut pas généraliser et de nombreuses discussions ont lieu pour tenter de définir ce que le terme « privacy » signifie techniquement, et quelles sont les données que l'on cherche à protéger (Kuhn et al. 2019). Dans la pratique, on s'entend néanmoins sur le principe que certaines informations ne devraient être lues que par certaines personnes, supposées légitimes.

L'**intégrité**, ou l'**authenticité**, concerne la manière d'attester du fait que l'information en question n'a été ni modifiée ni falsifiée. Il s'agit d'une fonctionnalité cruciale en termes de sécurité puisque sans garantie d'intégrité, il est possible de *forger* des informations qui paraissent authentiques mais qui ne le sont pas. Dans le cadre des emails, les spams et le *fishing* comptent parmi les problèmes les plus récurrents.

Mais un autre usage est encore plus présent dans l'écosystème d'OpenPGP : la signature des paquets open source par les développeuses. En effet, en signant un paquet avec sa propre signature ou celle de l'entreprise, les codeuses attestent de l'intégrité dudit code. Non seulement, ils et elles en sont responsables, mais surtout les utilisateurices peuvent ainsi s'assurer qu'aucun code n'a été introduit par un tiers malveillant. La signature numérique est l'une des possibilités pour garantir l'authenticité d'un message (Blanchette 2012)⁵⁹, mais d'autres formes peuvent être implémentées. Ainsi, dans le cha-

59 Certains protocoles comme le protocole OTR garantissent par exemple l'authenticité des propos durant une conversation mais sitôt celle-ci terminée, ces messages peuvent être modifiés par les deux parties, voire n'importe qui. On appelle cette propriété le déni plausible (*plausible deniability*, en anglais). Il s'agit d'une imitation d'une discussion que deux personnes peuvent avoir en face à face à l'écart d'oreilles indiscretes : sur le moment, il n'y a souvent aucun doute quant à la confidentialité et l'intégrité des propos tenus ; et après que la conversation s'est terminée, aucune

pitre 7 (« Lenteur et consensus à l'IETF », p. 256), il est question de l'AEAD qui est un mode de chiffrement symétrique qui garantit l'authenticité d'une information chiffrée sans avoir recours à la signature numérique.

Quant à l'**accessibilité**, il s'agit d'une propriété qui semble couler de source et qui, pourtant, fait souvent défaut. En effet, même le meilleur système cryptographique ne serait d'aucune utilité si l'information en question n'est pas disponible. De même, il s'agit aussi de garantir que les données chiffrées et/ou signées puissent être respectivement déchiffrées et vérifiées sans encombre. L'accessibilité paraît aller de soi, mais il s'agit d'un sérieux problème qui pose des questions très concrètes⁶⁰.

En résumé, ces trois propriétés – confidentialité, intégrité et accessibilité – semblent être les pierres angulaires de la sécurité informatique. Elles sont liées entre elles et une brèche dans un des aspects permet parfois de casser l'entier du système. Cette triade sert de socle commun et consensuel à la définition de la sécurité informatique des actrices rencontrées lors de ma recherche. Toutefois, si avoir trois mots (« confidentialité, intégrité et accessibilité ») plutôt qu'un seul (« sécurité ») peut déjà être considéré comme un début intéressant pour appréhender ce qu'est la sécurité informatique, il n'en demeure pas moins qu'il peut exister de nombreuses interprétations dans la manière de comprendre ces trois propriétés.

3.1.1. Premier niveau de traduction : établir un protocole de sécurité

Au-delà des anecdotes amusantes et des dessins XKCD concernant Alice, Bob et Ève, les micro-scénarios les impliquant représentent un outil conceptuel crucial qui permet de comprendre comment la sécurité est pensée puis traduite en protocoles et

preuve n'existe et seuls subsistent les souvenirs des deux personnes.

60 Dans de rares cas, l'inaccessibilité peut aussi s'avérer être une fonctionnalité : lors d'une conversation informelle, Philip Zimmermann, le créateur de PGP, m'a raconté sa stupéfaction quand il a compris qu'il pouvait détruire à distance ses archives sans même y avoir accès. Comme elles étaient chiffrées avec une clé privée, il lui suffisait de détruire cette clé pour que les archives soient également détruites – et cela même si elles continuaient d'exister sur des serveurs. Ces données devenaient inaccessibles de la même manière qu'on peut broyer des documents papiers confidentiels quand on cesse d'avoir besoin d'eux : la matière continue d'exister et peut d'ailleurs être recyclée, mais les informations sont, elles, perdues.

en codes informatiques. J'utilise à dessein le terme de traduction en pensant aux coquilles Saint-Jacques de la baie de Saint-Brieuc. Dans un article fondateur de la sociologie de la traduction, Michel Callon (1986) a étudié les controverses autour de la gestion des ressources desdits mollusques et a analysé les contributions des différent·e·s représentant·e·s de la pêche, des autorités publiques et de la science. Il a décrit comment la gestion des ressources de coquilles Saint-Jacques impliquait de nombreuses négociations, compromis voire conflits entre les différentes personnes engagées, et il a nommé « traduction » ce processus de transformation de différents récits pour s'accorder sur une réalité commune et partagée. En tâchant d'adapter sa vision au contexte de la sécurité informatique, je me suis permis de modifier la citation de Callon ci-dessous en remplaçant les mots « chercheurs » par « cryptographes », « expériences » par « expériences de pensée », et « les entités qu'ils étudient » par « les scénarios qu'ils et elles imaginent » :

Le travail des [cryptographes] consiste à mettre en place des expériences [de pensée] pour faire « écrire » les [scénarios qu'ils et elles imaginent], puis à mettre en forme ces inscriptions, et ensuite à les combiner, les comparer et les interpréter. Au terme de ces traductions successives, les [cryptographes] produisent des énoncés décrivant ce que sont capables de faire les [scénarios] sur lesquelles sont menées les expériences [de pensée]. (Callon 2006, 268s)⁶¹

Cette pseudo-citation est intéressante car elle permet de comprendre ce que font les scénarios imaginés d'Alice et Bob. Basées sur des hypothèses contextuelles, ces histoires sont des traductions simplifiées de scénarios qui produisent un « triple inventaire » : inventaire des actrices, inventaire des problèmes, inventaire des options/alternatives (Callon, Lascoumes, et Barthe 2001, 49). Il s'agit donc de réduire le réel à des conditions de laboratoire, en considérant que le laboratoire en question consiste en l'ensemble que forme la pensée des cryptographes ainsi que leur ordinateur, leurs notes, leurs manuels et leurs articles scientifiques.

61 La citation d'origine est la suivante : « Le travail des chercheurs consiste à mettre en place des expériences pour faire « écrire » les entités qu'ils étudient, puis à mettre en forme ces inscriptions, et ensuite à les combiner, les comparer et les interpréter. Au terme de ces traductions successives, les chercheurs produisent des énoncés décrivant ce que sont capables de faire les entités sur lesquelles sont menées les expériences. »

Le premier niveau de traduction consiste à passer d'une situation sociale donnée à un schéma simplifié mettant en scène Alice et Bob ; puis, de ce récit, à un véritable protocole cryptographique. Les élucubrations autour d'Alice et Bob sont en effet un moyen simple et relativement instinctif – bien que sérieux et efficace – de poser des hypothèses de travail et problématiser une situation. Ce processus repose sur la théorie des jeux [*game theory*] (Katz 2008; Kol et Naor 2008). La théorie des jeux est une branche de la science informatique qui étudie les comportements stratégiques dans des situations d'interactions. Elle examine les décisions possibles et actions envisageables pour en déduire des conséquences. Il s'agit ainsi de réduire une situation d'interaction ouverte à des paramètres objectivables.

On dira, par exemple, qu'Alice et Bob veulent communiquer de manière confidentielle, en s'assurant de l'authenticité des messages sans avoir à échanger au préalable un secret commun ; par ailleurs, on supposera qu'Ève peut intercepter tous les messages qu'Alice et Bob s'envoient, mais ne peut pas les modifier. Ces prémisses sont assez simples et ne présentent que peu de défis. Elles ne disent par exemple rien à propos du stockage de ces messages et de la manière dont Alice et Bob veulent se les envoyer (par email, par chat IRC, par une application de messagerie spécialisée)⁶². Alice, Bob et Ève sont donc mises dans des situations d'interactions spécifiques, et le rôle de concepteurices du protocole est de résoudre les différents obstacles. Les illustrations 10 et 11 qui suivent montrent deux exemples de tels schémas à différents niveaux d'élaboration.

62 Cette étape est pourtant cruciale car elle pose la question de l'architecture réseau que l'on veut utiliser. Dans un ouvrage récent, Ksenia Ermoshina et Francesca Musiani (2022) ont montré comment les choix de ces architectures étaient hautement stratégiques et politiques, et avaient un impact important sur le fonctionnement des infrastructures en question.

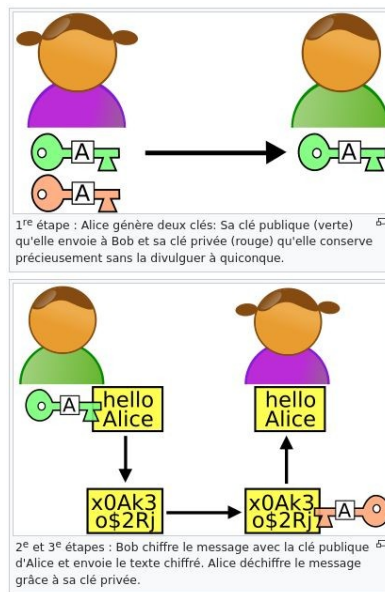


Illustration 10 : Schématisation du fonctionnement de la cryptographie à clé publique sur le site de Wikipédia. Source : https://fr.wikipedia.org/wiki/Cryptographie_asymétrique/.

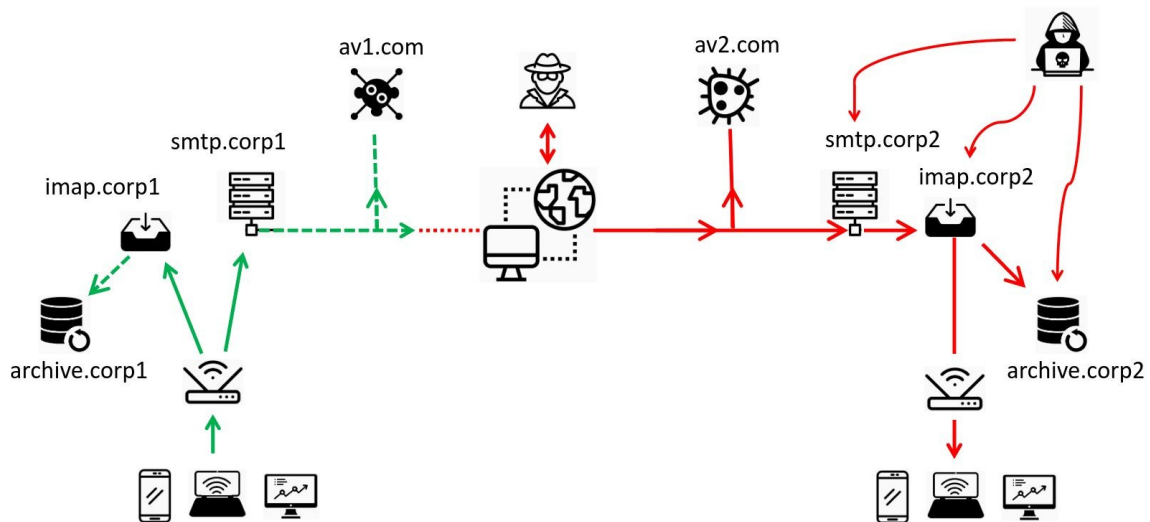


Illustration 11 : Schématisation du cheminement d'un email de l'ordinateur d'Alice (à gauche) à celui de Bob (à droite). Cette illustration est tirée de la présentation des vulnérabilités d'EFAIL de Sebastian Schinzel au Chaos Communication Congress de 2018. Les flèches vertes représentent ce qu'Alice peut plus ou moins maîtriser, les flèches rouges représentent ce sur quoi elle n'a aucun contrôle. Dans cette illustration, Alice et Bob ne sont pas directement mentionnés, mais ce schéma permet de compléter l'illustration 9 en montrant un niveau de détail différent. Source : https://media.ccc.de/v/35c3-9463-attacking_end-to-end_email_encryption.

Ces deux exemples, très différents l'un de l'autre, permettent de se rendre compte que les histoires d'Alice et Bob sont aussi des outils pédagogiques qui schématisent les opérations complexes dans des « boîtes noires ». Ces boîtes noires sont très nom-

breuses : « clé privée », « clé publique », « opérations de (dé)chiffrement », « signature » et « vérification de la signature », mais aussi « envoi de l'email sur le serveur SMTP », « transfert de l'email sur les différents serveurs jusqu'à destination », « stockage de l'email sur le serveur », etc. Ces boîtes noires permettent de formaliser les différentes étapes de la communication sans pour autant entrer dans des considérations techniques qui feraient perdre la vue d'ensemble. Elles rendent aussi plus facile l'identification des faiblesses de l'infrastructure et ce qui serait nécessaire pour protéger certaines données. Par exemple, dans l'illustration 11 ci-dessus, toutes les étapes où quelque chose pourrait mal se passer lors de l'envoi d'un email sont identifiées.

En résumé, les histoires d'Alice et Bob sont importantes pour comprendre ce que permet de faire un protocole de chiffrement. Elles nous informent des mécanismes de ce protocole et du type de situations pour lesquelles il a été pensé. Elles nous renseignent ainsi sur les ambitions de ce protocole, notamment en répondant aux questions concernant le type de conversation et de données que l'on cherche à protéger, ainsi que le type de menaces contre lesquelles on cherche à se protéger. Et tout cela sans présenter la moindre fonction mathématique ni la moindre ligne de code, car toutes les opérations sont représentées par des boîtes noires. Ainsi, pour les cryptographes, le processus de traduction implique de s'appuyer sur des situations sociales modélisées, pour prendre des décisions qui vont du choix de l'architecture du réseau à la conception de primitives cryptographiques.

À mesure qu'un protocole est développé, les différentes boîtes noires s'ouvrent. On doit par exemple spécifier comment sont formées les paires de clé en définissant l'algorithme utilisé, le format de ces clés et les autres informations qu'elles contiennent. Au final, un protocole prend souvent la forme d'un long document écrit, généralement en anglais, à destination des développeuses. Les spécifications du protocole OpenPGP recouvrent d'une police monospace un peu moins d'une centaine de pages A4 qui peuvent être consultées sur le site de l'IETF⁶³. Ce document décrit toutes les étapes nécessaires pour qu'Alice et Bob puissent s'envoyer des

63 <https://www.rfc-editor.org/rfc/rfc4880>.

messages en espérant que Ève ne trouve aucune faille dans le système. C'est une première traduction qui passe des scénarios d'interactions à des instructions techniques. L'étape suivante correspond à implémenter ces instructions dans un programme informatique qu'Alice et Bob pourront utiliser.

3.1.2. Deuxième niveau de traduction : du protocole au code

La conception d'un protocole n'est en effet que la première étape car il faut ensuite implémenter ce protocole dans un programme informatique. Ce passage peut se comprendre comme un passage de la théorie à un dispositif pratique fonctionnel. Le protocole énonce les opérations à exécuter en décrivant toutes les variables possibles et le code traduit cela dans des instructions qu'une machine est capable d'interpréter et d'exécuter. Autrement dit, le code est une interprétation des instructions du protocole dans un langage informatique où chaque opération est minutieusement décortiquée en opérations très simples. C'est pour cela qu'il peut y avoir plusieurs implémentations d'un même protocole, et que ces implémentations ont toutes différents avantages et inconvénients : certaines sont plus rapides, plus faciles à utiliser ou plus sujettes à planter. Ce passage du protocole au code n'a rien d'évident et il serait erroné de croire qu'il « suffit » de suivre les instructions du protocole : très proche du sens commun de traduction, cette étape consiste à interpréter les énoncés du protocole et notamment toutes les boîtes noires, ainsi qu'à tâcher de les reproduire au mieux avec les outils que fournit le langage informatique choisi. De nombreuses discussions existent quant à diverses spécifications du protocole et, pour bien comprendre les enjeux de gouvernance, il est important de se rappeler que chaque implémentation a tendance à interpréter les normes techniques selon les besoins spécifiques, qui varient bien entendu en fonction des usages de la technologie.

L'illustration 12 ci-dessous donne un exemple de code. Il s'agit du code écrit en python d'un petit programme très simple de cryptographie qui implémente le chiffre de Vernam⁶⁴. La lecture de ce code peut paraître assez déconcertante pour une personne

⁶⁴ Peut-être plus connu sous son nom anglais, *one-time pad* : https://fr.wikipedia.org/wiki/Masque_jetable.

non familière avec les langages informatiques et pourtant, je pense qu'il est aisé de se convaincre qu'une personne entraînée peut lire ce code comme d'autres lisent des PV d'observations ethnographiques, c'est-à-dire comme un outil de travail.

```

24 def get_plaintext():
25     plaintext = input('Please type your message : ')
26     return plaintext.lower()
27
28 def load_file(filename):
29     with open(filename, "r") as f:
30         contents = f.read()
31     return contents
32
33 def save_file(filename, data):
34     with open(filename, 'w') as f:
35         f.write(data)
36
37 def encrypt(plaintext, sheet):
38     ciphertext = ''
39     for position, character in enumerate(plaintext):
40         if character not in ALPHABET:
41             ciphertext += character
42         else:
43             encrypted = (ALPHABET.index(character) + int(sheet[position])) % 26
44             ciphertext += ALPHABET[encrypted]
45     return ciphertext

```

Illustration 12 : Extrait d'un petit programme écrit en python qui permet de chiffrer et déchiffrer un message à l'aide d'un masque jetable [One-Time Pad]. Ce petit programme était un exercice dans un programme d'apprentissage des bases du langage Python. Les quatre fonctions ci-dessus reproduites dans l'illustration permettent d'entrer le message en clair et d'enregistrer un fichier contenant le cryptogramme. Source: <https://github.com/RabihND/SecretMsg/blob/main/CRA-SYS.py>.

L'illustration 12 ne comporte qu'une partie des fonctions nécessaires au fonctionnement de ce petit programme. Mais ces quatre fonctions sont suffisantes pour se rendre compte que chaque petite étape doit être décrite avec minutie au vu du caractère borné et rigide des ordinateurs. Rien ne peut être laissé dans le flou. Par exemple, la fonction « `get_plaintext()` » à la ligne 24 a pour but de saisir le texte clair, et celui-ci est sauvé dans la mémoire sans majuscule. Un autre exemple concerne les instructions à suivre si un caractère inconnu est inséré dans le texte clair (lignes 40 et 41). Dans sa simplicité, cet extrait permet de se rendre compte que même si le cheminement doit être logique et explicite, le codage ne peut être réduit à une activité mécanique. Au contraire, coder nécessite souvent de la créativité car il existe souvent plusieurs possibilités d'arriver au même résultat. Par exemple, Paula Bialski (2019) a

décrit comment chaque codeuse a son propre style de codage qui peut être reconnu par ses collègues. Par ailleurs, l'esthétique du résultat final est souvent recherchée et valorisée dans certaines communautés (Coleman et Golub 2008).

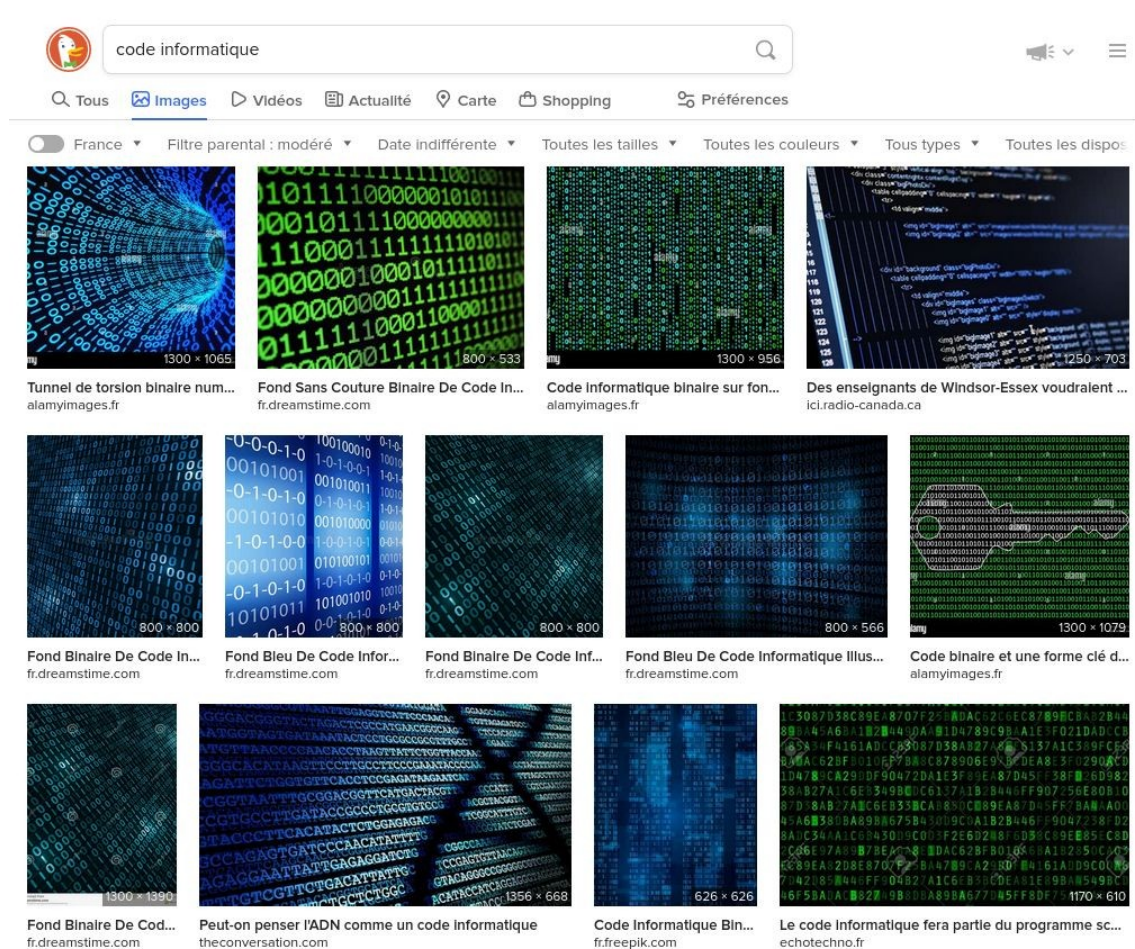


Illustration 13 : capture d'écran d'une recherche d'image sur duckduckgo.com avec les mots clés « code informatique ». Une fois de plus, les images des moteurs de recherche sont spectaculaires et décevantes. Les codes des programmeur-euse-s ne ressemblent pas au code de la Matrix. Source: copie d'écran sur duckduckgo.com effectuée le 12 mars 2023.

Finalement, il me semble aussi important d'affirmer que les codes ne correspondent pas aux images spectaculaires que l'on peut trouver en tapant « code informatique » sur un moteur de recherche (cf. illustration 13 ci-dessus). À part, éventuellement, la quatrième image, aucune de ces illustrations ne correspond à ce qu'écrivent les codeuses. En revanche, nombre d'entre elles montrent un code binaire, des 0 et des 1. Malgré l'exubérance des représentations en question, ces *bits* sont bel et bien une réalité car le code source écrit en langage informatique doit im-

pérativement être compilé en code binaire pour être interprété par le processeur de l'ordinateur. Cette nouvelle traduction est effectuée par un compilateur. Un compilateur est un logiciel qui traduit le code source écrit dans un langage de programmation en un code binaire, compréhensible par la machine qui doit l'exécuter. Ce deuxième niveau de traduction est donc composé de deux traductions simultanées : la première qui consiste à passer du protocole au code source et une deuxième pour passer du code source au code binaire. Contrairement à la deuxième, la première traduction n'est ni automatique, ni effectuée par une machine⁶⁵. Par ailleurs, cette première traduction du protocole au code est beaucoup plus sujette à débat et c'est pourquoi je vais particulièrement m'arrêter sur cette traduction-ci.

En somme, ce deuxième niveau de traduction consiste en l'explicitation de chaque étape décrite dans le protocole. Ainsi, toutes les boîtes noires, y compris les plus complexes comme l'implémentation des primitives cryptographiques, doivent être décortiquées en minuscules étapes, lignes après lignes, fonctions après fonctions. À la fin de cette traduction et si tout se passe bien, les codeuses ont un programme informatique qu'une machine peut interpréter pour exécuter ce qui est prévu dans le protocole.

À ce stade, nous avons donc un projet conçu et développé autour de ce que Madeleine Akrich (2006a) appelle le « script » de cette technologie. Ce script correspond au fonctionnement prévu du programme. Toutefois, ce script n'est pas encore « réalisé » :

Nous proposons d'appeler l'aboutissement de ce travail « script », ou « scénario » : cette mise en forme technique, par le concepteur, de son point de vue sur les relations nécessaires entre son objet et les acteurs qui doivent s'en saisir se veut une prédétermination des mises en scène que les utilisateurs sont appelés à imaginer à partir du dispositif technique et des prescriptions (notices, contrats, conseils...) qui l'accompagnent. Mais tant qu'il ne se présente pas d'acteurs pour incarner les rôles prévus par le concepteur (ou en inventer d'autres), son projet reste à l'état de chimère : seule la confrontation réalise ou irréalise l'objet technique. (Akrich 2006a, 163)

65 Bien que l'intelligence artificielle puisse coder des programmes simples, et qu'il soit dès lors possible d'imaginer un avenir où l'écriture du code sera de plus en plus assistée par des algorithmes d'intelligence artificielle, autrement dit d'autres codes.

En suivant ce cheminement, nous parvenons donc à un troisième niveau de traduction qui est celui de la sortie des conditions de laboratoires pour arriver dans le monde. Cette étape est loin d'être évidente, comme nous nous apprêtons à le voir dans la prochaine section.

3.1.3. Troisième niveau de traduction : du code aux usages

Jusqu'à maintenant, nous avons déjà parcouru un chemin qui paraît long ; nous sommes pourtant encore loin d'arriver à destination. À partir de scénarios simplifiés et idéalisés entre Alice et Bob qui désirent comploter à l'écart des oreilles attentives d'Ève, nous avons observé comment des ingénieur·e·s développaient des protocoles (premier niveau de traduction). Ces protocoles comprennent la description des algorithmes et des formats à utiliser, ainsi que l'ordre des opérations à effectuer, et ils sont destinés principalement aux codeuses qui s'en saisissent et les traduisent en code informatique (deuxième niveau de traduction). Il s'agit maintenant de comprendre comment Alice, Bob et le maladroit Johnny s'approprient en pratique ces programmes, autrement dit ; il nous reste à étudier une dernière étape, celle du passage des scripts des programmes à leur usage. Dans le cadre de mon étude, je ne me suis que peu intéressé à cette ultime étape et n'ai pas enquêté auprès d'utilisatrices d'OpenPGP car mes questions de recherche portaient sur d'autres éléments. Cependant, de nombreuses études ont été réalisées à ce sujet et il me semble important de les mentionner ici afin d'avoir une vue d'ensemble des trois niveaux de traduction.

Le champ des *usable security studies* est une sous-discipline des sciences informatiques. Il vise à comprendre comment les utilisatrices interagissent avec les systèmes de sécurité. Elles postulent que les utilisatrices sont souvent le maillon faible de la sécurité informatique et cherchent à améliorer la sécurité des systèmes en question en rendant les programmes faciles à utiliser correctement. Ainsi, les *usable security studies* cherchent non seulement à comprendre le comportement des utilisatrices, mais également à concevoir des interfaces plus intuitives qui aident à concevoir des programmes plus sûrs car plus faciles à utiliser. Parmi les études en

lien avec OpenPGP, j'ai dénombré de très nombreuses critiques qui s'en prennent non seulement à la complexité interne du protocole, mais également à sa difficulté d'usage, dont la série d'études suivantes, débutée en 1999, et dans laquelle Johnny fait sa star, crânement :

- *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0* (Whitten et Tygar 1999)
- *Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software* (Sheng et al. 2006)
- *Why Johnny Still, Still Can't Encrypt: Evaluating the Usability of a Modern PGP Client* (Ruoti et al. 2016)
- *Why Johnny STILL Can't Encrypt* (Gieter 2017)

Cette série peut être complétée par de nombreux autres articles qui critiquent encore les capacités de Johnny à utiliser correctement OpenPGP (ou parfois d'autres protocoles apparentés), ou l'incapacité des programmes de s'adapter aux modestes compétences de Johnny :

- *Johnny 2: a user test of key continuity management with S/MIME and Outlook Express* (S. L. Garfinkel et Miller 2005; N.B. Cet article est dédié à S/MIME et non OpenPGP)
- *Leading Johnny to Water: Designing for Usability and Trust* (Atwater et al. 2015)
- *Maybe Poor Johnny Really Cannot Encrypt: The Case for a Complexity Theory for Usable Security* (Benenson et al. 2015)
- *"Johnny, you are fired!"-Spoofing OpenPGP and S/MIME Signatures in Emails* (Müller, Brinkmann, Poddebniak, Böck, et al. 2019)
- *Re: What's Up Johnny?* (Müller, Brinkmann, Poddebniak, Schinzel, et al. 2019)
- *Work in Progress: Can Johnny Encrypt E-Mails on Smartphones?* (Schiller et Adamsky 2022)

Finalement, il faut tout de même mentionner que si Johnny ne parvient apparemment pas à utiliser OpenPGP, Sa Majesté le Roi George III et d'autres motivé·e·s y arrivent :

- *Why King George III Can Encrypt* (Tong et al. 2014)⁶⁶
- *The Motivated Can Encrypt (Even with PGP)* (Borradaile et al. 2021)

Ces études ne sont qu'un aperçu des différentes analyses sur l'usage des implémentations d'OpenPGP. Il semble toutefois s'en dégager un large consensus au sujet de la difficulté d'usage des programmes de chiffrement d'emails. Ces études viennent s'ajouter aux nombreuses critiques moins formalisées d'OpenPGP, dont les plus acerbes appellent tout bonnement à l'abandon du protocole (Valsorda 2016; 2021; Latacora 2019; Wong 2020). Je reviendrai dans les chapitres qui suivent sur ces critiques. Pour le moment, il me semble suffisant de comprendre que la sécurité informatique ne se résume pas seulement en l'élaboration de programmes de sécurité mais que leur adoption et usage importent également. Ils importent d'ailleurs aussi au niveau de la production de ces technologies de sécurité, puisque les développeuses que j'ai rencontrées discutent régulièrement de la manière de faciliter l'usage de leur technologie. Par exemple, un des objectifs de Sequoia est de proposer une implémentation d'OpenPGP difficile à mal utiliser [*hard to misuse*]. Toutefois – et c'est là un point crucial –, malgré ces critiques, de nombreuses personnes utilisent OpenPGP et comptent sur la fiabilité des programmes qui implémentent la norme. Ainsi, même si de nombreuses difficultés viennent compliquer l'usage du protocole, le script n'est pas totalement « irréalisé ».

Pour conclure cette première section au sujet des multiples niveaux de traductions successives, j'aimerais revenir sur la triade CIA et souligner une fois de plus le consensus qui existe dans le champ à propos des trois valeurs que sont la confidentialité, l'intégrité et l'accessibilité. Il ne fait pas de doute que toutes les personnes que j'ai interviewées dans le cadre de cette recherche considéreraient qu'elles représentent l'essence de la sécurité informatique. Il s'agit ainsi d'une définition émique de la sécurité qui est utilisée par de nombreuses personnes du terrain. Cette définition leur permet de penser la sécurité en des termes qui font consensus au sein de leur communauté de pratiques, et ainsi de s'entendre sur ce qui est attendu d'un système

66 Cet article a testé une nouvelle métaphore pour parler de la sécurité informatique. Plutôt qu'Alice et Bob, c'est du roi George III et de rébellion dans l'empire britannique qu'il s'agit.

cryptographique « sûr ». Cependant, cette définition émique ne peut me satisfaire pour mon analyse de la production sociale de la sécurité informatique car, comme le disent les personnes elles-mêmes, la sécurité dépend aussi toujours du contexte et des menaces contre lesquelles on veut se prémunir (Kazansky 2021).

Ce que je montre dans la suite de ce chapitre, c'est que la triade CIA se révèle beaucoup plus utile pour signaler une brèche que pour concevoir des systèmes sécurisés. En effet, les différentes étapes de traductions successives dont je viens de parler sont un signe évident que les choses ne sont pas aussi simples que poser trois propriétés au-dessus d'un système informatique pour le décréter sécurisé. Ces trois valeurs de la triade CIA peuvent être considérées comme les fondations de ce qu'on appelle « la sécurité informatique », mais elles ne font que de poser les bases d'un édifice qu'il reste à construire. Il faut maintenant s'intéresser aux briques, au mortier et à toutes les différentes couches qui permettent de construire un édifice solide. Dans la section suivante, je propose ainsi de réfléchir à deux autres façons d'entrevoir la sécurité.

3.2. Des processus d'(in)sécurisation et de soin au cœur de la machine

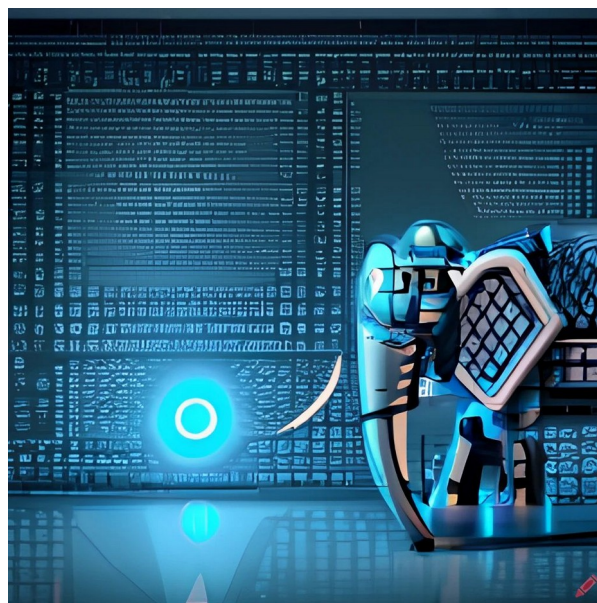


Illustration 14: Un éléphant dans la cyber-pièce. Image créée par un générateur d'images. Source : <https://www.craiyon.com/>.

Pour introduire cette deuxième section sur les approches plus socio-anthropologiques de la sécurité, j'aimerais pointer du doigt l'éléphant dans la pièce⁶⁷ et souligner un paradoxe déroutant : on ne sait pas exactement à quoi ressemble un bon système de chiffrement. Malgré la quantité d'études et de manuels dans le champ de la cryptographie ainsi que les multiples erreurs qui ont été étudiées depuis des siècles, voire des millénaires, la plus mathématique des disciplines informatiques n'a pas de théorie mathématique à proprement parler :

Un système de chiffrement est considéré comme sécurisé quand un-e adversaire ne peut pas le casser dans des circonstances raisonnables, dans un laps de temps raisonnable, à un prix raisonnable. Le terme "raisonnable" est forcément vague. Bien qu'étant le problème le plus important en cryptographie, l'évaluation des systèmes de chiffrement est la moins comprise. Une théorie mathématique adéquate de la cryptographie a déjoué les cryptographes pendant des siècles. (Diffie et Landau 2010, 26, ma traduction)

Diffie et Landau font ici référence à la sécurité computationnelle [*computational security*] qui consiste en un calcul de probabilité afin d'estimer le niveau de sécurité des algorithmes en fonction de deux valeurs : le nombre d'opérations à réaliser et la probabilité de succès (Aumasson 2018, 40ss). Jean-Philippe Aumasson (2018) précise que la sécurité computationnelle est agnostique en ce sens qu'elle considère non pas le temps nécessaire pour trouver la clé, mais le nombre d'opérations. Ainsi, ce calcul reste stable dans le temps, bien que son résultat puisse varier : ce qui est considéré aujourd'hui comme « raisonnable » ne le sera peut-être plus demain. Il est important de garder cela en tête car, comme je le montre dans le chapitre 9 (« Équiper OpenPGP face aux ordinateurs quantiques », p. 326), cela questionne la durée de la sécurité, une des dimensions temporelles clés de la sécurité. C'est donc en gardant cette incision fondamentale sur ce qui est « raisonnable » (et pour qui) que j'invite les lecteurs à prendre du recul par rapport aux définitions purement technique de la sécurité informatique et de s'emparer des visions plus dynamiques et relationnelles que j'abode dans cette deuxième section.

67 La locution « un éléphant dans la pièce » est une expression anglaise qui permet de désigner un sujet important et manifeste, généralement associé à un problème important, que tout le monde peut constater mais que personne n'ose mentionner.

3.2.1. Les processus d'(in)sécurisation

Si on considère les mots employés dans la grande majorité des discours sur la sécurité informatique, on voit apparaître de manière très nette un champ lexical tournant autour de l'attaque et de la guerre (Nissenbaum 2005; Dunn Cavelty 2007; 2014) (Nissenbaum 2005; Dunn Cavelty 2007; 2014; DeNardis 2014) : qui attaque qui, qui se défend de qui ?⁶⁸ On entend souvent parler d'armes (cyber-arsenal, virus, cheval de Troie) et des moyens de défense appropriés (armure, bouclier, pare-feu). Par ailleurs, ce vocabulaire guerrier met l'accent sur le spectaculaire et l'événementiel : on parle de désastres numériques (L. Hansen et Nissenbaum 2009), de *hacks* spectaculaires et audacieux (Coleman 2015; Tréguer 2015) ou de projets politiques « tueurs de secrets » (Greenberg 2012). Toutefois, et pour plusieurs raisons que je détaille dans la suite de ce chapitre, cette métaphore guerrière présente de nombreuses limites.

À la fin de la première décennie post-guerre froide, Barry Buzan, Ole Wæver et Jaap De Wilde, tous les trois chercheurs en relations internationales, ont publié un ouvrage majeur sur la sécurité : « *Security: A New Framework for Analysis* » (1998). Dans ce texte, ils proposent d'ouvrir l'étude de la sécurité à d'autres secteurs que les questions militaires en développant la théorie de la sécurisation. En se basant sur la notion d'acte de langage (Austin 1970), ils ont suggéré que la sécurité n'était pas réductible à un état objectif mais consistait en un processus. Ils ont proposé une vision constructiviste de la sécurité où la réception des discours sécuritaires primait sur l'état « objectif » de la situation. Il s'agit donc d'étudier les processus par lesquels quelque chose est transformé en une menace existentielle pour un groupe de personnes. La théorie de la sécurisation a par la suite été développée et discutée (Macleod 2004; Goldstein 2010; Balzacq 2011; Maguire, Fróis, et Zurawski 2014; Stritzel 2016). De ces études, je retiens notamment la définition de Thierry Balzacq qui considère :

[...] la sécurisation comme un agencement articulé de pratiques où des artefacts heuristiques (métaphores, instruments de politiques, répertoires d'images, analogies, stéréotypes, émotions, etc.) sont contextuellement mobilisés par une

68 On peut substituer « qui » par « quoi » dans ces questions.

acteurice de la sécurisation [*securitizing actor*] qui travaille à inciter un public à construire un réseau cohérent d'implications (sentiments, sensations, pensées et institutions) à propos de la vulnérabilité critique d'un objet référent, qui concorde avec les raisons des choix et actions de l'acteurice de la sécurisation, en revêtant l'objet référent d'une telle aura de menaces sans précédent qu'une politique sur mesure doit être immédiatement entreprise pour bloquer son développement (Balzacq 2011: 3, ma traduction)

Cette définition a l'avantage de souligner l'intersection entre quatre éléments clés :

1. La présence d'au moins deux types d'acteurices différentes : une acteurice de la sécurisation et un public spécifique. La définition de Balzacq ne suppose aucune légitimité institutionnelle a priori des acteurices de la sécurisation. C'est leur travail auprès d'une certaine audience qui produit une situation de sécurisation.
2. La performativité des moyens utilisés pour sécuriser : que ce soit par le biais de mots ou d'objets, l'acteurice de la sécurisation agit dans le but de transformer une situation donnée. Il n'y a pas, chez Balzacq, de menaces fondamentales intrinsèques : le caractère critique et imminent d'une situation sécuritaire découle de la trame narrative dans laquelle deux futurs sont considérés : un futur sombre et dangereux et un futur meilleur et sûr. Cela permet de convaincre du caractère insoutenable d'une menace donnée, contre laquelle il « faut » agir.
3. Dans la construction de cette trame narrative, l'objet référent mentionné par Balzacq joue un rôle crucial : il cristallise la menace et permet de donner corps à la vulnérabilité de la situation – même si ce corps est parfois symbolique et non matériel.
4. Les contraintes temporelles et contextuelles d'un tel dispositif : c'est de ce contexte qu'émerge l'imminence supposée de la menace et la nécessité d'une réaction immédiate.

Cette perspective peut facilement s'appliquer à la sécurité informatique. Imaginons la situation suivante : un activiste d'un mouvement anti-nucléaire et passionné de cryptographie entend parler d'un projet de loi qu'un sénateur états-unien veut

faire passer pour obliger les fournisseurs de messageries électroniques à livrer sur demande au gouvernement le texte clair des messages de leur clientèle. Cet activiste décrirait alors le scénario d'un État orwellien sur le point d'instaurer une surveillance systématique de ses citoyen·nes, contre laquelle il propose une porte de sortie : un programme garantissant une « assez bonne confidentialité »⁶⁹. Ce programme se présente alors comme une solution face à une menace considérée comme imminente et insoutenable⁷⁰.

Ce scénario décrit un processus de sécurisation qui va de l'identification d'une menace à la proposition d'une solution. Toutefois, l'inverse existe aussi : parfois, des personnes décrivent des vulnérabilités dans un monde où l'on se croyait en sécurité. Le cas typique en lien avec la sécurité informatique est celui d'une équipe de recherche qui présente une vulnérabilité qu'elle a trouvée dans le code d'un logiciel de chiffrement⁷¹. De telles recherches démontrent qu'il existe un moyen d'avoir accès à des données privées ; elles viennent insécuriser une situation qu'on croyait stable. Ces deux scénarios montrent bien comment la sécurité peut être perçue comme la construction d'une menace :

« En étant le résultat d'un processus de sécurisation, plutôt qu'une réalité objective, les problèmes de sécurité doivent être alors considérés comme toujours contingents, contestables et contestés, ainsi que politiquement, socialement et culturellement situés. La notion de sécurité est alors entendue davantage comme un idiome ou un outil politique à manier avec une extrême précaution que comme un concept scientifique permettant de saisir et d'analyser des réalités sociopolitiques variées. » (Bozzini 2017, 3)

69 La traduction française de « *pretty good privacy* » claque nettement moins, il faut l'avouer.

70 Le sénateur en question était un certain Joseph Biden, alors président de la commission judiciaire du Sénat ; l'activiste, un certain Philip Zimmermann. À propos de ce projet de loi (S 266) qui n'a finalement pas été retenu, Zimmermann a écrit : « C'est ce projet de loi qui m'a amené à publier gratuitement PGP sous forme électronique cette année-là, peu avant que la mesure ne soit rejetée après une protestation vigoureuse de la part des défenseurs des libertés civiles et des groupes industriels » (Zimmermann 1999, ma traduction). C'est ainsi que l'actuel président des États-Unis d'Amérique a joué un rôle crucial dans les débuts de PGP. Pour plus d'informations, je renvoie au chapitre « crypto anarchy » que Steven Levy a consacré aux débuts de PGP (Levy 2001, 187ss).

71 L'exemple de la révélation des vulnérabilités d'EFAIL peut illustrer ce propos (Besençon et Bozzini 2020; Besençon 2022).

De ce point de vue, sécurité et insécurité sont les deux faces d'une même médaille. Comme je l'ai montré dans un article sur l'écriture du code informatique (Besençon 2022), ces processus de sécurisation et d'insécurisation se répondent mutuellement et des révélations de failles peuvent permettre de sécuriser un dispositif alors que des correctifs se révèlent parfois problématiques en déstabilisant le code. J'ai proposé la notion de processus d'(in)sécurisation « pour qualifier toute pratique ou discours qui (in)sécurise différents aspects du code (son contenu, son design, les valeurs qu'il incarne, les primitives cryptographiques qu'il utilise) à travers de multiples moyens (recherche scientifique, rapports de bogue, commits, critiques) par de multiples acteurs (codeur·se·s, chercheur·se·s, cryptographes, journalistes, hackers) » (Besençon 2022, paragr. 51). Dans cet article, je parlais spécifiquement de codes, mais les processus d'(in)sécurisation peuvent concerner tout autre objet informatique, et notamment les normes, les primitives cryptographiques et les supports matériels.

3.2.2. La sécurité comme objet de sollicitude

J'aimerais maintenant me tourner vers une autre source d'inspiration : les études sur la maintenance et le soin porté aux technologies numériques. Alors que les théories de la sécurisation s'intéressaient particulièrement aux processus de construction d'objets menaçants (vulnérabilités, *exploits*, virus, *malwares*, etc), les études sur le soin des choses font prendre conscience de la fragilité du monde qui nous entoure. Par exemple, j'ai découvert que les programmes que j'utilisais étaient pleins de bogues et de failles et que des centaines de personnes travaillaient continuellement pour améliorer ces codes. Le numérique est fragile, et je ne parle même pas ici de la fragilité matérielle dont Jérôme Denis (2009), Nicolas Nova et Anaïs Bloch (2020) et Matthias Korn et Susann Wagenknecht (2017) ont décrit plusieurs aspects ; la fragilité dont je parle est informatique et concerne la manière dont les codes et les autres objets numériques fonctionnent.

Ma recherche parmi les différentes communautés d'OpenPGP permet de problématiser un aspect crucial de la sécurité que laissaient de côté les théories de la sécu-

risation : en m’inspirant des études sur le soin porté aux infrastructures techniques et aux choses du numérique (Star 1999; Star et Ruhleder 2010; Puig de la Bellacasa 2010; Mol, Moser, et Pols 2010; Kocksch et al. 2018), mes observations me permettent d’affirmer que la sécurité n’est pas pensée uniquement en termes de processus discursifs mais également de charge mentale et de pratiques « qui exigent un dévouement continu » (Kocksch et al. 2018, 92:4, ma traduction). Les processus de sécurisation attirent l’attention et une concentration particulière sur certains problèmes ; mais les études sur le soin et les éthiques du *care* mettent en valeur une différence fondamentale avec ce type d’attention : il s’agit d’une attitude éthique tournée vers la vigilance, le souci du détail et la ténacité (Brugère 2017). Par attitude éthique, j’entends d’une part souligner le fait qu’il s’agit de pratiques observables et, d’autre part, que ces pratiques sont orientées par des valeurs morales qui aiguillent sans déterminer le choix des personnes concernées. Ainsi, alors que les théories de la sécurisation permettent de comprendre la construction intellectuelle de la sécurité, une attention sur les pratiques de soin permettent de rendre compte des pratiques concrètes des actrices qui produisent des technologies de sécurité informatique. Il s’agit notamment de souligner le dévouement de certaines personnes qui, sans qu’un événement particulier insécurise la technologie, se soucient de ladite technologie.

Ce dévouement présuppose non seulement de la disponibilité et de l’écoute de ce qui est en train de se passer, mais également de la réactivité. Disponibilité, écoute, réactivité ne sont pas des valeurs abstraites mais s’observent dans des comportements concrets. Par exemple, lors de l’entretien que j’ai mené avec elles, plusieurs personnes gardaient en continu un œil sur Slack⁷² pour être certaines de ne pas manquer une information importante de leur équipe de travail. De même, lors de mon terrain en présence dans l’équipe de Sequoia, j’ai pu voir combien Twitter et d’autres flux RSS étaient constamment présents dans le quotidien de ces personnes. Par ailleurs, j’ai pu observer semaines après semaines combien les tâches qu’effectuaient les

72 Slack est une plateforme de *chat* utilisée en entreprise, et notamment dans le domaine de la technologie numérique : <https://slack.com/intl/fr-ch/>.

membres de l'équipe de Sequoia variaient constamment en fonction de ce qui arrivait : des projets étaient suspendus ou du moins ralentis le temps de corriger un bogue (quelques heures parfois, quelques jours voire semaines d'autres fois).

Cependant, une approche des technologies centrée sur le soin va beaucoup plus loin que le constat de la disponibilité et de la réactivité des personnes impliquées et il importe de s'intéresser aux différents types de tâches réalisées. Au risque de caricaturer le trait pour le moment, je dirais que considérer la sécurité en tant qu'« objet de sollicitude » dans une perspective liée au *care* permet de rendre compte de trois types de tâches complémentaires qui sont orientées vers trois horizons temporels distincts : la réparation, la maintenance et la préparation. Je propose de considérer l'ensemble de ces tâches comme des pratiques de soin. Ces catégories de tâches seront abordées plus en détail dans les différentes parties de la thèse et je n'en brosse un rapide schéma ici que pour avoir une vue d'ensemble approximative et provisoire.

Il y a d'une part la réparation, qui est orientée vers le passé puisqu'il s'agit de revenir à un état antérieur à une panne ou un problème identifié. Par exemple, en 2019, une attaque de déni de service⁷³ sur certaines clés publiques de personnes particulièrement actives de la communauté OpenPGP a causé un nombre important de problèmes. L'attaque a consisté à ajouter un nombre très important de signatures à ces clés sur les serveurs de clés en ligne, et ses multiples signatures faisaient ensuite planter les implémentations⁷⁴. Sans entrer dans les détails, il a fallu réparer cela en trouvant un moyen de « dépolluer » ces clés. Une réparation demande du temps, de la connaissance et parfois quelques bidouillages, mais elle peut s'achever. Quelque chose est passé de l'état de neuf à celui de cassé, puis de cassé à celui de réparé et le monde est stabilisé une fois de plus.

Cela s'oppose aux pratiques de maintenance qui sont profondément ancrées dans le présent. Ces pratiques se caractérisent par une « réalisation continue » [*ongoing achievement*] (Kocksch et al. 2018), un « bricolage permanent » [*persistent tinkering*]

73 Une attaque de déni de service [*denial of service* (DOS)] est une tentative délibérée de rendre un système, un service, ou un réseau informatique indisponible aux utilisateurices en submergeant la cible avec une grande quantité de demandes ou de trafic malveillant.

74 <https://gist.github.com/rjhansen/67ab921ffb4084c865b3618d6955275f>.

(Mol, Moser, et Pols 2010) et un « éternel recommencement » (Denis et Pontille 2022)⁷⁵. La maintenance est ce travail ancré dans le présent qui consiste à infléchir la course du temps pour la ralentir, voire parfois l'arrêter. Contrairement à la maintenance des choses matérielles, la maintenance des choses informatiques (un code, un protocole) ne vise pas forcément la conservation d'une esthétique ou d'une quelconque authenticité comme cela pourrait être le cas d'un bâtiment historique ou d'une œuvre d'art. La maintenance informatique vise avant tout la conservation de fonctionnalités : faire en sorte que tel code et tel protocole continuent de fonctionner. Ainsi, leur maintenance implique leur transformation continue et contrairement à la réparation qui suppose un retour à la normale après une panne, la maintenance postule que l'anormalité, la dégradation et les ennuis constituent la norme et qu'il faut en permanence s'adapter aux nouvelles conditions de ce monde sans cesse changeant.

Finalement, ce tableau sur le soin porté à OpenPGP ne serait complet sans mentionner l'incessant travail de projection que j'ai pu observer quotidiennement. Dans ce qui suit, je montre comment le travail de soin d'une technologie de sécurité informatique telle OpenPGP consiste non seulement à corriger les failles du passé et maintenir la fonctionnalité du programme aujourd'hui, mais aussi à réaliser un incessant travail de préparation de ce qui va arriver, et notamment des réparations à venir. Cette (p)réparation de l'avenir, que j'ai nommé la recherche de la « résilience prospective » (cf. chapitre 9 « Équiper OpenPGP face aux ordinateurs quantiques », p. 326) suit une triple logique. Il s'agit premièrement de réaliser un travail de mémoire en archivant les discussions et les prises de décision à propos de chaque transformation de la technologie. Cela se passe notamment à travers les archives des listes de diffusion ou des *commits* sur GitLab/GitHub. Deuxièmement, un travail de documentation du code est effectué afin que sa maintenance future soit la plus facile possible à réaliser : en effet, un code mal documenté sera plus difficile à déboguer, et par consé-

75 Les auteurices citées parlent parfois de maintenance, parfois de soin ; toutefois, comme j'utilise la catégorie « soin » pour un ensemble plus large de pratiques qui englobe aussi la réparation et l'innovation, j'ai décidé de regrouper ici ces pratiques en tant que pratique de maintenance.

quent à corriger et patcher. Finalement, un travail d'anticipation des ajouts à venir permet d'avoir suffisamment de marge de manœuvre pour des modifications futures de la technologie. Il s'agit par exemple de volontairement laisser des espaces libres dans le code pour permettre à de nouveaux algorithmes cryptographiques d'être ajoutés quand cela sera nécessaire. Cette triple logique de l'anticipation de l'avenir implique ainsi des activités que l'on réalise aujourd'hui, pour demain.

Une des contributions de cette thèse concerne la manière dont passé, présent et avenir sont présents dans le travail quotidien de soin des choses informatiques. Une attention sur les temps de la sécurité permet de distinguer les différentes modalités de ce soin, et ainsi contribuer à une compréhension plus fine de la manière dont des ingénieur·e·s, codeuses et cryptographes pensent et produisent de la sécurité informatique à travers des pratiques à la fois ancrées dans le présent, contraintes par le passé et tournées vers l'avenir. Ces horizons temporels seront largement analysés, discutés et nuancés dans les trois parties de cette thèse, intitulées respectivement « l'héritage du passé en mouvement », « les rythmes du présent » et « la durée de l'avenir ». Je laisse ainsi provisoirement de côté ces récits pour m'arrêter sur la notion de projet-sécurité qui permet de faire le pont entre la théorie de la sécurisation et les études sur le soin des choses en soulignant une dimension cruciale : l'attention tournée vers le devenir des technologies.

3.3. La sécurité comme projet : une attention tournée vers le devenir

Il ressort de ce qui précède qu'il n'est pas possible de définir la sécurité d'une seule et même manière qui fasse consensus parmi les actrices de ma recherche. Il y a de multiples façons de la concevoir. La triade CIA et les histoires d'Alice et Bob proposent de concevoir la sécurité en termes de propriétés et de garanties qui seraient vérifiables ; l'approche constructiviste de la théorie de la sécurisation suggère qu'il s'agit plutôt de résultats d'actes de langage qui créent un sentiment d'(in)sécurité ; les études sur la maintenance et le soin des infrastructures numériques considèrent quant à elles la sécurité comme un objet de sollicitude dont il s'agit de

s'occuper au quotidien, ce qui implique une attention constante, de l'improvisation et des bricolages.

Bien entendu, ces approches n'épuisent pas toutes les visions émiques et étiques de la sécurité et d'autres pistes pourraient être suivies pour comprendre ce qu'est la sécurité. Je pense notamment aux approches contextuelles qui concernent principalement les utilisatrices des technologies de sécurité. Ksenia Ermoshina et Francesca Musiani (2022) ont notamment montré l'importance d'identifier et comprendre les menaces pour établir un modèle de menace qui sert à orienter les personnes vers différentes solutions. Cette approche a aussi été décrite en détail par Becky Kazansky (2021) qui a analysé comment le travail de sécurité demandait également des pratiques anticipatrices pour faire face à « l'incertitude continuellement produite ». D'ailleurs, lors de ma recherche, j'ai souvent été confronté à des discussions plus ou moins houleuses qui cherchaient à affirmer ou infirmer le caractère (in)sécurisé d'un algorithme de chiffrement, d'un programme ou d'une situation alors que ces situations dépendaient du contexte. Un exemple parlant concerne l'extrait ci-dessous, tiré d'une discussion à propos de la sécurité des clés RSA :

> Est-ce que vous considérez toujours les clés RSA de 4096 bits de long
> comme sécurisées ?

Avant de vous poser la question de savoir si les clés RSA2048 ou RSA4096 sont plus sécurisées, vous devriez d'abord évaluer votre modèle de menace, sécuriser le hardware et le software de votre matériel informatique, et suivre une formation de sécurité opérationnelle.

(Werner Koch, email sur la liste de diffusion du groupe de travail OpenPGP, 22 août 2022)⁷⁶

En d'autres termes, il semble que la sécurité soit aussi multiple que le nombre d'actrices s'en réclamant et si je n'arrivais pas à répondre à la question de savoir ce qu'était la sécurité informatique, c'était peut-être que la question elle-même n'était

76 « Re: [openpgp] Upgrading PGP Keysize from 4096 to 8000 »: <https://mailarchive.ietf.org/arch/msg/openpgp/VgDQqWQIrtu6JSU8ib2ZDfsirJE/>. Les symboles « > » représente un extrait d'un email que Werner Koch citait dans sa réponse.

pas la bonne. Ces approches contextuelles permettent de rendre compte de manière pragmatique de la fluidité des interprétations de ce que peut signifier « sécurité » sans s'attarder sur des propriétés *a priori*. Concrètement, cela implique d'accepter qu'il soit question de sécurité simplement quand un individu ou un groupe se réclamant de la mise en œuvre de la sécurité agissait au nom de la sécurité pour soi-même et pour d'autres personnes⁷⁷. Cette perspective pragmatique et interactionniste incite ainsi à considérer un vaste panel d'activités qui vont bien au-delà d'une vision technique et guerrière de la sécurité axée autour de l'attaque et la défense, de la vulnérabilité et du patch, et ainsi d'ouvrir de nouvelles perspectives d'analyse à l'observateur que je suis.

J'ai notamment constaté que les différentes personnes que je côtoyais ne se contentaient pas de problématiser des enjeux sécuritaires et de chercher à y remédier comme l'ont montré les enquêtes centrées sur les vulnérabilités et leur cycle de vie (Arbaugh, Fithen, et McHugh 2000; Frei, Tellenbach, et Plattner 2008; Vyas 2013). Ils et elles se montraient inquiètes de ce qui pouvait arriver car, pour ces personnes, la sécurité était à tout moment susceptible d'être cassée. Même dans la perspective très positiviste et schématique des histoires d'Alice et Bob, la sécurité ne peut être qu'un état passager car il suffit de changer les hypothèses de travail (par exemple les privilèges d'Ève) et la sécurité était compromise. La sécurité ne peut être que provisoire et l'insécurité est toujours latente. Les technologies numériques sont à la fois fragiles à cause de leur héritage du passé, et vulnérables à cause des menaces que réserve l'avenir.

La posture pragmatique et interactionniste que j'ai adoptée m'a aussi montré d'autres dimensions, comme le soucis d'avoir une communauté solide autour d'une technologie. Cela ressort particulièrement bien du rapport « *Sustain. A one day*

77 Cette vision fait également penser à la définition du développement selon Chauveau (Chauveau 1985, 164) : « Il y a “développement” tout simplement là où il y a des développeurs, là où un des groupes se réclamant de la mise en œuvre du développement organise un dispositif d'intervention sur d'autres groupes sociaux ». Le développement présente en effet une similarité avec la sécurité dans le sens qu'il s'agit d'un ensemble de pratiques et discours hétérogènes et parfois contradictoires.

conversation for open source software sustainers » qui résume une journée de conversation entre une centaine de développeuses (Nickolls 2017). L'une des recommandations clés de ce rapport concerne l'importance de se soucier non seulement des ressources financières, mais aussi en terme de réseau et de communautés. Les infrastructures numériques dépendent en effet en grande partie des mainteneuses de ces technologies alors que l'accent est souvent mis sur les innovations et les nouveautés⁷⁸. Dans ce contexte, avoir une communauté solide, stable et dévouée devient un élément clé de la fabrique de la sécurité. Cette posture centrée sur les actrices permet ainsi de faire remonter des préoccupations centrales et néanmoins absentes des considérations uniquement techniques.

3.3.1. Le projet-sécurité pour penser les temps de la sécurité informatique

La notion de projet-sécurité permet de comprendre ce qui est entrepris au nom de la sécurité, et cela sans devoir nécessairement définir cette sécurité. Cette perspective m'a offert de nombreuses prises pour saisir différentes attitudes envers la sécurité, parmi lesquelles l'inquiétude et l'attention, mais aussi la curiosité et une certaine excitation à l'égard des nouvelles menaces ou des nouveaux défis. D'une manière intéressante, ces attitudes étaient ancrées dans le présent, dans le sens qu'elles étaient dirigées vers l'état actuel des choses. Mais leur motif, ce qui justifiait leur existence était orienté vers l'avenir, vers la question : « que va-t-il se passer ? »

C'est en tâchant de comprendre comment le souci de l'avenir s'articulait avec des pratiques sécuritaires qui concernaient le passé (la réparation), le présent (la maintenance) et le futur (la proposition de solutions) que j'ai proposé la notion de « projet-sécurité ». La sécurité n'est plus seulement un objet de sollicitude [*a matter of care*] mais un objet de préoccupation [*a matter of concern*]. Il s'agit notamment de se questionner non plus uniquement sur le présent mais aussi l'avenir : comment s'assurer que, malgré les nombreuses inconnues de l'avenir, notamment les vulnérabilités qui

78 Par exemple, la fondation NLnet (<https://nlnet.nl/>) est l'une des fondations vers lesquelles se tournaient régulièrement les développeuses que j'ai fréquenté·e·s lors de ma recherche pour obtenir un soutien financier. Cependant, cette fondation tend à financer les nouveaux projets, et non les projets de maintenance.

forcément émergeront tôt ou tard, la technologie puisse être suffisamment résiliente pour durer encore ?

Par projet-sécurité, j'entends l'ensemble de représentations personnelles induites par un souci de sécurité et qui s'observe par une attitude spécifique propre à chaque actrice vis-à-vis d'une situation sécuritaire. Il s'agit en somme de rendre compte de la manière dont chaque personne bâtit son univers sécuritaire à l'aide de pratiques, de discours et d'objets particuliers. Cette notion permet de saisir la sécurité non pas comme un état de fait mais comme un projet pour le présent et pour l'avenir que les actrices tenteraient de mettre en place. Le projet-sécurité vise à rendre compte des interprétations qui sont faites de ce que représente la sécurité et permet d'expliquer comment et pourquoi des actions sont entreprises en fonction de ces interprétations. Autrement dit, il ne s'agit pas d'étudier l'état d'un système, et encore moins les propriétés d'un objet, d'un code ou d'un algorithme, mais de comprendre ce que représentait la sécurité pour les cryptographes, codeuses et chercheuses que j'ai côtoyées durant la recherche, et comment ces personnes entreprenaient des actions en fonction de leurs interprétations.

Cette approche est très proche des pratiques que Becky Kazansky (2021) décrit à propos des modèles de menaces. Elle a identifié et décrit les stratégies que les utilisatrices de technologies de sécurité mettent en place pour pallier les risques qu'elles identifient. En répondant à un besoin d'approfondir la compréhension interdisciplinaire des formes de résistance et d'*agency* à l'égard de la surveillance numérique, elle analyse les pratiques explicites d'anticipation que sont l'identification des risques et des menaces qui planent sur les utilisatrices. Elle montre ainsi qu'au-delà des considérations purement technique, ces anticipations guident les utilisatrices vers certains outils techniques. Ainsi, selon elle, la sécurité n'est pas ces outils, mais ces pratiques d'anticipation qui cherchent à contrer la surveillance numérique. Dans mon cas, mon intérêt n'est pas porté sur les pratiques d'anticipation des utilisatrices, mais sur celles des conceptrices de ces technologies. Ma contribution participe ainsi à déplacer le regard plus en amont de la production de ces technologies.

De plus, la notion de projet-sécurité permet de réfléchir à la manière dont les actrices envisagent la sécurité dans le temps. Elle permet d'appréhender comment différents temps sont considérés d'un point de vue de la sécurité, mais aussi comment la sécurité fait émerger différents rythmes et différentes durées. La notion de projet-sécurité met l'accent sur le caractère non-définitif de la sécurité, voire même non accompli et irréaliste dans bien des cas. Je perçois en effet le projet-sécurité comme une asymptote qui tend vers l'infinitésimal sans jamais atteindre la valeur dont elle se rapproche : la sécurité est un projet qui tente de s'approcher au plus près d'un état que l'on sait pourtant inaccessible⁷⁹.

Peu importent la justesse des raisonnements des personnes concernées, le degré de risque impliqué ou leur niveau de littératie numérique : la sécurité émerge dès qu'on la mentionne et qu'on tâche d'en faire quelque chose. En ce sens, les descriptions académiques du fonctionnement de primitives cryptographiques n'ont pas plus de poids que le vague sentiment d'insécurité des utilisatrices ne pouvant pas voir le processus de chiffrement au moment du clic sur le bouton « envoyer »⁸⁰.

En m'inspirant de la notion de « concept sensible » de Blumer⁸¹, je conçois le projet-sécurité comme une façon de rendre compte de manière pragmatique de la fluidité des interprétations de ce que peut signifier « sécurité » sans avoir à m'attarder sur des propriétés *a priori*. J'insiste ici sur un point : le projet-sécurité n'est pas un synonyme de sécurité. Tout d'abord, comme je l'ai montré dans ce chapitre, la sécurité ne peut se réduire à une seule définition mais est, au contraire, un concept relationnel et

79 D'ailleurs, l'affirmation qu'un code ou qu'un système cryptographique serait parfaitement sûr serait pour beaucoup une raison de s'en méfier particulièrement. Après tout, même les systèmes cryptographiques les plus sécurisés ont été déchiffrés avec le temps (Kahn 1996; Singh 1999).

80 Pour une illustration de la méfiance de certaines utilisatrices à propos du chiffrement, voir l'article « *In Encryption We Don't Trust* » (Dechand et al. 2019).

81 « La notion de concept sensible chez Blumer renvoie au souci de ne pas objectiver les données saisies dans la fugacité du monde ou connaissant d'innombrables variations. La condition sociale des hommes, leurs interactions constantes, la matière de sens qui compose leur existence et leurs relations mutuelles n'autorisent guère des concepts clos ou définitifs. Les seuls susceptibles de rendre compte de la fluidité du sens dans les relations sociales sont des concepts sensibles dont le contenu n'est pas donné à l'origine de la recherche mais vaut comme proposition » (Le Breton 2008, 41).

dynamique. Par ailleurs, je considère comme réducteur de penser la sécurité au singulier alors que ma recherche m'a offert une telle palette d'interprétations toutes aussi pertinentes les unes des autres selon les situations. En analysant la manière dont les actrices donnent et créent du sens autour d'enjeux de sécurité, je montre comment l'idée de sécurité ou d'insécurité accompagne les actrices avec qui j'ai mené ma recherche.

Le projet-sécurité n'est pas non plus une idée abstraite qui découle d'un méta-raisonnement. Au contraire, je conçois cette notion comme particulièrement heuristique car elle peut être explicitée de manière très concrète à l'aide d'une triple interrogation :

1. Comment, selon la personne concernée, se présentent les choses ?
2. Comment, à son avis, devraient-elles être ?
3. Et que propose cette personne de faire pour améliorer la situation ?

Ces trois pistes de réflexion découlent de l'association des approches de la sécurisation et du soin des choses. Avant d'apporter un éclairage théorique sur cette association ci-dessous, j'aimerais donner un exemple concret d'un projet-sécurité. Pour cela, je vais prendre l'illustration d'une des contributions de l'équipe de développement de Sequoia à l'écosystème d'OpenPGP : la réalisation d'une série de tests pour identifier les problèmes d'interopérabilité entre les différentes implémentations : l'*OpenPGP interoperability test suite*⁸². Cette série de tests permet de vérifier que les différentes implémentations interprètent de la même façon les spécifications de la norme OpenPGP. Derrière cette idée se trouve un projet-sécurité qui m'a été explicité de la façon suivante par l'une des codeuses de Sequoia :

1. Dans un premier temps, elle m'a expliqué que l'écosystème d'OpenPGP était constitué de plusieurs implémentations concurrentes qui répondaient à différents besoins et usages. Ces implémentations avaient toutefois besoin les unes des autres car l'interopérabilité est justement la force des normes fédérées. Or, il s'avérait que les spécifications d'OpenPGP étaient non seulement

82 <https://tests.sequoia-pgp.org/>.

vieilles, mais également complexes à comprendre. Par conséquent, plusieurs problèmes d'interopérabilité existaient, ce qui n'était profitable pour personne, sauf les solutions indépendantes qui implémentaient leurs propres protocoles (Signal, Tutanota).

2. Pour améliorer cette situation, il fallait donc identifier ces problèmes et les résoudre. Selon cette développeuse, il était important de maintenir de telles normes fédérées et ouvertes car elles consistaient en une protection contre les systèmes de chiffrement propriétaires et obscurs.
3. La proposition de Sequoia a donc été de réaliser cette suite de tests qui permet de s'accorder sur une vision unanime des spécifications tout en identifiant certains bogues. Dans cette perspective, la suite de tests contribue à un renforcement de la sécurité d'OpenPGP par au moins deux leviers : d'une part, ces tests permettent de repérer et patcher des erreurs dans le code et des problèmes d'incompatibilité ; d'autre part, ils peuvent améliorer la collaboration entre les différentes implémentations en servant d'intermédiaires.

Ainsi, les trois questions posées permettent de mieux cerner le projet-sécurité qui a motivé la réalisation de cette suite de tests. Les réponses soulignent un souci d'interopérabilité qui est central pour le projet de Sequoia, qui se présente comme une librairie fédératrice et rassembleuse. Le projet-sécurité de Sequoia consiste non seulement à proposer le code le plus fiable possible, mais également d'améliorer l'entier de l'écosystème en identifiant et corrigeant certains problèmes au sein de l'écosystème.

Cet exemple permet de souligner la nécessité de considérer la sécurité comme un moteur qui fait faire des choses plutôt que comme un état passif que l'on ne pourrait que constater. Cet exemple montre aussi que la sécurité va bien au-delà des considérations mathématiques de la mécanique cryptographique et que d'autres valeurs que la solidité des algorithmes sont prises en compte : interopérabilité, collaboration, entraide. Une fois de plus, il importe de se souvenir qu'il ne s'agit pas d'établir une hiérarchie entre les différentes perspectives, mais de noter leur existence et leur

entrelacement dans la fabrique de la sécurité informatique. Maintenant que j'ai précisé ce que j'entendais par la notion de projet-sécurité, j'aimerais mettre en évidence la contribution théorique qu'elle constitue.

3.3.2. Le projet-sécurité et le temps

La notion de projet-sécurité m'a permis d'élaborer une conception de la sécurité informatique qui s'appuie à la fois sur la théorie de la sécurisation et sur les éthiques du *care*, avec en ligne de mire les temporalités de la sécurité. En discutant et nuancant ces deux approches dans le contexte des technologies numériques, cette thèse permet de comprendre d'une part les modalités du soin voué à ces technologies en fonction des différentes temporalités en jeu, et d'autre part l'organisation de ce travail de collaboration et de négociation de normes de sécurité informatique. Je m'appuie sur l'effet performatif des discours et pratiques d'(in)sécurisation, tout en considérant également le travail quotidien du soin qui s'apparente à un « infini recommencement » (Denis et Pontille 2022) et un « incessant bricolage » (Mol, Moser, et Pols 2010).

Les études sur la maintenance rendent compte du point de vue des personnes qui s'occupent de faire durer les technologies numériques au quotidien. Dans cette optique, j'ai décrit le travail habituel d'ingénieur·e-s et de codeuses qui conçoivent et codent des technologies liées à OpenPGP. Une part conséquente de leur travail consiste à patcher et améliorer le code (Vyas 2013), notamment en identifiant des vulnérabilités et en les patchant avant que des personnes malintentionnées n'utilisent ces failles de façon malveillante. Ainsi, il faut appliquer un patch à chaque fois qu'une erreur ou un problème est découvert. Cela signifie notamment qu'un système est supposé être sûr tant qu'aucune vulnérabilité n'est connue, ce qui ne signifie en aucun cas qu'aucune vulnérabilité n'existe. De cette façon, la sécurité est perçue à travers les connaissances du passé : la sécurité actuelle d'une technologie est analysée en fonction des connaissances déjà acquises. C'est ce qui se passe quand des scientifiques publient un rapport d'audit sur OpenPGP (par exemple, Barengi, Mainardi, et Pelosi 2017), ou quand un journaliste affirme que le protocole semble fiable

car la NSA n'arrivait pas à le casser (Brandom 2014). Les études sur la maintenance permettent de comprendre et décrire le travail des équipes de développement qui doivent sans cesse corriger leur code parce que le code du passé n'est plus à jour.

De leur côté, les théories de la sécurisation font de la sécurité un problème du présent à cause de menaces futures. En insistant sur la performativité des discours et des pratiques (in)sécurisantes, elles permettent de comprendre l'(in)sécurité en train de se faire. Pour ses partisan·es, la sécurité n'est pas un état objectif mais une construction d'abord discursive qui s'enracine dans certaines pratiques. Bien entendu, au centre des processus d'(in)sécurisation se trouve l'identification d'une menace existentielle et donc, par la force des choses, une anticipation de ce qui pourrait advenir. Toutefois, la force des théories de la sécurisation est justement de montrer que ce n'est pas cette anticipation qui importe le plus, et encore moins la réalité objective de ladite menace, mais le processus de construction et de problématisation qui est au cœur de la sécurité. En d'autres termes, c'est grâce à une projection dans l'avenir que la sécurité est construite. Ces deux approches se rejoignent dans la prégnance du présent : quand bien même passé, présent et avenir s'articulent, le moment privilégié de l'observation se fait sur le présent. Le projet-sécurité vient donc compléter ces perspectives en offrant une perspective orientée vers l'avenir : c'est non seulement l'état actuel de la technologie qui préoccupe les développeuses, mais aussi son état futur. La sécurité est appréhendée non seulement à travers les connaissances du passé (fonctionnement des primitives cryptographiques, existence de certains types de vulnérabilités, bogues identifiés dans le code), mais également à travers certaines projections de l'avenir.

L'intérêt de la notion de projet-sécurité repose ainsi sur sa capacité à sortir de la normativité d'une définition *a priori* de la sécurité pour mobiliser les représentations propres aux actrices et comprendre ce qu'elles en font, ou aimeraient en faire. Autrement dit, le projet-sécurité a pour but de montrer une attitude à l'égard d'enjeux considérés comme relevant d'une situation sécuritaire. Ainsi, la notion de « projet-sécurité » permet de cerner la perspective des personnes engagées dans une situation

sans projeter une vision normative de la sécurité, et ainsi de comprendre comment ces personnes créent du sens autour d'enjeux sécuritaires.

Il est courant d'insister sur la rapidité de l'évolution des technologies d'Internet qui, en l'espace de quelques décennies, ont radicalement transformé la manière d'être au monde de milliards d'individus (Cardon 2019). En effet, les discours sur les « nouvelles » technologies de l'information et de la communication sont souvent imbibés d'une idéologie de l'innovation, du progrès et surtout de la vitesse. Or, la cryptographie a cela de particulier que les temps qu'elle met en perspective sont souvent excessivement longs et lents par rapport aux temps de l'innovation tels qu'on se les représente généralement. Je vais donner trois illustrations de ces temps qui sont parfois particulièrement longs et parfois très courts. Ces différences d'échelles permettent de comprendre pourquoi le temps est un enjeu central de la sécurité informatique en général, et de la cryptographie en particulier. Le produit final en bout de chaîne (par exemple l'application Web de ProtonMail) encapsule ces différentes échelles de temps (et le souci des développeuses à leur égard) sans le laisser transparaître. Il est dès lors particulièrement important de bien comprendre en quoi le temps est un problème sécuritaire.

1. Des milliards d'années. Le premier exemple concerne un temps théorique qu'aucun humain ne connaîtra : depuis que la cryptanalyse s'est mécanisée avec les « bombes » de déchiffrement d'Enigma, il a été possible de calculer le temps pour tester toutes les possibilités d'une clé. Il suffisait de diviser le nombre de possibilités différentes d'une clé par le nombre d'opérations pouvant être réalisées en un certain temps. Une clé AES-128 compte, comme son nom l'indique, 2^{128} possibilités (soit 3.4×10^{38} ou plus précisément 340 282 366 920 938 463 463 374 607 431 768 211 456 possibilités). Par ailleurs, on peut savoir le nombre d'opérations que le réseau Bitcoin peut faire en une seconde (300×10^{18} , ce qui équivaut à environ 2^{68} opérations par seconde⁸³). En divisant 2^{128} par 300×10^{18} , on obtient $1,134274556 \times 10^{18}$ secondes, soit

83 Le réseau Bitcoin est utilisé comme référence ici car il s'agit d'un réseau d'ordinateur dont la puissance de calcul est actuellement l'une des plus importantes au monde, et que le nombre d'opération est documenté avec précision. Les opérations en question sont des opérations de hachage qui sont effectuées de très nombreuses fois. Le nombre de 300×10^{18} est une approximation pour

environ $3,6 \times 10^{10}$ années, ou 36 milliards d'années. Cela correspond à plus du double de l'âge supposé de notre univers, ce qui n'est pas peu. Ce temps est théorique car la chance ou d'autres facteurs peuvent faire en sorte qu'on tombe sur la bonne combinaison dès les premiers essais⁸⁴. Par ailleurs, je concède tout à fait que mes calculs peuvent être imprécis⁸⁵ mais peu m'importe car mon objectif est de donner un ordre de grandeur des temps extrêmement longs qui sont en jeu. Je prie donc les lectrices les plus scrupuleuses de ne pas donner trop d'importance aux détails de ces calculs et de ne retenir que l'ordre de grandeur qui, lui, est sans appel : les temps théoriques de la cryptographie sont immenses et c'est ce qui lui confère cet aura d'autorité.

2. Des décennies. La deuxième illustration des temps longs de la cryptographie est moins spectaculaire mais tout aussi informative. Il s'agit de la durée de vie des algorithmes qui sont aujourd'hui utilisés. En effet, si certaines primitives sont cassées très rapidement, comme ça a été le cas de Bass-O-Matic, l'algorithme mis au point par Zimmermann dans la version 1.0 de PGP (Garfinkel 1995), d'autres ont survécu beaucoup plus longtemps. Par exemple, l'algorithme RSA a été décrit en 1977 et reste très utilisé près de cinquante ans plus tard ; AES a été normalisé en 2000 et semble toujours être considéré comme une excellente solution 23 ans plus tard ; par ailleurs, un des candidats pour le concours de la nouvelle norme de cryptographie résistante aux ordinateurs quantiques est l'algorithme McEliece, développé en 1978 et toujours considéré comme très robuste. Ces durées de vie peuvent paraître étonnantes pour les personnes non-initiées aux temps des cryptographes, car les moyens informatiques ont grandement évolué depuis cette époque. Les protocoles cryptographiques

le mois de mars 2023: <https://www.blockchain.com/en/charts/hash-rate?scale=1&timespan=all>.

84 En se rappelant qu'une clé cryptographique est elle-même chiffrée par un mot de passe, on peut commencer par tenter sa chance en essayant les mots de passe les plus utilisés, voir par exemple : <https://nordpass.com/fr/most-common-passwords-list/>.

85 Il semble que ce soit le cas et que je sois particulièrement pessimiste. En effet, ChatGPT propose un temps « légèrement » supérieur de 10^{16} années pour casser une clé AES-128 avec une puissance de calcul de 10^{14} opérations par seconde (<https://chat.openai.com>), alors que l'entreprise de sécurité informatique Ubiq propose le chiffre nettement supérieur de 7×10^{25} années (<https://www.ubiqsecurity.com/128bit-or-256bit-encryption-which-to-use/>).

inventés durant cette décennie sont pourtant toujours fréquemment utilisés. D'ailleurs, les cryptographes ont tendance à avoir plus confiance en de vieux algorithmes qui ont survécu plutôt qu'en de nouvelles créations qui promettent une révolution :

En même temps, tous ces nouveaux algorithmes sont plutôt récents, n'est-ce pas ? La cryptographie est un monde très conservateur pour de bonnes raisons, parce que malheureusement, la manière dont sont construits ces algorithmes ressemble plus à de l'art qu'à de la science. Et peut-être que tu ne peux jamais vraiment savoir si ils sont sûrs. Le mieux que nous puissions faire est de faire des recherches tout autour de la terre, les étudier et tenter de les casser. Et cela prend du temps, du temps et de l'argent. Des années.
(Codeureuse d'une implémentation d'OpenPGP, entretien du 12 janvier 2023)

3. Quelques centièmes de secondes. Pour qu'un algorithme cryptographique soit adopté, il doit non seulement être fiable et réputé sûr, mais il doit aussi être rapide afin que l'usage de ces algorithmes fasse le moins possible attendre les utilisatrices qui auront alors une meilleure expérience du programme (Paquin, Stebila, et Tamvada 2020; Blanc et al. 2022). Cela s'appelle le *benchmarking* (analyse comparative, en français). Je me souviens qu'au sein de l'équipe de travail de Sequoia, de telles analyses comparatives étaient faites avec les autres implémentations car il s'agissait d'un argument important. Cela montre que les développeuses s'intéressent à l'expérience des utilisatrices notamment à travers le prisme du temps notamment, et que le temps consommé est un moyen de prendre certaines décisions. Par ailleurs, différentes attaques reposent sur la mesure de centièmes, voire millièmes de secondes. Ces attaques temporelles par canal auxiliaire consiste à estimer et analyser le temps mis pour effectuer certaines opérations cryptographiques dans le but de découvrir certaines informations⁸⁶.

Ainsi, les temps de la cryptographie sont à la fois gigantesques et minuscules. La notion de projet-sécurité permet de rendre compte de cet horizon temporel particulier à la cryptographie en mettant l'accent sur les projections et le sens qui est donné à ces périodes particulièrement longues et inconcevables. Pour certaines personnes,

86 Pour plus de détail, voir l'article de Wikipédia: https://fr.wikipedia.org/wiki/Attaque_par_canal_auxiliaire.

l'enjeu était de faire en sorte qu'OpenPGP continue d'être maintenu. Cela impliquait notamment de supposer qu'à tout moment, le pire pouvait arriver, tout en tâchant d'identifier les formes concrètes que ces menaces potentielles pourraient prendre. Cela impliquait aussi de chercher les meilleures solutions à ajouter à OpenPGP. Pour d'autres, il s'agissait de continuer à étudier les primitives, les protocoles et les codes pour en trouver des vulnérabilités importantes afin d'éviter que des failles inconnues soient utilisées. Finalement, certaines personnes étaient d'avis qu'OpenPGP n'apportait plus aucune solution face aux menaces que réserve l'avenir (et parfois même face à celles du présent) et devait donc être abandonné. Penser la sécurité en termes de projet permet de montrer son dynamisme et les projections qui sont faites en son nom, tout en regardant avec attention ce que créent ces projections.

3.3.3. Note sur l'inquiétude d'Alice et Bob

Dans ce chapitre, Alice et Bob nous ont emmené·e·s bien loin dans des considérations philosophiques sur ce qu'est ou représente la sécurité. Ce parcours était nécessaire pour souligner le caractère fluide et contextuel de la sécurité, et pour montrer les multiples façons de l'appréhender. Il me semble important d'insister sur le fait que la sécurité informatique est quelque chose de plus qu'une simple propriété d'un système et qu'elle implique différentes activités, pratiques et objets. La notion de projet-sécurité vient s'ajouter aux différentes théories mobilisées pour souligner aussi le travail des actrices pour projeter OpenPGP dans l'avenir, tâchant ainsi de lui conférer une pertinence future.

Cependant, tout cela ne doit pas faire oublier le cœur du problème, à savoir le besoin d'Alice et Bob (et Johnny) d'avoir un moyen fiable, sécurisé et facile à utiliser pour communiquer en ligne. Leur inquiétude qu'Ève (ou d'autres) n'entende leur conversation est à prendre au sérieux et il ne faut pas oublier que pour de nombreuses personnes, la confidentialité, l'intégrité et l'accessibilité des données est une question de vie ou de mort (Di Salvo 2020; Ermoshina 2023). Dans les chapitres suivants, je me concentre particulièrement sur la manière dont les actrices engagées dans des processus d'(in)sécurisation travaillent et collaborent en fonction de leur

projet-sécurité, et n'aborde que rarement le point de vue des utilisatrices. C'est pourquoi je voudrais souligner très clairement leur importance dans toutes les histoires que je vais relater et desquelles elles sont relativement absentes. En effet, toutes ces histoires n'ont de sens que parce que de réelles personnes sont directement concernées par la manière dont se fait et se défait la sécurité. J'aimerais donc prier les lectrices de ne pas oublier que ce sont les utilisatrices de ces technologies qui sont les premières impactées par ces processus d'(in)sécurisation.

Première partie : L'héritage du passé en mouvement

OpenPGP semble être un cas particulièrement intéressant pour questionner le vieillissement des technologies du numérique. En effet, l'âge du protocole est au cœur de nombreuses critiques. Beaucoup de personnes ont affirmé et affirment encore qu'OpenPGP n'est plus adapté à l'usage actuel d'Internet (Marlinspike 2016). D'autres ont également annoncé – ou du moins espéré – la mort d'OpenPGP (Katwala 2018; Latacora 2019; Wong 2020). Philip Zimmermann lui-même m'a affirmé préférer d'autres outils pour sécuriser ses conversations numériques. Pour lui, OpenPGP reste utile dans certains cas, mais au quotidien, il préfère utiliser d'autres outils plus simples à utiliser et mieux adaptés à l'usage actuel des smartphones (conversation personnelle)⁸⁷. Toutefois, la norme OpenPGP et ses implémentations sont toujours maintenues et utilisées par de nombreuses personnes. Il ne s'agit pas seulement de *geeks* nostalgiques d'une époque révolue, contrairement à ce qu'affirment les critiques. En effet, une grande partie de l'écosystème des logiciels libres et gratuits repose sur l'authentification des paquets par OpenPGP. Dans ce domaine, OpenPGP est un pilier de nombreuses communautés, à l'instar de la communauté Debian qui exige de ses contributeurices de posséder une paire de clés OpenPGP. Ainsi, OpenPGP n'est pas encore mort et enterré. Toutefois, le protocole fait face à de nombreuses et récurrentes critiques, dont voici un exemple :

⁸⁷ Voir aussi <https://www.vice.com/en/article/vvbw9a/even-the-inventor-of-gpg-doesnt-use-gpg>.

Pretty Good Privacy (PGP) est depuis longtemps la principale norme de l'IETF pour le chiffrement du courrier électronique, mais souffre de problèmes d'utilisation et de sécurité très répandus qui ont limité son adoption. Au fil du temps, le protocole cryptographique est devenu obsolète dans la mesure où PGP n'est pas authentifié pour chaque message et qu'il compresse les messages avant de les chiffrer. Les primitives de plus en plus obsolètes et les clients complexes utilisés par l'écosystème PGP ont fait l'objet d'un nombre croissant d'attaques. Cependant, les tentatives de mise à jour de la norme OpenPGP ont échoué à l'IETF, sauf en ce qui concerne l'ajout de primitives cryptographiques modernes. En dehors de la normalisation officielle, Autocrypt est une initiative communautaire "par le bas" visant à améliorer PGP, mais Autocrypt est toujours victime des attaques contre PGP qui impliquent l'authentification. La raison principale de l'incapacité à "fixer" PGP est l'absence d'une simple implémentation de l'AEAD⁸⁸ qui, à son tour, nécessite une infrastructure de clé publique décentralisée pour fonctionner avec le courrier électronique. Même si des normes telles que MLS remplacent PGP, le déploiement d'une infrastructure à clé publique décentralisée reste une question ouverte. (Halpin 2020, 1)

Il s'agit du résumé d'un article intitulé « *SoK: Why Johnny Can't Fix PGP Standardization* »⁸⁹. On peut y lire plusieurs critiques largement partagées à l'égard d'OpenPGP qui montrent l'ampleur des problèmes que pose le vieillissement du protocole : le nombre de vulnérabilités qu'il a cumulé dans son développement, la difficulté de le maintenir à jour dans un environnement qui change très vite, et les obstacles qui s'opposent aux initiatives tentant de le remplacer. À lire cela, on pourrait se dire qu'il serait effectivement bon qu'OpenPGP disparaisse bientôt – et je confesse ici que c'était d'ailleurs une des hypothèses que j'avais en commençant ma recherche. Or, force est de constater que la disparition d'OpenPGP continue d'être repoussée, jour après jour, patch après patch.

Dans cette partie, j'analyse le poids du passé et comment celui-ci influence le travail quotidien des codeuses. Il est question d'héritage, d'enlisement et d'ancrage infrastructurels (chapitre 4 « Enlisement et ancrage infrastructurels », p. 150), ainsi que de la complexité croissante d'un code qui vieillit en devenant toujours plus dense (chapitre 5 « Hériter, faire vieillir et transmettre le code », p. 186). Avant cela,

88 AEAD est l'acronyme de « *authenticated encryption with associated data* ». Il s'agit d'une manière de chiffrer des données tout en garantissant que celles-ci ne soient pas modifiées. C'est l'un des problèmes principaux identifiés par la recherche d'EFAIL.

89 « SoK » est l'abréviation de « *Systematization of Knowledge* » ; cet article propose un état de la littérature sur les problèmes de normalisation relatifs à OpenPGP.

j'aimerais toutefois donner un aperçu de ce que vieillir peut concrètement signifier dans le contexte d'une technologie informatique.

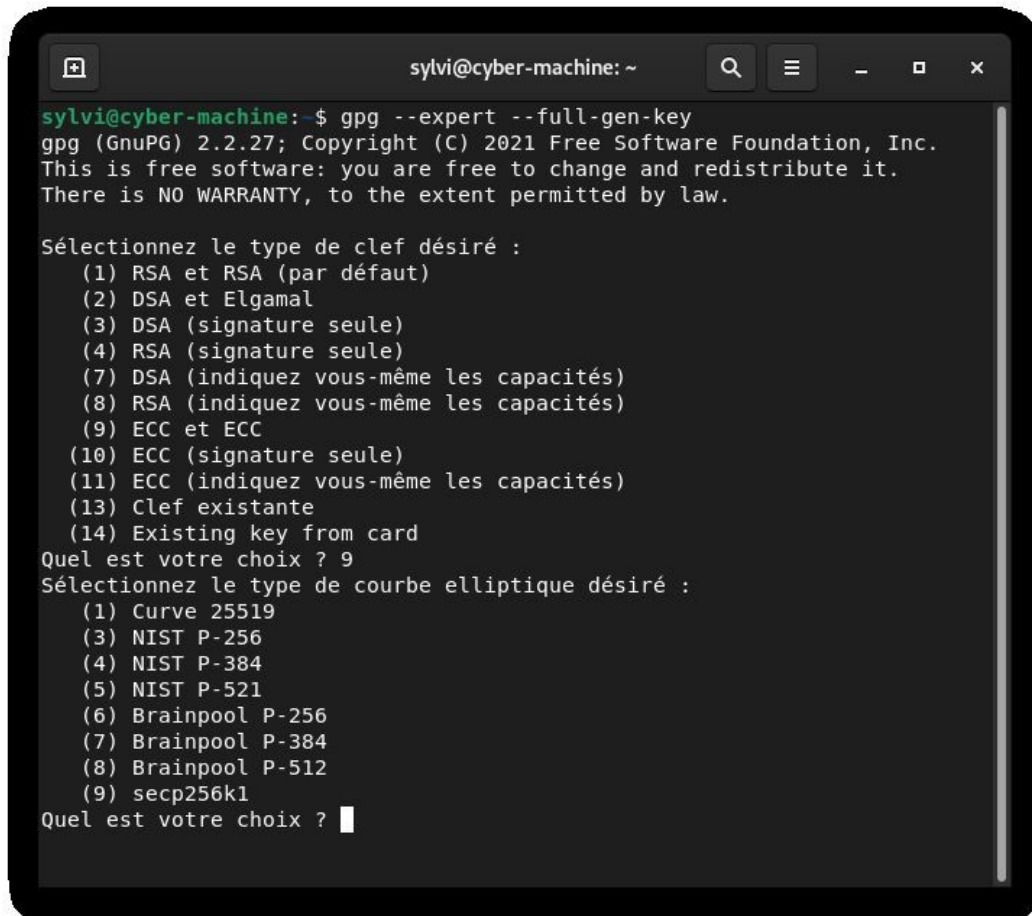
A screenshot of a terminal window titled 'sylvi@cyber-machine: ~'. The terminal shows the command 'gpg --expert --full-gen-key' being executed. The output displays the GnuPG version (2.2.27) and copyright information. It then prompts the user to select a key type from a list of 14 options. The user has selected option 9, 'ECC et ECC'. The terminal then prompts the user to select an elliptic curve type from a list of 9 options. The user has selected option 1, 'Curve 25519'. The terminal window has a dark background and a light-colored text. The window title bar shows standard Linux window controls (minimize, maximize, close) and a search icon.

Illustration 15 : Capture d'écran d'une fenêtre de terminal où GnuPG (v.2.2.27) est utilisé pour générer une nouvelle paire de clés OpenPGP en mode expert. Ici, les différents algorithmes de cryptographie sur les courbes elliptiques sont listés. Capture d'écran réalisée le 27 avril 2023.

Les différentes primitives cryptographiques disponibles dans GnuPG sont une trace visible de cet héritage du passé, mais cela n'est que la partie émergée de l'iceberg. En effet, ces différentes options ne concernent que le choix de l'algorithme de chiffrement asymétrique de la nouvelle clé et non pas tous les algorithmes supportés par le programme. Ce choix entre différents algorithmes n'est donc qu'un petit extrait des très nombreux paramètres que la norme OpenPGP décrit et qui ont évolué avec les années. Ainsi, au fil du temps, OpenPGP est devenu de plus en plus complexe et ramifié. Des changements profonds et importants ont été effectués et cette liste des algorithmes proposée par la version actuelle de GnuPG est l'une des traces visibles

des évolutions du protocole. Cet exemple permet ainsi d'avoir une illustration concrète des signes du vieillissement d'une technologie : il s'agit notamment, dans le cas présent, du foisonnement de protocoles utilisés dans cette technologie, couplé avec différentes façons de les implémenter.

Cette partie a pour but d'explorer les questions suivantes : comment une technologie numérique vieillit-elle ? Qu'est-ce que cela signifie ? Quels problèmes cela pose-t-il et quelles solutions sont apportées ? Et surtout, qui sont les personnes qui s'occupent de faire vieillir les technologies et que font-elles ? Cette partie explore comment des personnes ont pris le pari de faire durer cette technologie en l'adaptant à un environnement en perpétuel changement. Plus métaphoriquement, j'analyse une double tendance : comment les technologies s'enlisent dans l'infrastructure – qui s'enlise à son tour –, et ce qui est mis en œuvre pour contrer cet enlissement, c'est-à-dire pour maintenir OpenPGP en mouvement. Le paradoxe de la maintenance des choses informatiques, c'est justement le mouvement qu'on cherche à préserver, et non un état stable. Je montre ainsi l'incessant travail de correction qu'effectuent les mainteneuses d'OpenPGP qui tâchent de s'accommoder du passé tout en ne perdant pas de vue leur projet à long terme pour le protocole et ses implémentations.

4. Enlissement et ancrage infrastructurels

Comme je l'ai déjà répété plusieurs fois jusqu'ici, OpenPGP est un protocole qui est souvent considéré comme trop vieux et trop compliqué pour notre époque et qu'il vaudrait mieux le remplacer par d'autres outils plus « modernes ». Il est temps de s'arrêter sur cette critique et de comprendre ce qui est concrètement reproché à OpenPGP, pourquoi être vieux est un problème, et pourquoi il en est ainsi. Ce chapitre a pour but d'interroger le poids du passé dans le présent et le devenir d'OpenPGP. En développant les notions d'enlissement et d'ancrage infrastructurels, je montre que le passé est à la fois une source d'ennuis quand il s'agit de mettre à jour des technologies, et un facteur de stabilité quand il s'agit de se reposer sur une infrastructure enlisée. En termes de sécurité, la question est de savoir en quoi l'héritage du passé contraint certains développements à venir, et en favorise d'autres. Ce chapitre analyse ainsi comment le temps tend à enliser la technologie et la rendre immobile dans un environnement qui avance sans l'attendre, et comment la tâche des mainteneureuses peut être perçue comme une volonté de maintenir la technologie en mouvement.

Commençons par prendre connaissance des trois critiques suivantes qui permettent d'avoir une première impression du type d'arguments selon lesquels l'âge de la technologie engendre un problème sécuritaire.

Pour les communications sensibles, comme nous l'avons déjà écrit dans le *Motherboard Guide To Not Getting Hacked*, évitez d'utiliser PGP. Non pas à cause de

ces attaques en particulier [celles d'EFAIL], mais parce que PGP a des implémentations compliquées qui le rendent sujet à des bogues étranges, et qu'il est difficile d'apprendre à l'utiliser correctement.

(Franceschi-Bicchierai 2018, ma traduction)

Il y a [...] beaucoup de problèmes avec PGP. Heureusement, si vous n'êtes pas d'une curiosité morbide, il existe un méta-problème simple : il a été conçu dans les années 1990, avant l'avènement de la cryptographie moderne sérieuse. Aucun.e cryptographe compétent.e ne concevrait un système qui ressemblerait à PGP aujourd'hui, ni ne tolérerait la plupart de ses défauts dans un autre système. [...]

Un couteau suisse fait un tas de choses, toutes de manière médiocre. PGP fait un travail médiocre pour signer les fichiers, un travail relativement médiocre pour les chiffrer avec des mots de passe et un travail vraiment médiocre pour les chiffrer avec des clés publiques. PGP n'est pas un moyen particulièrement efficace de transférer un fichier en toute sécurité. C'est un moyen peu commode de signer des paquets. Il n'est pas très performant pour protéger les sauvegardes. C'est un moyen carrément dangereux de communiquer dans des messages sécurisés.

À l'époque de MC Hammer⁹⁰ dont PGP est issu, chiffrer était une chose à part ; il y avait un outil pour envoyer un fichier ou pour sauvegarder un répertoire, et un autre outil pour chiffrer et signer un fichier. La cryptographie moderne ne fonctionne pas de cette manière ; elle est conçue dans un but précis. La messagerie sécurisée nécessite une cryptographie différente des sauvegardes sécurisées ou de la signature des paquets.

(Latacora 2019, ma traduction)

[...] le véritable problème avec PGP n'est pas seulement la cryptographie ; c'est l'histoire que cela raconte à propos des dépendances et de comment fonctionne les communautés de logiciel. Le fait est qu'OpenPGP n'est pas véritablement un projet de cryptographie. C'est-à-dire, ce n'est pas la cryptographie qui donne de la cohésion à ce projet. Ce qui lui donne sa cohésion, c'est la compatibilité avec les systèmes hérités [*backwards compatibility*] et (de plus en plus) une espèce d'obsession avec l'idée de PGP comme une fin en soi plutôt que comme un *moyen qui rende effectivement les utilisatrices plus sûres*.

(Green 2018, souligné dans l'original, ma traduction)

Ces critiques se rejoignent sur un point : OpenPGP est décrit comme une chose du passé dont il vaudrait mieux se débarrasser au plus vite. À les lire, les technologies ne sont pas faites pour vieillir. D'un certain point de vue, ceci est assez logique. Chez les êtres vivants, le vieillissement mène à la mort. En est-il de même pour les technologies ? D'ailleurs de nombreuses personnes ont affirmé qu'OpenPGP était déjà mort –

90 Stanley Kirk Burrell de son vrai nom, MC Hammer est un rappeur états-unien né en 1962 et une figure majeure des années 1980 et 1990.

ou devrait l'être. J'ai déjà mentionné dans l'introduction l'article de Wired qui titrait « *We're calling it: PGP is dead* » (Katwala 2018) ; je peux ajouter d'autres exemples à la liste : « *A history of end-to-end encryption and the death of PGP* » est le titre d'un billet de blog du cryptographe David Wong (2020)⁹¹ ; « *I'm giving up on PGP* » est un billet de blog d'un cryptographe naguère grand défenseur d'OpenPGP⁹² ; ou encore un amer constat du magazine Motherboard qui a publié un article titré « *Even the Inventor of PGP Doesn't Use PGP* »⁹³.

Toutefois, la plus belle perle à propos du vieillissement d'OpenPGP, je l'ai trouvée dans un email reçu d'un chercheur de l'équipe d'EFAIL. Nous avons déjà échangé plusieurs emails, que je m'efforçais toujours de chiffrer le plus scrupuleusement possible. Puis, il m'a un jour envoyé un email non chiffré avec le commentaire suivant :

« J'ai supprimé PGP de mon client email, parce qu'il n'arrêtait pas de casser les choses [*it kept breaking things*]. Voici donc ma réponse en texte clair ! »

(Chercheur d'EFAIL, email personnel)

Désinstaller PGP parce qu'il était cassé était une chose ; le désinstaller parce qu'il cassait les choses autour de lui en était une autre. La première ne m'aurait pas tant étonné puisqu'il s'agissait d'un conseil qu'il avait donné sur Twitter, lors de ses conférences ainsi que lors de notre entretien. En revanche, sa formulation m'a généreusement fait sourire : est-ce qu'à l'instar de certains êtres humains avant de mourir, les technologie deviennent séniles ? Sans pouvoir répondre de manière définitive à cette dernière question, ce chapitre vise à comprendre ce que vieillir veut dire pour une norme de sécurité informatique.

Pour cela, il faut commencer par comprendre pourquoi cette technologie existe encore. Je commence donc par souligner l'importance infrastructurelle d'OpenPGP

91 <https://www.cryptologie.net/article/487/a-history-of-end-to-end-encryption-and-the-death-of-pgp/>.

92 <https://words.filippo.io/giving-up-on-long-term-pgp/>.

93 <https://www.vice.com/en/article/vvbw9a/even-the-inventor-of-pgp-doesnt-use-pgp>. Dans cet article, on apprend que Zimmermann lui-même a répondu à un email chiffré en demandant à ce qu'on lui envoie ledit message en texte clair.

en décrivant non seulement à quel point l'infrastructure contraint certains développements du protocole, mais également comment le protocole lui-même contraint le développement de l'infrastructure. Je prends ensuite l'exemple du Web of Trust pour illustrer de manière concrète comment l'héritage des idées du passé affecte le présent de la technologie. En parlant à la fois d'enlissement et d'ancrage infrastructurel, je montre comment l'infrastructure d'Internet contraint et en même temps facilite le vieillissement d'OpenPGP. J'émetts finalement une hypothèse qui sera confirmée au chapitre suivant : une technologie ne vieillit pas toute seule, mais grâce au travail de maintenance, de correction, d'adaptation et d'accompagnement de nombreuses personnes qui s'attachent à la faire durer.

Je m'inspire abondamment des études sur la maintenance des technologies pour décrire « l'art de faire durer les choses » (Denis et Pontille 2022), tout en contribuant à la discussion en mettant en avant deux spécificités liées aux technologies numériques : d'une part, les technologies informationnelles ne se dégradent pas toutes seules et il importe justement de souligner l'importance du travail de soin dont chaque opération contribue non seulement à faire durer un peu plus la technologie, mais également à la complexifier, à l'épaissir, à l'enliser – bref : à la faire vieillir. D'autre part, malgré l'usure causée par le vieillissement d'OpenPGP, je souligne également les opportunités d'innovation qu'offre un fort ancrage infrastructurel. Je nuance ainsi l'opposition entre maintenance et innovation, entre anciennes normes et innovation technologique.

4.1. L'importance infrastructurelle d'OpenPGP

Il me semble tout d'abord nécessaire de nuancer les critiques susmentionnées, car elles ne sont pas unanimes. En effet, dans certains milieux, OpenPGP est une technologie relativement banale – ou du moins elle est régulièrement utilisée sans que cela ne pose problème. Cela ressort très bien d'un entretien avec un·e cryptographe interviewé·e lors du 36^e Chaos Communication Congress en 2019 :

Je ne m'attends pas à ce qu'OpenPGP devienne célèbre et soit largement utilisé parce qu'il existe depuis... quand a-t-il été inventé ? Dans les années 90 ou quelque chose comme ça ? Cela fait donc au moins 20 ans [sic] qu'il est utilisé, et il a donc eu beaucoup d'occasions d'être utilisé par tout le monde. Il est compliqué à utiliser, il faut comprendre ce que sont les clés publiques et tout le reste pour pouvoir l'utiliser. Et il faut gérer soi-même toutes les clés. Je ne m'attends donc pas à ce qu'il soit largement utilisé. Toutefois, en ce qui concerne le fait qu'il meurt lentement, je ne pense pas que cela se produise, mais je n'ai pas de chiffres à ce sujet. Je pense que dans la communauté, où les gens sont familiers avec ce genre de sujets, les gens l'utilisent.
(Cryptographe, entretien du 28 décembre 2019)

La communauté que mentionne cette personne fait sans doute référence à la communauté du Chaos Computer Club : il s'agit principalement de groupements d'activistes des droits numériques, de *geeks* amateurices de nouvelles technologies et de développeureuses de logiciels libres qui utilisent très souvent OpenPGP à des fins d'authentification. Cette réflexion est intéressante et importante car contrairement aux opinions précédemment citées, la personne interviewée ne croit pas qu'OpenPGP puisse disparaître. Sa remarque semble à la fois pragmatique et résignée. D'un côté, elle soutient qu'OpenPGP n'est pas la meilleure solution face aux outils plus récents. De l'autre, elle nuance en affirmant qu'OpenPGP continue et continuera d'être utilisé car « les gens l'utilisent ». Cette tautologie remarquable pointe du doigt l'importance systémique d'OpenPGP dans certains milieux. En effet, il est intéressant de constater que les raisons invoquées dans cette citation ne sont ni idéologiques ni techniques, mais plutôt d'ordre social : dans un certain contexte et au sein de « la communauté », des gens savent se servir d'OpenPGP et s'en servent effectivement. Dans ces contextes, OpenPGP n'est donc pas un élément exotique et même si son usage n'est pas systématique, cette technologie reste actuelle.

Cette citation permet également de poser une question fondamentale : qu'est-ce qui autoriserait de décréter la mort clinique d'une technologie ? Cette question est en quelque sorte l'inverse de celle qui cherche à identifier ce qu'est une innovation. Or, comme dans le cas de l'innovation, il semble que l'adoption par des utilisateurices (Akrich 2006b) ainsi que la sanction positive d'un marché (Gaglio 2011) soient déterminantes. La remarque de cette cryptographe souligne l'importance de l'usage de la

technologie en question, et celle de la maintenance nécessaire pour faire en sorte que cet usage reste « sûr »⁹⁴. Elle permet d'aller au-delà de critères techniques qui pour juger si telle technologie est désuète. En soulignant la prégnance de l'usage, cette personne insiste sur la nécessité de considérer l'aspect social de la technologie. Les signes de la mort d'une technologie ne seraient ainsi pas techniques mais sociaux et liés aux usages.

Une autre remarque permet de nuancer encore davantage les critiques quant à l'âge d'OpenPGP. Pour que les utilisatrices continuent d'utiliser des technologies de sécurité informatique, il faut qu'elles aient confiance en ces technologies. Diverses études ont suggérés plusieurs sources de confiance dans les technologies de chiffrement, tenant en partie à la complexité et l'objectivité des mathématiques (Ruh 2018) ou à l'infrastructure (Bruun, Andersen, et Mannov 2020). Mais l'exemple d'OpenPGP permet d'ajouter que la confiance peut, peut-être paradoxalement, trouver sa source dans le « grand âge » de la technologie en question. En effet, pour certaines personnes, la vieillesse d'une technologie n'est pas un handicap mais au contraire un avantage certain en termes de sécurité. L'ancienneté d'une technologie peut représenter une certaine maturité, de nombreuses années d'expérience, un savoir-faire établi des développeuses. Cela ressort particulièrement bien de la citation suivante, tirée d'un entretien avec une codeuse de Sequoia :

Être vieux est globalement une bonne propriété pour un système de sécurité dans le sens que cela prouve qu'il suscite de l'intérêt, qu'il répond à un besoin, et qu'il a évolué et continue de le faire. OpenPGP va avoir 30 ans et toutes les primitives cryptographiques qui ont été introduites il y a 30 ans ont plus ou moins été supprimées. C'est comme un être humain qui voit toutes les cellules de son corps être remplacées sur une période de – je ne sais pas – 10 ans. Je pense que c'est une propriété qui est magnifique : remplacer véritablement toutes les parties et pourtant rester fonctionnel. Et je pense qu'en ce sens, le protocole s'est révélé très impressionnant.

(Codeuse d'une implémentation d'OpenPGP, entretien du 21 janvier 2021)

94 J'affine cette remarque dans le chapitre 8 (« Penser les fragilités à venir pour panser l'avenir », p. 300) dans lequel je soutiens l'importance capitale du soin dans les processus d'innovation technologique.

Cet extrait est particulièrement révélateur du travail effectué depuis trois décennies par les mainteneuses d'OpenPGP. Il souligne que, malgré toutes les critiques, OpenPGP peut être vu comme une (discrète) réussite en matière de soin et de maintenance. En effet, avec le temps, le programme de Zimmermann a été largement développé, donnant naissance à une norme de l'IETF ainsi qu'à une kyrielle d'implémentations dont chacune tente à sa manière d'utiliser le protocole du mieux possible pour effectuer différentes tâches. Il est dès lors intéressant de s'interroger sur les raisons qui ont incité les développeuses d'OpenPGP et de ses implémentations à s'obstiner à maintenir ces technologies plutôt que de réfléchir à un nouveau protocole, plus « moderne » et plus conforme à l'état de l'art actuel. En posant la question des motivations des mainteneuses, on en vient à se demander en quoi OpenPGP répond à leur projet-sécurité, et pourquoi elles souhaitent qu'OpenPGP fasse partie du projet-sécurité d'autres personnes. C'est en effet parce qu'OpenPGP fait partie du projet-sécurité de plusieurs personnes qu'il continue d'être maintenu.

En résumé, l'âge du protocole, qui est parfois critiqué, peut aussi s'avérer être un argument en faveur d'OpenPGP, car un grand âge signifie aussi des usages éprouvés et une bonne stabilité sur le long terme. D'un côté, l'enlisement décrié par certaines personnes devient un ancrage pour d'autres. Cela est à mettre en lien avec la notion d'infrastructure. Le concept d'infrastructure est en effet particulièrement utile pour comprendre les différentes imbrications de différentes technologies, normes et pratiques. D'une part, ce concept, tel qu'il a été défini par Susan L. Star (1999; Star et Ruhleder 2010)(1999; Star et Ruhleder 2010a) ou Geoffrey C. Bowker et ses collègues (2009), suggère différentes pistes heuristiques pour analyser les nombreuses interconnexions au sein d'un écosystème. D'autre part, l'étude des infrastructures en tant que « réseaux complexes de temporalités multiples » (Joniak-Lüthi 2019, 5, ma traduction) permet de saisir comment ces infrastructures évoluent dans le temps et font en retour évoluer la perception du temps.

Dans ce qui suit, j'analyse comment les technologies numériques ont tendance à s'enliser à cause de contraintes héritées du passé, tout en créant en même temps un

terreau fertile pour certaines innovations. Après un détour par une définition des infrastructures numériques, j'aborde les contraintes que représentent de telles infrastructures, notamment en ce qui concerne l'aspect sécuritaire de ces technologies, avant de clarifier en fin de chapitre l'idée d'ancrage infrastructurel. Faire évoluer l'infrastructure, c'est aussi faire évoluer les projets-sécurité qu'elle porte. Ainsi, en soulignant le potentiel d'innovation que représentent des technologies ancrées dans l'infrastructure existante, j'analyse la possibilité d'évolution de ces projets-sécurité.

4.1.1. Les infrastructures numériques

Commençons par poser une question centrale : qu'est-ce que la notion d'infrastructure apporte à la compréhension de la production de la sécurité informatique ? Pour répondre à cette question, il me faut d'abord préciser ce que j'entends par « infrastructure », puis exposer ce que cette approche permet de comprendre de crucial à propos d'OpenPGP. Comme l'affirme Natalia Buier (2023, 48, ma traduction), le terme « infrastructure » s'est largement répandu dans la littérature anthropologique au point où « tout est potentiellement infrastructurel ». L'autrice critique notamment les nombreux raccourcis dans les définitions utilisées, et particulièrement celles tirées de l'article de Brian Larkin (2013) qui est souvent cité comme référence dans la littérature sur les infrastructures⁹⁵. C'est également le cas d'une citation de Geoffrey Bowker et ses collègues, citation souvent utilisée pour définir les infrastructures :

Le terme « infrastructure » évoque de vastes ensembles d'équipements collectifs nécessaires aux activités humaines, tels que les bâtiments, les routes, les ponts, les voies ferrées, les canaux, les ports et les réseaux de communication. Au-delà des briques, du mortier, des tuyaux ou des fils, l'infrastructure englobe également des entités plus abstraites, telles que les protocoles (humains et informatiques), les normes et la mémoire. (Bowker et al. 2009, 97, ma traduction)

95 Par exemple, la première phrase de l'article de Larkin : « [l]es infrastructures sont des réseaux construits qui facilitent la circulation des biens, des personnes ou des idées et permettent leur échange dans l'espace » (2013, 328 ma traduction). Plus loin, Larkin ajoute que les « infrastructures sont de la matière qui permet le mouvement d'autres matières. Leur ontologie particulière réside dans le fait qu'elles sont des choses et aussi la relation entre les choses » (2013, 329 ma traduction). Ces points sont certes essentiels à la compréhension des infrastructures ; toutefois, isolés du reste de la définition, ils peuvent être appliqués à tant de contextes différents qu'ils en perdent leur sens premier.

Cette citation ne permet pas de comprendre ce que *sont* les infrastructures, mais ce qu'elles font et de quoi elles sont faites. D'une part, elles permettent la réalisation de nombreuses activités humaines ; d'autre part, elles sont composées d'éléments hétérogènes regroupant à la fois des matériaux concrets et des entités abstraites. La définition se précise mais ce qui relie ensemble ces éléments peut être difficile : où commence et où se termine une infrastructure ? Cette question est cruciale pour ne pas tomber dans le piège d'une notion polysémique qui se viderait de son sens. Ici, il est utile de remonter aux sources de cette notion, dans les années 1990. Conscientes de ce risque, Star et Ruhleder (2010, 117) ont posé la question « *quand* est-ce qu'il y a une infrastructure ? » plutôt que « qu'est-ce qu'une infrastructure ? ». En insistant sur le *quand* plutôt que le *quoi*, les chercheuses ont fait de la notion d'infrastructure un outil heuristique pour analyser les situations *in situ* où « l'infrastructure émerge pour des individus en pratique, liée aux activités et aux structures » (Star et Ruhleder 2010, 117). Elles ont établi une liste de huit propriétés (Star et Ruhleder 2010 [1996 pour l'article original]), à laquelle Star en a rajouté une neuvième quelques années plus tard (Star 1999). Dans l'encadré ci-dessous, je cite ces neuf propriétés et précise ce que ces propriétés permettent de voir à propos d'OpenPGP.

Les neuf propriétés des infrastructures du point de vue d'OpenPGP

1. **Encastrement (embeddedness).** OpenPGP est dépendante d'une grande quantité d'autres structures qui contraignent et façonnent son usage. Conçue à l'origine pour sécuriser des emails, elle est fortement liée au fonctionnement des emails eux-mêmes. Par exemple, les clés OpenPGP sont généralement rattachées à une adresse email. Pour reprendre les termes utilisés par Star et Ruhleder, OpenPGP est également coulée dans d'autres structures et liées à des arrangements sociaux, par exemple le fonctionnement de l'IETF. Finalement, d'autres technologies sont encastrees dans OpenPGP, comme par exemple les primitives cryptographiques.

2. **Transparence.** Selon Star et Ruhleder, l'infrastructure « est transparente pour l'utilisateur, c'est-à-dire qu'il n'y a besoin ni de la réinventer à chaque fois, ni de l'assembler pour chaque tâche, tout en restant un soutien invisible de ces tâches ». Il peut paraître étrange de considérer OpenPGP comme étant transparente, tant les critiques à son égard soulignent la difficulté de son utilisation. Toutefois, il est certain que dans certains contextes, comme la signature de paquets open source, par exemple, l'usage d'OpenPGP est transparent à l'usage. Certains logiciels et plateformes de partage de code permettent par exemple la signature automatique des commits avec OpenPGP.
3. **Portée ou étendue.** Il est évident qu'OpenPGP a une portée qui dépasse largement un événement isolé et une pratique unique. En effet, souvent comparée à un couteau suisse qui offre de multiples outils (voir par exemple Latacora 2019), OpenPGP est une technologie utilisée pour une grande diversité de situations, et ce depuis plus de 30 ans. Elle est par exemple utilisée pour chiffrer et signer des données au repos (archives, documents, etc.) ou des données en transit (emails, conversations instantanées) ; elle peut être utilisée comme outil de chiffrement asymétrique en utilisant une clé publique pour chiffrer une information, mais aussi comme outil de chiffrement symétrique en utilisant un mot de passe comme clé ; elle se retrouve dans des clients email, mais aussi sur les plateformes de partage de code GIT.
4. **Apprentissage comme bénéfice de l'adhésion.** Ce que Star et Ruhleder affirment en général par rapport aux infrastructures peut être appliqué sans problème au cas particulier d'OpenPGP : « Prendre pour acquis des artefacts et des arrangements organisationnels est une condition sine qua non pour faire partie d'une communauté de pratiques (Lave et Wenger, 1991 ; Star, 1996). Les étrangers et ceux qui viennent de l'extérieur rencontrent l'infrastructure comme un objet qu'il faut apprendre à utiliser. Les nouveaux participants acquièrent une familiarité naturalisée avec ses objets au fur et à mesure qu'ils y adhèrent » (Star et Ruhleder 2010b, 118). Les personnes que j'ai inter-

viewées qui avaient récemment commencé leur travail avec OpenPGP – par exemple parce qu’elles avaient été engagées au sein d’un projet – m’ont confirmé avoir passé de longs moments à essayer d’apprendre et comprendre comment fonctionnait le protocole, ainsi que de devenir familières avec ses différents artefacts, afin de pouvoir se sentir légitimes pour entrer en contact sur différents canaux de communication.

5. **Liens avec les conventions pratiques.** « L’infrastructure est à la fois façonnée et façonne les conventions d’une communauté de pratiques » (Star et Ruhleder 2010, 118). De nombreuses conventions de pratiques sont liées à OpenPGP. Par exemple, dans certaines communautés Linux (Debian, Arch, et beaucoup d’autres) ainsi que pour la communauté Wikipédia, posséder une paire de clés OpenPGP est un prérequis.
6. **Incorporation de normes et standards.** Normes et standards font parties des briques numériques nécessaires à la cohésion de l’infrastructure d’Internet. En tant que norme de l’IETF, OpenPGP participe ainsi à cette infrastructure. Par ailleurs, il faut aussi souligner qu’OpenPGP elle-même mobilise d’autres normes. Par exemple, il est possible de formater un message en format MIME ou en caractère ASCII, deux protocoles importants. Les implémentations d’OpenPGP ont quant à elles également recours à un grand nombre de conventions, notamment liée à l’écriture du code (par exemple le langage informatique ou le style d’écriture).
7. **Construite sur une base installée.** OpenPGP est bien entendu construite sur les bases de l’infrastructure d’Internet en général, et celle des emails en particulier. Elle a hérité à la fois des forces et des faiblesses de cette base. La décentralisation est un exemple qui a son importance. En effet, ce n’est que parce que l’écosystème des emails est décentralisé qu’OpenPGP l’est aussi. Or, cette décentralisation est justement l’un des points souvent mentionnés comme critique vis à vis de la sécurité d’OpenPGP en ce qu’elle empêche une réaction rapide et unilatérale lorsque des vulnérabilités sont divulguées. Cette décentralisation rend tout changement dans la norme difficile car nécessitant le

consensus de toutes les implémentations du protocole (ou du moins une grande majorité d'entre elles). Ces changements doivent être négociés au sein du groupe de travail de l'IETF, puis appliqués dans le code des implémentations. Cela dit, la décentralisation présente également de nombreux avantages, comme la flexibilité ou la résilience des solutions proposées sur le long terme.

8. **Devient visible au moment d'une défaillance.** La question de la visibilité est une question complexe. Comme mentionné plus haut, ce n'est pas forcément l'infrastructure même qui devient visible, mais son usage. Alors que beaucoup de gens considéraient OpenPGP comme peu utilisé et désuet en 2018, la révélation des failles d'EFAIL a en fait permis de voir une grande quantité d'utilisatrices inquiètes face à l'incertitude provoquée par cette révélation (voir chapitre 6 p. 226). La défaillance a permis de rendre visible non pas l'infrastructure même, mais l'ampleur de son usage.
9. **Est réparée par incréments modulaires, pas tous en même temps ni globalement.** C'est la dernière propriété rajoutée par Star trois ans après la publication du premier article (Star 1999). Elle a ici toute son importance pour comprendre combien il est difficile de changer OpenPGP. Les multiples usages du protocole, la communauté hétérogène du groupe de travail ainsi que la diversité des milieux dans lesquels est déployée OpenPGP font de cette norme un cas emblématique de cette neuvième propriété. Les changements ont lieu à différents lieux, au sein des équipes de développement des implémentations, au sein du groupe de travail, au sein d'équipe de recherche universitaire, etc. L'inertie causée par cette décentralisation est d'ailleurs un des arguments les plus souvent cités pour favoriser les systèmes non-fédérés qui peuvent réagir unilatéralement et donc beaucoup plus rapidement⁹⁶.

96 À ce sujet, voir la conférence que Moxie Marlinspike, fondateur de Signal, a faite au Chaos Communication Congress de 2019 intitulée : « The ecosystem is moving. Challenges for distributed and decentralized technology from the perspective of Signal development » (Marlinspike 2019).

Dans ses travaux sur la gouvernance d'Internet, Francesca Musiani (2018; voir aussi Musiani et al. 2016; DeNardis et Musiani 2016) a montré comment de nombreuses technologies qui font notre quotidien remplissent une « fonction infrastructurelle ». Parmi ces éléments, elle mentionne des objets physiques (des câbles, des centres de données) et des objets plus abstraits (des protocoles, des programmes). Elle affirme par ailleurs que ces « artefacts et [ces] entités qui peuplent et façonnent le réseau des réseaux [...] contribuent à structurer, façonner, modeler, permettre ou contraindre notre “être-ensemble” sur et avec l'Internet » (Musiani 2018, 162). Chercher à cerner les fonctions infrastructurelles d'OpenPGP m'a amené à interroger la manière dont différentes entités s'influençaient réciproquement car elles dépendaient les unes des autres. Par exemple, comme je le montre dans la prochaine section, le Web of Trust découle et dépend d'OpenPGP qui, en retour, a besoin de solutions telles que le Web of Trust pour être utilisé correctement. Ainsi, des changements chez l'une de ces entités provoquent forcément des modifications dans l'autre, d'où l'idée d'une dépendance réciproque qui se traduit souvent par un enlisement.

L'usage du terme « fonction infrastructurelle » peut cependant mener à une vision organique et fonctionnaliste d'Internet où chaque entité aurait une fonction établie qui réponde à un besoin clairement identifié. Or ce n'est à mon avis pas comme cela qu'Internet fonctionne : plusieurs normes cherchent à remplir la même fonction alors que d'autres « besoins » restent complètement délaissés. De plus, une vision fonctionnelle d'Internet impliquerait aussi que chaque élément ait une place définie et établie ; or, comme le montre très bien l'exemple d'OpenPGP, cela n'est pas le cas et une compétition acharnée existe entre les projets. En effet, Internet est une infrastructure qui est sans cesse en train d'être conçue et développée et qui repose sur des bases parfois surprenantes. Un cas extraordinaire avait d'ailleurs fait briller beaucoup de pixels quand Azer Koçulu, un programmeur, a effacé 17 lignes de code d'une librairie, ce qui a provisoirement rendu inutilisable des centaines de projets en JavaScript (Collins 2016; Gallagher 2016) :

L'un des modules supprimés par Koçulu s'est avéré être `left-pad`, un bout de 17 lignes de code en JavaScript très largement utilisé pour justifier un texte à droite. En fait, de nombreuses personnes ne savaient pas qu'elles l'utilisaient dans leur code, car il était enfoui dans les dépendances des outils qu'elles utilisaient. (Gallagher 2016, ma traduction)

Cet exemple illustre un problème bien connu des infrastructures du numérique, et notamment celles qui sont basées sur des technologies libres et ouvertes comme c'est le cas de la grande majorité des technologies d'Internet. Il s'agit de l'imbrication de différents projets dans des relations de dépendances : pour fonctionner, un programme fait appel à un bout d'un autre programme qui lui-même fait appel à quelques lignes de code d'une librairie qui elle-même est basée sur le code d'un autre logiciel, etc. Comme certains logiciels comptent des millions de ligne de code, il devient alors difficile pour les codeurs et codeuses d'avoir une vision complète de chaque lien de dépendance. Mieux que toutes autres explications, le dessin XKCD ci-dessous montre l'ampleur du problème⁹⁷. Internet n'est pas un ensemble cohérent et harmonieux qui est le résultat d'une concertation établie et cadrée. Au contraire, l'infrastructure ressemble à un empilement *ad hoc* de multiples projets hétérogènes qui ont des buts parfois très divergents. La « santé » de cette infrastructure dépend parfois de modestes projets et la bonne volonté de bénévoles qui maintiennent un code sans même forcément connaître l'impact que ce dernier a sur ladite infrastructure.

97 Je recommande d'ailleurs la lecture instructive et amusante de l'explication de cette caricature: <https://www.explainxkcd.com/wiki/index.php/2347: Dependency>.

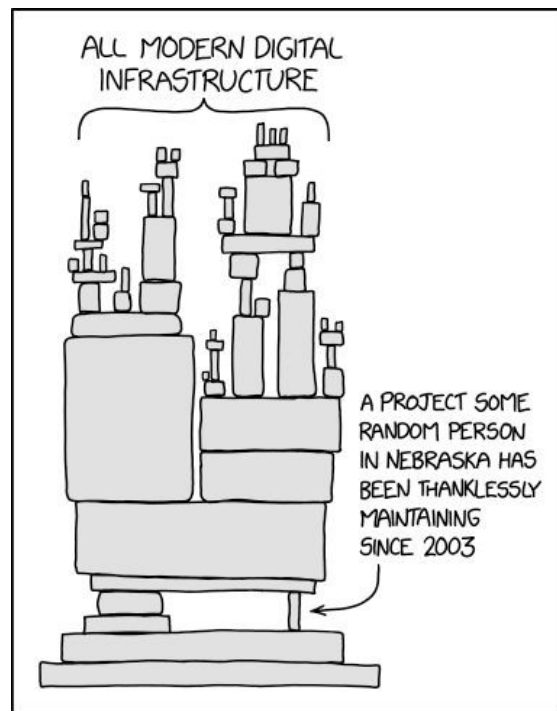


Illustration 16 : Caricature XKCD illustrant la fragilité des infrastructures open source et des liens de dépendances. Source: <https://xkcd.com/2347/>.

Ainsi, l'infrastructure n'est pas *une chose unique*, mais elle correspond à la manière dont les anthropologues appréhendent un ensemble de choses liées entre elles dans un contexte donné. Ces choses sont à la fois matérielles et immatérielles, vivantes et non-vivantes. Une fois de plus, je le répète, c'est dans ce qu'elle permet de voir que la notion d'infrastructure est utile, et non dans une quelconque essentialisation. Dans le cas de ma recherche, cette notion s'est révélée utile pour comprendre à la fois la manière dont une multitude d'éléments dépendaient les uns des autres et la manière dont cette dépendance réciproque affectait l'évolution de la technologie en question. L'illustration du Web of Trust dans la section suivante permet de comprendre en partie ce qu'apporte OpenPGP à l'infrastructure d'Internet et, inversement, comment cette infrastructure encadre, voire contraint, le développement du protocole. Cet exemple permet ensuite de mieux cerner un cas concret d'enlisement infrastructurel avec le cas du remplacement de SHA-1.

4.1.2. Le Web of Trust et l'héritage du passé

L'exemple du Web of Trust (WoT) qui suit permet de questionner ce qui est hérité du passé et la manière dont les protocoles encapsulent des idées du passé, et comment cet héritage ressurgit sans cesse sous forme de contraintes qui pèsent sur la maintenance et le développement d'OpenPGP. Même si plusieurs développeuses ne s'identifient plus à cette idée de pouvoir créer un véritable réseau de confiance entre les différentes utilisatrices d'OpenPGP, elles ne peuvent tout simplement pas ignorer cet aspect-ci de la technologie et doivent, d'une manière ou d'une autre, apporter une réponse adéquate aux demandes et besoins de fonctionnalités liées au WoT.

Le Web of Trust, littéralement la toile de la confiance, est une idée que Philip Zimmermann a développée parallèlement à PGP. Le WoT et PGP sont liés l'un à l'autre car ils ont été pensés et conçus en parallèle. Le WoT est en fait une solution à l'un des problèmes majeurs de la cryptographie à clé publique : la découverte et l'authentification des clés publiques. Il s'agit d'une étape fondamentale car chiffrer une information à destination de quelqu'un sans avoir eu accès à la bonne clé publique peut avoir de fâcheuses conséquences : au mieux, le message envoyé est indéchiffrable pour son ou sa destinataire ; au pire, c'était une attaque ciblée d'Ève s'était arrangée pour substituer la clé publique, et l'erreur commise lui a donné accès à des informations confidentielles. Pour pallier ce problème, Zimmermann a imaginé un système qui permettait non seulement de rechercher des clés publiques (comme un bottin de téléphone) mais également d'attester de la légitimité des clés en question en signant des clés publiques vérifiées et en pouvant consulter les signatures sur des clés récupérées sur un serveur. Zimmermann a décrit le principe du WoT dans le manuel d'utilisation de son programme PGP 2.0, en 1992 :

Au fil du temps, vous accumulerez des clés d'autres personnes que vous souhaitez peut-être désigner comme introducteurs de confiance. Toutes les autres personnes choisiront leurs propres introducteurs de confiance. Et chacun accumulera et distribuera progressivement avec sa clé une collection de signatures de certification d'autres personnes, en espérant que toute personne qui la recevra fera confiance à au moins une ou deux de ces signatures. Cela entraînera

l'émergence d'un réseau de confiance décentralisé et résistant aux pannes [*de-centralized fault-tolerant web of confidence*] pour toutes les clés publiques.⁹⁸

Le WoT a régulièrement été critiqué pour différentes raisons. Un email intitulé « *Why the Web of Trust Sucks* » envoyé sur une liste de diffusion liée au réseau TOR en 2013⁹⁹ résume certaines de ces critiques, notamment les trois points suivants : le WoT ferait fuiter certaines informations, en particulier le carnet d'adresses ; il existe de nombreux points de défaillances uniques qui peuvent causer la panne du système ; et ce système n'est pas adapté à l'usage massif d'Internet actuel. Par ailleurs, il y a eu de nombreuses attaques contre la manière de vérifier les clés¹⁰⁰ ou contre les serveurs eux-mêmes¹⁰¹. Finalement, la difficulté d'usage était aussi un problème majeur, comme l'explique Filippo Valsorda, qui s'est décrit lui-même comme un ancien utilisateur modèle de PGP : il connaissait bien la technologie, était enthousiaste à son sujet et était bien ancré dans la communauté. Il a toutefois changé d'avis et l'explique dans son billet de blog « *I'm giving up on PGP* » :

Je n'ai jamais utilisé avec succès le WoT pour valider une clé publique. Et n'oubliez pas que j'ai une clé bien connectée. Je n'ai pas réalisé d'étude formelle, mais je suis presque certain que toutes les personnes qui ont utilisé PGP pour me contacter ont fait ou auraient fait (si on le leur avait demandé) l'une des choses suivantes :

- extraire la clé la plus prometteuse d'un serveur de clé, probablement sans utiliser TLS
- utiliser une clé différente si j'avais répondu : « voici ma nouvelle clé »
- renvoyer l'email non chiffré si j'avais donné une excuse comme « je suis en voyage » (Valsorda 2021, ma traduction).

De nombreuses critiques ont évalué les difficultés des utilisatrices à s'approprier la gestion des clés avec OpenPGP (Whitten et Tygar 1999; S. L. Garfinkel et Miller

98 Ma traduction : <https://web.pa.msu.edu/reference/pgpdoc1.html>.

99 <https://lists.torproject.org/pipermail/tor-talk/2013-September/030235.html>.

100 La plus connue étant certainement l'attaque où des scientifiques ont créé des clés avec une empreinte très proche des clés originelles (les derniers caractères correspondaient). Pour prouver l'efficacité de leur attaque, les scientifiques ont posté ces clés sur les serveurs de clés et ces clés ont alors été confondues avec les clés originales. <https://evil32.com/>

101 En 2019, une attaque contre les clés OpenPGP a vu le jour d'abord de manière très discrète puis de manière virale : des clés étaient corrompues par des signatures trop lourdes et nombreuses, ce qui faisait crasher les implémentations des utilisatrices, rendant ainsi impossible leur usage d'OpenPGP. <https://nvd.nist.gov/vuln/detail/CVE-2019-13050>

2005; Sheng et al. 2006; Sousedek 2008). À en croire ces critiques, la situation semble claire : le WoT devrait être abandonné. Mais plusieurs éléments permettent de nuancer cette affirmation. Il s'agit des efforts fournis non seulement pour maintenir une telle infrastructure, mais également pour la développer. Plusieurs exemples existent. Sequoia a par exemple conçu un système appelé OpenPGP CA qui implémente le WoT sous la forme de certificat collectif au niveau d'une entreprise ou d'une organisation et l'entreprise Proton a également conçu un système équivalent et compatible¹⁰². En outre, en 2019, un codeur a par exemple mis en place un serveur de clés nommé Hagrid, relativement ambigu du point de vue du WoT¹⁰³. D'un côté, ce serveur supprime la partie des clés publiques où sont attachées les signatures, rejetant ainsi l'idée du WoT. Mais en s'assurant que les adresses emails attachées aux clés sont effectivement capables de déchiffrer les messages envoyés, ce serveur évite une série d'attaques possibles. Il propose ainsi une forme simplifiée de vérification.

Finalement, pour bien comprendre l'ampleur de la situation, il convient encore d'ajouter un point essentiel : OpenPGP est, comme je l'ai déjà mentionné, largement utilisé à des fins d'authentification, notamment dans les communautés open source. Souvent, avoir une clé OpenPGP pour signer ses contributions est un prérequis pour faire partie de telles communautés¹⁰⁴. L'authentification des contributions grâce aux signatures OpenPGP remplit donc une fonction infrastructurelle majeure pour l'usage actuel des technologies numériques et les solutions pour proposer des alternatives plus « modernes » sont souvent très controversées¹⁰⁵.

102 Voir <https://openpgp-ca.org/> et <https://proton.me/blog/why-we-created-protonca>. Voir aussi section 4.3 (p. 176) pour plus de détails sur OpenPGP CA.

103 <https://keys.openpgp.org/> (voir aussi le billet de blog de Sequoia par Walfield 2019).

104 Il en est ainsi des majeures communautés Linux (Debian, Arch, Fedora, etc.), ainsi que de Wikipédia.

105 L'exemple d'une nouvelle solution pour signer les paquets Debian a été proposée en 2021 et une ébauche de protocole avait été postée sur le wiki de Debian sous le titre : « *Ditching OpenPGP, a new approach to signing APT repositories* » (<https://wiki.debian.org/Teams/Apt/Spec/AptSign>). Cette proposition a suscité de nombreux commentaires, parmi lesquels se trouvent des très optimistes et des très critiques. Voir par exemple : https://lobste.rs/s/srsazh/ditching_openpgp_new_approach_signing.

Ainsi, le WoT n'est pas un héritage complètement délaissé et, malgré de nombreuses critiques, il continue d'être développé. Il en ressort ainsi un paradoxe important et intrigant : cette infrastructure semble pleine de défauts et en même temps paraît irremplaçable. Dans la section suivante, je prends l'exemple du remplacement de SHA-1 pour montrer en quoi la forte dépendance réciproque entre différents éléments d'une même infrastructure peut entraîner des complications dans le vieillissement d'une technologie. Dans le cas présent, on parle d'un algorithme de hachage qui est utilisé notamment pour calculer, à partir des informations contenues dans une clé publique, l'identité de cette clé, aussi appelée empreinte. Il s'agit d'une série de 32 caractères hexadécimaux¹⁰⁶. Cette empreinte est publique et utilisée pour vérifier les identités des clés. C'est donc une fonction centrale dans le fonctionnement d'OpenPGP et du WoT. L'exemple qui va suivre montre comment un changement qui paraît évident d'un certain point de vue peut simultanément sembler une dangereuse aberration d'un autre point de vue. Cet exemple illustre donc une des raisons qui montre pourquoi une norme telle OpenPGP ne peut évoluer que lentement : elle s'enlise dans sa propre maintenance et son propre développement, coincée entre la nécessité de s'adapter aux nouvelles conditions de son existence et la nécessité tout aussi impérieuse de rester compatible avec le système actuel – hérité du passé.

4.1.3. Travailler avec le passé, travailler en dépit du passé : remplacer SHA-1

En 2020, des négociations ont eu lieu au sein du groupe de travail OpenPGP de l'IETF pour négocier une nouvelle version des clés OpenPGP (Key V5). Ces négociations ont duré plusieurs années, ce qui n'est pas exceptionnel au sein de ce groupe de travail de l'IETF. Parmi les changements proposés par la 5^e version des clés se trouvait un changement important dans la manière de former l'empreinte d'une clé. Les clés V4 utilisent l'algorithme de hachage SHA-1 : or celui-ci présente de nombreuses vulnérabilités connues¹⁰⁷ et certaines personnes souhaitaient le supprimer totalement

¹⁰⁶ On peut voir un exemple d'une telle empreinte sur la page de titre de cette thèse.

¹⁰⁷ Dès 2005, des attaques d'abord théoriques puis pratiques ont été élaborées par des scientifiques. Voir notamment : <https://fr.wikipedia.org/wiki/SHA-1#Attaques>. Des vulnérabilités spécifiques à l'écosystème OpenPGP ont aussi été théorisées : <https://nvd.nist.gov/vuln/detail/CVE-2019-13050>.

des spécifications du protocole pour le remplacer par un autre plus puissant (SHA-256). Alors que de ce point de vue, cette proposition paraît tout à fait logique, elle a fait l'objet d'une forte résistance pour une raison également très logique : toutes les clés de l'écosystème OpenPGP étaient identifiées grâce à l'algorithme SHA-1. Ne plus l'utiliser signifiait potentiellement « casser toutes les empreintes digitales existantes sur la planète », comme l'a affirmé un contributeur du groupe de travail OpenPGP dans l'extrait du mail ci-dessous :

La première chose à faire pour "réparer" les empreintes digitales SHA1, c'est-à-dire casser toutes les empreintes digitales existantes sur la planète, est de définir les propriétés qu'elles doivent avoir. Je ne vois pas pour quelle raison SHA-256 est acceptable alors que SHA-1 ne le serait pas. Donc, avant d'ajouter arbitrairement SHA-256, il faudrait définir ce qui est nécessaire pour un algorithme d'empreinte digitale afin de voir pourquoi -1 ne répond pas aux exigences, et si -256 y répond.

Pour anticiper la réponse inévitable, la résistance aux collisions n'est pas une propriété que je considère comme nécessaire, étant donné que X.509 et S/MIME utilisent des chaînes de texte statiques comme identifiants depuis bien avant que PGP n'existe, sans que personne n'ait trouvé de vulnérabilités avec ces chaînes. Un algorithme d'empreinte digitale n'est qu'un moyen de faire correspondre une clé à un identifiant de longueur fixe, pour lequel presque n'importe quoi peut faire l'affaire.

(Peter Gutmann, email sur la liste de diffusion du groupe de travail OpenPGP, 5 juin 2021)¹⁰⁸

Cet exemple montre comment le passé restreint le potentiel d'évolution d'une technologie à cause de problème d'interopérabilité. Après de nombreux échanges d'emails sur la liste de diffusion, un compromis a été trouvé : il s'agissait d'opérer ce changement tout en insérant dans les spécifications de la norme un court texte expliquant pourquoi SHA-1 avait été remplacé. Voici la proposition de texte qui a été discutée sur la liste de diffusion :

Il n'y a pas de problème cryptographique introduit par [l'usage de SHA-1] puisque l'empreinte digitale est simplement une valeur opaque de longueur fixe utilisée pour identifier les données structurées de longueur variable qui constituent une clé publique. En particulier, le passage à SHA-256 pour les empreintes digitales V5 n'a pas été fait pour remédier à une quelconque vulnérabilité cryptographique, mais pour éviter de donner l'impression que quelque chose d'insécurisant pourrait se produire en raison de l'utilisation de SHA-1.

(Peter Gutmann, email sur la liste de diffusion du groupe de travail OpenPGP, 7 juin 2021)¹⁰⁹

108 « Re: [openpgp] v5 in the crypto-refresh draft, ma traduction » : <https://mailarchive.ietf.org/arch/msg/openpgp/UrPnPm-BIEDIUBdUDo3ZGLPSOWU/>

Cet exemple illustre le fait qu’il n’y a pas une seule logique à l’œuvre dans la maintenance d’OpenPGP ou, pour le dire en termes plus précis, que la sécurité n’est pas qu’une question d’attaque cryptographique mais aussi de compatibilité et d’usage. En ce sens, le passé contraint non seulement le présent (la maintenance du protocole au jour le jour) mais également son avenir (son développement sur le long terme) car il confronte les négociateurices à leur projet-sécurité respectif. Pour certaines personnes, l’interopérabilité avec le passé prime, alors que pour d’autres, la résistance à de potentielles attaques est plus importante. Comme ces deux logiques ne se justifient pas par les mêmes valeurs, elles entrent régulièrement en friction. Cette tension était d’autant plus grande que l’authentification est de loin la fonction d’OpenPGP la plus utilisée, notamment parmi les communautés de logiciels open source. Cet exemple permet d’illustrer un problème de taille concernant les infrastructures numériques : leur inertie. Dans la section suivante, j’aborde la question des blocages causés par le poids du passé dans l’infrastructure d’OpenPGP sous le prisme de l’« enlisement infrastructurel ».

4.2. L’enlisement infrastructurel

J’appelle « enlisement infrastructurel » l’inertie causée par la relation d’interdépendance entre les protocoles et l’infrastructure. L’exemple du remplacement de SHA-1 en est une illustration : l’état de l’art de la cryptographie « moderne » voudrait que SHA-1 ne soit plus utilisé depuis longtemps, mais le remplacer pose de gros problèmes de compatibilité non seulement avec les anciennes versions du protocole, mais également avec d’autres protocoles de l’infrastructure (comme la vérification des logiciels open source). Les discussions autour d’OpenPGP sont particulièrement sujettes à de telles tensions tant le protocole est utilisé pour une multitude d’usages dans différents contextes (le chiffrement et l’authentification d’emails, le chiffrement symétrique d’archives ou encore l’authentification des paquets dans plusieurs communautés Linux). Toutefois, ce genre de blocages se retrouvent dans la plupart des

109 « Re: [openpgp] v5 in the crypto-refresh draft » : <https://mailarchive.ietf.org/arch/msg/openpgp/poTul49MejKBk2pUf4FafwyrOEY/>

projets collectifs où il existe une tension entre des opinions conservatrices et des opinions tournées vers l'innovation. J'en donne une illustration avec la citation suivante tirée d'un entretien avec une contributeurice régulière de plusieurs groupes de travail à l'IETF, dont le groupe de travail OpenPGP :

Comme je l'ai dit, il y a différents types de groupes [à l'IETF]. Il y a certains groupes où les gens sont juste heureux de bosser ensemble pour une nouvelle technologie et tout va bien. D'autres sont rassemblés parce qu'il y a de nouvelles choses à voir mais avec quelques conflits. Et finalement, il y a des groupes comme OpenPGP, avec de vieilles technologies. Et si tu veux proposer n'importe quel changement dans les emails, tu vas avoir ce genre de choses ; si tu proposes n'importe quel changement dans http, tu vas avoir ce genre de choses. (Membre du groupe de travail OpenPGP, entretien du 1^{er} juillet 2021)

Le « genre de choses » en question faisait référence à la lenteur et la complexité des négociations au sein du groupe de travail OpenPGP. Aucun changement qui affecte négativement une application déjà utilisée ne saurait être accepté tel quel. De manière isolée, un tel blocage est logique et acceptable. Toutefois, comme chaque implémentation a interprété à sa manière les spécifications, il était devenu très difficile de trouver des changements qui permettent à la fois d'avancer dans la direction du projet-sécurité des participant-e-s tout en restant compatible avec les anciennes versions du protocole.

En outre, l'enlissement infrastructurel permet de cerner une double relation d'interdépendance : les contraintes de l'infrastructure dans laquelle OpenPGP est utilisé enlissent le protocole, et l'inertie du protocole enlisse à son tour l'infrastructure en y apportant son lot de contraintes, comme le fait de devoir maintenir un algorithme de hachage obsolète dans le code de programmes de sécurité informatique. Autrement dit, plus le temps passe, plus il est difficile de remplacer les éléments vieillissants de l'infrastructure existante et plus cette même infrastructure contraint les possibles évolutions de ces mêmes éléments vieillissants. L'infrastructure ne peut être transformée que par incréments modulaires (Star et Ruhleder 2010) mais ces corrections incrémentales sont rendues encore plus difficiles à opérer du fait même de l'inertie de cette infrastructure. Dans le cas d'OpenPGP, cela signifie que non seulement le proto-

cole s'enlise dans une infrastructure dont l'inertie grandit (pensons par exemple aux emails dont on peut très difficilement se passer), mais lui-même participe à l'enlissement de cette infrastructure en proposant des outils ou des concepts qui s'enracinent dans le passé.

Ce que j'appelle l'« enlissement infrastructurel » correspond donc à ce double blocage où avancer ne peut se faire que lentement et petit à petit, et où bifurquer n'est plus une option car cela causerait trop de dégâts à l'infrastructure (par exemple en rompant la manière d'authentifier les paquets logiciels de la majorité des distributions Linux). L'exemple du remplacement de SHA-1 pour le calcul de l'empreinte de la clé et les exemples que je donne dans la dernière section de ce chapitre (4.3 p. 176) montre que l'enlissement n'est pas insurmontable – et heureusement, car comme je l'ai déjà suggéré, ne pas évoluer, c'est mourir. Avant de présenter ces exemples, j'aimerais toutefois m'arrêter sur deux conséquences de cet enlissement infrastructurel : l'usure qu'il cause et la prolifération de normes.

4.2.1. L'usure et la lenteur

À l'instar d'une roue prise dans un amas de sable, l'enlissement infrastructurel nécessite un arrêt complet, l'examen de la situation, et le bricolage d'une solution sur mesure pour repartir. Aucune procédure systématique et toujours efficace permet de faire face à ce genre de cas. L'analogie de la roue d'un véhicule prise dans une dune de sable fin qui s'emballe à tourner dans le vide en empirant toujours plus la situation met aussi en lumière un autre point important : le gaspillage d'énergie et l'usure causée. Outre les questions d'écologie, je parle également de l'énergie psychologique d'un tel travail. En effet, en discutant de cet échange d'email avec une codeuseuse d'OpenPGP durant un entretien, j'ai clairement senti sa grande lassitude, voire son exaspération, et ce autant dans ses propos que le ton et le rythme de sa voix :

Sylvain : Et est-ce que tu as des craintes par rapport au travail qui est fait dans ce groupe de travail ?

Codeuseuse : Non, pas vraiment. Je veux dire... bien sûr, je suis préoccupé-e par des suggestions comme celle de cette personne qui dit qu'il n'y a pas de problème avec SHA-1. Mais ensuite, je pense [long silence] ouais, je pense que ce

n'est toujours pas idéal mais cela n'a aucune conséquence pratique. La seule chose est juste que... je veux dire, d'abord, je pense que passer de SHA-1 à SHA-256 en 2021 ne devrait pas nécessiter de justification. Mais à part ça, je ne crois pas que ça va vraiment dans une mauvaise direction ; mais ça pourrait aller plus vite, et nous pourrions faire plus. Je ne sais pas par exemple si nous allons finir par avoir une nouvelle fonction de dérivation de clé. Je crois que [le président du groupe de travail] a dit quelque chose à propos d'en ajouter une, mais je ne sais pas si c'est dans le planning.

(Codeureuse d'une implémentation d'OpenPGP, entretien du 7 juin 2021)

Cette citation rend explicite deux aspects de l'enlisement : l'encastrement dans une idée du passé et l'incapacité à se projeter dans l'avenir de manière claire – ou du moins, la conscience de devoir « repousser » les échéances de son propre projet-sécurité. Pourtant, la confiance dans la volonté du groupe de travail d'aboutir à une normalisation des spécifications n'était pas entamée malgré la lenteur du processus. Sa frustration était liée à la lenteur du processus et la difficulté d'avancer. Ce point de vue est à souligner car il ajoute une nuance qui souvent disparaît dès lors que l'on s'éloigne du milieu où se font ces négociations. En effet, la plupart des critiques formulées publiquement à l'encontre d'OpenPGP considèrent simplement qu'OpenPGP appartient au passé et qu'il faut passer à autre chose. Or, la citation ci-dessus souligne implicitement qu'OpenPGP ne va pas disparaître car il y a non seulement des besoins mais aussi des idées pour faire évoluer le protocole. L'enlisement est, de ce point de vue, à mettre en rapport avec l'idée de Paula Bialski (2020) qui décrit le passé comme un frein qui ralentit la production, mais qui ne la bloque pas.

4.2.2. Faut-il remplacer OpenPGP et créer une nouvelle norme ?

Face à l'enlisement, deux postures semblent possibles. La première consiste à s'entêter à maintenir la norme en question, malgré l'usure que cela engendre et malgré les critiques. C'est cette approche que suit le groupe de travail de l'IETF. Elles a certes de nombreux inconvénients, mais elle présente également quelques avantages que j'aborde plus loin. La deuxième approche consiste à faire table rase et proposer un nouveau protocole qui soit plus adapté aux normes actuelles d'Internet – du moins c'est ce que l'on souhaite et suppose. C'est notamment ce qu'a proposé de faire Moxie Marlinspike en créant Signal : il a exprimé à plusieurs reprises son point

de vue à propos des changements de fond dans la manière de penser la sécurité d'Internet, préférant une architecture centralisée et réactive en cas de découvertes de nouvelles vulnérabilités à une architecture décentralisée et très lente (Marlinspike 2016). C'est aussi la solution choisie par l'entreprise d'email sécurisé Tutanota qui a développé sa propre solution cryptographique. L'entreprise donne trois raisons : la ligne de l'objet de l'email n'est pas chiffré avec OpenPGP, les algorithmes utilisés par OpenPGP ne peuvent pas être facilement mis à jour et ne sont pas résistants aux ordinateurs quantiques, et OpenPGP ne propose pas de solution pour la confidentialité persistante¹¹⁰. Ainsi, Tutanota a opté pour mettre au point sa propre solution.

En agissant ainsi, Signal et Tutanota se sont à la fois simplifié et compliqué la vie. En effet, mettre au point son propre système de chiffrement signifie ne pas bénéficier de l'expertise d'autres implémentations. Toutefois, n'étant plus dépendantes du choix des autres, il devient beaucoup plus facile de faire des mises à jour car il n'y a pas besoin de négocier chaque aspect des changements opérés au sein d'une communauté aux intérêts parfois divergents. Autrement dit, cela permet de s'aligner plus facilement à son propre projet-sécurité sans avoir à s'occuper de celui des autres¹¹¹. Cela permet aussi de vendre son expertise exclusive en proposant d'adapter un produit que l'on a créé à des entreprises qui en désirent des variations, comme Signal l'a notamment fait (Ermoshina et Musiani 2019). Toutefois, cette solution pose aussi des problèmes de taille. Par exemple, elle mène à la situation cocasse que m'a exposée une codeuse lors d'un entretien :

Je pense que ce serait possible de remplacer OpenPGP, mais ce serait un cas similaire à cette caricature XKCD à propos des normes¹¹². Tu as 14 normes concurrentes et toutes sont imparfaites. Alors tu en publies une nouvelle et bientôt, il y a 15 normes concurrentes. Je pense que c'est la même chose avec OpenPGP. Ce serait probablement possible d'aboutir à quelque chose de plus simple, de mieux, mais il y aurait toujours des gens qui n'en aimeraient pas certains aspects. Et d'une manière générale, PGP a prouvé qu'il était stable, qu'il était en grande partie sûr et qu'il pouvait être utilisé de manière sûre. Il y a beaucoup d'outils pour lui, il y a du matériel *hardware* qui lui est dédié, le cas des *Smart-*

110 <https://tutanota.com/fr/encryption>.

111 Sauf si, évidemment, avoir un protocole interopérable fait partie dudit projet-sécurité.

112 Voir l'illustration 17 ci-dessous.

*Card*¹¹³. Donc oui, il pourrait être remplacé, mais cela mènerait certainement à une sorte de segmentation. Je pense que probablement, un projet utiliserait cette solution et un autre projet laisserait tomber les signatures cryptographiques et n'utiliserait que https. Il y aurait sûrement une espèce de fragmentation, de segmentation de l'écosystème. Et ce serait triste, sans doute.

(Codeuseuse d'une implémentation d'OpenPGP, entretien du 26 mai 2021)



Illustration 17 : Comment les normes prolifèrent. Source: <https://xkcd.com/927/>.

Cette citation et cette caricature me paraissent très intéressantes car elles mettent en avant un aspect crucial d'une infrastructure enlisée. L'inertie du protocole et de l'infrastructure se nourrissent réciproquement. En effet, au vu de la difficulté que poserait sa disparition subite, il est improbable qu'OpenPGP cesse simplement d'exister. Au contraire, devenu un élément clé d'une infrastructure enlisée (et donc difficile à contourner en pratique), OpenPGP nécessite d'être maintenu et mis à jour. Ainsi, l'obligation morale de la maintenance dont Denis et Pontille parlent (2022) naît entre autre de cette impossibilité de contourner l'enlissement infrastructurel. Il s'agit donc d'une sorte de cercle vicieux : la maintenance d'un protocole est nécessaire parce que d'elle dépend la survie de ce protocole, et cette maintenance participe à enliser

113 Une *smartcard* est une carte à puce sécurisée qui permet de stocker des clés cryptographiques privées de manière sécurisée par un code PIN. Il peut s'agir d'une carte à puce standard qui ressemble à une carte de crédit, ou d'une clé USB. Le principe est que les opérations cryptographiques se passent sur la carte et non dans l'ordinateur. Ainsi, la clé secrète n'est jamais lue par l'ordinateur et encore moins stockée dans sa mémoire ; l'ordinateur peut seulement demander à la clé d'exécuter des opérations de déchiffrement ou de signature. C'est pourquoi les clés publiques ne sont pas accessibles via la *smartcard* et doivent être récupérées ailleurs, par exemple sur un serveur de clé. L'accès aux données de la carte est dit « *write-only* » est une protection supplémentaire importante car il rend impossible la copie de la clé privée.

davantage ledit protocole ainsi que, par ricochet, l'infrastructure. L'enlissement est donc une forme de garantie de survie : « trop enlisé pour disparaître »¹¹⁴. C'est en ce sens que je propose de prolonger cette réflexion sur l'enlissement infrastructurel en développant son pendant positif : l'ancrage. Dans la section suivante, je montre comment l'infrastructure peut être à la fois une source de contrainte et une source d'ancrage et donc d'opportunités pour les nouvelles technologies.

4.3. Ancrage et opportunités

En utilisant le terme « enlissement » pour décrire les contraintes qui pesaient sur l'infrastructure, j'ai surtout insisté sur la situation de blocage, et donc de « non-mouvement » qui résultait du vieillissement des technologies. J'aimerais maintenant renverser cette perspective et analyser comment le passé peut aussi être une source d'opportunités. Cela contribue à nuancer la dichotomie entre maintenance et innovation, entre anciennes technologies et nouvelles possibilités techniques. En effet, en parlant d'ancrage infrastructurel, je rappelle une propriété étonnante des normes technologiques : malgré les tempêtes, malgré les vagues de critiques qui ne cessent de déferler, et malgré les sables mouvants qui enlisent, les normes techniques ont tendance à persister d'elles-mêmes, un peu dans la logique de l'obstination décrite par Denis et Pontille (2022). Cela est particulièrement vrai concernant OpenPGP qui a jusqu'ici résisté à sa mort sans cesse annoncée. Mais cela est aussi vrai pour de nombreuses normes techniques comme le protocole IPv4 qui demeure, bien qu'une mise à jour existe depuis des années (DeNardis 2006). Malgré cela – ou peut-être grâce à cela –, un ancrage infrastructurel peut représenter de réelles opportunités d'innovation, même si les changements sont parfois modestes. La notion d'ancrage infrastructurel laisse apercevoir trois dimensions importantes du soin des choses numériques : le symbolisme de certaines entités et l'obligation morale qui en découle, l'évolution forcément lente des communs du numérique et les opportunités d'innovation.

114 À l'instar de l'argument « *too big to fail* » au sujet des banques d'importance systémique. D'ailleurs, en anglais, la phrase « *too stuck to fail* » me semble très *catchy*.

4.3.1. Symbole et obligation morale

L'ancrage infrastructurel peut s'observer de multiples façons. Dans certains cas par exemple, il peut s'agir d'un élément qui pousse à s'intéresser à un protocole non pas à cause de ce protocole, mais à cause d'un autre élément de l'infrastructure qui repose sur ce protocole. C'est le cas notamment d'une codeuseuse que j'ai interviewée et qui travaillait pour une entreprise qui vendait des *smartcards*. Cette personne m'a raconté que son intérêt pour OpenPGP lui venait de son engagement bénévole pour la fondation Wikipédia qui utilisaient OpenPGP pour authentifier ses membres. Par ailleurs, la manière dont cette codeuseuse s'est procuré sa première *smartcard* est révélatrice :

En fait, je suis devenu·e membre de la Free Software Foundation Europe parce que, bien sûr, j'aime ce qu'ils font, mais aussi parce qu'ils offraient aux nouveaux membres une *smartcard* OpenPGP. C'est la raison pour laquelle j'ai décidé de devenir membre.

(Codeuseuse d'une implémentation d'OpenPGP, entretien du 26 mai 2021)

Il ne me semble pas anodin que la fondation Free Software Foundation Europe attirent de nouvelles personnes en offrant à ses membres un symbole fort. Cela résonne particulièrement avec une citation d'un·e professeur·e engagé·e partiellement dans le groupe de travail de l'IETF qui affirme justement qu'OpenPGP est aussi un symbole fort dans certaines représentations d'Internet en tant qu'infrastructure décentralisée, libre et open source :

Je pense que, oui, [les groupes de travail OpenPGP et PKIX¹¹⁵] sont attachés à leur écosystème, mais je pense que la vraie, la véritable raison pour laquelle ces choses ne sont pas parties, c'est qu'elles font maintenant partie d'une culture, et qu'elles ont en fait été conçues pour refléter le type de culture sur lequel elles travaillaient.

(Codeuseuse d'une implémentation d'OpenPGP, entretien du 26 mai 2021)

Cette citation permet d'aller au-delà de l'anecdote de la citation précédente en soulignant l'importance culturelle et symbolique d'OpenPGP. C'est bien plus qu'un simple protocole technique. Il s'agit d'un symbole fédérateur et rassembleur sur lequel de nombreuses personnes projettent différentes choses. Les participant·e·s à

115 Le PKIX est un groupe de travail de l'IETF dédié aux infrastructures à clés publiques.

ma recherche seraient probablement toustes d'accord sur le fait qu'OpenPGP n'est pas une norme comme les autres de par son passé et ce qu'elle représente. Malgré ses défauts et les problèmes qu'elle pose, cette norme continue d'alimenter l'infrastructure d'Internet, elle continue d'être maintenue et d'être développée dans diverses directions parce qu'il s'agit d'un élément fédérateur. Cela explique aussi les raisons de la survie d'OpenPGP : étant un symbole fédérateur au sein d'une infrastructure enlisée, il *doit* être maintenu. Il n'y a naturellement pas d'obligation formelle, mais une obligation morale importante. En s'ancrant dans une infrastructure qui l'utilise, OpenPGP est un acte d'intéressement d'un grand nombre de personnes. Du point de vue des communs du numérique, cela a une importance majeure. Par cet acte d'intéressement, un protocole enlisé se donne les conditions nécessaires à sa maintenance : une foule de mainteneureuses prêtes à en prendre soin. Cet aspect est approfondi dans la sous-section suivante qui montre un deuxième avantage des technologies enlisées.

4.3.2. Évolution plutôt que révolution

L'une des questions qui me taraudait le plus lorsque j'ai commencé à collaborer avec l'équipe de Sequoia au début de ma recherche était liée à un paradoxe que je n'arrivais pas à comprendre : malgré les nombreuses et redondantes critiques à l'égard de la norme, l'équipe de développement de Sequoia avait décidé en 2017 de développer une nouvelle implémentation d'OpenPGP pour faire directement concurrence à l'implémentation la plus implantée dans l'écosystème, GnuPG. Je voulais tenter de comprendre pourquoi il leur semblait si important d'« améliorer l'écosystème OpenPGP » alors que beaucoup d'expert·e·s affirmaient que le protocole serait mieux mort que vivant. La réponse de l'un des fondateurs est très parlante :

L'âge [du protocole] n'est pas important. Je dirais que simplement créer un nouveau protocole pour créer quelque chose de neuf n'est pas la bonne solution pour aller de l'avant. Et si tu regardes l'Internet, il se base sur des protocoles qui ont une longue histoire. Regarde TCP/IP, regarde SMTP. Est-ce qu'ils sont parfaits pour l'Internet d'aujourd'hui ? Non, mais ils sont largement déployés. Et quand je dis « largement déployés », je dis qu'ils sont déployés par tout le monde ; même Google et Facebook, qui aimeraient bien avoir leurs propres normes pro-

priétaires, les utilisent. Tout le monde utilise ces choses et elles ne vont pas disparaître. Elles pourraient être meilleures, c'est sûr, mais elles ne sont pas horribles. Donc je dirais qu'il vaut mieux une évolution plutôt qu'une révolution. (Codeureuse d'une implémentation d'OpenPGP, entretien du 24 juillet 2020)

Tous les protocoles mentionnés dans cette citation (OpenPGP, TCP/IP, SMTP¹¹⁶) ont vieilli ; ils ont pu représenter un jour une révolution – c'est certainement le cas d'OpenPGP – mais depuis, ils se sont enlisés et n'ont continué d'évoluer que lentement. Toutefois, ils n'appartiennent pas au passé mais au présent. Ce dernier point est peut-être la clé pour comprendre le paradoxe mentionné ci-dessus : le fait est qu'OpenPGP fonctionne, c'est-à-dire que le protocole est utilisé pour des opérations variées, de l'authentification de paquets Linux au chiffrement d'emails en passant par le chiffrement d'archives et la messagerie instantanée DeltaChat. Le protocole n'est pas parfait et présente une importante marge de progrès, certes ; mais de nombreuses personnes l'utilisent de manière régulière. D'ailleurs, le scandale lié à la manière dont les vulnérabilités d'EFAIL ont été révélées publiquement a montré que l'usage de ce protocole n'était pas réservé aux seul·e·s *geeks* mais que de nombreuses personnes l'utilisaient. De ce point de vue, on peut dire que ce n'est pas l'infrastructure qui a été rendue visible par la défaillance provoquée par EFAIL mais bien son utilisation qui dépassait largement ce que certaines personnes imaginaient. Comme le disait le·a cryptographe que j'ai cité·e en début de chapitre, le gens utilisent OpenPGP, et parce que les gens l'utilisent, OpenPGP est maintenu.

4.3.3. Ancrage et innovation

Maintenir une norme numérique telle OpenPGP, c'est la faire co-évoluer avec l'écosystème mais aussi en fonction des attentes et des différents projets-sécurité des ac-

116 Le *Transmission Control Protocol* (TCP) date de 1973 et les prémisses de l'*Internet Protocol* (IP) datent de l'année suivante. Le RFC 1122 est le document de référence pour la suite des protocoles Internet (y compris le protocole TCP/IP) : <https://datatracker.ietf.org/doc/html/rfc1122>. Il est intéressant ici de rappeler les discussions qui ont cours à l'IETF depuis une trentaine d'années pour remplacer IPv4 (DeNardis 2009). Le *Simple Mail Transfer Protocol* (SMTP) date quant à lui des années 1980 et c'est le RFC 772 qui fait référence en la matière : <https://datatracker.ietf.org/doc/html/rfc772>.

teurices impliquées dans cette maintenance. Il y a d'une part un mouvement vers l'avenir qui contrecarre l'enlisement¹¹⁷, et d'autre part un ancrage sociotechnique dans l'infrastructure qui consolide cette construction (tout en la contraignant). J'aimerais donner ici deux exemples qui montrent comment deux développements ont récemment fait évoluer les possibilités offertes par OpenPGP dans deux directions opposées. Ces deux exemples montrent que l'enlisement n'est automatiquement synonyme de paralysie mais qu'au contraire, l'ancrage dans une infrastructure établie peut représenter de réelles opportunités.

Exemples d'innovation dans l'infrastructure du Web of Trust

1. Autocrypt. Autocrypt est une norme ouverte et gratuite qui cherche à offrir une alternative au Web of Trust en simplifiant l'échange des clés et le chiffrement. La version 1.0 a été publiée en 2017 et Autocrypt est aujourd'hui (printemps 2023) une norme implémentée par différents clients email et applications de *chat-over-email*¹¹⁸. Le but de ce protocole est de rendre le plus transparent possible l'usage d'OpenPGP pour les utilisatrices. Le site Internet l'affirme :

« Pour les utilisatrices, Autocrypt Level 1 offre un chiffrement en un seul clic, facilite les communications chiffrées au sein d'un groupe et permet de configurer le chiffrement sur plusieurs appareils¹¹⁹.

Autocrypt cherche ainsi à contourner les étapes fastidieuses de la recherche et de la vérification des clés publiques en attachant celles-ci directement aux emails envoyés. Dès le deuxième message échangé, la conversation est chiffrée sans qu'aucune action ne soit nécessaire. Bien entendu, le fait qu'il ne soit pas demandé aux correspondant·e·s de vérifier l'authenticité de la clé est un problème de taille dans certains cas, et plusieurs personnes considèrent que l'authentification des clés est une étape incontournable pour un bon ni-

117 Cette projection vers l'avenir est analysée plus en détail dans la dernière partie de cette thèse (chapitre 8 « Penser les fragilités à venir pour panser l'avenir », p. 300).

118 Une liste de ces implémentations peut être consultée sur le site Internet du protocole : <https://autocrypt.org/dev-status.html>.

119 <https://autocrypt.org/>, ma traduction.

veau de sécurité (Walfield 2021) : en effet, on peut aisément imaginé que dans certain cas, ce manque de vérification peut causer du tort et notamment envoyer un message chiffré avec la mauvaise clé. Cependant, plusieurs participant·e·s à ce projet m'ont affirmé que leur but n'était pas de minimiser le niveau de sécurité mais bien de renforcer considérablement le nombre d'emails chiffrés en simplifiant au maximum le mode opératoire. Par ailleurs, Autocrypt n'empêche pas la vérification des clés mais en fait une étape optionnelle. Il me paraît important ici de citer un·e des personnes qui a géré le projet dès ses premiers instants :

Autocrypt, le protocole dans lequel j'étais impliqué·e, a commencé avec une série de sessions annexes. Il y a eu une année, je crois que c'était en 2016, où il y a eu plusieurs événements OpenPGP durant lesquels j'ai rencontré les mêmes personnes. Et à côtés des présentations, on a eu quelques discussions. [...] Et l'idée était vraiment très simple : ajouter un entête aux emails¹²⁰ – tu connais tout à ce sujet. Et la chose intéressante est que même si cette idée était très basique, il a fallu plusieurs expert·e·s qui savaient vraiment bien comment faire les choses : différent·e·s implémenteuses. Je dirais qu'il a fallu une centaine d'heures pour comprendre vraiment les détails précis, le fonctionnement précis, pour faire en sorte que ce soit interopérable. C'est pour ça que si tu compares juste une chose avec une autre, c'est une chose. Mais Autocrypt n'est pas juste une idée, c'est une spécification. Matrix également. Ils ont beaucoup, beaucoup de protocoles. Et Signal, WhatsApp, etc, aussi Threema de ce que je connais ; aucun n'offre cette interopérabilité, ce qui n'est pas facile à faire et habituellement, il y a plus de sens à l'implémenter quand il y a au moins 2 implémentations différentes. Parce qu'autrement, il n'y a pas besoin de discussion. Mais si tu as plusieurs implémentations, alors évidemment, elles ont des opinions différentes, elles voient les choses sous un autre angles, leur programmes emails font les choses à leur façon ; et alors, il y a de nombreuses discussions. (Codeureuse et conceptrice d'Autocrypt, entretien du 26 août 2021)

Cette citation est importante car elle souligne deux choses cruciales pour comprendre l'enlissement infrastructurel et la manière dont on peut parfois le dépasser. La première est qu'une idée n'est jamais simple dans ce genre de

120 Autocrypt ajoute en effet la clé publique ainsi que quelques autres informations dans les entêtes des emails. Ces entêtes ne font pas partie du corps du message que le·a·récipiendaire va lire ; ils contiennent généralement des métadonnées comme les champs d'envoi (destinataire / cc / cci), des informations sur les clients email, et autres.

contexte car il faut non seulement comprendre comment les choses fonctionnent et comment les changer pour qu'elles continuent de fonctionner en parallèle de l'innovation proposée. La centaine d'heures que mentionne cette développeuse est un indice révélateur de l'importance de comprendre finement tous les mécanismes en jeu. Cela nécessite une grande expertise, de nombreuses discussions et surtout un esprit suffisamment créatif pour trouver une bonne solution indolore et efficace.

2. OpenPGP CA. Ce deuxième exemple concerne une autre solution pour faciliter l'authentification des clés. L'équipe de Sequoia a développé un système d'autorité de certification (CA) pour faciliter la certification des clés OpenPGP d'une organisation¹²¹. Cette idée peut paraître étrange à cause de l'incompatibilité idéologique entre le système décentralisé d'OpenPGP et du Web of Trust et l'idée même d'autorité de certification. Par ailleurs, le protocole de chiffrement d'email concurrent à OpenPGP, S/MIME, largement plus utilisé dans les milieux des entreprises, propose déjà une centralisation avec autorité de certification. Mais une difficulté de taille existe : dans le cas de S/MIME, ce ne sont pas les utilisatrices qui créent leur propre paire de clés, mais l'autorité de certification. Cela facilite la gestion des clés pour l'organisation, et facilite la découverte et l'authentification des clés publiques. Cependant, l'inconvénient pour les utilisatrices est l'impossibilité de créer et configurer leur propre paire de clés qui soit réellement privée. L'équipe de Sequoia a ainsi « hacké » à la fois l'infrastructure et la norme en proposant un système qui reste compatible avec l'indépendance d'OpenPGP – les utilisatrices gèrent elles-mêmes leurs clés – et qui permette en même temps une centralisation qui facilite la recherche et l'authentification de ces dernières. L'autorité de certification d'OpenPGP CA a pour mission de vérifier que les propriétaires des clés sont bien les personnes prétendues, mais n'interfère ni dans la création des clés, ni dans leur gestion.

121 Il s'agit du projet OpenPGP CA : voir par exemple le billet de blog « Hello, OpenPGP CA » : <https://sequoia-pgp.org/blog/2021/05/12/202105-hello-openpgp-ca/>.

Ce système offre ainsi une ouverture intéressante à l'idée de Web of Trust sans pour autant l'annuler ou la contraindre. En effet, OpenPGP peut sans autre s'insérer dans l'infrastructure actuelle qui n'est pas affectée *a priori* par OpenPGP CA. OpenPGP CA peut même être utilisé en parallèle à l'usage classique des serveurs de clés publiques, par exemple. À terme¹²², OpenPGP CA pourra avoir des impacts sur le fonctionnement de l'infrastructure, par exemple en offrant une alternative fonctionnelle et plus aisée de trouver les clés publiques de membres d'organisation sans recourir aux serveurs traditionnels de clés du type SKS¹²³. Mais OpenPGP CA ne nécessite aucun changement sur cette infrastructure pour pouvoir être déployé.

Ces deux exemples d'innovation grâce à un fort ancrage dans une infrastructure enlisée nous apprennent deux choses importantes. Premièrement, ils montrent que de vieux protocoles peuvent évoluer tout en restant les mêmes – ce qui est probablement une caractéristique propre des technologies numériques. En effet, malgré l'ajout de nouvelles extensions, la technologie demeure tout à fait compatible avec le fonctionnement « normal » d'OpenPGP. Pour celles et ceux qui n'utilisent pas ces nouvelles fonctionnalités, OpenPGP reste totalement identique. Pour les autres, ces nouvelles fonctionnalités font largement évoluer l'usage de la technologie. Il n'y a donc pas de révolution mais une lente évolution qui passe par l'adoption ou non de nouvelles fonctionnalités. Ainsi, le protocole lui-même devient un objet byzantin, à l'image de la caricature XKCD de l'illustration 16 (p. 164).

Deuxièmement, au lieu de mourir, le protocole évolue mais sans changer ses fonctionnalités de base. Ainsi, une part de son développement est enlisée dans le passé et ne peut que difficilement se dépêtrer de certaines breloques. Mais d'un autre côté, le protocole peut aussi s'appuyer sur cette stabilité pour proposer des prolongements

122

123 Les serveurs SKS sont des serveurs OpenPGP dont l'objectif est de fournir une synchronisation fiable, décentralisée et facile à déployer. En juin 2019, une attaque massive contre ces serveurs de clés a fait la une des sites spécialisés qui ont été plusieurs à considérer ces serveurs comme morts, allant jusqu'à écrire « RIP SKS. It's been pretty good, SKS keyserver network. Rest in peace » (Lee 2019; voir aussi R. J. Hansen 2019; Spring 2019; Böck 2021).

dans différentes directions. Comme je l'analyse dans le chapitre suivant, ces prolongements ont pour corollaire d'augmenter toujours plus la complexité du protocole à cause des nouvelles fonctionnalités qui, une fois qu'elles sont utilisées, doivent également être maintenues. Ce chapitre a montré que loin de n'être qu'un handicap pour la sécurité informatique, l'enlisement infrastructurel est ce qui permet justement aux normes techniques de survivre dans un domaine où les développements techno-scientifiques se font de manière très rapide. C'est en s'appuyant sur cette stabilité de l'infrastructure qui ne peut évoluer que très lentement et de manière incrémentale que les développeuses peuvent proposer des innovations qui se rapprochent de leurs projets-sécurité.

En guise de conclusion à ce chapitre, j'aimerais revenir sur les critiques avec lesquelles je l'ai ouvert. Ces critiques affirment qu'OpenPGP est un vieux protocole, enclin à d'étranges bogues et dont la cohésion tient avant tout à sa compatibilité avec les systèmes hérités. Ces critiques sont à prendre au sérieux car elles pointent des vulnérabilités importantes du standard. Toutefois, le parcours proposé par ce chapitre à travers les fonctions infrastructurelles d'OpenPGP montre qu'au-delà des risques d'un écosystème embourbé dans des pratiques passéistes, c'est justement cet enlisement non seulement des normes mais également des usages qui est à la base de la survie des technologies numériques. L'attachement à OpenPGP des personnes qui en prennent soin n'est pas uniquement émotionnel, comme le prétend Mathew Green en parlant d'« une espèce d'obsession avec l'idée de PGP comme une fin en soi » (Green 2018, ma traduction). Certes, beaucoup de ces mainteneuses sont attachées à OpenPGP émotionnellement, mais ce chapitre a aussi montré un devoir moral à perpétuer certains usages de l'infrastructure. Autrement dit, ce n'est pas uniquement par amour que ces codeuses ont à cœur de faire évoluer le standard, mais c'est pour garantir certains usages actuels. Ainsi, maintenir OpenPGP ne signifie pas figer des pratiques découlant directement de l'Internet des années 1990 mais de s'attacher à apporter de nouvelles solutions tout en assurant le fonctionnement des acquis. Les transformations du protocole depuis son invention ont montré une im-

pressionnante capacité d'adaptation du standard. Dans le prolongement de ces réflexions, le chapitre suivant est dédié au soin du code informatique et à la manière dont le code est à la fois le véhicule de l'héritage du passé et de ce que l'on transmet aux générations futures.

5. Hériter, faire vieillir et transmettre le code

Le chapitre précédent a montré que l'infrastructure ne pouvait évoluer que lentement, ce qui représentait de nombreuses contraintes d'un point de vue de la sécurité, mais également quelques opportunités intéressantes. Ce chapitre prolonge la réflexion à propos du poids du passé en se focalisant non plus sur l'infrastructure en général mais sur le code informatique. Il nourrit trois objectifs. D'une part, il apporte un éclaircissement sur l'architecture et le fonctionnement du code informatique, informations qui sont cruciales pour comprendre en quoi consiste la maintenance de ce code informatique. D'autre part, il contribue à une réflexion plus large au sein des études sur la maintenance en pointant du doigt le dialogue qui s'opère entre les développeuses du passé, celles du présent et celles de l'avenir. Finalement, ce chapitre est également l'occasion de réfléchir de manière à ce que vieillir veut dire pour une technologie du numérique. Il montre en particulier comment le code, objet inerte et insensible au passage du temps, tend à l'immobilité. Or, s'il cesse d'avancer en même temps que l'écosystème dans lequel il s'insère, il perd sa pertinence. Le travail de maintenance consiste donc à maintenir le code en mouvement pour entretenir sa pertinence.

Parmi les nombreuses difficultés auxquelles font face les mainteneuses d'un logiciel informatique se trouve la gestion du code hérité, appelé « *legacy code* » en anglais. Il existe de nombreuses définitions de ce terme mais elles s'accordent généralement sur le fait que le code hérité est un fardeau car il n'est pas écrit selon

les normes « actuelles ». Plusieurs livre sur le *legacy code* le définissent de manière pragmatique en fonction d'une caractéristique technique : l'absence de *unit tests*¹²⁴ (Feathers 2005; Hardy 2021). Les *unit tests* sont des tests intégrés au sein du code qui permettent de vérifier que les différentes parties du code restent fonctionnelles et cohérentes entre elles. Les *unit tests* permettent ainsi de vérifier qu'un changement dans une partie du code ne compromette pas d'autres parties du code. Bien qu'apparemment essentiels, les *unit tests* sont fastidieux à coder, ce qui explique peut-être pourquoi les intégrer au code n'a pas toujours fait partie des bonnes pratiques du codage. Mais l'expression « *legacy code* » est aussi abondamment utilisée par des codeuses que j'ai rencontrées lors de ma recherche pour exprimer le fait qu'ils et elles n'aiment pas ce code hérité du passé, ou qu'ils et elles ne le comprennent pas, ou encore qu'ils et elles en ont peur car le modifier peut causer des bogues aussi divers qu'inattendus¹²⁵.

Dans certains cas, le code paraît si vieux, si difficile à comprendre et si pénible à maintenir qu'il est tentant de le réécrire complètement « *from scratch* », à partir de zéro. Dans un fameux billet de blog, Joel Spolsky (2000) qualifie cette idée comme étant « la pire erreur stratégique qu'une entreprise puisse faire ». Dans un style coloré et humoristique, il propose un argument crucial que de nombreuses personnes ont également évoqué lors de discussions à ce sujet :

L'idée que le nouveau code est meilleur que l'ancien est complètement absurde. L'ancien code a été utilisé. Il a été testé. De nombreux problèmes ont été découverts et corrigés. Il n'y a rien qui cloche avec lui. Il n'acquiert pas de bogues simplement en restant sur votre disque dur. *Au contraire*¹²⁶, bébé ! Les logiciels sont-ils censés être comme une vieille Dodge Dart qui rouille rien qu'en restant dans le garage ? Les logiciels sont-ils comme un ours en peluche qui est un peu répugnant s'il n'est pas fait d'un matériau neuf ? (Spolsky 2000, ma traduction)

Dans ce billet, Spolsky insiste particulièrement sur tout le savoir incorporé dans le code hérité dont la complexité est justifiée par le fait qu'il contient tous les correctifs

124 Des *unit tests* sont des tests intégrés au sein du code et qui permettent de vérifier que les différentes parties du code restent fonctionnelles et cohérentes après des changements.

125 Ces différentes nuances sont très bien présentées dans un billet de blog d'Eli Lopian (2018).

126 En français dans l'original.

de bogues, ce qui représente des années de travail de programmation et de maintenance. Il souligne également les risques d'un point de vue commercial car réécrire un code prend du temps et cela empêche l'évolution du logiciel, donnant ainsi un avantage important aux entreprises concurrentes. Sa perspective va dans le même sens que celles d'auteurs comme Feathers (2005) et Hardy (2021) qui encouragent à remanier [*refactoring*] le code plutôt que de le réécrire complètement.

Le témoignage de Werner Koch, fondateur de GnuPG, semble particulièrement bien représenter cette attitude envers l'héritage du passé. Celui qui, durant des années, a porté seul la seule implémentation complète, libre et gratuite d'OpenPGP, a une vision très positive du code hérité. Lors d'un entretien, il a insisté sur le fait que c'était tout à fait normal d'avoir un code complexe pour un programme qui a été maintenu et corrigé pendant des années. Lors de mes observations et mes entretiens avec lui, j'ai été frappé par son immense connaissance de l'historique des changements opérés. Quand on évoquait un paramètre, un algorithme ou un bogue, il avait en effet l'habitude d'expliquer quand avait été introduit telle fonctionnalité, pourquoi telle décision avait été prise et quels avaient été les changements opérés. Ce faisant, il dévoilait la mémoire du code et son histoire de négociation, de compromis, d'essais et d'erreurs. Ainsi, pour lui, la référence au passé est un élément positif. Dans sa perspective, vieillir, ce n'est pas un synonyme de décadence, mais de maturité. De même que dans certaines cultures, des personnes âgées sont considérées comme des bibliothèques vivantes (et non des êtres inutiles et séniles), le code hérité est parfois perçu comme un réservoir de savoir qui offre fiabilité, solidité et sécurité :

D'un point de vue des affaires, [réécrire le code à partir de zéro] n'est pas une bonne idée. Il y a beaucoup de maturité dans le code, ce qui n'est pas facile à remplacer. C'est quelque chose que tu apprends avec l'expérience dans la conception de logiciel : tu ne peux pas tout réécrire. Tu auras des défaillances du système et ce n'est pas bien. Nous avons aussi une communauté établie et de nombreuses utilisatrices dépendent de ce logiciel pour travailler.
(Werner Koch, entretien du 30 juillet 2020)

Réécrire le code à partir de zéro, c'est pourtant ce qu'ont décidé de faire les trois co-fondateurs de Sequoia en 2017 alors qu'ils travaillaient tous les trois pour GnuPG

à l'époque. La version 1.0 de Sequoia a été publiée en décembre 2020. Également en 2017, rPGP, une autre librairie en RUST, est développée pour l'application de messagerie instantanée DeltaChat dont la version 1.0 a été publiée en décembre 2019¹²⁷. En 2020, la nouvelle version de Thunderbird (78.0) a remplacé l'extension Enigmail¹²⁸ par une implémentation native d'OpenPGP¹²⁹. En décembre 2021, la version 1.0 de PG-Painless, une implémentation en Python, a également été publiée et est notamment utilisée par FlowCrypt¹³⁰. Bien sûr, chacune de ces implémentations a une raison d'être différente et répond à des besoins précis. Cela fait toutefois beaucoup de nouvelles implémentations pour un protocole que beaucoup considèrent comme trop vieux et désuet (Green 2014; Katwala 2018; Latacora 2019; Wong 2020). Comment interpréter cette effervescence ? Cela signifie-t-il simplement que les nouvelles implémentations viennent combler des lacunes dans les besoins actuels de l'infrastructure, ou qu'il existe une véritable impasse avec GnuPG et sa librairie GPGME ?

Ce chapitre vise à analyser ce que le temps fait au code et, partant, ce que le code permet de comprendre quant au travail de la sécurité. J'y analyse le lien entre l'âge du code, l'augmentation de sa complexité et les tensions causées par l'adoption de nouveaux projets-sécurité. Pour ce faire, il me semble important de bien comprendre ce qu'est le code informatique, ce que je présente dans la première section du chapitre en donnant quelques clés de compréhension à propos de son fonctionnement, son architecture et le choix du langage utilisé. Puis, je dédie la section suivante à une analyse de ce que fait la sécurité au code informatique¹³¹. Je questionne notamment

127 <https://github.com/rpgp/rpgp> et <https://delta.chat/fr/>.

128 Enigmail utilisait la librairie GPGME de GnuPG, alors que la nouvelle implémentation de Thunderbird utilise la librairie RNP.

129 <https://support.mozilla.org/en-US/kb/openpgp-thunderbird-howto-and-faq>.

130 Une extension pour utiliser OpenPGP dans gmail : <https://flowcrypt.com/>.

131 La deuxième partie, et notamment le chapitre 6 « EFAIL et le temps de l'urgence » (p. 226), analyse la situation inverse, à savoir ce que le code informatique (sous la forme de vulnérabilité) fait à la sécurité. Par ailleurs, plusieurs études ont montré comment les technologies numériques compromettent la sécurité informatique sous la forme de virus et de *malware* (Balzacq et Dunn Cavelty 2016; Stevens 2020), sous la forme de logiciel technologie d'espionnage (Diffie et Landau 2010) ou de surveillance de masse à des fins de sécurité nationale ou économiques (Lyon 2018; Zuboff

la complexité croissante du code et le fait qu'il devienne toujours plus difficile à comprendre pour les personnes qui s'en occupent. Dans la dernière section, j'analyse la manière dont le code passe du statut d'un objet plein de défauts hérités du passé à un objet de sollicitude que l'on s'applique à faire vieillir du mieux possible afin de préparer l'avenir. Je reviens sur la question de savoir ce que vieillir veut dire pour une technologie du numérique et je propose de voir les pratiques d'écriture, mais surtout de lecture, de réécriture et de relecture du code comme des pratiques de soin qui produisent de manière quotidienne voire parfois routinière la sécurité informatique.

5.1. Composer le code source

Le code informatique est souvent représenté de manière fantasque et extraordinaire, des représentations à *la Matrix* constituées de symboles verts sur fond noir aux écrans couverts de caractères multicolores d'une page HTML, en passant par des enchaînements infinis de 0 et de 1 (voir illustration 13 p. 117). Toutefois, ces images ne représentent que très mal ce qu'est et fait le code informatique. Surtout, elles ne permettent pas de rendre compte de la manière dont les codeuses façonnent et travaillent avec cet objet. J'ai déjà expliqué dans l'encadré « Les mots pour le dire : norme, protocole, code, programme » (p. 61) les différences entre code source et code binaire. J'aimerais maintenant préciser un peu les choses du côté du code source en m'arrêtant sur la manière dont il est écrit, organisé, relu et réécrit.

Commençons donc par le début. Il se présente généralement sous la forme d'un document texte, souvent dans une police monospace avec différentes couleurs selon le type d'information afin de faciliter la lecture. Il comprend à la fois des instructions destinées à être interprétées par une machine et des informations adressées aux humains qui liront et peut-être transformeront le code en question. Dans cette section, j'analyse l'exemple d'un fragment du code de Sequoia afin de montrer concrètement comment s'organise un code source typique. Cela permet de comprendre ce qui est nécessaire de faire pour en prendre soin.

2020).

L'illustration 18 ci-dessous donne un exemple de code source. Elle correspond à la simplification d'un module réellement existant du programme en ligne de commande SQ de Sequoia, mais qui a été simplifiée pour l'occasion¹³². Ce module permet d'analyser et formater une indication de date et d'heure selon la norme ISO 8601¹³³, par exemple pour déterminer une date d'expiration d'une nouvelle clé publique (voir illustration 19 p. 195). J'utilise l'exemple de cette fonction simplifiée `parse_iso8601` comme illustration de code source pour en souligner la structure, la logique et le fonctionnement.

132 Le code source du programme SQ peut être consulté sur la page GitLab du projet Sequoia, dont une copie sauvegardée du 6 mars 2023 existe ici : <https://web.archive.org/web/20230306065207/https://gitlab.com/sequoia-pgp/sequoia-sq/-/blob/main/src/sq.rs>. Sur cette sauvegarde, la fonction originale se trouve entre les lignes 750 et 818 du code. La fonction simplifiée est fonctionnelle et peut être testée sur le site du langage informatique RUST : <https://play.rust-lang.org/?version=stable&mode=debug&edition=2021&gist=68de83f2de25bbb0eb5814a952622ef0>.

133 <https://www.iso.org/fr/iso-8601-date-and-time-format.html>.

```

1 use chrono::{DateTime, NaiveDate, NaiveDateTime, NaiveTime, Utc}; // For date and time handling.
2 use anyhow::{anyhow, Result}; // For error handling.
3
4 /// Parses the given string depicting a ISO 8601 timestamp.
5 ///
6 /// If the given timestamp 'input' does not include a time of day,
7 /// this function returns the date given in 'input' combined with the
8 /// time of day given in 'time_of_day'.
9 pub fn parse_iso8601(input: &str, time_of_day: NaiveTime)
10     -> Result<DateTime<Utc>>
11 {
12     // Formats including the time of day.
13     for &format in &[
14         "%Y-%m-%dT%H:%M:%S%#z",
15         "%Y-%m-%dT%H:%M:%S",
16         "%Y-%m-%dT%H:%M",
17         "%Y-%m-%dT%H",
18         // Some formats omitted for clarity.
19     ] {
20         if format.ends_with("%#z") {
21             // Format includes a time zone.
22             if let Ok(d) = DateTime::parse_from_str(input, format) {
23                 // Success!
24                 return Ok(d.into());
25             }
26         } else {
27             // Format does not include a time zone.
28             if let Ok(d) = NaiveDateTime::parse_from_str(input, format) {
29                 // Success!
30                 return Ok(DateTime::from_utc(d, Utc));
31             }
32         }
33     }
34
35     // Formats with just the date.
36     for &format in &[
37         "%Y-%m-%d",
38         "%Y-%m",
39         "%Y",
40         // Some formats omitted for clarity.
41     ] {
42         if let Ok(d) = NaiveDate::parse_from_str(input, format) {
43             // Success!
44             return Ok(DateTime::from_utc(d.and_time(time_of_day), Utc));
45         }
46     }
47
48     // Failed to parse the given timestamp.
49     Err(anyhow!("Malformed ISO8601 timestamp: {}", input))
50 }
51
52 #[test]
53 fn test_parse_iso8601() -> Result<> {
54     let z = NaiveTime::from_hms_opt(0, 0, 0)
55         .expect("this to be a valid time of day");
56     parse_iso8601("2017-03-04T13:25:35Z", z)?;
57     parse_iso8601("2017-03-04T13:25:35+08:30", z)?;
58     parse_iso8601("2017-03-04T13:25:35", z)?;
59     parse_iso8601("2017-03-04T13:25", z)?;
60     parse_iso8601("2017-03-04", z)?;
61     // Some tests omitted for clarity.
62     Ok(()) // Tests completed successfully.
63 }

```

Illustration 18 : Exemple de code source écrit en RUST. Cette fonction est une simplification d'une réelle fonction du programme en ligne de commande SQ de Sequoia. Les différentes couleurs reprennent un code couleur classique : le texte en noir correspond au code lui-même, les variables sont en rouges et vert, et les commentaires sont en gris. Cette fonction a pour but de formater correctement l'indication d'une date et heure selon la norme ISO8601.

Source : capture d'écran de la page de test de code RUST : <https://play.rust-lang.org/?version=stable&mode=debug&edition=2021&gist=68de83f2de25bbb0eb5814a952622ef0>.

Anatomie d'un module écrit en RUST : exemple de la fonction **parse_iso8601**

L'exemple de l'illustration 18 est écrit en RUST. Des différences importantes existent entre les différents langages informatiques. Toutefois, ce que je présente dans cet encadré représente des généralités qui se retrouvent dans de nombreux codes sources quel que soit le langage informatique utilisé.

On notera tout d'abord que chaque ligne est numérotée, ce qui permet de mieux s'orienter dans le texte. Par ailleurs, certains mots sont mis en évidence (gras, couleur). Cette mise en page est générée automatiquement par le programme utilisé pour coder et elle sert à repérer rapidement certaines informations. Ainsi, les éléments en gras permettent par exemple de repérer la syntaxe du code rapidement. Ils soulignent en effet le type d'instructions qui sont données. On repère ainsi plus aisément la structure du code et les différentes fonctions présentées. Par exemple, les deux premières lignes commencent par le mot **use** en gras, ce qui indique que deux bibliothèques externes sont nécessaires pour ce module. Chacune de ces deux premières lignes est suivie d'un commentaire indiqué par les doubles barres **//** et représenté en gris dans l'exemple. Ils sont uniquement destinés aux humains qui lisent le code. Lorsqu'ils sont saisis de la sorte au milieu d'une ligne de code, ces commentaires donnent des informations sur la ligne en question. Lorsqu'ils sont en début de ligne, comme c'est le cas aux 4 à 8 de l'exemple, ils sont précédés d'une triple barre **///** et commentent l'entier de la fonction. Ils permettent de se faire une idée de ce que fait cette fonction, comment elle le fait et dans quel but elle le fait. D'autres parties du code sont mises en évidence avec du rouge et en vert : ce sont différentes variables qui sont utilisées par les fonctions. **%Y-%m-%d** indique par exemple le format de la date avec l'année (4 chiffres), puis le mois (2 chiffres), puis le jour (2 chiffres) séparés par des traits d'union : 2023-02-23.

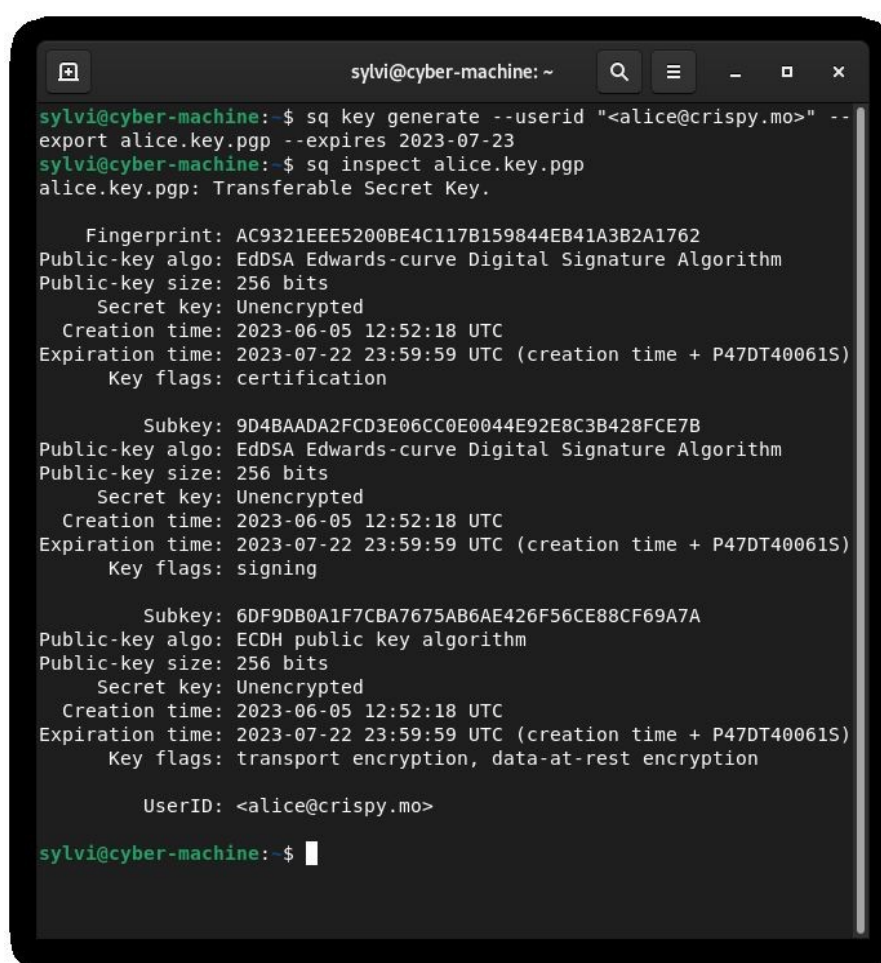
Maintenant que nous avons une meilleure compréhension de ce qu'on voit dans ce module, tâchons de comprendre ce qu'on y lit. La structure du module est la suivante : les lignes 1 et 2 signalent que la fonction utilise deux li-

brairies externes. En l'occurrence, la première librairie s'appelle **chrono** et permet de gérer le temps et la date, alors que la deuxième librairie s'appelle **anyhow** et permet de gérer les erreurs. On peut concevoir ces librairies comme du code externalisé qui propose un catalogue de fonctions utiles à différents contextes. Comme de nombreux programmes utilisent des fonctions similaires indépendamment du contexte, comme celle d'interpréter une heure et une date, des codeuses ont développé des bouts de codes que l'on appelle des librairies et qui peuvent être importés dans différents programmes. On peut imaginer cela comme si Sequoia avait copié et collé certaines fonctions de **chrono** et **anyhow** dans son propre code, à la différence près qu'il s'agit d'un copier-coller dynamique : en effet, si des changements sont fait dans ces librairies et qu'une mise à jour est publiée, le programme qui y fait appel peut mettre à jour le lien vers la nouvelle version de la librairie et les changements sont dynamiquement adoptés. Ces librairies externes sont ce qu'on appelle les « dépendances » du code : autrement dit, le code dépend de ces librairies. Ces liens de dépendance ne posent pas problème en soi, mais peuvent s'avérer problématiques quand la librairie en question n'est pas fiable. De plus, le chemin des dépendances peut-être très long : SQ fait appel – entre autres – à la librairie Sequoia, qui appelle à son tour la librairie Nettle, qui appelle encore plusieurs autres librairies¹³⁴. Ainsi, le choix des dépendances est un enjeu majeur dans le travail de codage. Comme nous le verrons tout à l'heure, l'importance des librairies est particulièrement saillante pour l'implémentation de fonctions cryptographiques. Il en existe de nombreuses différentes, en différents langages, et le choix pour l'une ou l'autre n'a rien d'anodin.

Revenons pour le moment à notre exemple du clone de la fonction **parse_iso8601**. Les lignes 4 à 8 sont des lignes de commentaires qui documentent le module. Puis, on trouve des lignes 9 à 50 l'unique fonction du module nommée **parse_iso8601**. On peut constater que la logique de la construction de cette fonction peut se résumer ainsi : « si X, alors Y ; si X', alors

134 Notons au passage que Sequoia est elle-même une librairie et que le programme en ligne de commande SQ utilise cette librairie pour exécuter les instructions demandées.

Y' ; sinon Z ». Dans ce cas très simplifié, le module ne compte qu'une seule fonction ; mais en général, les modules en comptent plusieurs. Finalement, à partir de la ligne 52, il y a une section de tests, indiquée par **#[test]**, qui est là afin de s'assurer que la fonction réagisse comme il se doit. Comme je l'ai indiqué, ce module est le clone d'un réel module appliqué dans le code dont on peut voir un usage type dans l'illustration 19.



```
sylv@cyber-machine: ~  
sylv@cyber-machine:~$ sq key generate --userid "<alice@crispy.mo>" --  
export alice.key.pgp --expires 2023-07-23  
sylv@cyber-machine:~$ sq inspect alice.key.pgp  
alice.key.pgp: Transferable Secret Key.  
  
Fingerprint: AC9321EEE5200BE4C117B159844EB41A3B2A1762  
Public-key algo: EdDSA Edwards-curve Digital Signature Algorithm  
Public-key size: 256 bits  
Secret key: Unencrypted  
Creation time: 2023-06-05 12:52:18 UTC  
Expiration time: 2023-07-22 23:59:59 UTC (creation time + P47DT40061S)  
Key flags: certification  
  
Subkey: 9D4BAADA2FCD3E06CC0E0044E92E8C3B428FCE7B  
Public-key algo: EdDSA Edwards-curve Digital Signature Algorithm  
Public-key size: 256 bits  
Secret key: Unencrypted  
Creation time: 2023-06-05 12:52:18 UTC  
Expiration time: 2023-07-22 23:59:59 UTC (creation time + P47DT40061S)  
Key flags: signing  
  
Subkey: 6DF9DB0A1F7CBA7675AB6AE426F56CE88CF69A7A  
Public-key algo: ECDH public key algorithm  
Public-key size: 256 bits  
Secret key: Unencrypted  
Creation time: 2023-06-05 12:52:18 UTC  
Expiration time: 2023-07-22 23:59:59 UTC (creation time + P47DT40061S)  
Key flags: transport encryption, data-at-rest encryption  
  
UserID: <alice@crispy.mo>  
sylv@cyber-machine:~$
```

Illustration 19 : Capture d'écran d'une fenêtre de terminal créant une paire de clés PGP avec le programme SQ (1ère commande) puis affichant ladite paire de clés (2e commande). La fonction `parse_iso8601` est utilisée pour interpréter la date dans la première commande, en l'occurrence le 23 juillet 2023.

Maintenant que l'organisation d'un module de code a été présentée, il est intéressant de s'arrêter sur ce que l'on peut apprendre à partir d'un tel exemple simplifié. Dans ce qui suit, je précise un point crucial pour la compréhension du code informa-

tique : il s'agit de l'interprétation particulièrement pointilleuse que font les ordinateurs de ce qu'on leur donne à lire. Cela permet de comprendre une des raisons pour lesquels le code est si vulnérable et si enclin aux bogues.

5.1.1. La méticulosité des ordinateurs

L'exemple de ce module simplifié a été « fabriqué » en collaboration avec une développeuse de Sequoia. Après avoir copié-collé la « vraie » fonction **parse_iso8601** du code de SQ, la codeuse a commencé à la simplifier. Il s'agissait notamment de supprimer des cas de figures superflus à la compréhension du code. Puis, après avoir beaucoup épuré la fonction, elle m'a dit qu'il était temps d'aller voir si ce qu'elle avait fait tenait la route : c'était l'heure de la « danse du compilateur ». Ce moment correspond à la traduction du code source en code binaire. La codeuse l'appelait ainsi parce que cette étape consistait en de nombreux allers-retours entre le code source et le compilateur. En effet, il est très rare que le code source soit parfait dès sa première version : il y a presque toujours des fautes de frappe, des symboles manquants (comme des parenthèses qui ouvrent ou ferment des fonctions), ou parfois, des parties incomplètes. Or, le compilateur de RUST refuse de compiler un code qui comporte des erreurs. Il envoie donc un message d'erreur à la personne qui code, en signalant les lignes ne fonctionnant pas. Lorsqu'elle m'avait fait la démonstration avec un véritable bout de code lors d'un séjour en présentiel, le compilateur avait d'abord indiqué plus d'un millier d'erreurs. En faisant des allers-retours entre le code source et le retour du compilateur, la codeuse était parvenue à corriger toutes ces erreurs. Dans le cas du clone de la fonction **parse_iso8601**, elle m'a écrit sur IRC être plutôt satisfaite :

je n'ai foiré que trois choses. Les ordinateurs sont si pointilleux.
(Codeuse de Sequoia, *chat* IRC, 3 mars 2023)

Pointilleux : voici ce qui définit bien la manière dont les ordinateurs interprètent le code. En effet, lorsque l'on écrit un programme informatique, il s'agit d'envisager toutes les possibilités et de coder chaque petite étape l'une après l'autre. Cela signifie que chaque boîte noire du protocole doit être décortiquée en une séquence d'opéra-

tions explicites. Il ne suffit plus de dire : « Alice utilise la clé publique de Bob pour chiffrer son message que Bob déchiffra avec sa clé privée ». Il faut préciser ce que sont les clés publiques et privées, comment se passe une opération de chiffrement, comment savoir quels algorithmes de chiffrement symétrique et asymétrique utiliser pour former un message chiffré, comment formater ce message, etc. Chaque opération doit être scrupuleusement explicitée. Si j'insiste là-dessus, c'est que la méticulosité des ordinateurs est souvent source de conflits entre eux et les humains, quand ces derniers appellent « bogues » ce que les premiers n'arrivent pas à comprendre.

Un autre élément que j'aimerais souligner à propos de cet exemple de module concerne l'anticipation de dysfonctionnements. Tout d'abord, il est intéressant de noter que l'erreur n'est pas exclue et que le code spécifie que faire en cas d'erreurs quand celles-ci sont prévisibles (ligne 49). Cela dit, il est probable que certaines erreurs empêcheront le fonctionnement du programme et il faudra alors modifier la fonction. Dans le cas présent, on peut par exemple imaginer qu'une manière d'écrire la date n'ait pas été prise en compte par la première codeuse et qu'il serait préférable de l'ajouter dans le code. Pour se faire, la mainteneuse pourra modifier cette fonction en ajoutant la nouvelle combinaison. C'est ce qui s'appelle un patch, ou un correctif. Les patches ajoutent, suppriment ou transforment des lignes de code ; ils sont généralement ajoutés au code complet sous forme de *commit* sur GitLab ou GitHub, et, s'ils sont validés par une *merge request*, ils seront alors disponibles lors de la nouvelle mise à jour du programme¹³⁵.

En résumé, cet exemple du module de la fonction **parse_iso8601** permet de rendre compte de plusieurs points importants dans la compréhension du code. Tout d'abord, « le code » n'est pas que « le code » : même une fonction aussi simple que celle de l'exemple fait appel à des bibliothèques externes – et donc du code externe. De plus, le code n'est pas composé uniquement d'instructions à destination de la machine, mais contient de nombreuses informations sous forme de commentaires qui s'adressent aux personnes qui l'utiliseront ou le modifieront plus tard. Comme on le

135 Je suggère la lecture du très bon article de Stéphane Couture (2012) à propos de l'écriture collective du code pour en apprendre plus sur l'anthropologie du *commit*.

voit dans l'exemple, le code n'est pas non plus magique : il ne s'agit « que » d'instructions très précises et très basique qui détaillent chaque étape à réaliser en vue d'obtenir un certain résultat. Finalement, remarquons aussi que ce module simplifié `parse_iso8601` semble très éloigné des savants calculs mathématiques de la cryptographie asymétrique ; pourtant, sans ce module, le programme SQ ne pourrait pas fonctionner. Cela souligne le fait qu'un programme de chiffrement consiste toujours en quelque chose de plus que simplement des fonctions cryptographiques. Du point de vue de la sécurité, cette complexité du code est un élément important : même en utilisant les primitives cryptographiques réputées pour être les plus performantes, un code peut être vulnérable à cause d'une erreur dans une fonction aussi simple que **`parse_iso8601`**.

La section suivante continue l'exploration du code informatique en explorant deux éléments centraux pour la sécurité informatique que l'on cherche à produire : le choix du langage informatique d'une part, et le choix des bibliothèques cryptographiques d'autre part.

5.1.2. Le choix du langage informatique

Il me faut commencer par mentionner la grande diversité des langages informatiques. Chacun d'eux offre quelque chose de particulier et il n'est pas rare que les codeuses soient amenées à en utiliser plusieurs, des dizaines parfois. La citation ci-dessous est tirée d'une conversation par email avec une des codeuses de ma recherche. Elle me semble particulièrement illustratrice de la diversité de ces langages.

Professionnellement, j'ai travaillé avec Java, JavaScript, TypeScript, Python, Rust, Kotlin, Ruby, plusieurs variants SQL. Quand j'étudiais, j'ai aussi utilisé C, C++, C#, MATLAB et Perl. Pour mes projets personnel, j'utilise aussi PHP et Lua et j'ai aussi expérimenté dans mon temps libre avec d'autres pour savoir ce qu'il y avait : Go, C#, Haskell, PureScript. J'ai aussi écrit mon propre langage de programmation pour mon blog secret. J'ai aussi écrit des interpréteur pour Prolog et Lips. Quand j'étais petit, j'ai commencé à véritable programmation avec Delphy (Object Pascal).
(Conversation par email, 23 octobre 2020)

Dans ce qui suit, je me focalise uniquement sur deux langages informatiques : C et RUST, car ce sont des langages informatiques utilisés dans plusieurs librairies cryptographiques, et notamment GnuPG et Nettle (pour C) et Sequoia et rPGP (pour RUST). Mon objectif n'est pas de faire une comparaison exhaustive de ces deux langages mais de souligner le type d'enjeux que présente le choix d'un langage informatique.

C a été développé en 1972. Il est basé sur le langage B, développé en 1969 et lui-même basé sur BCPL. C est l'un des langages les plus utilisés, notamment parce qu'il est très flexible et permet une gestion fine d'une grande quantité d'opérations. Il a notamment servi à écrire Unix, encore actuellement codé en C. RUST est, quant à lui, un nouveau langage développé par Mozilla depuis 2010. Il est considéré par plusieurs des codeuses interviewées comme l'un des meilleurs langages informatiques actuels. Son nom provient du fait qu'il repose sur des concepts uniquement connus et éprouvés sans inclure des éléments nouveaux et non testés. L'un de ses points forts est son compilateur qui permet d'éviter de nombreuses erreurs, notamment celles dues à une mauvaise gestion de la mémoire particulièrement fréquentes en C.

Cette présentation des deux langages informatiques que je viens de faire pourrait faire croire que RUST serait largement supérieur à C. C'est d'ailleurs ce qu'affirme une codeuse de la communauté OpenPGP qui avait l'habitude d'utiliser RUST :

C et C++ ne sont pas des langages que tu devrais utiliser pour implémenter de la crypto, plus aujourd'hui. Nous avons un monde au-delà de C, et Microsoft et Google sont littéralement en train de dire au monde : « Arrêtez d'écrire des codes cryptographiques, ou des codes sensibles, en C et C++. » Donc s'il-vous-plaît, arrêtez. Ils ont analysé que plus de 50 % des vulnérabilités dans Windows qui ont été trouvées sont dues à des erreurs de type dépassement de la mémoire ou d'accès à la mémoire qui aurait pu facilement être évitée en utilisant un autre langage que C.

(Codeuse d'une implémentation d'OpenPGP, entretien du 2 septembre 2021)

Le fait que Sequoia et rPGP soient écrits en RUST fait d'ailleurs partie des arguments sécuritaires en leur faveur – du moins pour de nombreuses personnes. Ce choix de RUST est considéré par beaucoup comme un choix intelligent, voire logique, au vu du caractère moderne et performant attribué à RUST. Ainsi, le simple fait d'utiliser RUST est perçu comme une garantie d'avoir un plus haut niveau de stabilité du

code, qui est un élément clé de la sécurité. La question que pose ce consensus autour de l'usage de RUST est donc la suivante : pourquoi utiliser d'autres langages ?

La réponse peut être historique. En effet, dans les années 1990, C a lui aussi été considéré comme « le langage de l'avenir ». C'est par exemple ce qui ressort d'un entretien avec Werner Koch qui m'a expliqué qu'il a décidé d'utiliser C pour écrire GnuPG en 1997, car non seulement Unix était codé en C, mais que c'était aussi un langage réputé comme étant un langage moderne et performant :

J'ai écrit GnuPG pour Unix, et Unix est écrit en C. [...] J'ai commencé assez tôt à utiliser C, en 1985, pour un job d'été. Je disais des choses comme : « Oh, tu peux faire ces trucs qu'on ne peut plus faire avec BASIC », parce qu'à l'époque, nous utilisions encore BASIC pour certains programmes commerciaux. Donc j'ai commencé à faire bouger l'entreprise pour qu'elle travaille avec C, jusqu'à ce qu'on fasse finalement tout en C. La raison pour laquelle on utilise C est que c'est un programme de bas niveau. Bon, pour moi, c'est encore un haut niveau mais c'est suffisamment de bas niveau pour pouvoir coder toute sorte de logiciel. Et c'est toujours le cas.

(Werner Koch, entretien du 30 juillet 2020)

Cette citation est intéressante car elle souligne un argument important lié à la compatibilité. En effet, Koch précise d'emblée que GnuPG a été écrit dans le langage du système avec lequel il voulait interagir, Unix. Étant donné que deux codes écrits dans le même langage peuvent facilement être intégrés l'un avec l'autre, cet argument a un poids important. C'est par exemple pour avoir une implémentation complète en RUST que rPGP a été développé en parallèle à Sequoia : alors que Sequoia utilise Nettle en C, rPGP n'utilise que des bibliothèques écrites en RUST.

Mes données de recherche me permettent d'ajouter un autre argument en faveur de C. Lors d'un entretien avec une personne très active dans l'écriture de la 4^e version du protocole OTR¹³⁶, nous avons parlé du choix de C comme langage informatique utilisé pour cette nouvelle implémentation. D'abord, la codeuseuse m'a dit, sur le ton de l'humour, que C avait un aussi grand nombre de fans et que de personnes qui le détestaient. Elle m'a ensuite expliqué durant une dizaine de minutes les types d'er-

136 OTRv4 n'est pas une implémentation d'OpenPGP. J'ai eu l'occasion de m'entretenir avec cette personne dans la phase exploratoire de ma recherche.

reurs les plus fréquentes avec C, la grande majorité étant en lien avec la gestion de la mémoire. Finalement, malgré ces critiques, elle a conclu :

Nous avons finalement opté pour C parce que C est un langage que tout le monde comprend. Je dirais même : tout le monde et tous les systèmes d'exploitations comprennent C, et aussi, la plupart des bibliothèques sont écrites en C. Donc si tu veux parler avec d'autres bibliothèques, c'est très facile parce qu'elles parlent aussi C. [...] Avec C, tu peux toujours ponter tous les autres langages informatiques [...]. C'est pour ça qu'on a décidé d'utiliser C.
(Codeuseuse d'OTRv4, entretien du 19 juillet 2019)

Ce dernier argument est intéressant : parfois, ce ne sont pas toujours les qualités ou les défauts intrinsèques qui guident le choix du langage, mais l'environnement dans lequel le code s'insère. Nous avons donc affaire à une autre forme de l'enlèvement infrastructurel présenté dans le chapitre précédent. Dans le cas d'OTRv4, qui d'ailleurs revendique aussi une volonté de pouvoir également fonctionner sur des ordinateurs anciens et vétuste que l'on trouve dans le Sud Global, l'omniprésence de C dans l'infrastructure avec laquelle le programme veut interagir était un argument majeur. Dans d'autres cas, d'autres langages sont favorisés pour mieux s'adapter à ce milieu. Ainsi, les applications mobiles de ProtonMail sont codées en Go, un langage haut niveau développé par Google pour les environnements Android notamment, et l'application web de ProtonMail est codée en JavaScript, le langage utilisé dans l'immense majorité des sites web. Ainsi, l'infrastructure peut donc parfois, mais pas toujours, être un argument majeur dans le choix (ou la contrainte) de l'utilisation d'un langage informatique spécifique.

Dans tous les cas, on retiendra que le choix du langage informatique présente plusieurs enjeux stratégiques, mais aussi sécuritaires. Comme je l'ai montré en soulignant les différences entre RUST et C, les problèmes ne seront pas les mêmes selon le langage que l'on choisit pour coder son logiciel. Que ce soit pour se prémunir contre des attaques liées à une mauvaise utilisation de la mémoire, pour favoriser un langage que « tout le monde connaît » ou pour être le plus compatible possible avec différentes bibliothèques, le choix du langage informatique est influencé par le projet-sécurité des développeuses et la manière dont ils et elles perçoivent leur mission.

5.1.3. Le choix d'une librairie cryptographique

Finalement, il convient d'examiner une autre source d'insécurité dans le code : les librairies (que l'on appelle parfois aussi les « dépendances »). Je l'ai mentionné plus haut : les codes font souvent appel à des librairies cryptographiques externes pour implémenter certaines parties du code. Dans l'exemple de la fonction `parse_iso8601`, il s'agissait de deux librairies de RUST. L'usage de ces librairies pour cette fonction allait de soi à cause de la simplicité des fonctions concernées ; mais dans certains cas, les codeuses peuvent se retrouver face à un choix difficile à faire, par exemple quand il s'agit de sélectionner la librairie pour les primitives cryptographiques. Dans cette section, j'aimerais revenir sur une discussion qui a eu lieu au de l'équipe de Sequoia à ce sujet. Cette discussion permet de saisir les enjeux sécuritaires dans le choix de telle ou telle librairie, qui est une étape clé de la production de technologie numérique. Cet exemple permet aussi de saisir la négociation d'un projet-sécurité commun au sein d'une équipe de développement.

Lors d'une séance hebdomadaire de travail d'automne 2021, la discussion s'est orientée vers la possibilité d'utiliser une deuxième librairie cryptographique, RustCrypto¹³⁷, en plus de Nettle¹³⁸ qui était la librairie qui avait été choisie au début du projet. Cette discussion n'est pas née d'une remise en question de l'usage de Nettle utilisée depuis le début du projet : l'équipe la trouvait fiable, efficace, stable et assez réactive quand il y avait des problèmes à régler. Cependant, elle était écrite en C, ce qui la rendait plus difficile à utiliser dans l'écosystème de RUST et minimisait également l'argument de sûreté que fournissait l'usage de RUST pour Sequoia. Une alternative possible pour fournir un environnement entièrement en RUST était la librairie RustCrypto qui implémentait en RUST la plupart des algorithmes nécessaires à OpenPGP. La question était de savoir s'il fallait ou non utiliser RustCrypto. Cette discussion a été l'occasion de redéfinir au sein de l'équipe de Sequoia non seulement ce qui était attendu d'une telle librairie, mais aussi quel était le niveau de responsabilité de Sequoia par rapport à ces librairies et aux potentiels bogues qu'elles contenaient.

137 <https://github.com/RustCrypto/>.

138 <https://www.lysator.liu.se/~nisse/nettle/>.

Ainsi, lors de cette discussion, quatre critères ont été énoncés pour juger de la qualité d'une librairie :

- La librairie implémente les algorithmes nécessaires ;
- La librairie est correctement maintenue ;
- Les mainteneureuses sont réactives ;
- La librairie a été auditée plusieurs fois.

Parmi ces critères, seul le premier semble relever de critères complètement objectifs. Il est en effet facile de dresser une liste des algorithmes nécessaires et de cocher lesquels sont implémentés par les potentielles librairies. Les autres critères, en revanche, dépendent de l'évaluation subjective des membres de l'équipe. Or, comme l'a dit un·e des membres, la plupart des personnes présentes ne se sentaient pas capable d'évaluer la qualité d'un code cryptographique, ni d'ailleurs celle d'un audit¹³⁹. En revanche, certaines vulnérabilités étaient identifiées et posaient problème. Par exemple, RustCrypto ne fournissait pas d'opérations en temps constant [*constant-time operation*], ce qui la rendait probablement vulnérable aux attaques par canal auxiliaire [*side-channel attack*]. À cet argument, une codeureuse a affirmé :

Nous ne sommes pas des expert·e·s en cryptographie, nous n'avons donc pas les clés pour comprendre les fonctions cryptographiques [*crypto stuff*]. Ce que nous pouvons faire, c'est évaluer le travail des autres en fonction de ce que nous jugeons utile, comme la réactivité des mainteneureuses.
(Codeureuse de Sequoia, propos retranscrits de mémoire après la discussion)

Lors de cette discussion, la question de la responsabilité que Sequoia avait vis à vis de ses utilisateurices a souvent été évoquée. En cas de problèmes liés à la librairie en question, par exemple si une vulnérabilité sérieuse était découverte dans le code d'une librairie, Sequoia pouvait-elle être tenue pour responsable ? Certaines codeureuses affirmaient que oui et que c'était de leur responsabilité d'éviter aux utilisateurices de prendre des risques inutiles en choisissant une librairie plus facile à utiliser (car en RUST) mais moins sécurisée (car des vulnérabilités étaient identifiées) ;

139 À propos de la difficulté de juger de la qualité d'un audit, je recommande la lecture du chapitre 5 « *What is 'good' security* » de Ksenia Ermoshina et Francesca Musiani (2022, 183ss).

d'autres soutenaient l'inverse en affirmant qu'il fallait éviter de dire ce que les utilisatrices devaient faire ou pas. Un·e des membres a par exemple affirmé :

Vous voulez le faire de la manière sûre, mais « sûr » n'a de sens que dans un contexte donné, en fonction du modèle de menace. Or, ce n'est pas notre responsabilité de définir ce que cela doit être.
(Codeureuse de Sequoia, propos retranscrits de mémoire juste après la discussion)

Cette discussion m'a semblé très intéressante car elle montre différents enjeux autour du choix de l'usage d'une librairie externe. Dans le cas présent, la librairie idéale n'existait simplement pas et l'enjeu était donc de décider d'un compromis acceptable entre deux choix imparfaits : d'un côté, une librairie auditée et réputée comme étant fiable, mais codée dans un langage informatique plus contraignant avec des inconvénients majeurs quant à la facilité d'usage et des vulnérabilités potentielles typique des codes en C et, de l'autre, une librairie encore immature, non auditée, mais codée dans un langage informatique réputé beaucoup plus sûr et qui permet une meilleure compatibilité, rendant l'utilisation de la librairie beaucoup simple. L'absence de solutions idéales souligne le caractère négocié de la sécurité même sur des choix aussi technique que le choix d'une librairie.

Maintenant que nous avons vu de quoi est fait un code informatique et plusieurs sources de bogues au sein du code, nous pouvons nous pencher sur les raisons qui explique la complexité du code qui ne cesse de s'épaissir. Cela permet de comprendre pourquoi la maintenance de codes anciens est souvent problématique et toujours fastidieuse.

5.2. La complexité du code, de l'incompréhension au dialogue

On dit souvent que la cryptographie est un domaine hautement complexe. Les livres sur l'histoire de la cryptographie insistent par exemple systématiquement sur les adroites complications des systèmes utilisés et sur la difficulté de les casser. Cette complexité-ci concerne la sophistication de l'ingénierie cryptographique et est connotée positivement. Au contraire, OpenPGP est souvent qualifié de complexe et

cela n'est pas un compliment. Derrière cette complexité-là se cache une critique quant à certains aspects dans la conception même du protocole, mais aussi au fonctionnement des algorithmes cryptographiques supportés et aux multiples cas d'utilisation. Pour bien comprendre la différence entre ces deux types de complexité, il est important de se rendre compte que la première est recherchée car considérée comme un atout pour la sécurité alors que la seconde est subie et considérée comme un problème sécuritaire. Je commence par expliquer d'où vient cette deuxième complexité puis établis un lien évident avec les temporalités en jeu. En effet, en évoluant, c'est-à-dire en étant maintenu, le code devient nécessairement plus complexe, accroissant ainsi la difficulté de le comprendre et de le modifier. Une fois de plus, nous nous retrouvons devant une forme d'enlisement inévitable des technologie du numérique : pour survivre, elles doivent être maintenues ; or cette maintenance les rend plus complexes, et donc plus difficiles à maintenir par la suite, et donc plus vulnérables. Cela augmente encore le besoin de maintenance. Au final, nous avons un cercle vicieux où les actes actuels de maintenance rendent la maintenance future plus difficile à mettre en œuvre.

5.2.1. L'inévitable complexité et les petits poils qui poussent sur le code

Commençons par questionner la raison de cette complexité. Partant du principe empiriquement éprouvé que les bogues ne cessent jamais d'émerger, on peut se demander ce que l'on peut et doit faire pour prendre soin du code. Tout d'abord, les personnes qui maintiennent un code doivent non seulement comprendre comment ce code est organisé et ce qu'il est censé faire, mais aussi ce qu'il fait vraiment et d'où vient le bogue en question. Il s'agit donc de comprendre à la fois le code et le bogue. Une fois que ces deux aspects sont compris, la codeuse peut alors expérimenter diverses solutions, en espérant que l'une d'elles fonctionne. À chaque modification, le code gagne en complexité, d'autant plus que les modifications elles-mêmes peuvent contenir des erreurs, voire même représenter elles-mêmes un nouveau

bogue¹⁴⁰. Dans son fameux billet de blog déjà cité précédemment, Joel Spolsky justifie cette complexité de manière très claire et imaginée :

Pourquoi est-ce un vrai désordre ? [en parlant du code hérité]

“Eh bien”, disent-ils, “regardez cette fonction. Elle fait deux pages ! Rien de tout cela n'a sa place là-dedans ! Je ne sais pas à quoi servent la moitié de ces appels à l'API.”

[...]

Revenons à cette fonction de deux pages. Oui, je sais, ce n'est qu'une simple fonction pour afficher une fenêtre, mais elle a accumulé des petits poils [*it has grown little hairs*] et d'autres trucs, et personne ne sait pourquoi. Je vais vous dire pourquoi : ce sont des corrections de bogues. L'un d'entre eux corrige le problème que Nancy a rencontré lorsqu'elle a essayé d'installer le logiciel sur un ordinateur qui n'était pas équipé d'Internet Explorer. Un autre corrige le bogue qui se produit lorsque la mémoire est faible. Une autre encore corrige le bogue qui se produit lorsque le fichier se trouve sur une disquette et que l'utilisateur arrache la disquette au milieu. L'appel à la librairie de chargement est affreux, mais il permet au code de fonctionner sur les anciennes versions de Windows 95. (Spolsky 2000, ma traduction)

Bien que cet exemple date d'il y a plus de vingt ans et que les disquettes font désormais partie des accessoires *vintage*, il décrit une réalité qui est encore actuelle : un code qui vieillit devient forcément complexe, et la complexité est mal vue parce qu'elle fait peur et qu'elle est difficile à corriger. Avec le temps, le code devient trop compliqué à comprendre, trop long, trop dense, trop mal écrit, trop lent, trop contraignant, et finalement trop facile à mal utiliser. Autrement dit, plus un code est travaillé, maintenu et corrigé, plus il devient compliqué ; et plus il est compliqué, plus il devient difficile de le comprendre, de travailler avec et de le corriger. Cela semble inévitable, comme le montre la discussion entre deux développeuses sur la plateforme GitHub reportée dans l'illustration 20 ci-dessous :

140 Je l'ai notamment montré dans un article au sujet des vulnérabilités d'EFAIL (Besençon 2022).

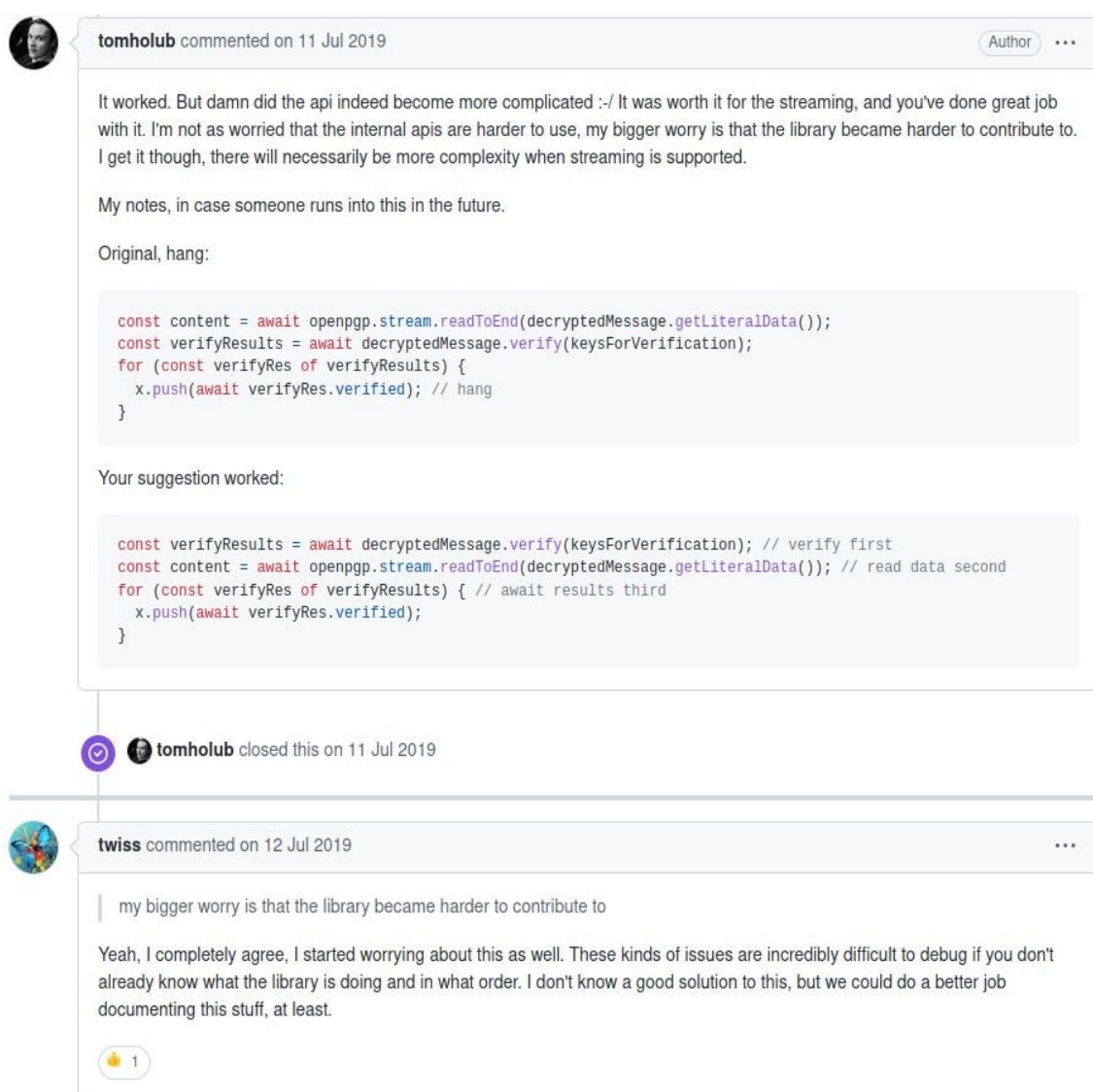


Illustration 20 : Conversation entre deux contributeurices d'OpenPGP.js. Source: <https://github.com/openpgpjs/openpgpjs/issues/916>.

Cette discussion a eu lieu sur la plateforme d'échange de code GitHub d'OpenPGP.js, qui est l'une des librairies que maintient l'entreprise Proton pour son service d'email sécurisé. La première personne avait constaté un bogue qu'elle voulait tâcher de résoudre, et la deuxième personne l'avait aidé à comprendre le problème et à le résoudre. La dernière intervention me paraît essentielle : répondant au commentaire « mon plus gros soucis est qu'il devienne de plus en plus difficile à contribuer à cette librairie », la deuxième personne a répondu :

Oui, je suis complètement d'accord, j'ai aussi commencé à m'inquiéter à ce propos. Ce genre de problèmes sont très difficiles à déboguer si tu ne sais pas déjà

ce que fait la librairie et dans quel ordre. Je ne connais pas la bonne solution à ce problème, mais nous pourrions au moins mieux documenter ces choses.

Ici, il paraît évident que la complexité est inévitable, contre laquelle le seul remède envisagé semble être une meilleure documentation¹⁴¹. Le lien entre l'âge du code et sa complexité est ici explicite : de par les multiples correctifs qui sont appliqués au code au fur et à mesure que des bogues ont été trouvés et que de nouvelles fonctions ont été ajoutées, le code gagne en épaisseur : il n'y a pas forcément plus de fonctions, mais celles-ci deviennent toujours plus denses et plus complexes. J'ai ainsi établi que le code, en vieillissant, devenait nécessairement plus complexe. La sous-section suivante analyse les conséquences de cette complexité.

5.2.2. Le passé comme source d'incompréhension

Examinons de plus près la question de l'héritage du passé et regardons comment les personnes qui s'occupent du code en parlent. Lors de mes entretiens, j'ai constaté que le lexique utilisé pour parler du code hérité avait de fortes connotations morales ancrées dans une représentation linéaire du temps. La citation ci-dessous donne un exemple particulièrement saillant de cette rhétorique où le passé est désigné par des termes péjoratifs alors que l'avenir est connoté positivement :

[...] il y a en ce moment une discussion sur la liste de diffusion [d'OpenPGP] sur la manière de procéder pour moderniser OpenPGP. Mais le travail ici n'est pas unique : tu dois introduire de nouveaux éléments qui n'étaient pas là quand le vieux protocole OpenPGP a été conçu (par exemple le chiffrement authentifié AEAD) et tu dois commencer à considérer comme obsolète [*deprecating*] les mauvaises choses et finalement les supprimer [...]. C'est comme si tu maintenais un vélo : tu montes des nouvelles et meilleures pièces, et tu nettoies ce qui s'est accumulé sur les choses [*the cruft on things*] qui se sont abîmées quand tu l'utilisais.
(Codeuseuse d'une implémentation d'OpenPGP, email personne, 22 octobre 2020)

L'analogie avec la maintenance d'une bicyclette n'est pas anodine et souligne l'aspect manuel voire salissant du travail de maintenance. J'aimerais insister sur les

¹⁴¹ L'effort de documentation sera discuté dans la 3^e partie de cette thèse au chapitre 8 « Penser les fragilités à venir pour panser l'avenir » (p. 300).

termes utilisés dans cet extrait : « moderniser », « introduire de nouveaux éléments », « *to deprecate* », « *cruft* »¹⁴², « s’abîmer ». Aux éléments du passé considérés comme inutiles, défectueux ou qui font obstacle (« *cruft* ») s’oppose la modernisation du projet-sécurité qui nettoierait, polirait et huilerait les engrenages rouillés d’une infrastructure vieillissante bien qu’encore fonctionnelle. Une autre citation tirée d’un entretien confirme ce clivage entre ce qui est hérité et ce qui est projeté en suggérant l’idée d’un dialogue :

C’est très dur d’innover dans OpenPGP à cause de ses vieilleries [*old craft*] et de ses vieilles idées. Ce sont surtout de vieilles idées. Coder, c’est comme écrire des idées. Si tu réécrits un logiciel, tu jettes des idées du passé. Donc de mon point de vue, écrire du code, c’est comme communiquer avec les personnes qui ont écrit avant toi.

(Codeuseuse d’une implémentation d’OpenPGP, entretien du 25 août 2020)

Une traduction littérale de l’expression « *old craft* » traduite ci-dessus par « vieilleries » serait « ancien savoir-faire ». Ces mots renforcent l’analogie de la maintenance des choses numériques à un art manuel ancien où le vieux est fondamentalement mauvais et doit être remplacé par du neuf. Ces deux citations expriment très bien l’inconfort face au code hérité tout en admettant une certaine violence à son égard : « jeter les idées du passé », « considérer comme obsolètes les mauvaises choses et finalement les supprimer ». Cette manière de décrire le travail de maintenance souligne le fossé qui peut exister entre l’état actuel du code en partie déterminé par le code hérité d’une part et le projet-sécurité des développeuses d’autre part.

Ces deux extraits d’entretiens sont intéressants car ils montrent une certaine aversion à l’égard du code qui vient du passé, tout en concédant que le changement ne va pas se faire d’un coup mais petit à petit. Ici, on constate que les codeuseuses actuelles entame une négociation entre leur propre projet-sécurité et celui des codeuseuses de l’époque, en tâchant de se rapprocher le plus de leur idéal sans pour autant réécrire totalement le passé. Bien qu’il faille parfois « jeter des idées du passé », il

142 « Cruft » signifie, dans le jargon hacker, « tout ce qui est en trop, redondant et qui gêne. Le terme est utilisé en particulier pour les éléments défectueux, dépassés, inutiles, superflus ou dysfonctionnels d’un logiciel informatique » (ma traduction) : <https://web.archive.org/web/20220130185435/https://en.wikipedia.org/wiki/Cruft>.

n'est souvent pas possible de les renier dans leur intégralité puisque, rappelons-le, l'infrastructure est utilisée et donc qu'elle ne peut pas, doit pas mourir. Au final, nous voyons que non seulement la sécurité complexifie le code à cause des nombreux patchs qui s'additionnent au fil du temps, mais la sécurité contribue aussi à le « moderniser » petit à petit en tâchant de remplacer les anciennes pièces par des nouvelles et en y ajoutant de nouvelles fonctions.

Qu'est-ce que cela signifie concrètement ? Pour illustrer ce point, j'aimerais revenir à l'exemple de la fonction `parse_iso8601` dans la section précédente. Les lectrices les plus attentives auront remarqué que le nom de certaines variables n'est pas le même que dans la fonction originale. Ainsi, le « **f** » original a été remplacé par « **format** », le « **s** » par « **input** » et « **pad_date_time** » par « **time_of_day** ». Cela avait pour but de rendre cette fonction plus facile à comprendre, bien entendu, mais traduisait un certain malaise de fond. En effet, les codeurs et codeuses n'ont pas tous été formés à la même école et il existe différentes conventions qui fluctuent avec le temps et le contexte. Lors de la production de cet exemple, la codeuseuse m'a écrit sur IRC :

dans les premiers temps de la programmation, les gens utilisaient souvent des noms de variables très courts
certains langages de programmation limitaient même la longueur des noms à quelques caractères
mais c'est très difficile à saisir pour les lectrices
et le code a tendance à être plus souvent lus qu'écrit
(Codeuseuse de Sequoia, *chat* IRC, 3 mars 2023)

Cette citation me paraît très importante car elle signale deux choses fondamentales à comprendre à propos du code source. D'une part, l'acte de coder implique autant de lecture que d'écriture et, d'autre part, le code n'est pas uniquement destiné à être interprété par des machines, mais il est également lu et relu par des contributeurices actuelles et futures du code. Dans un article paru en 2022, je parlais des « pratiques d'écriture et de réécriture du code informatique » (Besençon 2022). Cette formulation permettait de rendre compte que le code n'était pas un objet fixe mais qu'il évoluait au gré des modifications, ce qui est important à comprendre. Cependant, j'oubliais de préciser qu'avant d'être modifié, le code devait être lu et compris.

J'irais même plus loin : avant d'écrire la moindre ligne de code d'un programme, une codeuse doit d'abord lire et comprendre les protocoles à implémenter, ainsi que les codes des bibliothèques à utiliser et la documentation qui s'y rapporte. Par ailleurs, s'il est prévu d'intégrer ce code dans une infrastructure plus large, il est important de connaître les conventions propres à cette infrastructure. Ainsi, si le code est effectivement un élément clé des infrastructures scripturales (Denis 2018), ce n'est pas uniquement parce qu'il est écrit mais aussi parce qu'il est lu, et relu, et rere lu. Dès lors, la pratique du codage ne correspond pas à une simple traduction à sens unique du protocole au code source, mais une multitude d'allers-retours entre les différentes étapes de codage et de *dé*-codage, entre lecture et écriture, entre l'état actuel du code hérité du passé et le projet-sécurité des codeuses qui se projettent dans l'avenir.

On comprend alors qu'à travers l'incompréhension qu'il suscite, le code travaillé, maintenu et corrigé instaure un dialogue entre les personnes qui l'ont écrit hier, celles qui le maintiennent aujourd'hui et celles qui, demain, le maintiendront. Ce dialogue, qui consiste parfois à simplement « jeter des idées du passé », mais le plus souvent à les actualiser selon de nouveaux projets-sécurité, associe différentes personnes et différents objets que j'aimerais examiner dans la dernière section de ce chapitre.

5.3. Du vieillissement à l'ancrage

La discussion sur le code hérité me paraît essentiel pour comprendre comment certaines codeuses sont confrontées à deux « ennemis complémentaires » qui les font toujours plus travailler. D'une part, la nécessaire maintenance du code complexifie forcément le code, transformant de simples fonctions en « vrai désordre », selon les termes de Joel Spolsky ; d'autre part, la complexité du code le rend toujours plus difficile à maintenir. Autrement dit, à force d'être maintenu, le code devient plus long, moins homogène et donc moins intelligible, ce qui contribue à rendre la maintenance à venir de plus en plus difficile et faisant appel à des solutions de plus en plus

bricolées – solutions qui rendront à leur tour le code de plus en plus difficile à comprendre et réparer. Si on ne le maintient pas, le code devient inutilisable voire dangereux ; si on le maintient, il devient de plus en plus complexe et donc de plus en plus difficile à maintenir. C'est ce cercle vicieux paradoxal que j'appelle « le vieillissement » du code que je considère comme un geste volontaire de survie.

Contrairement aux êtres vivants, le code n'est pas inéluctablement soumis à la tyrannie du temps qui burine la peau, rouille les mécaniques et fissures les murs. Il est de nature informationnelle, ce qui signifie que la dégradation de son support n'est pas un problème en soi car les informations peuvent être recopiées à l'identique sur un nouveau support et l'information demeure accessible. En revanche, garder ces informations intactes et inchangées n'a pas beaucoup de sens car elles ont besoin d'être mises à jour pour rester fonctionnelles. En d'autres termes, si on n'y touche pas, le code ne vieillit pas, il « meurt » en cessant d'être fonctionnel. Ce qui rend le code obsolète n'est pas l'action du temps qui passe à proprement parler, mais la transformation de l'environnement dans lequel le code s'insère. Autrement dit, le code ne vieillit pas *à cause* du temps qui passe, mais à cause de son propre enlissement dans une infrastructure elle-même enlisée. Ce sont les actes de soin qui prolongent la durée de vie du code et ce sont donc ces actes qui sont responsables du vieillissement du code. Dans ce sens, vieillir, pour une technologie numérique, c'est se transformer ; vieillir, c'est lutter contre l'enlissement ; vieillir, c'est rester en mouvement. De ce point de vue, il est possible de voir le vieillissement et ses corollaires de manière positive : un code vieux est un code qui a été lu, étudié, relu et corrigé à de nombreuses reprises : il véhicule des idées du passé qu'il transmet aux générations futures qui les mettront peut-être à jour.

Ce travail de mise à jour du code hérité est comparable au « bricolage permanent dans un monde plein d'ambivalences complexes et de tensions en mouvement » dont parlait Mol et ses collègues (Mol, Moser, et Pols 2010). Ici, le mot bricolage n'est bien entendu pas péjoratif : il souligne au contraire la dévouement constant qu'exige le travail de maintenance. Le vieillissement de la technologie est en effet constitué de

multiples expérimentations qui contribuent à rendre le code hérité toujours plus complexe à comprendre. Et comme le code ne pourra jamais être complet et totalement terminé, le problème du code hérité ne cessera pas de faire travailler codeuses et développeuses. En guise d'illustration, je mentionne la page GitHub du journaliste et cryptographe Hanno Böck où ce dernier a listé une série de plus d'une vingtaine de failles qui ont affecté le protocole depuis la création de PGP en 1991¹⁴³. Certaines de ces failles sont critiques mais aucune n'a « tué » le protocole, pour reprendre un mot régulièrement utilisé. Une autre manière de lire la situation est de considérer que les codeuses ont réussi à bricoler des solutions à tous ces problèmes.

Ainsi, le fait qu'il y ait tant de problèmes avec l'héritage d'OpenPGP serait la preuve que le protocole n'est pas en état de mort cérébrale. Le fait qu'OpenPGP continue de faire parler de lui est en soi un signe de vitalité et d'intérêt, mais c'est surtout le fait que des failles soient patchées qui indique que le protocole est maintenu. Cela ne signifie pas qu'il est 100 % sécurisé ni que des failles futures ne lui seront pas fatales ; cela signifie que depuis sa création, de nombreuses personnes ont adapté le programme de Zimmermann à un environnement toujours changeant. Se faisant, elles ont retravaillé le code et les spécifications à mesure que les problèmes se présentaient, contribuant ainsi à ancrer toujours plus le protocole dans l'infrastructure qui le soutient et à laquelle il contribue en même temps. Autrement dit, le protocole vit parce qu'autour de lui s'affairent d'autres vies.

Considérer le vieillissement d'une technologie de la manière dont je le propose ici fait surgir un paradoxe étonnant : alors que vieillir est souvent considéré comme un rapprochement inéluctable d'une fin (la mort), certaines technologies, en vieillissant, gagnent en importance et s'éloignent justement d'une létale disparition. Leur vieillesse, leur survie, devient un ancrage dans une infrastructure elle-même vieillissante et néanmoins inéluctable. Ainsi, en vieillissant – ou plutôt en « se faisant vieillir » par des personnes dévouées – OpenPGP crée un pont entre différentes géné-

¹⁴³ <https://github.com/hannob/pgpbugs>.

rations de codeureuses qui s'y sont consacrées, s'y consacrent et s'y consacreront peut-être encore. Ce travail de mise à jour, de négociation et de consolidation est étudié dans la partie suivante de la thèse, dédiée au présent et aux rythmes qui l'habitent.

Hériter, faire vieillir et transmettre le code

Deuxième partie : Les rythmes du présent

La première partie de cette thèse a montré l'importance du passé dans le devenir d'OpenPGP. Il a été question de souligner l'enlisement inévitable des technologies du numérique qui contraint leur maintenance. J'ai montré comment prendre soin d'une technologie consistait à la maintenir en mouvement dans un univers toujours changeant, ce qui contribue irrémédiablement à la rendre de plus en plus complexe, épaisse et enlisée à force d'y apporter des corrections. Parallèlement, j'ai aussi montré les opportunités d'ancrage qu'offrait une infrastructure certes enlisée, mais stable et utilisée : non seulement une telle infrastructure permet certaines innovations, mais, étant « trop enlisée pour faire faillite », elle représente surtout une importante promesse de survie.

Afin de poursuivre l'analyse de la production de la sécurité informatique, il faut désormais analyser une autre dimension du temps, à savoir le rythme. Le rythme, c'est à la fois la fréquence, mais aussi l'intensité de ce mouvement de déséquilibre que j'ai déjà évoqué. Pour ce faire, je me tourne vers cette période liminaire qu'est le présent. Liminaire, car il se situe entre le passé et l'avenir, c'est-à-dire entre ce qui ne pourra pas être modifié et ce qu'il reste à devenir (Lübbe 2009), mais également entre la sécurité et l'insécurité, la vulnérabilité et le patch. Les deux chapitres qui suivent rendent ainsi compte des différents rythmes qui façonnent le travail des codeu-

reuses, développeuses et ingénieur·e-s qui prennent soin de la sécurité informatique : parfois rapide, parfois lent, le présent des mainteneuses d'OpenPGP n'est jamais uniforme.

L'objectif de cette partie est d'interroger concrètement comment les différentes communautés se mettent au travail pour « faire vieillir » cette technologie, et donc « faire durer » la sécurité qu'elle cherche à offrir. Il s'agit de comprendre comment les défaillances du code et du protocole influencent le travail des mainteneuses, et comment leur projet-sécurité doit toujours être réadapté en fonction des failles qui ne cessent d'émerger. Je m'appuie pour cela sur le constat de l'oscillation incessante et pourtant imprévisible entre sécurité et insécurité informatique, qu'ont décrit Laura Kocksch et ses collègues (2018, 12, ma traduction) :

Assurer la sécurité informatique, c'est se préoccuper des oscillations constantes et inattendues entre sécurité et insécurité [...], un effort permanent qui ne trouve jamais de fin et qui doit s'appuyer sur le bricolage et l'expérimentation. [...] la sécurité est dynamiquement liée à l'insécurité. Les efforts déployés pour sécuriser les technologies entraînent généralement de nouvelles insécurités.

Je montre comment le projet-sécurité des développeuses et ingénieur·e-s doit sans cesse être renégocié dans des temporalités fort variables, créant non seulement une oscillation entre sécurité et insécurité, mais également entre urgence et lenteur. Pour ce faire, je prends l'exemple des vulnérabilités d'EFAIL. EFAIL est le nom donné à une série de vulnérabilités étudiées par une équipe d'universitaires (tous des hommes, allemands pour la plupart). Cette série de vulnérabilités affectent OpenPGP et S/MIME, les deux protocoles de chiffrement d'emails les plus répandus. Les scientifiques ont trouvé et mis au point plusieurs moyens d'accéder aux données déchiffrées en insérant dans le cryptogramme un code malicieux. Concrètement, les techniques présentées dans l'article d'EFAIL permettent à l'attaquante Ève de modifier l'email chiffré qu'Alice a envoyé à Bob en ajoutant certains « gadgets de malléabilité » [*malleability gadgets*] qui permettent d'exfiltrer le texte clair. Par exemple, un de ces gadgets de maléabilité consiste à ajouter des parties de code HTML faisant croire au client email de Bob qu'il faut ouvrir un lien. L'ouverture de ce lien envoie l'entier

du contenu déchiffré de l'email sur un serveur dont Ève a le contrôle. Plus généralement, EFAIL joue avec le fait que les données chiffrées par OpenPGP ne sont pas systématiquement authentifiées, ou que certaines implémentations affichent des données dont l'intégrité n'a pas été vérifiée. Une des mesures pour pallier ce problème est l'usage d'un mode de chiffrement authentifié, appelé AEAD, que j'expose dans le chapitre 7 (« Lenteur et consensus à l'IETF », p. 256). Avant cela, il est nécessaire de s'attarder sur l'événement qu'a été la divulgation publique d'EFAIL dans le chapitre 6 (« EFAIL et le temps de l'urgence », p. 226).

EFAIL est un exemple qui n'est paradoxalement pas exemplaire car il ne correspond en rien à un cas typique de divulgation et de correction de vulnérabilités. En effet, pour différentes raisons qui deviendront claires à la lecture de ce qui suit, EFAIL est un contre-exemple de ce qui se passe d'ordinaire, et c'est précisément en cela que cette série de vulnérabilités est intéressante à analyser : elle a donné lieu à de nombreuses discussions à propos de ce qui était en train de se passer et de ce qui aurait dû se passer. La crise d'EFAIL m'a également permis de saisir un paradoxe intéressant : c'est en effet une « panne » qui a causé une fulgurante accélération du rythme de soin de la technologie.

D'une manière plus générale, cette partie interroge ce que le code et ses vulnérabilités font à la sécurité informatique. Plusieurs notions clés des sciences sociales et particulièrement du champ des STS y sont mobilisées : controverse technique, panne, mais aussi gouvernance des normes techniques. Étudier les controverses techniques permet notamment de comprendre comment les connaissances se constituent et sont diffusées. Comme l'a mentionné Michel Callon (2006, 135s), « [d]es mots comme rationalité, logique, preuve, démonstration, cachent [...] l'infinie variété des stratégies et des rapports de force à travers lesquels les sciences se fabriquent réellement ». Les controverses représentent des moments dont l'issue est incertaine et où se cristallisent des tensions entre les différentes actrices en présence. L'étude de tels moments d'incertitude permet également de comprendre comment se cristallisent certaines connaissances (Dousset 2019).

Dans cette optique, les figures de la panne, du dysfonctionnement technique et du bogue informatique sont particulièrement intéressantes à passer à la loupe car elles permettent non seulement de mieux cerner l'étendue des dépendances infrastructurales (Star 1999), mais également de documenter les multiples accidents de parcours propres à toutes innovations. Dans son livre *The Shock of the Old*, Edgerton (2008) affirme que l'innovation sur laquelle se concentre habituellement toute l'attention n'est pas la caractéristique principale des technologies ; bien plus, ce sont les dysfonctionnements, les pannes et la casse en général qui sont la norme. Un appel à développer une « pensée du monde cassé » [*broken world thinking*] (Jackson 2014) s'est d'ailleurs fait entendre et les articles, conférences et colloques sur les pannes et la maintenance qu'elles exigent se sont multipliés lors de la dernière décennie. L'introduction de Mathilde Bourrier et Nicolas Nova du numéro spécial de « Techniques et Culture » sur les pannes s'ouvre par une belle définition de ces pannes :

Panne, incident, accroc, accident, dysfonctionnement, imprévu, bug, épine dans le pied, os, pépin, tuile, sans oublier des termes plus triviaux : emmerdes, couilles, court-jus. Ce sont les marins qui ont donné au XVI^e siècle son origine au mot. « Mettre en panne », c'est orienter la voilure pour que le bateau n'avance plus, mais aussi faire pencher le bateau pour pouvoir colmater une voie d'eau. La « panne sèche », contrairement à ce que l'on croit, ne renvoie pas à la voiture. Là encore, l'expression était en usage dans la marine, quand la mise en panne s'obtenait en l'absence totale de voilure, en utilisant le gouvernail, alors que la panne avec voilure portait le nom de « panne courante ». Si cette terminologie peut s'appliquer à toutes sortes de circonstances du quotidien, c'est à l'action technique, riche en dérapages en tout genre, qu'elle est le plus communément associée : événement imprévu, défaut de fabrication, erreur de fonctionnement. (Bourrier et Nova 2019, 14)

La panne apparaît donc comme une rupture dans le cours normal des choses et le temps de la panne est de l'ordre de l'imprévu – même s'il est quasiment certain qu'une panne aura lieu, on ne peut que supposer le moment, le lieu, l'ampleur et les dégâts qu'elle causera. Bourrier et Nova affirment d'ailleurs que « les cas où les pannes sont acceptées, non comme une calamité éternellement surprenante, mais comme relevant du registre du normal et de l'attendu, demeurent l'exception » (2019, 14). Toutefois, dans le domaine de la programmation et peut-être de manière encore

plus saillante dans la sécurité informatique, les bogues, vulnérabilités et erreurs semblent non seulement courantes, mais surtout attendues, voire recherchées. En effet, le paradigme dominant de la sécurité informatique est construit autour de l'idée de « la sécurité par l'insécurité », que je décris plus en détail dans l'encadré ci-dessous.

Faire et défaire la sécurité en identifiant et patchant des failles : le paradigme de la sécurité par l'insécurité

Le paradigme de la sécurité par l'insécurité fait largement consensus dans le champ de la sécurité de l'information. Cette logique très pragmatique considère qu'il n'est vraisemblablement pas possible de prouver la sécurité informatique¹⁴⁴ et que le seul moyen de sécuriser des logiciels ou des protocoles serait empirique : il s'agit de découvrir les vulnérabilités et de les corriger avant qu'elles ne soient exploitées de manière offensive¹⁴⁵. Cette vision se rapproche du « scepticisme organisé » théorisé par l'un des fondateurs de la sociologie des sciences, Robert King Merton (1942) : tout comme la science, la sécurité informatique avancerait grâce à ses erreurs et surtout au fait que ces dernières soient identifiées et corrigées¹⁴⁶. La divulgation de vulnérabilités permet donc de faire progresser la sécurité informatique dans ce sens : au lieu que ces vulnérabilités restent inconnues, voire qu'elles soient même exploitées par des actrices malveillantes, elles peuvent être corrigées.

Comme le montre David Bozzini (2023) dans son histoire de la divulgation de vulnérabilités, cette manière de divulguer les failles des programmes a cristallisé des tensions entre différentes communautés, par exemple quand les com-

144 Le champ de la sécurité démontrable [*provable security*] se veut une alternative à cette méthode empirique de la sécurité. Toutefois, elle semble bien difficile à atteindre. Voir par exemple la présentation de FJW et Lukas « *Provable Security. How I learned to stop worrying and love the backdoor* » : https://media.ccc.de/v/35c3-9517-provable_security.

145 En faisant notamment recours à des entreprises d'audit, de hacking éthique ou de *bug bounty* (Bozzini 2023).

146 D'ailleurs, les notions de répliquabilité et d'empirisme sont aussi centrales dans la sécurité informatique. En effet, lors d'un rapport de bogue, des informations sont données pour que les développeuses puissent répliquer le bogue sur leur machine.

pagnies poursuivaient en justice les chercheuses de vulnérabilités (voir aussi Goerzen et Coleman 2022). Alors que certaines personnes prônaient la divulgation complète des failles [*full disclosure*] (par exemple Schneier 2007), d'autres arguaient pour un compromis qui respecte mieux le travail des développeuses : la divulgation responsable [*responsible disclosure*]. L'idée est de donner un délai suffisant aux développeuses pour leur permettre de corriger le système, tout en les mettant sous pression avec un délai à partir duquel les vulnérabilités seraient divulguées. En adoptant en 2014 cette stratégie dans son *Project Zero*, Google a participé à institutionnaliser cette pratique et entériner le consensus à propos d'un délai de 90 jours¹⁴⁷ pour corriger la vulnérabilité avant la divulgation publique¹⁴⁸.

La divulgation de vulnérabilités ou de bogues n'est pas toujours un événement aussi médiatisé que l'a été EFAIL. Dans la plupart des cas, les bogues peuvent être résolus en corrigeant le bout du code problématique. Ces processus sont quotidiens et l'on peut en voir la trace sur les pages GitHub et GitLab des projets¹⁴⁹. Ces processus de divulgation se déroulent généralement loin de l'attention générale des utilisateurices, qui, dans le meilleur des cas, laissent échapper un soupir quand il s'agit de mettre à jour un logiciel¹⁵⁰. Cela n'en fait pas pour autant des processus invisibles. Au contraire, les divulgations de vulnérabilités peuvent faire l'objet de réelles mises en scène – que ce soit en ligne comme le cas d'EFAIL ou lors de conférences comme le Chaos Communication Congress, Black Hat ou DefCon (Bozzini 2023)¹⁵¹. Ces processus sont importants pour comprendre la manière dont la sécurité est faite *en*

147 Ce délai a d'ailleurs été récemment revu à la hausse.

148 <https://web.archive.org/web/20230815000511/https://googleprojectzero.blogspot.com/p/vulnerability-disclosure-faq.html>.

149 La page des « *issues* » du projet Sequoia sur GitLab permet de voir quels sont les problèmes auxquels sont confronté-e-s les développeuses : <https://gitlab.com/sequoia-pgp/sequoia/-/issues>. Les solutions sont visibles sous la forme de « *commits* ».

150 Ce qui se fait d'ailleurs de plus en plus automatiquement, sans que l'utilisateurice ne doive intervenir. La plupart du temps, la mention « corrige des failles de sécurité » dans la notice de mise à jour pas totalement inaperçu et n'est en cela que très peu révélatrice du travail que cette correction a nécessité.

151 Notons toutefois que les divulgations lors des conférences sont souvent faites après que les vulnérabilités ont été reportées aux codeuses.

même temps que défaite : en identifiant des failles, on insécurise un système tout en donnant les clés pour l'améliorer. Toutefois, ces processus sont toujours complexes et souvent particuliers.

Finalement, il me semble important de terminer cet encadré sur ce paradigme si important pour la fabrique de la sécurité informatique par une remarque méthodologique : au centre de ces processus se trouve l'objet-frontière¹⁵² probablement le plus important de la sécurité informatique : la vulnérabilité. À la frontière entre les communautés de chercheuses en sécurité (des scientifiques, mais aussi des *pentesteurs* et des *bug hunters*), des codeuses, des participant-e-s au groupe de travail d'OpenPGP, et des utilisatrices, se trouvent les vulnérabilités qui sont au cœur des processus d'(in)sécurisation que j'ai décrit dans le troisième chapitre (« Alice et Bob aux pays des bogues », et en particulier la section 3.2.1 p. 124). Il est donc tout à fait indiqué de tâcher de comprendre les trajectoires de ces vulnérabilités, de leur recherche à leur correction, en passant par leur divulgation et les discussions qu'elles suscitent.

Cette partie propose ainsi d'étudier la trajectoire des vulnérabilités d'EFail, qui permettent d'analyser deux rythmes fort contrastés. D'un côté, il y a eu une divulgation spectaculaire et très controversée desdites vulnérabilités en mai 2018 qui a suscité un sentiment de peur partagé par de nombreuses utilisatrices. Les codeuses ont été sous une grande pression pour proposer des solutions immédiates et la communauté s'est embrasée durant quelques jours. De l'autre côté, il y a eu de longues et nombreuses négociations au sein du groupe de travail de l'IETF afin de mettre à jour les spécifications du protocole. Comme je vais le montrer, ces négo-

152 Pour rappel, voici la définition d'un objet-frontière selon Star et Griesemer (1989, 393, ma traduction) : « Les objets-frontière sont des objets suffisamment plastiques pour s'adapter aux besoins locaux et aux contraintes des divers groupes qui les utilisent, tout en étant suffisamment robustes pour maintenir une identité commune d'un site à l'autre. [...] Ils ont différentes significations dans différents mondes sociaux mais leur structure est suffisamment commune à plusieurs mondes pour les rendre reconnaissables, un moyen de traduction. La création et la gestion d'objets-frontières est un processus clé dans le développement et l'entretien de la cohérence à l'intersection de plusieurs mondes sociaux. »

ciations s'inscrivent dans un processus très long qui durent depuis des décennies et qui ne s'arrêtera probablement pas tout de suite.

Le contraste est saisissant. Il ne doit toutefois pas faire croire qu'il s'est passé beaucoup de choses du côté des implémentations et très peu du côté de l'IETF. En fait, il s'agit justement de questionner l'impermanence de la sécurité et de l'insécurité, en saisissant les rythmes du soin des technologies numériques. Dans certains cas, quelques heures représentent un temps d'attente beaucoup trop long ; dans d'autres, des années de négociations sont tout à fait acceptables.

Les deux logiques – celle de l'urgence et celle de la lenteur – ne s'opposent pas, elles se complètent et font toutes les deux partie intégrante de ce qu'il y a lieu de nommer « la marche du soin de la sécurité informatique ». J'ai cité dans le chapitre méthodologique un extrait du livre « Chronotopie » de François Laplantine (2023, 39) où il est question de la marche en tant que mouvement de « déséquilibre compensé » (cf. p. 95 de la présente thèse). Je propose de considérer les pratiques de réparation et de maintenance des technologies de sécurité informatique comme des mouvements de panne compensée : une panne (ou un bogue, ou une découverte scientifique) cause un déséquilibre dans l'écosystème, déséquilibre qui nécessite un mouvement pour retrouver l'équilibre, comme le codage d'un patch ; or, cette réparation produit très souvent de nouveaux déséquilibres qui nécessitent à leur tour de nouvelles interventions. Il s'agit d'une variation de l'infini recommencement dépeint par Denis et Pontille (2022), du bricolage persistant décrit par Mol et ses collègues (2010), ou encore du dévouement continu qu'exige la sécurité informatique (Kocksch et al. 2018). Ce que mon analyse permet de cerner concerne notamment l'enchevêtrement du passé, du présent et de l'avenir dans cette marche du soin de la sécurité. Ancrée dans le passé et dépendante des événements du présent, cette marche bascule continuellement vers l'avenir, vers les failles à venir et les patches à développer. La notion de rythme permet de rendre compte non pas du mouvement entre un avant et un après, mais de la vitesse et de la manière dont se passent ces changements. Autrement dit, il ne s'agit plus de regarder ce qui enlise l'infrastructure, ni vers

quoi elle tend, mais les multiples gestes effectués pour retrouver l'équilibre, ou du moins en avoir l'impression. Il s'agit donc d'observer la sécurité en train de se faire (et se défaire).

Cette vision contribue également à affiner la compréhension anthropologique de la sécurité informatique en décentrant le regard des discours et des objets pour insister sur les pratiques de lecture, écriture, relecture et réécriture du code informatique et des spécifications des protocoles de sécurité. Mettre au centre de l'attention la tension entre, d'une part, le projet-sécurité et, d'autre part, les pratiques des actrices qui prennent soin des technologies numériques permet de comprendre en quoi la sécurité est une affaire de constante négociation, de compromis, voire, peut-être, de compromission. Il s'agit de négocier à un niveau individuel et collectif une vision pragmatique de la sécurité. Les participant·e·s à ma recherche se sont toujours montré·e·s très conscient·e·s des limites de leur travail. Elles m'ont régulièrement montré de manière explicite comment leur travail consistait à faire en sorte que ces technologies continuent d'être étudiées, patchées et, finalement, utilisées.

6. EFAIL et le temps de l'urgence

En mai 2018, OpenPGP n'avait déjà pas une très bonne réputation et plusieurs cryptographes s'en prenaient régulièrement au protocole pour en dénoncer de nombreux problèmes. Les principales critiques étaient toutefois liées à sa difficulté d'usage et les utilisateurices qui réussissaient à comprendre et maîtriser son fonctionnement pouvaient donc raisonnablement se croire en sécurité : non seulement les primitives cryptographiques semblaient solides, mais la technologie était aussi recommandée par l'Electronic Frontier Foundation (EFF) et Reporters sans frontière (RSF) par exemple. Edward Snowden avait en outre utilisé OpenPGP pour contacter la journaliste Laura Poitras et un tutoriel en ligne lui est même attribué¹⁵³. Finalement, les documents qu'il a révélés indiquaient qu'OpenPGP restait indéchiffrable pour les agences de sécurité états-uniennes (Brandom 2014).

OpenPGP avait donc la réputation d'être une technologie certes vétuste et difficile, voire pénible à utiliser au quotidien, mais fiable malgré tout. Puis, soudain, voilà qu'au matin (en Allemagne) du 14 mai 2018, un professeur en sécurité informatique a publié quatre tweets inquiétants : son équipe de recherche avait trouvé dans OpenPGP et S/MIME des vulnérabilités qu'il qualifiait de sérieuses sans plus donner

153 La vidéo « *GPG for Journalists - Windows edition* | *Encryption for Journalists* | *Anonymous 2013* » postée par « anon108 » est attribuée à Edward Snowden : <https://vimeo.com/56881481>. Un récit des échanges entre Snowden et Poitras a été fait par Micah Lee pour le journal The Intercept sous le titre « *Ed Snowden Taught Me To Smuggle Secrets Past Incredible Danger. Now I Teach You* » : <https://theintercept.com/2014/10/28/smuggling-snowden-secrets/>.

de détails. Comme ces vulnérabilités n'étaient pas encore patchées, les tweets affirmaient que les explications complètes seraient données le lendemain afin de donner le temps à toutes les utilisatrices de prendre des mesures de protection. Un des conseils était par exemple de désactiver les clients OpenPGP et d'utiliser d'autres moyens de communication (illustration 21 ci-dessous). Ces quatre tweets alarmistes ont créé un climat de panique au sein de différentes communautés et une controverse a éclaté.

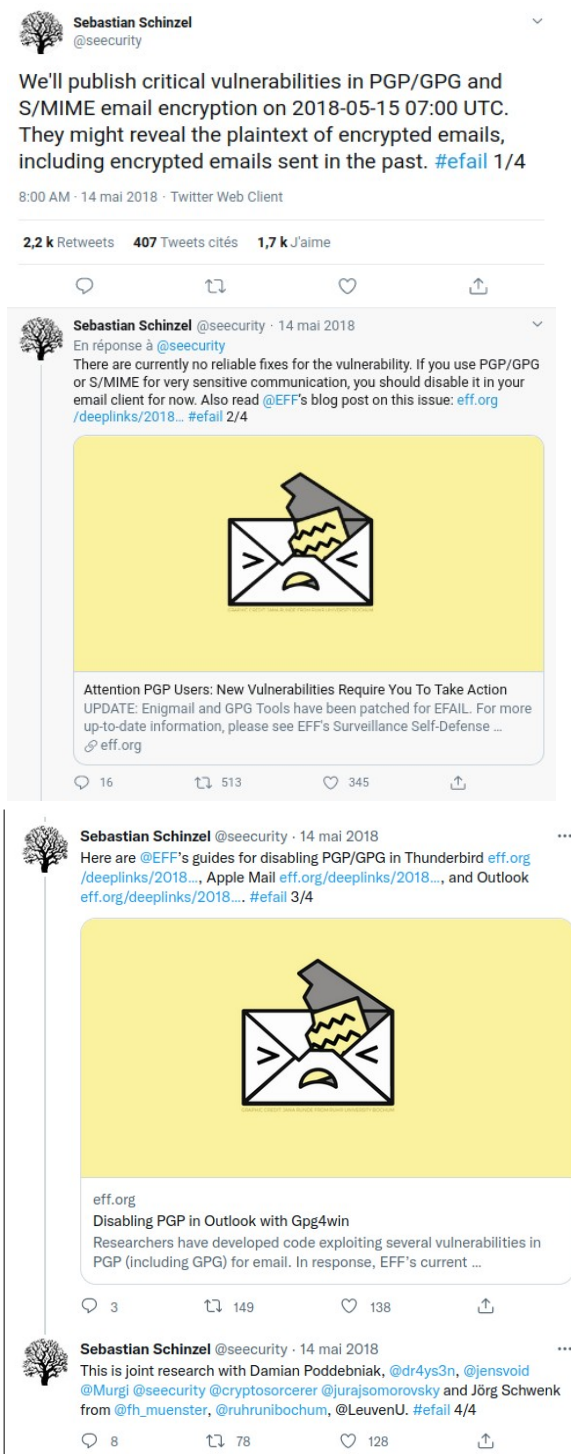


Illustration 21: Le thread du Professeur Sebastian Schinzel annonçant la découverte d'une série de vulnérabilités critiques dans OpenPGP et renvoyant au blog d'EFF. Source : Twitter, consultable sur : <https://cva.unifr.ch/content/sebastian-schinzel-tweet-about-efail-vulnerability>.

Ce chapitre questionne comment la communauté OpenPGP a fait face aux vulnérabilités d'EFAIL et comment le projet-sécurité de nombreuses personnes a été ébranlé par cette divulgation publique. L'exemple de la révélation publique de cette série de vulnérabilités et de la controverse qui a suivi permet de comprendre une nouvelle dimension de la temporalité de la sécurité informatique. En effet, alors que les projets-sécurité des développeuses s'inscrivent dans un temps très long (souvent en termes de décennies), ce temps est sans cesse interrompu par des événements insécurisants. Ces événements peuvent être des nouvelles découvertes, l'émergence d'un virus, le développement d'une nouvelle technologie ou, comme c'est le cas ici, des vulnérabilités. Les développeuses ne sont pas dupes et savent très bien que de tels événements sont à prévoir ; seulement, on ne peut ni prévoir quand ces événements auront lieu, ni quelle(s) partie(s) de la technologie ils affecteront. Aussi, pour tenir la promesse de sécurité qu'elles ont faites tacitement aux utilisatrices de leur logiciel, les codeuses sont sans cesse obligées de réviser leur projet-sécurité, ainsi que le code de leur application et les spécifications du protocole, en fonction d'événements extérieurs à leur travail, les contraignant entre autres à devoir effectuer une veille technologique continue pour s'assurer qu'aucune vulnérabilité connue n'affecte leur code ou celui des bibliothèques utilisées.

Ce chapitre, ainsi que le suivant, analysent les rythmes du quotidien des développeuses d'OpenPGP. EFAIL a en effet été une importante rupture dans la routine des codeuses et étudier cet exemple permet de rendre compte du sentiment d'urgence qui a soudain enflammé une communauté de pratique d'ordinaire plutôt discrète. Dans ce chapitre, j'analyse comment EFAIL a exigé un remaniement rapide du projet-sécurité de nombreuses personnes. Pour cela, il est nécessaire de détailler dans un premier temps les différents épisodes liés à cette série de vulnérabilités, notamment ce qui est survenu durant la période de gestation d'EFAIL, la période de la divulgation et la controverse qui a suivi. De là, je propose une analyse de cette controverse en axant ma réflexion autour de la notion d'incertitude afin de montrer comment la sécurité informatique est une affaire de constants compromis entre dif-

férents projets-sécurité d'une part, et différentes attentes d'autre part. Je soulève également plusieurs questions morales importantes qui touchent à la responsabilité des différentes actrices vis-à-vis des utilisatrices des technologies, à la répartition du travail et, plus généralement, aux « bonnes » et « mauvaises » façons de contribuer à la sécurisation des technologies numériques.

6.1. EFAIL, un (contre-)exemple extraordinaire

Plusieurs éléments me permettent d'établir une chronologie d'EFAIL. Il existe tout d'abord les nombreuses traces laissées sur la toile et ailleurs : tweets, billets de blog, sites Internet, articles de presse, forums, emails, etc. Il y a aussi les articles scientifiques et conférences données par les chercheurs (tous des hommes) qui expliquent les vulnérabilités et répondent aux critiques qui leur ont été adressées. Il existe également des chronologies établies par d'autres personnes, notamment celle de Ptacek (2018)¹⁵⁴ qui retrace les événements publics connus avant la divulgation d'EFAIL. Finalement, je complète ce matériel en ligne par des entretiens que j'ai menés avec différentes chercheuses et développeuses¹⁵⁵.

6.1.1. La période de gestation d'EFAIL

Je n'ai pas beaucoup d'informations quant à cette période. Elle est relativement peu documentée par des sources publiques et les chercheurs ont été relativement avarés en information à ce sujet lors de mes entretiens, sûrement car cela n'avait sans doute rien d'extraordinaire pour eux. En effet, la recherche de vulnérabilités fait partie intégrante de leur quotidien de chercheurs, au même titre que l'enseignement et la rédaction d'une thèse de doctorat et d'articles scientifiques. Je crois pourtant

¹⁵⁴ Il s'agit d'un pseudonyme d'un chercheur que je n'ai pas rencontré.

¹⁵⁵ Cette chronologie peut être consultée dans un essai PECE, sur la plateforme CVA sous le lien suivant : <https://cva.unifr.ch/efail-timeline>. Cet essai regroupe les différentes sources que je mentionne dans ce chapitre, notamment des copies d'écran de différents tweets. Pour des raisons de confidentialités, certains tweets ont été anonymisés, car c'était leur contenu qui m'importait et pas spécialement les auteures. Pour cette raison, je n'ai pas contacté individuellement les auteures de ces tweets pour demander leur autorisation de reproduire ces tweets ici ; de plus, s'ils et elles décidaient de supprimer ces publications de la plateforme, on ne pourrait pas remonter jusqu'à eux et elles.

avoir assez d'informations pour faire le récit de ce qu'il s'est passé avant la divulgation publique.

Lors d'une discussion informelle (et non enregistrée) durant le 35^e Chaos Communication Congress, en décembre 2018, un des chercheurs m'a dit que la recherche d'EFAIL en tant que telle s'était précisée en été 2017. Je sais également qu'un des chercheurs principaux a rejoint l'équipe en 2016 pour un contrat doctoral et qu'EFAIL tient une place centrale dans sa thèse. Je n'ai donc pas de « date de naissance » précise d'EFAIL et c'est pourquoi je me contente de faire arbitrairement démarrer cette chronologie le 15 août 2017 – 9 mois avant la divulgation publique, temps de gestation commun chez l'*homo sapiens sapiens*.

Les personnes directement concernées par la recherche d'EFAIL sont le professeur Sebastian Schinzel qui a dirigé l'équipe en question, ainsi que Damian Poddebniak, Christian Dresen et Fabian Ising de la Münster University of Applied Sciences, Jens Müller, Juraj Somorovsky et Jörg Schwenk de la Ruhr University Bochum, et Simon Friedberger de NXP Semiconductors en Belgique (Poddebniak et al. 2018). À ces co-auteurs de l'article d'EFAIL, il faut encore ajouter Marcus Brinkmann qui a rejoint l'équipe en cours de route et qui a participé aux autres articles publiés dans la suite d'EFAIL (Müller, Brinkmann, Poddebniak, Böck, et al. 2019; Müller, Brinkmann, Poddebniak, Schinzel, et al. 2019; Müller et al. 2020; Schwenk et al. 2020). Rappelons-nous aussi que les chercheuses ne produiraient pas grand-chose sans le soutien essentiel du personnel administratif et d'entretien des locaux, bien souvent absent du « générique » de la recherche scientifique (Knorr-Cetina et Mulkay 1983; Knorr-Cetina 1995; Latour et Wooglar 1988).

EFAIL est le fruit d'un travail collectif qui a commencé à mûrir dès l'été 2017. La raison pour laquelle les chercheurs ont décidé de rechercher des vulnérabilités spécifiquement dans OpenPGP et S/MIME sont variables : par exemple, un des chercheurs m'a dit qu'il aimait OpenPGP et qu'il voulait contribuer à sa santé en recherchant des vulnérabilités sérieuses afin qu'elles puissent être patchées avant que des personnes mal intentionnées puissent les utiliser à des fins malveillantes. Un autre chercheur

m'a dit qu'il aimait la cryptographie et qu'il trouvait intéressant d'un point de vue scientifique et intellectuel d'y chercher des vulnérabilités, bien que dans la plupart des cas – et c'est également le cas d'EFail – les vulnérabilités trouvées n'attaquent pas directement les primitives cryptographiques mais la manière dont celles-ci sont implémentées. Un des chercheurs m'a également donné la justification suivante par email :

Nous avons déjà pas mal analysé TLS et y avons trouvé des vulnérabilités critiques¹⁵⁶. OpenPGP, S/MIME et TLS partagent certaines constructions cryptographiques et nous voulions savoir s'il y avait des attaques similaires qui fonctionneraient contre OpenPGP et les emails.
(Chercheur d'EFail, email personnel, 27 septembre 2019.)

Cette citation montre un aspect intéressant de la recherche de vulnérabilités : les chercheurs semblent avoir procédé par déduction, par « instinct ». Ils ont deviné qu'en tant que protocole d'une même époque, OpenPGP, S/MIME et TLS avaient de grandes probabilités de partager certaines vulnérabilités et ils ont donc creusé dans cette direction. Quoi qu'il en soit, à l'automne, les premiers résultats semblaient prometteurs et les chercheurs ont commencé à en parler discrètement à différents milieux concernés. Il s'agissait toutefois d'être très prudents, car non seulement ils ne voulaient pas que l'existence de ces vulnérabilités s'ébruite pour ne pas compromettre la sécurité des utilisatrices, mais ils redoutaient également que d'autres chercheuses universitaires « concurrentes » leur volent leurs idées.

Le 25 octobre 2017, l'équipe d'EFail a pris contact avec l'équipe de développement de Thunderbird concernant Efail¹⁵⁷. Le rapport de bogue numéro 1411592 marque la première prise de contact de l'équipe d'EFail avec des développeuses. Il est intéressant de constater qu'à ce stade, Efail est traité comme un bogue ordinaire et c'est à travers le canal usuel que l'équipe d'EFail a contacté Thunderbird. Ce rapport de bogue est toujours confidentiel à ce jour (été 2023) mais sa trace est tout

156 Pour rappel, nous utilisons des dizaines de fois le protocole TLS par jour : il s'agit notamment du « s » de « HTTPS ». (Pour plus de détails à propos de la recherche en question, voir : Aviram et al. 2016).

157 <https://cva.unifr.ch/content/mozilla-foundation-security-advisory-2018-13>.

de même visible et documentée (Ptacek 2018). Par ailleurs, un mois plus tard (dès le 23 novembre 2017), d'autres échanges sont documentés et intégralement accessibles¹⁵⁸.

Il me semble important de revenir ici sur un cas particulier qui a fait couler beaucoup d'encre – ou plutôt briller de nombreux pixels sur nos écrans d'ordinateurs et de téléphones. Il s'agit de la manière dont la vulnérabilité a été traitée au sein des équipes de développement de GnuPG, Thunderbird (client email de Mozilla) et Enigmail (plugin OpenPGP de Thunderbird). En effet, les chercheurs ont été accusés de n'avoir pas divulgué ces vulnérabilités à ces implémentations – ce qui n'est pas le cas. Werner Koch, fondateur et principal mainteneur de GnuPG, a lui-même confirmé que Sebastian Schinzel lui avait écrit à propos d'EFAIL le 24 novembre 2017¹⁵⁹. Dans un email envoyé sur la liste de diffusion des utilisatrices de GnuPG, Werner Koch explique pourquoi il a considéré qu'EFAIL n'affectait pas GnuPG (ni la librairie GPGME). Il a notamment précisé qu'un message d'erreur était affiché lorsque les attaques d'EFAIL étaient testées et qu'il considérait donc que la faute revenait aux implémentations qui semblaient ignorer ledit message d'erreur.

Puis, en février 2018, l'équipe d'EFAIL a contacté la plupart des implémentations qui étaient vulnérables. Les équipes de développement de ces implémentations ont reçu certaines informations sur les vulnérabilités d'EFAIL ainsi qu'un délai de 90 jours pour sécuriser leur programme, avant que les vulnérabilités d'EFAIL soient rendues publiques. Les enjeux de cette période de divulgation privée sont très sensibles : il faut à la fois aviser un maximum de personnes concernées et en même temps réduire au minimum le nombre de personnes ayant connaissance de cette recherche afin de minimiser le risque de fuites d'information. En effet, si des détails venaient à tomber dans les mains de personnes mal intentionnées, ces dernières pourraient les utiliser dans de véritables *exploits* avant que la divulgation n'ait eu lieu¹⁶⁰. Un chercheur

158 <https://www.mozilla.org/en-US/security/advisories/mfsa2018-13/>.

159 <https://cva.unifr.ch/content/efail-or-openpgp-safer-smime-w-koch-email>.

160 Selon plusieurs personnes familières avec OpenPGP, cela n'a visiblement pas été le cas durant la longue période entre le 25 octobre 2017 et le 14 mai 2018.

d'EFAIL m'a raconté les réactions très variées de différentes équipes de développement :

[Une entreprise] nous a par exemple répondu : «Non, c'est un problème théorique, nous n'allons pas corriger quoi que ce soit » ; ce qui est une formulation malheureuse mais je peux comprendre parce que... qu'est-elle supposée faire ? Elle ne peut pas corriger le protocole à elle seule, l'IETF doit le faire. [Une autre entreprise], par exemple, a reproduit la vulnérabilité dans les deux semaines dans leur service email et nous a même payé une récompense [*bounty*], même si nous n'avons pas demandé. Donc c'est un autre extrême.
(Chercheur d'EFAIL, entretien du 27 décembre 2018)

Finalement, il me faut aussi mentionner que durant cette période, les chercheurs ont finalisé leur article scientifique qu'ils ont envoyé au comité de rédaction de la conférence USENIX¹⁶¹. Je ne retrouve aucune trace documentée concernant cette étape : elle est pourtant cruciale car les publications scientifiques sont centrales dans l'agenda des chercheurs. Par ailleurs, il était important que l'article scientifique soit validé par la communauté scientifique avant que la vulnérabilité devienne publique, notamment pour des raisons d'embargo.

Il s'agit, jusqu'ici, d'un cas relativement normal de procédure pour une divulgation de vulnérabilités par une équipe de scientifique : une équipe a recherché une vulnérabilité dans des systèmes cryptographiques, en a trouvées, puis a cherché un équilibre entre préserver le secret autour de leur découverte tout en tâchant d'en faire part à toutes les personnes concernées, afin que celles-ci corrigent les codes vulnérables. Certaines équipes de développement des logiciels vulnérables se sont montrées intéressées et reconnaissantes, et d'autres n'ont pas considéré que les vulnérabilités trouvées étaient sérieuses. Il n'y a pour le moment rien de surprenant ni d'étonnant ni dans la démarche des chercheurs, ni dans les réactions variées des développeuses car ce type de procédure existe depuis les années 1990 (Bozzini 2023). L'étonnement et la controverse sont arrivées par la suite, lors de la divulgation publique d'EFAIL.

¹⁶¹ <https://www.usenix.org/conference/usenixsecurity18/presentation/poddebniak>.

6.1.2. L'accouchement d'EFAIL et la stupéfaction générale

Pour une grande majorité de personnes, dont je fais partie, les vulnérabilités d'EFAIL sont « nées » le 14 mai 2018, à 8 heures du matin en Allemagne. C'est à ce moment que le professeur Sebastian Schinzel a publié les quatre tweets reproduits dans l'introduction de ce chapitre (illustration 21 p. 228). Il est crucial de noter que ces tweets ne disent rien de la nature de ces vulnérabilités bien que leur ton soit alarmiste. L'usage des termes « vulnérabilités critiques » et le conseil de désactiver (temporairement) le module PGP/GPG ou S/MIME dans les clients email renforcent ce sentiment de gravité. Par ailleurs, le titre du premier billet de blog d'EFF conseille aux utilisatrices d'agir, ce qui est rarement le cas : d'ordinaire, les utilisatrices n'ont que peu de marge de manœuvre, si ce n'est mettre à jour leur logiciel. Pour comprendre pourquoi les chercheurs ont choisi de divulguer EFAIL de cette façon, il faut se figurer le dilemme face auquel les chercheurs se sont retrouvés, qui a très bien été décrit par Matthew Green :

La divulgation de bogues affectant PGP est particulièrement délicate. Cela tient au fait qu'il n'existe rien de tel que "PGP". Ce que nous avons à la place, c'est une communauté vaste et distribuée qui tourne autour du protocole OpenPGP. Le pilier de cette communauté est le projet GnuPG, qui maintient l'outil principal GnuPG et les bibliothèques sur lesquelles de nombreux clients s'appuient. Il existe ensuite toute une série de clients de niche basés sur une interface graphique et de projets de plugins de messagerie. Enfin, il y a les vendeurs commerciaux comme Apple et Microsoft. [...]

La solution évidente au problème de la divulgation est d'utiliser une divulgation échelonnée. Vous informez d'abord les grands fournisseurs commerciaux, car c'est là que se trouvent la plupart des utilisateurs concernés. Ensuite, on descend dans la "longue file" des projets open source, sachant qu'inévitablement, l'embargo pourrait être rompu et que tout le monde devra appliquer des correctifs dans l'urgence. Et vous gardez à l'esprit que, *quoi qu'il arrive, tout le monde vous reprochera d'avoir fait échouer la divulgation*. (Green 2018, ma traduction, en italique dans le texte original)

Concrètement, après avoir repoussé plusieurs fois la date de la divulgation publique, et alors qu'ils m'ont dit avoir craint que des attaques utilisant leur recherche ne voient le jour, les chercheurs ont décidé d'alerter les utilisatrices pour éviter qu'elles soient mises en danger par cette divulgation (conversation privée non enre-

gistrée). De plus, la date de la présentation de la conférence USENIX approchait et il était important de faire avancer la recherche. Or, afin de prévenir les utilisatrices sans leur faire courir un trop grand risque, ils ont opté pour une stratégie en deux temps. Dans un premier temps, ils ont voulu avertir la communauté *via* Twitter qu'ils avaient trouvé des vulnérabilités, sans révéler aucun détails à leur sujet mais en conseillant une mesure d'atténuation : la désactivation temporaire des modules OpenPGP des clients email. Cette solution, bien que drastique, avait le double avantage d'être efficace sans pour autant dévoiler des informations à propos de la nature des vulnérabilités. Puis, dans un deuxième temps, le plan prévoyait de publier les détails des vulnérabilités sur un site web, 24 heures après cet avertissement. Dans un email privé, Sebastian Schinzel me donnait la raison de ce choix :

Nous avons des exploits triviales qui permettaient d'exfiltrer des centaines de cryptogrammes grâce à un seul email malicieux, et beaucoup de vendeurs ne semblaient pas considérer cela sérieusement. Nous avons déplacé déjà deux fois le délai et nous voulions révéler les failles publiquement avant qu'elles ne fuitent. Pour prévenir des attaques immédiates après la révélation, nous avons décidé de publier un avertissement public 24 heures avant la révélation en tant que telle. Cet avertissement disait de désactiver temporairement le déchiffrement automatique dans les clients email. Beaucoup de personnes ont compris que nous recommandions d'envoyer des emails en clair.
(Sebastian Schinzel, email privé)

Pour améliorer les chances de succès de cette stratégie, les chercheurs ont coordonné cette annonce avec l'Electronic Frontier Foundation (EFF), l'une des organisations les plus réputées en matière de protection des droits sur Internet, qui a publié un billet de blog expliquant comment se protéger provisoirement contre EFAIL¹⁶². Lors d'un entretien avec un des chercheurs, ce dernier m'a raconté en détail ce qu'il s'était passé. Cette idée de coordonner la révélation de la faille avec EFF en deux temps a été d'abord considérée comme une mauvaise idée par EFF et les chercheurs

¹⁶² Une question légitime se pose : pourquoi les chercheurs ont choisi de coordonner la divulgation avec une ONG comme EFF plutôt qu'avec un CERT. Un CERT (acronyme de *Computer Emergency Response Team*) est un centre d'alertes et de réactions aux attaques informatiques, destiné aux entreprises ou aux administrations, mais dont les informations sont généralement accessibles publiquement. Je n'ai pas la réponse, mais fais l'hypothèse que les chercheurs ont considéré qu'étant proche des milieux journalistiques et activistes, EFF aurait plus de poids et de visibilité qu'une institution officielle et gouvernementale comme les CERT.

avaient renoncé à le faire. Cependant, le lendemain, les personnes d'EFF avaient changé d'avis et ont décrété que c'était finalement une bonne idée. Je ne dis pas cela pour blâmer EFF d'avoir grandement influencé la décision des chercheurs mais pour insister sur le fait que ni les chercheurs, ni EFF, ne savaient vraiment quelle était la meilleure solution. Leur objectif commun était d'agir au mieux pour protéger les utilisatrices, malgré les incertitudes et les contraintes. Plusieurs chercheurs m'ont ainsi dit avoir agi en aveugle, uniquement guidés par le souci de protéger les utilisatrices.

Le choix de cette stratégie en deux temps en collaboration avec une organisation à but non lucratif semble relativement insolite dans le champ de la sécurité de l'information et ne correspond pas aux schémas traditionnellement utilisés. Ni les chercheurs d'EFAIL, ni EFF, ni toutes les autres personnes avec qui j'ai eu l'occasion d'en parler n'avaient déjà opéré de la sorte – et une des leçons tirées par Sebastian Schinzel est d'ailleurs de ne plus jamais procéder ainsi (Schinzel 2018). Il est indiscutable que cette stratégie a très mal été comprise et qu'elle s'est révélée plus délétère qu'efficace.

Il faut aussi dire qu'EFAIL était un cas spécial. Tout d'abord, EFAIL ne correspond pas à une seule vulnérabilité dans une seule implémentation et à un seul niveau ; il s'agit de plusieurs vulnérabilités qui affectent le code de certains logiciels (niveau des implémentations), deux protocoles de chiffrement (OpenPGP et S/MIME), et le fonctionnement du code HTML dans les emails (niveau de l'infrastructure email en général). Contactées séparément, les implémentations ne pouvaient répondre qu'à leur propre niveau, et ne corriger par conséquent qu'une partie des problèmes. Le reste devait être géré au niveau du protocole lui-même, ce qui prend nécessairement beaucoup de temps. Par ailleurs, il semble quasiment impossible de supprimer l'usage du code HTML dans les emails tant cet usage est devenu totalement institué¹⁶³. Cela

¹⁶³ Et ce, parfois même à des fins de sécurité. Depuis une année, l'Université de Fribourg ajoute une bannière HTML à tous les emails reçus de l'extérieur de l'institution informant les lectrices que cet email est potentiellement dangereux. Ayant configuré mon client email pour qu'il affiche uniquement les versions textes des emails (et non le code HTML), je me retrouve avec des emails « sabotés » par l'usage de cette bannière.

montre les contraintes auxquelles étaient confrontées l'équipe de recherche et les équipes de développement.

Comme je l'ai déjà mentionné, cette stratégie de communication n'a pas été comprise et de nombreuses utilisatrices, développeuses et expert·e·s en sécurité ont fortement critiqué cette annonce à cause du flou qu'elle a fait planer. En effet, de nombreuses questions ont suivi cette annonce, comme par exemple : est-ce que mes anciens emails chiffrés sont compromis ou seulement les nouveaux ? Est-ce qu'il s'agit de faille dans les algorithmes utilisés ? Quelles sont les implémentations vulnérables ? À ces questions, les chercheurs répondaient laconiquement : « plus d'informations demain ». Toutefois, des informations – plus ou moins exactes et précises selon les cas – ont rapidement commencé à circuler, et des pans entiers de l'article à fuiter. Ainsi, quelques heures à peine après la publication de l'avertissement, les chercheurs ont décidé d'anticiper la divulgation complète des failles prévue le lendemain et de publier l'article et le site internet, toujours *via* Twitter (voir illustration 22 ci-dessous) :



Illustration 22: Tweet annonçant la divulgation complète des vulnérabilités. On remarquera que la raison invoquée par Sebastian Schinzel est le fait que leur article ait commencé à fuiter. Source : <https://twitter.com/seecurity/status/995964977461776385>.

Les chercheurs ont donc divulgué publiquement et dans tous leurs détails les vulnérabilités d'EFAIL 3 heures et 52 minutes après les premiers tweets. Quelques minutes plus tard, un autre chercheur d'EFAIL a publié sur Twitter le *pre-print* de leur article avec pour seul commentaire la mention « divulgation complète des attaques sur S/MIME et OpenPGP »¹⁶⁴. Bien entendu, ce rétropédalage a amplifié la confusion

¹⁶⁴ <https://twitter.com/jensvoid/status/995966114466234369>.

autour de cette divulgation, d'autant plus qu'EFF et qu'une partie des implémenteurs d'OpenPGP se trouvaient en Amérique du Nord où il faisait encore nuit. Pour celles et ceux qui ont découvert tout cela au réveil, le choc a dû être particulièrement rude.

Le site Internet d'EFAIL contenait une explication générale du contexte des failles d'EFAIL, des explications sur le fonctionnement technique des différentes attaques, des clarifications sur les mesures d'atténuation proposées par les chercheurs et finalement, une FAQ et sur la manière de s'en prémunir¹⁶⁵. Quelques heures plus tard, visible sur la capture de 15h15 de la page, une entrée supplémentaire « *Responsible Disclosure* » a été ajoutée dans la FAQ pour répondre aux accusations de divulgation immorale et imprudente. Cette rubrique montre sous forme de tableau quelles sont les implémentations vulnérables, quand celles-ci ont été contactées et, le cas échéant, quel est le numéro du rapport de bogue en question (voir illustration 23 ci-dessous).

165 Le site Internet d'EFAIL est toujours en ligne à l'heure actuelle, mais a été mis à jour avec, notamment, les nouvelles publications et liens vers les conférences des chercheurs. Il peut être consulté dans sa version du 14 mai 2018 à midi sous le lien suivant : <https://web.archive.org/web/20180514120008/https://efail.de/>.

Attacks on PGP clients :

Product	First contact	Case number
Outlook 2007 / GPG4Win	Out of support	
Outlook 2010	–	
Outlook 2013	–	
Outlook 2016	–	
The Bat!	–	
Postbox / Enigmail	2018-03-21	
eM Client	2018-02-27	
Thunderbird / Enigmail	2017-10-25	Bugtracker: 1411592
Evolution	–	
Trojitá	–	
KMail	–	
Claws	–	
Mutt	–	
Apple Mail / GPGTools	2018-02-16	
MailMate	–	
Airmail / GPGTools	2018-02-16	
Canary Mail	–	
K-9 Mail	–	
R2Mail2	2018-03-10	
MailDroid / Flipdog	2018-02-27	
Nine	–	
United Internet	–	
Mailbox.org	–	
ProtonMail	–	
Mailfence	–	
Roundcube / Enigma	2018-03-28	
Horde IMP / GnuPG	2018-03-21	
AfterLogic	–	
Rainloop	–	
Mailpile	–	

■ Exfiltration channel (no user interaction required)
■ Not vulnerable

Direct exfiltration attacks :

Product	First contact	Case number
Thunderbird	2018-02-10	Bugtracker: 1419417
Apple Mail	2018-02-10	Follow-up: 684760367
iOS Mail	2018-02-10	Follow-up: 684760367
Postbox	2017-11-21	Request: 114513
MailMate	2018-02-10	

■ Exfiltration channel (no user interaction)
■ Exfiltration channel (with user interaction)

Illustration 23: Tableaux des différents clients email vulnérables aux attaques d'EFail. Source: <https://efail.de/>.

Sur le site Internet d'EFail, les chercheurs ont précisé quatre mesures d'atténuation des vulnérabilités. La première mesure à court terme est celle qui a créé la polémique en premier lieu : elle recommande de temporairement ne plus déchiffrer d'emails dans les clients email vulnérables. En lisant correctement et jusqu'au bout cette recommandation, il est évident que les chercheurs ne recommandaient pas d'arrêter d'utiliser OpenPGP, mais de déchiffrer les emails en question en dehors des clients email (par exemple en utilisant directement l'outil GnuPG et non un client email). La deuxième mesure, également à court terme, concerne le code HTML dans les emails que les chercheurs recommandaient d'arrêter d'utiliser. Cette mesure sem-

blait aller de soi pour bon nombre de commentateurices, mais elle est en même temps inapplicable tant le code HTML est institué dans l'infrastructure email. Désactiver le code HTML implique certes une diminution drastique de la surface d'attaque mais également une perte importante de fonctionnalités des emails. La troisième mesure d'atténuation, à moyen terme, était de patcher les logiciels vulnérables. Finalement, à long terme, les chercheurs ont préconisé une quatrième mesure : modifier le protocole, ce que je détaille dans le chapitre suivant.

Ainsi, ces quatre mesures d'atténuation semblent légitimes et cohérentes. Toutefois, la réalité est toujours plus complexe que la théorie¹⁶⁶. L'exemple d'Enigmail est parlant. En effet, l'extension Enigmail avait déjà patché son logiciel avant la révélation des failles et avait été avertie de la divulgation publique. Or, le logiciel restait vulnérable sous certaines conditions. Après la divulgation publique et face à l'ampleur de la controverse, Enigmail a republié une nouvelle version qui devait répondre aux vulnérabilités d'EFAIL. Mais Hanno Böck, un cryptographe et journaliste, a encore trouvé une manière de contourner ces mesures d'atténuation et a affirmé qu'Enigmail demeurerait encore vulnérable et que l'extension seule ne pouvait pas résoudre tous les problèmes puisque certaines mesures devaient être prises dans le client email Thunderbird (voir illustration 24 ci-dessous). En ce sens, patcher ne pouvait se faire qu'en coordonnant les différents niveaux du code imbriqués les uns dans les autres (la librairie GPGME, l'extension Enigmail et le client email Thunderbird).

¹⁶⁶ Et c'est pourquoi je me dis parfois qu'il vaudrait mieux vivre en Théorie, car en Théorie, tout se passe toujours bien.



Illustration 24: Tweet du journaliste et cryptographe Hanno Böck à propos d'un correctif d'Enigmail. Le deuxième tweet est important car il souligne que l'extension Enigmail ne peut pas proposer de solution à elle seule.

Source : <https://twitter.com/hanno/status/997138771194859521>.

6.1.3. La controverse d'EFAIL dans la peur, l'incompréhension et la colère

La première annonce d'EFAIL a suscité de très nombreuses réactions sur les réseaux sociaux et forums spécialisés. Comme les détails des attaques en question restaient dans un premier temps inconnus, de nombreuses spéculations ont été émises. Les chercheurs étaient déjà critiqués à ce moment, d'autant plus qu'ils refusaient catégoriquement de répondre aux questions qui leur étaient posées en affirmant simplement que tout deviendrait plus clair le lendemain. Puis, la divulgation complète a eu lieu et a amplifié cette controverse naissante. Certaines personnes ont félicité les chercheurs pour leurs découvertes, d'autres posaient des questions, et d'autres encore commentaient de manière plus ou moins critique la situation. L'illustration 25 ci-dessous donne quelques exemples de réactions. J'ai choisi cet extrait car il donne un aperçu assez nuancé des différentes réactions. J'encourage cependant les lecteurices curieuses à ouvrir le lien de référence et à lire les différentes réactions, dont certaines semblent particulièrement agressives.

EFAIL et le temps de l'urgence

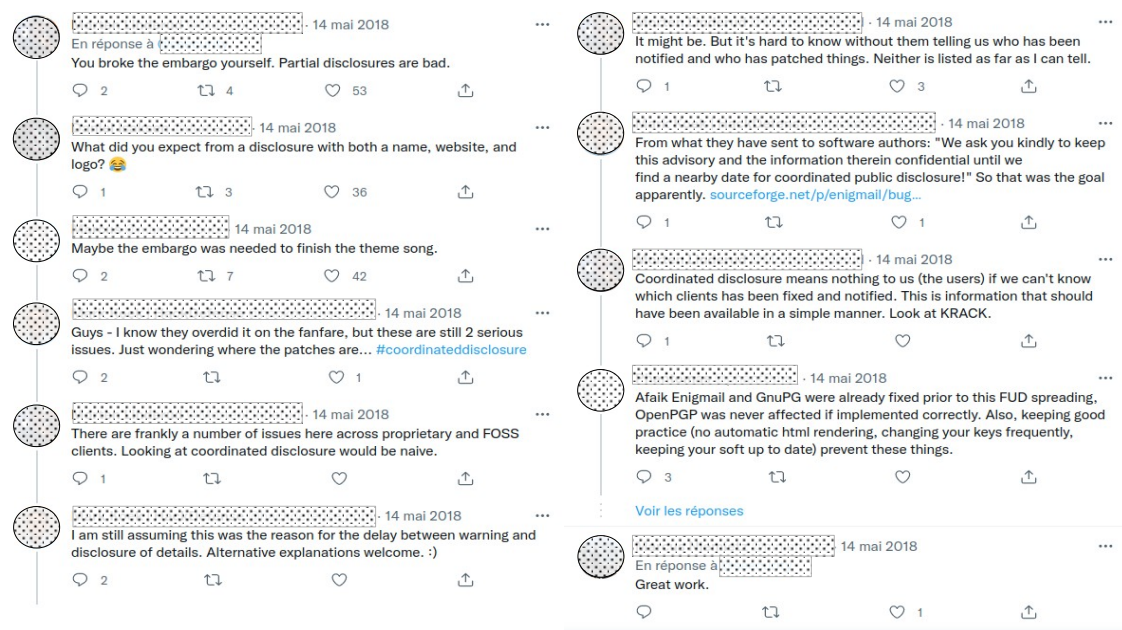


Illustration 25: Exemples de tweets après la révélation complète des vulnérabilités d'EFAIL. On constate que le débat n'est pas aussi tranché et qu'il existe des critiques, des appréciations positives et des questionnements sur la stratégie des chercheurs.

Source : <https://twitter.com/seecurity/status/995964977461776385>.

Il est intéressant de constater qu'un autre type de réactions a consisté à minimiser la portée de ces failles. Ces tentatives de minimisation contrebalançaient – sans pour autant l'annuler – le mouvement de panique suscité par les premiers tweets de Schinzel. Un exemple typique est celui d'un *thread* sur Twitter de Dan Guido affirmant que les attaques d'EFAIL sont élégantes du point de vue intellectuel mais stupides du point de vue opérationnel car très faciles à détecter et archivées dans la boîte mail¹⁶⁷. Un autre exemple de minimisation est illustré par la citation ci-dessous, extrait d'une déclaration officielle de GnuPG envoyée sur la liste de diffusion des utilisateurs de GnuPG, toujours ce 14 mai 2018 :

Cet article est mal nommé. Ce n'est pas une attaque sur OpenPGP. C'est une attaque contre des clients email cassés qui ignorent l'avertissement de GnuPG et qui font des choses folles après avoir été prévenus.

Cette attaque vise des clients email bogués. L'utilisation correcte du MDC empêche complètement cette attaque. GnuPG supporte le MDC depuis l'été 2000.

Les auteurs ont fait une liste de clients email bogués. Ça vaut la peine de regarder cette liste de clients email (qui se trouve à la toute fin) pour voir si le vôtre est vulnérable. Mais soyez

¹⁶⁷ Voir <https://cva.unifr.ch/content/example-reactions-efail-twitter-dan-guido>.

prudent·e·s, parce qu'il se pourrait que ce ne soit pas exact – par exemple, MailPile affirme ne pas être vulnérable alors que l'article indique que MailPile a certaines vulnérabilités.
(R. J. Hansen 2018, ma traduction¹⁶⁸)

La formulation désinvolte de la citation ci-dessus est intéressante en ce qu'elle contraste fortement avec le ton alarmiste et spectaculaire de l'annonce conjointe des chercheurs et d'EFF. Hansen contredit les chercheurs en affirmant que les failles d'EFAIL ne concernent pas OpenPGP mais des implémentations « cassées ». L'expression « clients email bogués » contraste d'ailleurs avec les « vulnérabilités critiques » des chercheurs. Par ailleurs, il prévient ironiquement ses lecteurices en leur demandant d'être prudent·e·s non pas par rapport à leurs emails, mais à la manière de lire l'article car il affirme que les chercheurs n'ont pas été précis dans leur rapport.

Ce type d'affirmations contradictoires s'est d'ailleurs répété ailleurs : par exemple, le célèbre magazine Wired titrait un article en annonçant la mort d'OpenPGP : « *We're calling it: PGP is dead* » (Katwala 2018) dont le sous-titre enjoignait les utilisateurices à utiliser Signal à la place des emails chiffrés. Parallèlement, ProtonMail affirmait l'inverse : « *No, PGP is not broken, not even with the Efail vulnerabilities* » (Yen 2018). Ainsi, suite à la divulgation complète des vulnérabilités d'EFAIL, les articles de presse, billets de blog et autres publications se sont multipliées et EFAIL est restée sur le devant de la scène médiatique du champ de la sécurité de l'information durant plusieurs jours, voire plusieurs semaines. Avant de passer à la section suivante, je résume dans l'encadré ci-dessous les principaux points contentieux de la controverse.

Les principaux points contentieux de la controverse d'EFAIL

- **Gravité des failles d'EFAIL :** la gravité des failles d'EFAIL n'a cessé d'être remise en question, de même que la cible même de ces attaques. Comme le montre l'exemple de la citation de Hansen ci-dessus, plusieurs personnes ont affirmé que l'article des chercheurs était mal nommé et que les attaques ne concernaient pas OpenPGP et S/MIME mais l'usage du

168 Dans la citation ci-dessus, j'ai souligné en gras les formulations qui ont été reprises dans différents billets de blogs et tweets par la suite.

code HTML dans les emails, les mauvaises implémentations de clients email défectueux ou encore une version très ancienne de GnuPG. À l'inverse, d'autres personnes ont insisté sur l'importance et la sévérité de ces vulnérabilités, notamment pour le domaine des entreprises et du journalisme d'investigation. Finalement, certaines implémentations ont reconnu les risques liés à EFAIL tout en affirmant ne pas être vulnérables à ces attaques.

- **La responsabilité et l'éthique des chercheurs :** la manière dont a été gérée la divulgation publique d'EFAIL a été abondamment et parfois brutalement critiquée. D'une part, en révélant ces vulnérabilités sans qu'elles ne soient patchées, les chercheurs ont été accusés de faire primer leur intérêt personnel et leur carrière au détriment de la sécurité des milliers d'utilisatrices d'OpenPGP et S/MIME. Cette critique a été renforcée par une communication à ce sujet jugée excessive de par les moyens mobilisés (nom et logo amusants pour les vulnérabilités, site internet, tweets alarmistes, etc.). D'autre part, cette stratégie de communication des chercheurs a été critiquée également en vertu des mesures de mitigation notamment relayées par l'EFF. En effet, le conseil de désactiver (temporairement) les modules OpenPGP et S/MIME dans les clients email a été compris comme une injonction à ne plus du tout chiffrer ses emails. Or, de nombreuses personnes m'ont affirmé qu'elles considéraient que dans la plupart des cas – et notamment le cas d'EFAIL où l'attaque demande déjà un certain niveau de privilège, comme l'accès aux emails chiffrés de la victime – un mauvais chiffrement était préférable à aucun chiffrement. L'illustration 26 donne un exemple de ces attaques sur Twitter. Il faut toutefois noter que les chercheurs ont également été soutenus, notamment par les cryptographes Bruce Schneier (2018) et Matthew Green (2018) qui accusent respectivement l'organisation de l'infrastructure et de la communauté OpenPGP d'être à la source du problème. Green a aussi écrit dans un *thread* sur Twitter :

Donc en résumé, les clients PGP sont vulnérables parce que 17 ans après la découverte d'une vulnérabilité, les mesures d'atténuation n'ont pas été intégrées par défaut dans GnuPG et la défense a été "laissée aux clients PGP", qui sont aussi un bouc émissaire com-mode lorsque les choses tournent mal.¹⁶⁹

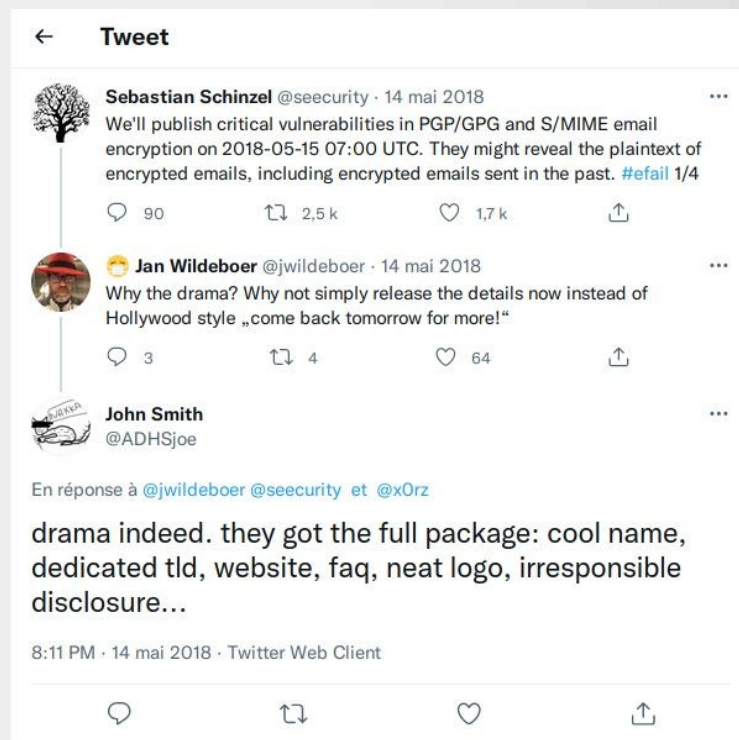


Illustration 26: Exemple de critiques à la suite de la révélation publique d'EFAIL. Le « marketing » autour de la divulgation des failles a été notamment très critiqué et les chercheurs accusés d'avoir dramatisé la situation alors qu'ils auraient pu agir de manière plus discrète et efficace, selon ces critiques.

Source : <https://twitter.com/ADHSjoe/status/996090542751657984>.

- **La surmédiation d'EFAIL :** une autre critique a été faite autour de la surmédiation d'EFAIL. En cela, on peut dire que d'une certaine manière, la communication des chercheurs a été efficace dans le sens où leur recherche a été comparée aux grandes découvertes qui sont sorties des cercles confidentiels du champ de la sécurité de l'information (Brook 2020). Cependant, de nombreuses personnes ont critiqué ce succès pour des raisons variées. Par exemple, on peut comprendre l'agacement, voire le découragement, ressenti par certaines développeuses, notamment

¹⁶⁹ https://twitter.com/matthew_d_green/status/995996706457243648, ma traduction.

de GnuPG et d'Enigmail (qui étaient les principales implémentations touchées par la controverse) : engagées parfois bénévolement depuis des années en faisant un travail « dans l'ombre », elles se sont tout à coup retrouvées sur le devant de la scène. Leur contre-attaque, si on peut l'appeler ainsi, peut alors être considérée comme une réponse sur la défensive à une attaque considérée comme illégitime. D'autre part, lors d'une discussion informelle, un-e cryptographe m'a dit regretter qu'EFAIL ait pris tant de place car d'autres vulnérabilités toutes aussi sérieuses ont été occultées. Cette personne a par exemple mentionné deux vulnérabilités découvertes l'une à la suite de l'autre, affectant la messagerie Signal (voir par exemple : Khandelwal 2018a; 2018b; Haworth 2018). Cette importante médiatisation est aussi intéressante en ce qu'elle contraste avec le mode opératoire dominant propre aux communautés de pratiques de la sécurité informatique qui demeurent d'ordinaire « invisible » pour la plupart des utilisatrices ; cependant, cette médiatisation n'est pas nouvelle dans le domaine de la sécurité de l'information et de nombreuses révélations complètes (*full disclosures*) avaient déjà attiré l'attention des médias *mainstream* (Goerzen et Coleman 2022; Bozzini 2023). Ce qui change avec EFAIL est le format hybride du mode opératoire, qui n'est ni une révélation complète puisque les chercheurs ont tenté de donner 24 heures aux utilisatrices pour prendre des mesures de précaution, ni une révélation responsable puisqu'ils n'ont pas attendu que tous les systèmes soient patchés. L'illustration 26 ci-dessus montre bien l'incompréhension de leur stratégie de communication.

- **La communication entre les différentes communautés :** il s'agit d'un quatrième « méta-point » qui, d'une certaine manière, englobe les trois points ci-dessus. Il a été dit que les chercheurs n'ont pas bien communiqué, ni avant la divulgation, ni pendant, ni après ; certaines personnes ont également affirmé qu'EFF aurait également dû communiquer aussi bien avec les chercheurs qu'avec les codeuses¹⁷⁰ ; GnuPG et Enigmail ont

170 Voir par exemple le tweet de Robert J. Hansen qui affirme que : « [...] EFF n'a apparemment consulté que les auteurs d'EFAIL avant de rédiger son communiqué de presse. Elle n'a consulté aucun membre de la communauté OpenPGP. » <https://twitter.com/robertjhansen/status/>

aussi été critiquées pour n'avoir pas donné plus d'informations à leur utilisatrices¹⁷¹ ; et, d'une manière générale, on m'a aussi régulièrement dit, bien que de manière informelle, que tous les problèmes venaient du fait de la « toxicité » de la communauté OpenPGP « avec qui tu ne peux pas discuter calmement » (discussion informelle avec un-e cryptographe de la communauté OpenPGP). La communication semble avoir été un problème majeur durant cette révélation.

6.2. L'incertitude comme moteur de la controverse, et le sentiment d'urgence comme combustible

L'exemple d'EFAIL permet de comprendre trois fonctionnements cruciaux de la sécurité informatique : (1) l'oscillation permanente et persistante entre sécurité et insécurité, (2) l'assignation de rôles moraux et l'élaboration en conséquence de pratiques de soin à court et long termes et (3) la co-construction du savoir et du non-savoir. Ces trois aspects sont discutés dans cette section en dialogue avec un article intitulé « *Caring for IT Security* » de Laura Kocksch et ses collègues (2018). À travers quatre vignettes, cet article identifie trois enjeux de la sécurité informatique, à savoir la permanente oscillation entre sécurité et insécurité, la négociation de responsabilités et la « logique du soin » (en mobilisant notamment Mol 2009). Cet article décrit la sécurité informatique comme un « objet de sollicitude » [*a matter of care*] qui se construit de manière très concrète à travers des pratiques où bricolage, expérimentation et improvisation se mêlent à des réglages techniques :

La sécurité informatique n'est pas un problème technique qui peut être élaboré et implémenté d'une manière simple. La sécurité informatique est rarement une tâche en soi. La sécurité informatique est plutôt le résultat d'efforts dispersés parmi un ensemble hétérogène d'acteurs, une réalisation fragile et tempo-

[996388432812355589](https://doi.org/10.1017/9781009055589.009).

171 Par exemple, Matthew Green a affirmé « Une mauvaise communication de la part de GnuPG et d'Enigmail. À la date de la divulgation, toute personne cherchant des informations précises sur la sécurité de deux projets majeurs - GnuPG et Enigmail - n'aurait pas été en mesure de les trouver. » (Green 2018, ma traduction).

raire qui se déroule sur différents sites et à travers des frontières organisationnelles, légales et professionnelles. En d'autres termes, le soin de la sécurité informatique se situe généralement dans l'*entre-deux* [*in-between*] : elle a lieu entre les personnes et les choses, car la prise en charge est, par nature, une question d'établissement de relations. (Kocksch et al. 2018, 15, ma traduction)

Cette vision a profondément influencé ma manière de donner du sens à mes observations ethnographiques. La controverse qui a suivi la divulgation d'EFAIL est un cas particulièrement adapté pour questionner ces différents points. Comme je l'ai exposé dans un article (Besençon 2022), et comme je le montre dans le chapitre suivant, les pratiques d'écriture et de réécriture du code informatique qui ont suivi la divulgation d'EFAIL signalent effectivement une constante oscillation entre sécurité et insécurité : des patches étaient par exemple codés pour corriger certaines vulnérabilités, mais ces correctifs se sont révélés eux-mêmes sources d'insécurité. Par ailleurs, la controverse d'EFAIL souligne également de nombreuses discussions sur la moralité des différentes actrices, tant dans ce qui est attendu d'elles que ce qu'elles-mêmes considèrent comme leur devoir ou leur motivation à poursuivre leur travail de soin. J'aimerais ici apporter quelques nouveaux arguments à l'analyse de Laura Kocksch et ses collègues.

Une des mes propositions concerne la négociation du savoir et du non savoir qui permet de comprendre comment se mettent en place certaines logiques de soin. En effet, au cœur du paradigme de la sécurité par l'insécurité, se trouve la connaissance des insécurités. Une première étape consiste à établir son propre « projet-sécurité », autrement dit définir ce que l'on désire protéger, et comment. Pour les développeuses, il s'agit de comprendre comment traduire la triade CIA en protocole puis en code informatique (cf. section 3.1 p. 108) ; pour les utilisatrices, il s'agit d'établir un modèle de menaces (Kazansky 2021), puis d'identifier quels outils permettent de se protéger contre les menaces en questions. La deuxième étape consiste à connaître les vulnérabilités qui affectent le protocole, les codes et/ou les usages, et d'y pallier. Cela nécessite une veille permanente pour surveiller l'état de santé des technologies concernées, et également la correction des problèmes le cas échéant. Cela implique également un lien de confiance entre les utilisatrices et les développeuses des

technologies en question, d'où l'assignation de rôle dans les diverses tâches de maintenance.

Le cas de la divulgation des vulnérabilités d'EFAIL et de la controverse qui a suivi est intéressant car il met en évidence un élément clé qui montre la fragilité de la sécurité informatique en tant que procédure : même s'il existe certains consensus au sein des communautés de développement à propos de la meilleure manière de gérer les bogues et vulnérabilités, les « *best practices* » ne sont jamais qu'un modèle qu'il s'agit d'adapter à la situation en question. Ces procédures *ad hoc* sont dépendantes du contexte, de la gravité des vulnérabilités, des ambitions des personnes qui divulguent et de la réaction des développeuses et des utilisatrices. Ces nombreuses variables, qui sont autant d'inconnues pour les personnes engagées, font qu'il est souvent difficile de savoir quelle procédure marchera le mieux.

6.2.1. L'assignation des responsabilités des différentes actrices

Quand des problèmes émergent – ce qui ne manque jamais d'arriver –, la réaction doit être rapide et efficace. J'utilise ici le verbe « devoir » pour souligner à la fois un devoir moral exprimé par de nombreuses codeuses (« il faut protéger les utilisatrices ») et les attentes des utilisatrices elles-mêmes (« il faut que le code soit mis à jour »). Il existe aussi d'autres justifications, comme un sens du devoir professionnel (« c'est mon travail, je dois le faire correctement pour que mon entreprise ne subisse pas de conséquences »), un devoir lié à l'image de marque de l'entreprise ou de l'équipe de développement (« nous devons tenir nos promesses de sécurité envers nos utilisatrices ») ou encore de la propre réputation de la codeuse en charge d'implémenter le patch (« si je corrige ce code très vite et de manière efficace, j'aurai tenu mes promesses et secouru de nombreuses personnes »). Mon objectif n'est pas tant de lister toutes les motivations personnelles mais de montrer qu'à l'instar des actions du quotidien, il y a plusieurs logiques de justifications (Boltanski et Thévenot 1991)¹⁷², et que ces justifications sont d'ordre moral. Il s'agit de faire le bien, d'agir

172 Luc Boltanski et Laurent Thévenot ont appelé ces logiques des « cités ».

non seulement en fonction de son éthique personnelle, mais aussi en fonction des attentes des autres, qui jugeront si l'on a bien ou mal agi.

Comprendre ce qui est attendu des différentes actrices impliquées est essentiel pour saisir un élément crucial de la production de la sécurité informatique : la répartition des tâches ne fait pas consensus. L'une des façons de s'en rendre compte est d'analyser les critiques qui ont été émises lors de cette controverse.

Commençons par prendre la perspective des codeuses et des utilisatrices qui ont critiqué la manière de faire des chercheurs. Ces derniers ont été accusés, entre autres, de n'avoir pensé qu'à leur propre carrière et d'avoir fait passer leur intérêt avant celui des milliers d'utilisatrices qui se retrouvaient potentiellement en danger. Concrètement, la majorité de ces critiques portait sur la manière d'avoir gérer la communication avec les différentes communautés. Les chercheurs ont été accusés d'avoir divulgué des vulnérabilités graves de manière inconsidérée et irresponsable, en usant de moyens de mise en scène particulièrement décriés (annonce alarmiste sur Twitter, nom de la vulnérabilité, site Internet, logo). Ces critiques font ainsi voir une attente morale vis-à-vis des scientifiques, qui ne devraient pas tirer un profit personnel de leur recherche. Le rôle de recherche de vulnérabilités à l'intérieur du système n'est pas remis en cause ; c'est plutôt la manière de les communiquer qui, elle, a posé problème.

Prenons maintenant un autre point de vue, celui de la communauté scientifique justement. Tout d'abord, il faut remarquer que l'accueil de l'article d'EFAIL et de ceux qui ont suivi a été plutôt favorable, laissant penser que la contribution de ces recherches était utile et importante d'un point de vue scientifique. De plus, deux cryptographes – tous deux issus du monde académique et qui ont l'habitude d'épiloguer sur l'actualité de la sécurité informatique – ont commenté la divulgation d'EFAIL en affirmant que ce n'était pas de la faute des chercheurs si la controverse avait été si grande (Green 2018; Schneier 2018). Matthew Green a par exemple affirmé que « malgré ce que vous avez entendu, étant donné la complexité de la divulgation, très peu de choses ont mal tourné » (Green 2018, ma traduction).

Selon les deux cryptographes, et mes entretiens avec avec les chercheurs semblent le confirmer, l'équipe d'EFAIL a effectivement tenté de divulguer ces vulnérabilités de la manière qu'ils considéraient la plus sûre et responsable pour les utilisatrices, à savoir une divulgation coordonnée suivant le modèle de la divulgation responsable. Ils ont contacté certaines implémentations plus de 200 jours avant la divulgation publique et ils ont repoussé à plusieurs reprises cette dernière voyant que les logiciels étaient toujours vulnérables. Finalement, ils ont coordonné la divulgation avec EFF. Or, cela n'a été ni vécu de la sorte ni vécu comme une divulgation responsable par de nombreuses développeuses et certaines utilisatrices qui ont eu très peur de ce qui pouvait arriver.

Ils insistent notamment sur un changement important qu'EFAIL laisse percevoir dans le champ de la sécurité de l'information : l'infrastructure, l'imbrication des différents niveaux de code et les technologies elles-mêmes sont devenues si complexes que les mécanismes qui, jusqu'ici, semblaient garantir une certaine stabilité dans le champ de la sécurité de l'information ont commencé à ne plus fonctionner. Selon Bruce Schneier, « l'application de patch commence à échouer » (Schneier 2018, ma traduction), dans le sens où il devient de plus en plus compliqué de corriger des logiciels et de gérer la complexité des technologies tant différents niveaux de codes sont imbriqués. Ce problème nous apprend quelque chose de fondamental sur la fabrique de la sécurité, que Bruce Schneier a très bien explicité :

Tout cela témoigne de la difficulté de coordonner la divulgation des vulnérabilités lorsqu'elle implique un grand nombre d'entreprises ou - ce qui est encore plus problématique - de communautés sans attribution claire de leur responsabilité. Et c'est ce qui se passe avec OpenPGP. La situation est encore pire lorsque le bogue implique l'interaction entre différentes parties d'un système. Dans ce cas, PGP ou S/MIME ne posent aucun problème en soi. La vulnérabilité est plutôt due à la manière dont de nombreux programmes de messagerie électronique traitent les messages cryptés. GnuPG, une implémentation d'OpenPGP, a décidé que le bogue n'était pas de sa faute et n'a rien fait pour y remédier. On peut dire que c'est vrai, mais ce n'est pas pertinent. GnuPG devrait corriger le problème. (Schneier 2018, ma traduction)

Ce qui ressort de cette citation est la difficulté de plus en plus importante de parvenir à assigner des responsabilités claires en matières de tâches de sécurité. Cela re-

joint un argument de Moxie Marlinspike (2016), le fondateur de Signal : l'écosystème est en train de changer et les fédérations sont arrivées à la limite de ce qu'elles pouvaient offrir en terme de sécurité. Selon lui, les infrastructures fédérées sont devenues si complexes et si imbriquées dans des normes interdépendantes qu'il est préférable d'opter pour une architecture centralisée où les changements peuvent être rapidement implémentés. Cet argument fait certes l'impasse sur un point crucial, à savoir qu'il n'est pas si facile d'abandonner une infrastructure sur laquelle repose de nombreux programmes utilisés quotidiennement par des milliers, voire des millions d'utilisateurs (par exemple les emails ou la signature des *commits*). Toutefois, il souligne avec force un fait notable : dans un écosystème aussi complexe qu'OpenPGP, le manque d'assignation formelle des responsabilités vis-à-vis de certaines vulnérabilités fait que cette assignation émerge donc de critères éthiques par définition très personnels.

6.2.2. Le savoir et le non savoir dans la gestion de la controverse

Que s'est-il passé dans les jours qui ont suivi la divulgation publique d'EFAIL ? Une manière de poser un regard anthropologique sur cette période de quelques jours est de mobiliser la notion heuristique d'incertitude, une notion clé des études sur les controverses sociotechniques. Je me réfère ici notamment à la définition proposée par Laurent Dousset (2019). Il suggère que « l'incertitude est d'abord et avant tout un état individuel ou un état d'individu qui se trouve confronté à une dissonance entre son horizon des attentes et la réalité observée ou vécue » (Dousset 2019, 25s). Contrairement à la notion de risque qui est socialement et culturellement construite (Beck 2008; Adam 2000), l'incertitude ne s'observe qu'au niveau des individus bien que « les raisons pour lesquelles des incertitudes émergent dans la conscience et la vie des gens sont fondamentalement sociales » (Dousset 2019, 26). Dousset précise que l'incertitude naît d'une carence de convention et qu'elle est le produit d'un fait social inattendu. Dousset indique aussi que si l'incertitude est individuelle, la situation d'incertitude est, elle, sociale. Elle amène les individus à la résoudre lors de moments d'élucidation : qualifier l'objet ou la situation inconnue, l'expliquer, mesurer les

risques et finalement « trouver un consensus autour de la redéfinition d'un rapport stable entre objet et chose » (Dousset 2019, 29). C'est alors que se fixer sur de telles situations d'incertitude est porteur pour les sciences sociales : en effet, l'élicitation de la situation permet de repérer ce qui ne correspond pas à la marche « normale » du monde et ce qui *aurait dû* se passer si tout avait été normal. EFAIL a représenté une rupture dans le projet-sécurité de nombreuses personnes, qui ont non seulement eu peur pour leurs données, mais qui ont aussi eu besoin de comprendre ce qu'il fallait faire pour améliorer la situation. Ces personnes se sont donc concertées et ont tenté d'éliciter ce fait social inattendu et surtout incompris qu'était EFAIL.

L'incertitude portait sur de multiples points. En effet, des utilisatrices ne savaient pas si leurs données chiffrées étaient vulnérables, et sous quelles conditions ; des codeuses d'implémentations ne savaient pas comment réagir face à ces annonces ; et de nombreuses observatrices dont je faisais partie ont montré beaucoup de curiosité quant aux failles ou à la situation elle-même, ne sachant pas ce dont il en retournait. Cette curiosité était d'autant plus attisée que les attaques de cryptographie exercent souvent une grande attirance et le lexique utilisé dans les tweets de Schinzel laissait présager le pire. Il est important de noter ici que l'incertitude n'était pas la même pour chacun et chacune : était-ce sérieux ? était-ce si grave que cela ? À qui devait-on faire confiance ? Qui devait-on écouter ? La situation d'incertitude a été créée à partir de l'accumulation des différentes incertitudes individuelles qui touchaient différents aspects de la divulgation d'EFAIL, et elle a été alimentée par l'urgence de savoir, de comprendre, de se rassurer et de se croire à nouveau en la sécurité. Ainsi, le savoir et le non savoir jouent un rôle crucial dans la compréhension de ce qu'est la sécurité informatique, de la manière dont elle est créée et ce qu'elle crée en retour.

Il y avait encore une incertitude sur ce qui allait résulter de cette situation d'incertitude : est-ce qu'OpenPGP résisterait au choc ? Est-ce que des solutions à long terme seraient apportées ? Des éléments de réponses à ces questions sont donnés dans la prochaine section, et surtout dans le prochain chapitre.

7. Lenteur et consensus à l'IETF

Dans leur ouvrage « Le soin des choses » (2022), Jérôme Denis et David Pontille ont montré le dévouement, l'attention sensible et le corps à corps que le « peuple de la maintenance » engage pour prendre soin des choses. Ils soulignent aussi le manque de reconnaissance envers le travail de maintenance qui est décrit comme ordinaire, non-événementiel et consistant en un « infini recommencement » des mêmes tâches de soin : par exemple, pour maintenir la signalétique du métro parisien, il ne suffit pas de simplement remplacer un panneau abîmé par un panneau neuf, il faut souvent faire des bricolages sur mesures en fonction de la situation (Denis et Pontille 2010). À propos de ce manque de reconnaissance, ils citent les propos de John Oliver, l'animateur d'une émission populaire aux USA, qui propose d'imaginer le jeu Lego de la maintenance. Sous la forme d'un pont déjà assemblé, ce jeu serait offert à un enfant par un parent qui lui expliquerait le but en ces termes : « Il est livré construit. Tu le maintiens, et si tu le fais correctement, rien ne se passe. Et à la fin, tu meurs » (2022, 46). Le fait que « rien ne se passe » est dans ce cas quelque chose de tout à fait positif car cela montre que la maintenance fonctionne. Cette image du jeu Lego consistant à faire en sorte que « rien ne se passe » est donc très efficace pour montrer à la fois la difficulté de *voir* le travail de maintenance, puisque rien ne se passe, et la durée forcément très longue de ce travail, puisqu'il est sans fin :

[L]e travail de maintenance est sans fin. Il ne fait jamais événement, il ne s'organise pas autour d'une disjonction entre deux états du monde, il se déploie dans

les interstices des jours et des nuits, où rien ne semble se passer. Il n'est pas l'exception à la routine, l'écart à la trajectoire linéaire du monde, mais ce qui est continuellement effectué afin que la routine et la linéarité soient possibles. Il génère la continuité, il la cultive. (Denis et Pontille 2022, 51)

Ce chapitre cherche à contribuer à cette réflexion sur la continuité offerte par la maintenance, continuité qui n'est ni donnée, ni facile, ni gratuite. Dans le chapitre précédent, j'ai analysé la manière dont les différentes communautés liées à OpenPGP ont réagi suite à la divulgation publique d'EFAIL, en insistant particulièrement sur le sentiment d'urgence et la controverse qui en a découlé. EFAIL semblait être une rupture radicale dans la « vie » d'OpenPGP. Or, à l'époque de cette divulgation publique, une chose m'avait particulièrement frappé : alors que la confusion régnait à la suite à cette divulgation, le groupe de travail OpenPGP de l'IETF restait totalement impassible à l'agitation ambiante. Quelques emails continuaient d'être échangés, mais ils semblaient sans lien direct avec EFAIL et la liste de diffusion du groupe de travail OpenPGP de l'IETF n'a pas été utilisée pour discuter de ces failles dans les jours et semaines qui ont suivi la révélation. La seule allusion à EFAIL que j'y ai trouvée était ironique : un des membres de la liste avait envoyé quelques mois plus tôt un email présentant une potentielle menace affectant OpenPGP et proposé une solution. Il n'avait reçu qu'une seule réponse. Le 21 mai 2018, trois jours après la divulgation d'EFAIL, il a renvoyé un email commençant ainsi :

Pas de retour à ce sujet ? Peut-être devrais-je créer un site Internet et un logo pour une furtive attaque de transfert ?¹⁷³

(Membre du groupe de travail OpenPGP, email sur la liste de diffusion, 21 mai 2018)

Mise à part cette remarque ironique, EFAIL était absente de cette liste de diffusion. Ici, le fait qu'« il ne se passait rien » n'était pas un point positif car une réaction était attendue. Je me souviens avoir été très étonné du fait que les chercheurs affirmaient que la solution à long terme se situait au niveau du protocole lui-même et que le groupe de travail ne semblait pas réagir à cette divulgation. Ce manque de réaction semblait paradoxal. Cela peut certainement s'expliquer en partie par un certain malaise face à EFAIL, en partie à cause des multiples niveaux où cette série de vulnérabi-

173 <https://mailarchive.ietf.org/arch/msg/openpgp/BFPS8sikd1dF-7QKrJrwJruScU/>.

lités devait être patchée : affirmer qu’il fallait faire quelque chose, c’était sans doute concéder qu’EFAIL devait être prise au sérieux, ce que les développeuses n’étaient sans doute pas toutes prêtes à admettre. Mais cette lenteur n’est pas que fortuite et semble au contraire structurelle. D’ailleurs, cinq ans plus tard, un autre paradoxe me frappe : contrairement à ce qui a été affirmé à maintes reprises par beaucoup de personnes différentes, OpenPGP n’est pas (encore) mort à l’heure où j’écris ces lignes. Bien au contraire, il s’est passé beaucoup de choses – et cela semble être une bonne chose. De nouvelles implémentations ont vu le jour, des nouvelles vulnérabilités ont été divulguées et corrigées, le groupe de travail a été reformé et s’est donné une nouvelle mission, et de nombreuses propositions ont été faites pour continuer à améliorer OpenPGP¹⁷⁴. Et en même temps, la solution à long terme pour corriger EFAIL n’a toujours pas été approuvée et adoptée par consensus au sein du groupe de travail de l’IETF. Cela indique la capacité de persistance d’OpenPGP malgré des vulnérabilités qui elles-mêmes persistent. Toutefois, contrairement à la logique de l’obstination de Denis et Pontille (2022), la sécurité d’OpenPGP n’est pas donnée. Elle est le fruit de ce qui convient d’appeler le travail du soin.

Dans ce chapitre, j’analyse d’une part le travail de négociation des spécifications du protocole, et d’autre part les outils de gouvernances du groupe de travail. Denis et Pontille ont affirmé qu’« [en] faisant durer les choses, on y fabrique toujours de l’histoire » (2022, 180). L’analyse des activités du groupe de travail OpenPGP après la divulgation d’EFAIL permet de raconter l’histoire de l’évolution d’une technologie déjà vieille et qui continue de vieillir. Ce chapitre vise ainsi à décrire un élément clé de la production de la sécurité informatique : sa lenteur. En effet, il est coutume d’associer sécurité et urgence. Or, ce chapitre montre comment la sécurité se construit aussi dans l’épaisseur d’un temps long, notamment en négociant différents projets-sécurité.

174 Dans la troisième partie de cette thèse, j’en donne quelques exemples notamment autour de la confidentialité persistante et de l’implémentation d’algorithmes résistants aux ordinateurs quantiques.

7.1. Le groupe de travail OpenPGP en prise avec EFAIL

Avant de regarder en détail ce qu'il s'est passé à l'IETF après la divulgation d'EFAIL, il me semble nécessaire de présenter le groupe de travail tel qu'il se présentait en 2018 avant la divulgation. En effet, bien que la liste de diffusion continuait d'être utilisée comme moyen de communication et de discussion entre les différentes implémentations d'OpenPGP, ce groupe de travail était officiellement dissout au sein de l'IETF. Pour mieux comprendre la situation, il faut remonter dans le temps, en commençant par le début. En 1996, l'IETF a publié le RFC 1991, qui n'est autre que la description par Zimmermann et Atkins du fonctionnement de PGP¹⁷⁵. Un groupe de travail a alors été constitué et son travail a débouché sur la publication du RFC 2440 en 1998, qui est devenu la première norme OpenPGP, implémentée par PGP et GnuPG. Ce RFC a été mis à jour en 2007 (RFC 4880¹⁷⁶), qui est toujours la version officielle de la norme à l'heure où j'écris ces lignes. Cette version a néanmoins été actualisée par deux RFC : le RFC 5581 datant de 2009 qui décrit l'usage d'OpenPGP avec le Camellia Cipher¹⁷⁷, et le RFC 6637 datant de 2012 qui définit l'usage de la cryptographie sur des courbes elliptiques pour OpenPGP¹⁷⁸.

En 2015, le groupe de travail a été une nouvelle fois constitué afin de mettre à jour la norme. Cependant, n'ayant pas réussi à aboutir aux objectifs fixés dans le temps imparti, il a été officiellement dissout en novembre 2017. Sa liste de diffusion a toutefois continué de servir de moyens de communication entre les différentes implémentations et Werner Koch, en tant qu'éditeur du groupe de travail, publiait également régulièrement une nouvelle version du brouillon du document nommé RFC4880bis, qui correspondait au brouillon des nouvelles spécifications¹⁷⁹. Ce brouillon n'avait pas de statut officiel, mais était largement implémenté par les différentes implémentations. La plupart des éléments dans ce document était discutée, notamment sur la

175 Ce RFC n'est pas encore un standard mais une simple description. <https://www.ietf.org/rfc/rfc1991.txt>.

176 <https://www.ietf.org/rfc/rfc4880.txt>.

177 <https://www.rfc-editor.org/rfc/rfc5581>.

178 <https://www.ietf.org/rfc/rfc6637.txt>.

179 <https://datatracker.ietf.org/doc/draft-ietf-openpgp-rfc4880bis/>.

page GitHub du document, même si certains changements étaient contestés au sein du groupe de travail.

Par ailleurs, il n'y avait plus de réunions du groupe de travail lors des rencontres de l'IETF et aucun nouveau RFC ne pouvait être publié tant que le groupe n'était pas à nouveau reconstitué officiellement. Ainsi, lors de la divulgation publique d'EFail, la norme OpenPGP était dans un état de « status quo ». Il s'agit d'une formule émiqque pour désigner l'état dans lequel se trouvait alors le groupe de travail. Ce dernier continuait d'échanger, mais sans espoir d'obtenir de grands changements à long termes car le groupe était dissout et aucun changement ne pouvait être officialisé par une nouvelle norme.

En même temps, malgré le status quo, les implémentations suivaient tout de même la plupart des spécifications du document rfc4880bis comme norme officielle. Cela n'est au final pas très étonnant. En effet, l'IETF, l'organe de normalisation d'OpenPGP et de nombreux autres protocoles d'Internet, n'a pas vocation à édicter des normes contraignantes et suivre les spécifications d'OpenPGP (ou tout autres normes de l'IETF) n'est pas une obligation mais une opportunité : celle d'utiliser un système que d'autres utilisent également. Cette opportunité ne vient toutefois pas sans contraintes : il faut se mettre d'accord avec ce que font d'autres personnes, même si tout n'est pas consigné dans la norme officielle. Le document rfc4880bis représentait ainsi l'état actuel du compromis pragmatique auquel étaient parvenues les différentes équipes de développement qui participaient au processus. Certains points étaient certes contestés, mais il était préférable de s'y astreindre que d'être incompatibles. Cela ouvre nécessairement à des manipulations de la part des implémentations les plus puissantes, mais assure en même temps l'interopérabilité.

Quand une entreprise comme Proton décide d'utiliser une telle norme pour implémenter le chiffrement de bout en bout dans les emails, il ne s'agit pas d'une décision contrainte par une obligation. D'autres entreprises, comme Tutanota¹⁸⁰, ont décidé de mettre au point leur propre système de chiffrement ce qui leur permet d'être très in-

180 <https://tutanota.com/fr/encryption>.

dépendantes dans les choix architecturaux, mais qui les empêchent en même temps d'être compatibles avec d'autres systèmes comme OpenPGP ou S/MIME. Ainsi, suivre une norme de l'IETF, c'est mettre la priorité sur l'interopérabilité avec d'autres services. D'un point de vue sécuritaire, cela implique d'accepter de ne pas pouvoir intervenir sans consultation des autres implémentations sur le protocole, ainsi que de suivre les changements opérés – même s'ils ne sont pas officialisés. C'est donc le pragmatisme qui domine : si certaines implémentations particulièrement influentes comme GnuPG ou ProtonMail opèrent certains changements, il est probable que d'autres implémentations doivent s'y plier. Cela donne beaucoup d'importance à la coordination entre les différents partis en présence et les relations de pouvoir entre eux. Finalement, je souligne que dans une configuration telle le groupe de travail OpenPGP, la contrainte ne vient pas tant de la norme elle-même, mais de la coordination entre les différents partis.

Dans le reste de ce chapitre, je décris plus en détail les différents points évoqués rapidement ci-dessus. L'exemple d'EFail et des négociations qui ont suivi est une porte d'entrée idéale qui permet d'analyser différentes dynamiques de gouvernance au sein du groupe de travail. Il est donc nécessaire de bien comprendre quels problèmes a soulevé EFail, ce que je décris dans la section suivante.

7.1.1. L'identification d'un des problèmes d'EFail : la *chunk size* de l'AEAD

Comme je l'ai déjà mentionné, la publication publique d'EFail n'a dans un premier temps guère suscité de réactions au sein du groupe de travail OpenPGP de l'IETF. Rappelons-nous qu'à ce même moment, plus d'une centaine d'emails étaient par exemple échangés sur la liste des utilisatrices de GnuPG¹⁸¹, sans parler de Twitter et des autres sites mentionnés dans le chapitre précédent. EFail brillait donc par son absence sur cette liste de diffusion et il a fallu plus d'un mois avant qu'un premier message ne soit envoyé à propos d'EFail. Cet email a été écrit par Marcus Brinkmann, l'un des chercheurs d'EFail¹⁸², qui m'a dit lors d'un entretien que l'absence de

181 <https://lists.gnupg.org/pipermail/gnupg-users/2018-May/thread.html#start>.

182 Son nom ne figure pas parmi les auteurs de l'article scientifique car il a rejoint l'équipe en cours de route ; cependant, il m'a assuré lors d'une interview avoir participé à ces recherches et a d'ailleurs

réactions au sein du groupe de travail le préoccupait et que c'était la raison qui l'avait poussé à envoyer cet email. Ainsi, le samedi 30 juin 2018, à 16h00 précise, Marcus Brinkmann a cliqué sur « envoyer ». Dans la ligne de l'objet de l'email : une série de mots mystérieux pour l'anthropologue que je suis : « *AEAD mode chunk size* ». Je cite ci-dessous ce message dans son intégralité et en anglais afin que les lecteurices puissent se faire une idée de la prose « IETFienne » :

Subject: [openpgp] AEAD mode chunk size
From: Marcus Brinkmann <adresse.email@university.com>
Date: Sat, 30 June 2018 16:00 UTC

Hi,

RFC4880bis contains this segment:

The chunk size octet specifies the size of chunks using the following formula (in C), where c is the chunk size octet:

$$\text{chunk\_size} = ((\text{uint64\_t})1 \ll (c + 6))$$

An implementation MUST support chunk size octets with values from 0 to 56. Chunk size octets with other values are reserved for future extensions.

This allows chunk size up to $2^{(6+56)} = 4 \text{ EiB}$. It is impossible to implement AEAD correctly with chunk sizes larger than can be buffered in RAM. A large chunk size would require output of unverified plaintext, enabling attacks like EFAIL but also others.

To implement AEAD correctly, chunk size must be limited to reasonable sizes. TLS uses a chunk size up to 2^{14} (16 KiB), but any reasonable limit will do, for example 64 KiB. I suggest to change the text to this:

The chunk size octet specifies the size of chunks using the following formula (in C), where c is the chunk size octet:

$$\text{chunk\_size} = ((\text{uint64\_t})1 \ll c)$$

An implementation MUST support chunk size octets with values from 0 to 16. Chunk size octets with other values are reserved for future extensions.

Thanks,
Marcus Brinkmann¹⁸³

co-écrit les autres articles de l'équipe de recherche, publiés plus tard (Müller, Brinkmann, Poddebniak, Böck, et al. 2019; Schwenk et al. 2020).

Certes, cet email mentionnait EFAIL et c'est ainsi que je l'ai repéré, mais mon intérêt pour cet email n'était alors pas partagé, à en croire l'unique réponse sur la liste, à laquelle Brinkmann a encore répondu en retour, portant ainsi à trois le nombre d'emails de ce fil de discussion. Il ne faut toutefois pas se fier aux apparences et ne surtout pas juger avec trop de hâte cette situation car l'AEAD a occupé durant des années le groupe de travail. Toutefois, avant de s'engager dans l'analyse de ces discussions, il me semble important de mentionner un point important : comprendre cette discussion dans tous ses détails n'est pas aisé, du moins pour l'anthropologue que je suis. À l'instar de l'email présenté ci-dessus, la prose IETFienne ne serait pas si efficace sans son jargon hautement technique qui permet, à celles et ceux qui le maîtrisent, d'aller droit à l'essentiel. Pour les autres, comme moi-même et peut-être certaines lectrices de ces lignes, la lecture de ces emails peut être frustrante. J'aimerais donc tenter de vulgariser ce qu'est l'AEAD et ce qu'est la *chunk size*.

L'AEAD et l'authentification des données chiffrées

L'AEAD est l'acronyme de *Authenticated Encryption with Associated Data*, qui, comme son nom l'indique, est un mode de chiffrement qui combine à la fois le chiffrement et l'authentification des données. Ce mode de chiffrement permet de garantir que les données chiffrées n'ont pas été modifiées et il est utilisé dans des schémas de chiffrement symétrique, comme AES. Il s'agit d'une fonction au sein de l'algorithme de chiffrement qui permet de vérifier que la donnée qui a été chiffrée correspond bien à celle qui est déchiffrée. Le mécanisme repose sur une signature de blocs de certaines tailles [*chunk*] de texte clair, ce qui permet de détecter s'il y a eu des modifications du cryptogramme. Cette vérification automatique de l'intégrité des données nécessite un mode de chiffrement par bloc [*bloc cipher*] qui s'oppose au chiffrement de flux [*stream cipher*]¹⁸⁴.

183 Marcus Brinkmann, email « [openpgp] AEAD mode chunk size » du 30 juin 2018, à 16:00 UTC sur la liste de diffusion du groupe de travail OpenPGP : <https://mailarchive.ietf.org/arch/msg/openpgp/t79iRZ80KHuVTEyVVLaoCLl4Rwc/>. Je n'ai pas traduit cet email car je suis incapable de traduire la plupart des explications techniques qui le composent.

Dans le contexte du chiffrement AEAD, la *chunk size* correspond à la taille des blocs qui sont (dé-)chiffrés et vérifiés les uns après les autres. Plus la taille de ces blocs est réduite, moins il y a de possibilités d'en altérer le contenu et plus l'opération prendra du temps et nécessitera des ressources. Au contraire, plus la *chunk size* est grande, plus il existe de risques que des attaques du type EFAIL existent. Ce problème était déjà connu en 2000 et le groupe de travail OpenPGP avait mis en place un mécanisme nommé « code de détection de modification » [*Modification Detection Code*] (MDC)¹⁸⁵.

Finalement, il est important de souligner que l'AEAD ne doit pas être confondu avec la signature électronique qui permet d'authentifier la provenance du cryptogramme. OpenPGP permet non seulement de chiffrer mais également de signer des données, pour en attester l'intégrité et surtout l'origine. En signant un message avec ma clé privée 9584 92A9 80DC C8FE F5C5 EE91 56F7 ACD4 B1FD E56F, j'atteste que ce message est bien écrit par moi, sylvain.besencon@unifr.ch, unique détenteur présumé de cette clé. Cette information peut être vérifiée par toute personne qui détient ma clé publique. Celle-ci peut notamment être téléchargée sur Internet (<https://keys.openpgp.org/search?q=sylvain.besencon@unifr.ch>). La signature produit ainsi des preuves cryptographiques qui attestent l'origine et l'intégrité d'un message chiffré. Cependant, dans certains cas, il peut être intéressant de ne pas signer des données chiffrées, tout en souhaitant qu'elles soient authentifiées, c'est-à-dire que leur intégrité soit garantie. L'AEAD permet de garantir l'intégrité des données sans authentifier son auteure.

184 Les deux modes utilisés dans OpenPGP.

185 Pour implémenter l'AEAD, il existe plusieurs solutions. OpenPGP utilise un mécanisme qui s'appelle le code de détection de modification [*Modification Detection Code*] (MDC). Il s'agit d'informations qui sont ajoutées au cryptogramme et qui contiennent des données permettant de vérifier l'intégrité des données chiffrées. EFAIL a su tirer profit de cette particularité en « jouant » avec le MDC grâce aux outils que les chercheurs ont appelés les « malleability gadgets ». Si ce niveau d'explications ne vous satisfait toujours pas, je vous recommande d'aller consulter la page Internet d'EFAIL (<https://efail.de/>), la page anglophone Wikipédia sur le chiffrement authentifié (https://en.wikipedia.org/wiki/Authenticated_encryption) ou certains emails de la liste de diffusion du groupe de travail (notamment celui Neal H. Walfield datant du 28 février 2019 : https://mailarchive.ietf.org/arch/msg/openpgp/kl2wziH-odvtFbIXA3yLMS_fldk/).

7.1.2. Un vieux problème inscrit dans l'histoire d'OpenPGP

Loin d'être anecdotique, l'AEAD concerne un sujet débattu depuis longtemps au sein du groupe de travail. Dans un email envoyé quelques heures après la divulgation publique d'EFAIL, Werner Koch s'est adressé aux utilisatrices de GnuPG *via* la liste de diffusion pour expliquer pourquoi, selon lui, GnuPG n'était pas vulnérable aux attaques d'EFAIL. Dans cet email, il cite notamment un email privé qu'il avait envoyé à l'équipe d'EFAIL en novembre 2017, lorsque celle-ci l'avait contacté. Dans cet email, il a établi une chronologie de l'instauration du *Modification Detection Code* (MDC), qui implémente justement une forme d'AEAD :

[...] Voici la chronologie de l'introduction du MDC.

AES conference, en mars 2000

* Rencontre entre [Philip Zimmermann], Jon Callas, et moi pour discuter comment améliorer la robustesse de notre chiffrement sans exiger du contenu chiffré.

GnuPG 1.0.3 (2000-09-18)

* Les chiffrements améliorés Twofish et MDC sont maintenant utilisés. PGP 7 supporte cette fonctionnalité. Les anciennes versions de GnuPG ne la supportent pas, donc elles devraient être mises à jour au minimum jusque la version 1.0.2.

GnuPG 1.0.7 (2002-04-29)

* L'indicateur de fonctionnalité MDC est pris en charge et peut être défini à l'aide de la commande "updpref".

GnuPG 1.1.92 (2002-09-11)

* L'usage du MDC a augmenté. Un MDC sera utilisé si les destinataires l'ont directement exigé, si les destinataires ont AES, AES192, AES256 ou TWOFISH dans leur préférences d'algorithmes de chiffrement, ou si l'algorithme de chiffrement a un chiffrement par bloc qui n'est pas égal à 64 bits (actuellement, il s'agit aussi d'AES AES192, AES256, et TWOFISH).

* GnuPG ne désactivera plus automatiquement la compression lors du traitement d'un fichier déjà compressé, à moins qu'un MDC ne soit utilisé. Ceci a pour but de donner au message une certaine résistance à l'attaque par texte chiffré choisi [chosen-ciphertext attack] lors de la communication avec d'autres

programmes (le plus souvent PGP antérieur à la version 7.x) qui ne supportent pas les MDCs.

GnuPG 2.1.9 (2015-10-09)

* gpg: Échec avec une erreur au lieu d'un avertissement si un algorithme de chiffrement moderne est utilisé sans MDC.

(Werner Koch, email sur la liste de diffusion de gnupg-users, 14 mai 2018)¹⁸⁶

On le voit, en 2000 déjà, la question du chiffrement authentifié était débattue entre les deux implémentations d'OpenPGP : PGP de Zimmermann et GPG de Koch. Cette chronologie montre comment les différentes modifications des spécifications et les patches du code ont « inscrit » l'histoire du vieillissement d'OpenPGP et comment cette histoire resurgit dès qu'il s'agit de comprendre comment adapter les spécifications ou les codes à un problème qui date. Par exemple, en 2002, des scientifiques avaient déjà divulgué des attaques qui exploitaient des messages qui ne faisaient pas usage du MDC. À cet article intitulé « *Implementation of Chosen-Ciphertext Attacks against PGP and GnuPG* » (Jallad, Katz, et Schneier 2002), le groupe de travail a répondu par un renforcement de l'usage du MDC dès 2002 ainsi que sa formalisation dans le RFC 4880 qui stipule l'importance du MDC :

Cette attaque [*Implementation of Chosen-Ciphertext Attacks*] peut être déjouée par l'utilisation de la détection de modification, à condition que la mise en œuvre ne permette pas à l'utilisateur de renvoyer naïvement les données à l'attaquant. Une implémentation DOIT traiter une défaillance du MDC comme un problème de sécurité, et pas seulement comme un problème de données. (RFC 4880, section 14, p. 82, ma traduction).

Cette extrait du RFC 4880 est intéressant à plusieurs égards. D'une part, des failles exploitant des vulnérabilités similaires avaient déjà été découvertes par le passé et le groupe de travail était conscient des risques liés à des défaillances concernant l'AEAD. L'équipe d'EFAIL a contribué à mettre à jour les connaissances scientifiques sur ce problème déjà identifié¹⁸⁷. D'autre part, la dernière phrase pointe une intéres-

186 « Efail or OpenPGP is safer than S/MIME » : <https://lists.gnupg.org/pipermail/gnupg-users/2018-May/060320.html>.

187 Je ne sous-entends pas qu'EFAIL n'a rien apporté de nouveau et la réception positive de la communauté scientifique de ces failles me semble être une preuve suffisante pour considérer l'apport scientifique des chercheurs. Souvenons-nous que leur recherche a notamment abouti à la publi-

sante distinction à opérer entre un problème de sécurité et un problème de données. On l'a vu dans le chapitre précédent, EFAIL a été très controversé, notamment parce qu'il n'y avait pas de consensus clair sur le problème d'EFAIL. Nous sommes ici face à un problème similaire et le fait que le RFC stipule qu'il s'agit bien d'un problème de sécurité est probablement un signe qu'identifier une défaillance du MDC comme un problème de sécurité n'allait pas de soi et il a fallu l'inscrire dans le RFC.

Comme je l'ai déjà mentionné, le groupe de travail OpenPGP a été reformé en 2015 pour travailler sur une nouvelle version du protocole, connue sous le nom de rfc4880-bis. Dans cette nouvelle version des spécifications, le groupe de travail discutait d'instaurer un autre mode de chiffrement authentifié, nommé OCB. Pourtant, loin de faire consensus, de nombreuses discussions avaient lieu quant à l'usage de ce mode. Werner Koch le laisse entendre dans l'email déjà cité ci-dessus :

Nous travaillons progressivement au sein du groupe de travail sur le rfc4880bis afin d'introduire un nouveau mode de chiffrement. Malheureusement, les avis sont partagés sur l'utilisation du mode OCB et nous devons donc peut-être choisir entre deux nouveaux modes. Sur la base de l'expérience du MDC, je pense que le déploiement d'un nouveau mode prendra au moins trois ans. D'ici là, j'espère que le bricolage autour du MDC [the MDC hack] fera l'affaire.¹⁸⁸

Cet extrait me semble très intéressant et important à considérer, car il montre une logique temporelle complètement différente que celle de l'urgence vue dans le chapitre précédent. Ici, pour Werner Koch, il est tout à fait acceptable d'attendre trois ans pour obtenir un début de changement. L'expression « *the MDC hack* » – où « *hack* » est à comprendre comme « bricolage », dans le sens noble du terme – montre que Koch et probablement d'autres développeuses étaient conscientes des faiblesses du MDC mais que cette solution fonctionnait tant bien que mal. Cette fenêtre « d'au moins trois ans » pour trouver une solution aux vulnérabilités d'EFAIL paraît incroyablement longue eu égard au déferlement de réactions de toutes sortes dans les heures qui ont suivi la divulgation d'EFAIL. Toutefois, présenté comme l'a fait Koch dans cet email, ce délai semble à la fois raisonnable et pragmatique : il existait de fortes dissensions dans le groupe de travail à ce propos et trois années lui semblaient

cation de 5 articles dans des revues ou des conférences réputées.

188 <https://lists.gnupg.org/pipermail/gnupg-users/2018-May/060320.html>.

un délai raisonnable¹⁸⁹. Il est intéressant de noter que Koch semblait conscient du danger que posaient ces attaques mais qu'il interprétait ce danger dans la logique du mode de production propre à l'IETF et au groupe de travail : des négociations lentes et complexes qui aboutissent généralement à des compromis. Il n'y a dans ses propos ni urgence ni panique, mais confiance et sérénité.

Il a été reproché à Werner Koch de n'avoir saisi ni la gravité ni le danger d'EFAIL quand les chercheurs l'ont contacté. Certes, les attaques d'EFAIL se sont révélées sérieuses et surtout efficaces contre le « *MDC hack* » et il aurait certainement été adéquat de donner un peu plus de crédit à ces attaques. Cependant, l'email de Koch laisse entrevoir une autre interprétation du problème qui écarterait la piste de la simple négligence : dans cet email, Koch affirme être conscient de la nécessité de changer de mode de chiffrement. En estimant à trois ans le temps nécessaire pour implémenter de manière interopérable un nouveau mode de chiffrement, il place ces vulnérabilités de manière pragmatique dans le temps long de la norme. Ce que les chercheurs, la communauté d'utilisatrices et les médias ont perçu dans l'instantanéité de l'urgence et que j'ai décrit dans le chapitre précédent, Werner Koch l'a situé dans la lente marche de la standardisation et des longues négociations qui l'accompagnent généralement.

7.1.3. Négocier les propriétés de l'AEAD, encore et toujours

Revenons maintenant à la manière dont le groupe de travail a saisi les failles d'EFAIL. Quelques semaines après l'email de Marcus Brinkmann du 30 juin 2018 qui n'avait presque pas suscité de réactions, Werner Koch a envoyé le 24 juillet 2018 un autre email sur la liste de diffusion à propos de la nécessité de publier la version 05 du brouillon du RFC-4880bis, car la version 04 allait bientôt expirer¹⁹⁰. Parmi les changements suggérés, plusieurs avaient trait à une implémentation plus rigoureuse de l'AEAD, qui devait réduire la surface d'attaque d'EFAIL. Il est cependant intéressant de

189 Son estimation n'était d'ailleurs pas si erronée puisqu'en été 2021 (trois ans et demi après son email), un accord est trouvé au sein de la *design team* quant à la taille de la *chunk size* de l'AEAD, comme on le voit plus loin.

190 https://mailarchive.ietf.org/arch/msg/openpgp/yan7x39PoDXsOmMXMvGmsR_6ReU/.

constater qu'EFAIL n'est pas mentionnée explicitement dans cet email. Ce n'est que dans les réponses que le nom d'EFAIL apparaît : certaines personnes affirmaient que ces changements ne corrigeaient que trop faiblement les vulnérabilités. Pourtant, là encore, la discussion a été brève et le 26 juillet 2018, à peine deux jours plus tard, la nouvelle version du brouillon du RFC4880bis est publiée. Les sections 5.8 (p. 53) et 15 (p. 103) proposaient des mesures, notamment le passage ci-dessous¹⁹¹ :

*[...] l'implémentation NE DEVRAIT PAS permettre à l'utilisateur d'accéder aux données erronées et DOIT avertir l'utilisateur des problèmes de sécurité potentiels si ces données sont renvoyées à l'expéditeur.*¹⁹²

Après la publication de ce document, la discussion concernant l'AEAD est à nouveau suspendue pour être relancée six mois plus tard, le 27 février 2019 avec un email de Neal H. Walfield, un des fondateurs de Sequoia. Son email porte le même mystérieux (pour moi) objet : « AEAD chunk size ». La discussion a généré plus de 80 emails en l'espace de quelques semaines. Durant cet échange d'emails du printemps 2019, de nombreux aspects de l'AEAD ont été discutés. Plusieurs personnes ont proposé des explications (parfois divergentes) de ce qu'est la *chunk size*, des risques et opportunités que représentent une grande ou petite *chunk size*¹⁹³. Certaines personnes ont émis des doutes quant à la capacité de l'AEAD et de la modification de la *chunk size* d'avoir quelques effets positifs¹⁹⁴ alors que d'autres ont affirmé qu'il était crucial de s'accorder sur une limite supérieure tout en proposant certains compromis¹⁹⁵.

Puis, la discussion est à nouveau retombée... et a encore repris plusieurs mois plus tard. Par la suite, cette question de la *chunk size* a refait surface à plusieurs reprises durant plusieurs années. On en débattait de temps en temps, sans pourtant

191 Entre le 5 et le 8 août, la page Internet d'efail.de est mise à jour pour signifier que ces changements ont été faits.

192 <https://datatracker.ietf.org/doc/draft-ietf-openpgp-rfc4880bis-05>.

193 Voir par exemple l'email de Neal H. Walfield datant du 28 février 2019 en réponse à Jon Callas, l'une des personnes qui a participé à la standardisation du tout premier RFC d'OpenPGP : https://mailarchive.ietf.org/arch/msg/openpgp/kl2wziH-odvtFbIXA3yLMS_fldk/.

194 Voir par exemple l'email de Peter Gutmann datant du 29 mars 2019 : <https://mailarchive.ietf.org/arch/msg/openpgp/MQ-sPn0DeNYZFyCFKbvmJrpkTKQ/>.

195 Voir par exemple l'email de Bart Butler datant du 28 mars 2019 : https://mailarchive.ietf.org/arch/msg/openpgp/4bNWNzTjRRjE8F6pwszMKW_ZhvQ/.

qu'elle ne soit vraiment prioritaire et surtout sans parvenir à mettre d'accord le groupe de travail. En août 2021, cependant, un début de consensus semblait se former autour d'une taille maximale de 4MB, ce qui dépassait la valeur suggérée par Marcus Brinkmann mais qui semblait raisonnable – bien que non idéale – à de nombreuses personnes consultées¹⁹⁶. Cependant, cet accord a à nouveau été remis en question par la suite. À l'heure où j'écris ces lignes au printemps 2023, les nouvelles spécifications d'OpenPGP sont encore en attente d'être publiées dans un nouvel RFC officiel, ce qui indique que, d'une certaine manière, la discussion n'est pas encore close à ce sujet.

Cette difficulté à trouver un consensus représente aussi une sorte d'enlisement récurrent dans les architectures fédérées : l'enlisement dans le social. Dans le cas présent, l'adoption de l'AEAD à OpenPGP a certes été controversée à cause de certaines contraintes du passé, comme l'usage dans certains cas du mode de chiffrement continu [*stream cipher*], incompatible avec l'usage d'un mode AEAD. Mais elle a aussi et peut-être surtout souffert d'un désaccord profond entre différents projets-sécurité. Il a donc fallu négocier non plus seulement au niveau des spécifications techniques, mais au niveau des projets-sécurité respectifs. Rappelons-nous également que, comme je l'ai déjà mentionné au début de ce chapitre, l'interopérabilité représente l'un des avantages les plus importants et revendiqués des architectures fédérées. Nous voyons pourtant que cet avantage n'est pas donné et qu'il nécessite un grand investissement temporel. Il faut d'abord explorer différentes possibilités, négocier et convaincre les implémentations à la fois partenaires et concurrentes, et finalement trouver un moyen commun de les implémenter.

En résumé, les tâtonnements techniques ne représentent qu'une partie du travail de maintenance d'OpenPGP. Une autre immense partie de ce travail de maintenance a consisté en la gestion même du groupe de travail. Il s'agit par exemple de procé-

¹⁹⁶ Ce début de consensus a été trouvé dans une formation restreinte du groupe de travail. Le PV de la séance de travail du 13 août 2021 est disponible sur le site de l'IETF : <https://mailarchive.ietf.org/arch/msg/openpgp-dt/3tnlRWJ0UT6JLdjlIVJyioOvwgl/>; cette décision a été en partie contredite par les discussions qui ont eu lieu la semaine suivante : https://mailarchive.ietf.org/arch/msg/openpgp-dt/UFQCaxFIW4ga_KnDiGSJVsEYTW0/.

dures bureaucratiques propres au mode de gouvernance de l'IETF pour améliorer le processus de négociation et d'adoption de consensus. Il s'agit également de gérer les conflits interpersonnels entre différents membres du groupe de travail. La prochaine section aborde cet aspect du soin des choses numériques.

7.2. Prendre soin du groupe de travail OpenPGP

J'ai déjà mentionné l'historique du groupe de travail avant la divulgation d'EFAIL. J'aimerais à présent revenir sur ce qui s'est passé après cette divulgation durant le temps de ma recherche ethnographique. Je n'ai malheureusement pu observer aucune réunion in situ de ce groupe de travail à cause de la pandémie de coronavirus. Toutefois, ma recherche m'a fait voir comment le processus de négociation des spécifications d'OpenPGP dépendait non seulement de l'attention portée aux aspects techniques d'OpenPGP, mais aussi du soin de la cohésion du groupe de travail lui-même.

D'un point de vue institutionnel, l'IETF est une organisation ouverte régie par un mode de gouvernance bureaucratique qui a pour but d'assurer à la fois la transparence et la qualité des normes dont elle est responsable. Il s'agit notamment d'avoir des procédures strictes pour édicter une norme. Par exemple, chaque email est archivé et publiquement accessible en ligne, de même que l'enregistrement vidéo de chaque réunion des différents groupes de travail. Pour le groupe de travail OpenPGP, cela constitue des milliers d'emails consignés, et des heures de vidéos. En outre, chaque étape des négociations est documentée en ligne, et il est possible de retourner dans les archives pour voir quels changements ont été opérés à quel moment et pour quelles raisons. Finalement, il existe aussi des procédures formelles de vérifications des instances supérieures de l'IETF, auquel le groupe de travail soumet sa proposition de RFC.

Dans le RFC 3935, intitulé « *A Mission Statement for the IETF* », l'IETF clarifie ses cinq principes cardinaux : processus ouverts, compétence technique, base bénévole,

rough consensus et code fonctionnel, et propriété des normes¹⁹⁷. Des règles relativement strictes encadrent les activités des nombreux groupes de travail. Les processus ouverts ont pour objectif de garantir la transparence de la gouvernance et exigent de fait une rigueur dans les procédures, ce qui se traduit notamment par une bureaucratie importante qui prend nécessairement du temps. Parallèlement, l'institution n'a pas de système d'adhésion et n'importe qui peut potentiellement participer aux discussions : il suffit d'envoyer un email à une liste de diffusion pour y contribuer¹⁹⁸. Il n'y a pas de statut de membre, pas de cotisation à payer ni de compte à rendre. Dans ces conditions, il est structurellement impossible d'établir une liste de membres et donc de calculer une majorité. C'est pourquoi les décisions sont prises par consensus approximatif [rough consensus], en « bourdonnant » [*humming*]¹⁹⁹. En pratique, cela signifie qu'une décision est prise uniquement quand il n'y a plus aucune objection²⁰⁰. Le consensus approximatif est également lié à la volonté de prendre des décisions qui fonctionnent dans la pratique, d'où l'idée du *running code* : plutôt que des systèmes théoriquement bons mais pratiquement difficiles à mettre en place, l'IETF prend le parti des décisions qui se veulent pragmatiques, fonctionnelles et surtout interopérables. Ainsi, il est souvent exigé qu'il y ait au moins deux implémentations différentes pour qu'un protocole soit normalisé dans un RFC.

Je vais illustrer ci-dessous comment ces différents aspects ont des conséquences très concrètes sur les temporalités de la maintenance du protocole en m'arrêtant sur un épisode marquant de ma recherche : la reconstitution officielle du groupe de travail OpenPGP dès 2020. Je vais résumer cet épisode sans entrer dans tous les détails mais en insistant particulièrement sur les démarches entreprises pour avoir « tout le monde à bord ».

197 <https://datatracker.ietf.org/doc/rfc3935/>.

198 Pour rappel, il suffit de suivre la procédure sur cette page pour s'inscrire à la liste de diffusion et pouvoir ensuite y contribuer : <https://www.ietf.org/mailman/listinfo/openpgp>.

199 Il y a même un RFC (le numéro 7282) sur la pratique du bourdonnement : <https://www.rfc-editor.org/rfc/rfc7282.txt>. Voir aussi la courte présentation « Please Hum Now: Decision Making at the IETF » de Niels Ten Oever sur le site Hackcur.io : <https://hackcur.io/please-hum-now/>.

200 Décréter qu'il y a un consensus n'est pas aussi évident qu'il n'y paraît et le cas d'OpenPGP est un excellent exemple dont j'expose une illustration plus loin.

7.2.1. La refonte du groupe de travail

En 2020, plusieurs failles avaient vu le jour (dont EFAIL) qui n'avaient été que partiellement résolues au sein du groupe de travail. Ainsi, constatant que les choses n'évoluaient que trop lentement, plusieurs voix se sont faites entendre pour demander à Werner Koch, alors l'unique éditeur du RFC4880bis, de céder sa place à d'autres personnes dans l'espoir de donner un nouveau départ au groupe de travail et parvenir à un consensus sur les nouvelles idées à intégrer dans le protocole. Le vendredi 17 avril 2020, Vincent Breitmoser, une personne active depuis plusieurs années pour OpenPGP, a envoyé un email sur la liste de diffusion d'OpenPGP dont le sujet était explicite : « *Requesting the editor to step down* »²⁰¹. Dans ce long email, il demande à ce que le groupe de travail soit ré-institué [*re-chartered*] et qu'un nouvel éditeur soit nommé. Il donne plusieurs exemples qui montrent, selon lui, que Koch ne remplissait pas son cahier des charges en tant qu'éditeur. Un des points fondamentaux concerne le fait que l'état actuel du brouillon « RFC4880bis » ne reflétait plus le consensus du groupe de travail :

[...] J'apprécie le travail effectué par Werner [Koch] dans le RFC4880bis, et bien sûr dans OpenPGP en général. Cependant, un document publié au nom de l'IETF a un caractère officiel, et on s'attend à ce qu'il reflète le consensus du groupe de travail. Les documents draft-ietf-openpgp-rfc4880bis-* ne répondent pas, à mon avis et à celui de plusieurs autres personnes à qui j'ai parlé, à cette attente.

En conséquence, je demande officiellement à Werner Koch de démissionner de son poste d'éditeur de la spécification OpenPGP.

(Membre du groupe de travail OpenPGP, email sur la liste de diffusion, 17 avril 2020)²⁰²

Dans la deuxième partie de cet email, Breitmoser suggère quelques pistes pour poursuivre le travail de normalisation des spécifications. Il propose notamment de repartir de la charte originale du RFC4880bis (datant de 2015) et de recommencer ce travail de négociation en veillant à ce que le consensus soit assuré au sein du groupe de travail. Un tel message a suscité un nombre important de réponses venant de différentes personnes, à commencer par le principal concerné : à cet email et ces accu-

201 <https://mailarchive.ietf.org/arch/msg/openpgp/XxZt89Eh7XUenuVRajbgtcWzWdA/>.

202 « [openpgp] Requesting the editor to step down » : <https://mailarchive.ietf.org/arch/msg/openpgp/XxZt89Eh7XUenuVRajbgtcWzWdA/>.

sations, Werner Koch a répondu trois jours plus tard, le lundi 20 avril 2020 qu'« à l'exception de quelques détails sans importance, nous avons déjà depuis longtemps un consensus approximatif [...] »²⁰³. Cette attitude à ignorer le désaccord d'une partie du groupe de travail lui avait déjà été reprochée et l'a été également par la suite. En effet, plusieurs personnes ont affirmé et répété qu'il n'y avait pas de consensus dans le groupe de travail à propos du manque ou non de consensus²⁰⁴.

La situation était compliquée : certaines personnes étaient très heureuses de la collaboration avec Koch et ne se sentaient pas concernées par la plainte de Breitmoser²⁰⁵. D'autres ont soutenu la proposition de Breitmoser. Il est aussi utile de constater qu'un grand nombre de personnes ne s'est pas exprimé sur cette liste à ce moment. Ayant mené des entretiens avec certaines d'entre elles, je sais que certaines ne se sont pas manifestées pour éviter de s'immiscer dans un conflit jugé comme interpersonnel. Je peux aussi faire l'hypothèse que d'autres personnes ont aussi été préoccupées et prises par les différentes mesures contre la pandémie de COVID-19, et OpenPGP n'était pas au centre de leurs préoccupations. À la lecture des réponses, il semble se dégager un rejet tacite de la proposition de Breitmoser, qui n'a été soutenue explicitement que par deux personnes de l'équipe de Sequoia. Cependant, plusieurs autres personnes ont suggéré de résoudre certains points litigieux mentionnés par Breitmoser dans le brouillon actuel²⁰⁶.

Après la discussion engagée à la suite de la demande de Breitmoser quant à la démission de Koch, il n'y a presque plus eu d'emails envoyés sur la liste entre avril et septembre 2020. Selon les différents échanges que j'ai eus avec certaines codeuses à cette époque, il semble qu'une certaine lassitude mêlée à de la résignation frappaient plusieurs membres du groupe de travail, le tout dans un contexte où ces personnes avaient beaucoup d'autres choses auxquelles penser. À la toute fin du

203 « *Except for some unimportant details we already had rough consensus on the I-D a long time ago* » : <https://mailarchive.ietf.org/arch/msg/openpgp/VVxFe0kbTCVxf6LFybsU4PfKNc/>.

204 Cette affirmation a d'ailleurs été formulée plus tard dans un email envoyé sur la liste : https://mailarchive.ietf.org/arch/msg/openpgp/0VMNQHFd9dvvljlyd-S_Wppl4uk/.

205 Voir par exemple les emails de Ronald Tse, et notamment celui-ci : https://mailarchive.ietf.org/arch/msg/openpgp/DchlXXv_NV9pVvPeHQBDaQI741U/.

206 <https://mailarchive.ietf.org/arch/msg/openpgp/BjqpSmSGGpRC0EGhGAij55lW1Y0/>,

mois d'août 2020, Koch envoie un nouveau message sur la liste de diffusion. Cet email était destiné à « simplement » valider une nouvelle version du brouillon (la version 10) :

Bonjour!

Comme la date d'expiration était proche, j'ai mis à jour une nouvelle version révisée. Les changements collectés du dépôt GitHub depuis la version -09 sont les suivants :

b5c48cf Clarifie l'usage des deux octets de hash dans les paquets de signature.
750b19f Corrige la description des paquets de signature pour EdDSA
6c0bd58 Ajoute les valeurs actuelles à la liste des types d'attributs d'utilisatrices.
76177bb Usage de majuscules cohérentes pour l'ID de l'utilisatrice et l'attribut de l'utilisateur.
722ed9e Corrige des fautes de frappe
30d8397 Introduit le sous-paquet Key Block pour aligner OpenPGP sur les CMS.
669f73f Corrige des fautes de frappe
6fd718d Rétablit l'exigence du RFC4880 d'avoir un ID d'utilisatrice.
(Werner Koch, email sur la liste de diffusion du groupe de travail OpenPGP, 31 août 2020)²⁰⁷

Tout d'abord, on constate que Koch continuait de modifier le brouillon et qu'il gardait en tête les différents délais. Toutefois, le ton informatif de sa première phrase suggère qu'il n'a effectivement pas cherché à obtenir l'aval du groupe de travail pour publier cette version. Ainsi, quelques participants du groupe de travail lui ont demandé si deux des points mentionnés ci-dessus avaient obtenu le consensus du groupe. L'échange a progressivement glissé de la discussion des points en question à des considérations plus générales quant à la gouvernance du groupe de travail. Une fois encore, certaines personnes affirmaient être tout à fait d'accord avec les changements proposés alors que d'autres s'y opposaient. La discussion s'est achevée 15 jours plus tard sans qu'il n'y ait de véritable décision.

Pourtant, sous ce silence trompeur, un événement majeur était déjà en négociation de manière informelle : la ré-officialisation du groupe de travail. En effet, depuis quelques mois, des discussions avaient lieu dans d'autres canaux car Daniel Kahn Gilmor s'était annoncé prêt à reprendre les rênes en tant que président du groupe de travail si une autre personne se joignait à lui.

207 « [openpgp] [internet-drafts@ietf.org] New Version Notification for draft-ietf-openpgp-rfc4880bis-10.txt » <https://mailarchive.ietf.org/arch/msg/openpgp/LcsG6VA6WW0XDR1gvpO-KUXgCG8/>.

C'est le 20 octobre qu'un email intitulé « *heads-up: re-chartering the OPENPGP WG* » est envoyé sur la liste de diffusion²⁰⁸. Ce mail annonce la reconstitution officielle du groupe de travail, avec deux présidents. Le premier, Stephen Farrell était un habitué de l'IETF et n'était pas impliqué dans une implémentation d'OpenPGP alors que le second, Daniel Kahn Gillmor, était un grand connaisseur d'OpenPGP et impliqué dans plusieurs projets d'implémentation. Werner Koch restait l'éditeur du groupe et était désormais secondé par Paul Wouters, familier du groupe de travail mais guère impliqué dans une implémentation. Le groupe de travail était prêt à être reconstitué officiellement et le premier pas était d'établir sa mission sous la forme d'une charte. Entre cette annonce et la réouverture officielle du groupe de travail, plusieurs mois ont été nécessaires. Cette « lenteur » est notamment due à la bureaucratie propre à l'IETF, qui est d'ailleurs explicitement mentionnée dans certains emails :

Juste au cas où vous croyez qu'il ne se passe rien à propos de ça, ce n'est pas le cas : la bureaucratie ordinaire suit son cours [*normal bureaucracy is happening*] :-)
(Co-président du groupe de travail OpenPGP, liste de diffusion du groupe de travail, 30 octobre 2020)²⁰⁹

Une fois de plus, on constate que l'IETF n'est pas outillée pour répondre à l'urgence, dans le sens que ses mécanismes internes sont longs et lents. En l'occurrence, il s'agissait de faire approuver la décision de rouvrir le groupe de travail OpenPGP auprès du directeur de domaine de l'IETF [*IETF area director(s)*] et du *Internet Engineering Steering Group* (IESG) qui devaient vérifier si le groupe de travail remplissait bien les exigences formelles décrites au point 2.1 du RFC 2418²¹⁰.

Mais au-delà des procédures bureaucratiques, c'est un autre point qui a surtout marqué les esprits dans ce processus : plusieurs personnes ont insisté lors de mes entretiens sur la volonté affichée de ne froisser personne dans ce processus délicat qui consistait à recommencer depuis le début un travail qui avait été entamé cinq ans

208 <https://mailarchive.ietf.org/arch/msg/openpgp/6ro7-NcJMx6K-qPtLfX0uajb51A/>.

209 « [openpgp] likely timeline for WG (re)formation » : <https://mailarchive.ietf.org/arch/msg/openpgp/1bahjn2ISb1pOXJ9HQeSTK4bX-k/>.

210 <https://www.rfc-editor.org/rfc/rfc2418.txt>. On peut aussi consulter les pages du site Internet de l'IETF dédié au groupe de travail OpenPGP pour se convaincre du nombre important de documents relatifs au processus de normalisation : <https://datatracker.ietf.org/wg/openpgp/about/>.

plus tôt et qui n'avait pas abouti. Cela impliquait non seulement de ne désigner aucun responsable de l'échec des précédentes négociations, mais aussi de montrer que la voix de chacune et chacun était importante et comptait. Les deux présidents ont fait un travail remarquable de cohésion à ce propos, visible notamment par leur manière à la fois de cadrer les discussions et de donner de l'importance à chaque intervention. L'objectif était de chercher le soutien d'un maximum de personnes, et notamment les représentant·e·s des implémentations principales. Ainsi, le travail de négociation autour de la charte du nouveau groupe de travail illustre très bien cette volonté d'inclure un maximum de personnes dans les discussions. Cela figure d'ailleurs explicitement dans la charte elle-même :

Le groupe de travail sera en charge des travaux suivants :

- Réviser le RFC4880. L'intention est de démarrer avec le brouillon actuel rfc4880bis.
- D'autres travaux en lien avec OpenPGP pourront être entamés par le groupe de travail tant que cela n'interfère pas avec l'achèvement de la révision du RFC4880. Comme la révision du RFC4880 est l'objectif premier du groupe de travail, d'autres travaux peuvent être menés à condition que :
 1. Le travail ne retardera pas excessivement la dissolution du groupe de travail une fois la révision terminée (à moins que le groupe de travail ne soit reformé).
 2. Ce travail bénéficie d'un large soutien au sein du groupe de travail.²¹¹

Lors d'entretiens ou de conversations informelles, de nombreuses personnes ont par exemple loué le sens de la diplomatie et la patience des deux présidents, ainsi que de Paul Wouters, le nouvel éditeur au côté de Werner Koch. Cette charte a été formalisée le 23 mars 2022, une après la première nouvelle réunion du groupe de travail et peut être consultée sur le site de l'IETF²¹².

²¹¹ Ma traduction : <https://datatracker.ietf.org/doc/charter-ietf-openpgp/>.

²¹² <https://datatracker.ietf.org/doc/charter-ietf-openpgp/>. Les discussions qui ont été nécessaires pour parvenir à l'établissement de cette charte sont documentées dans la liste de diffusion, notamment dans les fils de discussions « *heads-up: re-chartering the OPENPGP WG* » (note 208 p. 276), « [openpgp] likely timeline for WG (re)formation » (<https://mailarchive.ietf.org/arch/msg/openpgp/1bahjn2ISb1pOXJ9HQeSTK4bX-k/>) ; « OpenPGP wg notes for saag » (<https://mailarchive.ietf.org/arch/msg/openpgp/on08F3GBqMjZY6gdZsOAysACLG8/>) et « *WG Review: Open Speci-*

7.2.2. Nouveau souffle, nouvel essoufflement, et nouvelles solutions

Ce nouveau départ a motivé un grand nombre de personnes à participer à la session de l'IETF de mars 2021 (qui a eu lieu en ligne uniquement à cause de la pandémie de COVID-19). L'ordre du jour de cette séance est informative de la direction que voulait prendre le groupe de travail car une grande partie était déjà dédiée à des considérations techniques de différentes implémentations : ProtonMail présentait une vulnérabilité trouvée, Sequoia présentait sa suite de tests²¹³ et GnuPG une proposition de formatage de clé²¹⁴. Par ailleurs, les présidents et éditeurs ont eu le temps de présenter les changements à faire et comment ils prévoyaient de les faire.

Par ailleurs, il semble que la reformation de ce groupe de travail lui a véritablement donné un nouveau souffle car dès les premiers mois de l'année et avant même la première rencontre officielle du nouveau groupe de travail en mars 2021, de nombreuses discussions avaient lieu sur beaucoup d'aspects des spécifications. Après la réunion de mars 2021, beaucoup de sujets ont été « ouverts ». Voici la liste des fils de discussions entre le 15 mars 2021 et le 4 mai²¹⁵ :

- [openpgp] Working Group reportback from IETF 110
- [openpgp] incomplete/confusing guidance around "Hash" Armor header for cleartext signing framework
- [openpgp] [RFC4880bis PATCH] Clarify CRC-24 C example implementation
- [openpgp] [RFC4880bis PATCH] Drop "Compatibility Profiles" section.
- [openpgp] Algorithm-specific data: problems with Simple Octet Strings, and possible alternatives
- [openpgp] Intended Recipient observation

ification for Pretty Good Privacy (openpgp) » (https://mailarchive.ietf.org/arch/msg/openpgp/Xv-ZI4Vljdi_A2l1V26T2kYD7bXY/). La décision de l'IESG a été rendue publique le 18 décembre 2021 dans l'email intitulé « WG Action: Rechartered Open Specification for Pretty Good Privacy (openpgp) » (<https://mailarchive.ietf.org/arch/msg/openpgp/u3mpJcFTGBFN-JRW8EHoWdbx5Ucl/>).

213 Cette suite de tests est accessible sur le site de Sequoia : <https://tests.sequoia-pgp.org/>. Les principales implémentations ont été testées par l'équipe de Sequoia, qui a aussi divulgué de manière responsable des bogues identifiés grâce à ces tests : <https://gitlab.com/sequoia-pgp/openpgp-interoperability-test-suite#hall-of-fame>. Cette suite de tests étant open source, chacun-e peut y contribuer et l'utiliser pour tester tout ou une partie de son implémentation.

214 <https://datatracker.ietf.org/doc/agenda-110-openpgp/>.

215 <https://mailarchive.ietf.org/arch/browse/openpgp/>.

- [openpgp] Ed25519 in a draft
- [openpgp] image-29-04-21-06-01.jpeg
- [openpgp] v5 fingerprints and public keys
- [openpgp] I-D Action: draft-ietf-openpgp-crypto-refresh-03.txt
- [openpgp] A comment on draft-ietf-openpgp-crypto-refresh-03, section 5.5.2
- [openpgp] don't emit version armor header by default
- [openpgp] OpenPGP certificate structure: multiple binding signatures on subkeys? (MR 43)
- [openpgp] Ed25519 and digest choices (issue 31)
- [openpgp] [RFC4880bis PATCH] WIP: bind wire format representations to specific pubkey algorithms
- [openpgp] v5 in the crypto-refresh draft

Cette liste de nouveaux sujets abordés au sein du groupe de travail montre l'engouement suscité par la refonte dudit groupe et le foisonnement d'idées. Les problèmes liés à EFAIL ne tiennent qu'une place secondaire dans cette liste de sujets qui ne les mentionnent d'ailleurs pas directement. Lors des entretiens que j'ai conduits avec plusieurs membres du groupe de travail, un élément était particulièrement récurrent : la volonté d'avancer et la nécessité de trouver des compromis. Pourtant, malgré cette volonté commune et partagée par le plus grand nombre, un autre élément revenait sans cesse : la lenteur avec lesquelles les choses avançaient. Certaines personnes déploraient notamment que des personnes s'opposent par principes à certaines suggestions sans apporter de solutions constructives. La discussion à propos de la révision de la 5^e version des clés OpenPGP (v5 key) est un bon exemple de cette situation de blocage (cf. chapitre 4 « Enlisement et ancrage infrastructurels », p. 150).

Face à ces blocages, les présidents du groupe de travail ont proposé une nouvelle mesure : la mise en place d'une *design team*. Il s'agit d'une petite équipe uniquement formée par des représentants (tous des hommes) des différentes implémentations principales (GnuPG, OpenPGP.JS, Sequoia et RNP) ainsi que les éditeurs et présidents du groupe de travail. Cette *design team* s'est vu attribuer une liste de diffusion dont les archives sont publiques mais à laquelle seules les personnes habilitées peuvent contribuer²¹⁶. L'objectif était d'améliorer le dialogue entre les implémentations princi-

²¹⁶ <https://mailarchive.ietf.org/arch/browse/openpgp-dt/>.

pales en se concentrant sur des compromis constructifs pour ensuite soumettre les différentes idées au groupe de travail. La *design team* a tenu sa première séance le 16 juillet 2021 et a tenu un rythme plus ou moins hebdomadaire jusqu'au 14 juin 2022 où une proposition a été faite au groupe de travail. Les comptes-rendus de ces séances étaient publics et le progrès des négociations semblait convenir aux intervenants. C'est notamment au sein de cette *design team* qu'un consensus autour de la taille minimale de la *chunk size* a été trouvé en août 2021²¹⁷.

Si ni la constitution ni la composition de cette *design team* n'a été remise en question par le groupe de travail dans son ensemble, force est de constater que le choix des différents participants introduit une nouvelle hiérarchie parmi les implémentations : celles qui peuvent participer activement à ce processus ont plus de marge de manœuvre dans les négociations de certains paramètres techniques. En effet, même si le dernier mot revient toujours au groupe de travail dans son ensemble, il est peu probable que ce dernier reviennent sur des paramètres adoptés au sein de la *design team* avec une nouvelle proposition meilleure. Près de deux ans plus tard, en automne 2022, cet aspect aura son importance car Werner Koch, pourtant co-rédacteur du groupe de travail, a publié sur le site de l'IETF une nouvelle version de l'ancien brouillon (rfc4880bis) qui annulait toutes les nouveautés négociées au sein de la *design team*, auquel il a participé puis a été remplacé par une autre personne de GnuPG. Comme j'avais déjà terminé de faire ma recherche ethnographique, je n'ai suivi que de loin cette affaire, mais les discussions informelles que j'ai pu avoir avec différentes personnes (malheureusement pas avec Werner Koch) indiquent que cela a beaucoup retardé l'avancée de la *design team* qui venait de soumettre ses propositions au groupe de travail.

217 Le PV de ladite séance de la *design team* du 13 août 2021 est disponible sur le site de l'IETF : <https://mailarchive.ietf.org/arch/msg/openpgp-dt/3tnIRWJ0UT6JLdjlIVJyioOvwgl/> ; cette décision a été en partie contredite par les discussions qui ont eu lieu la semaine suivante : https://mailarchive.ietf.org/arch/msg/openpgp-dt/UFQCaxFlW4ga_KnDiGSJVsEYTW0/.

7.2.3. Trois sources de blocages

Lors de ma recherche, j'ai constaté plusieurs fonctionnements pouvant poser problème au sein du groupe de travail OpenPGP. Je les ai regroupés dans les trois catégories suivantes : un mode d'interaction favorisant l'opposition plutôt que la discussion, les problèmes liés à la compatibilité avec les systèmes hérités et la cohabitation de plusieurs projets-sécurité parfois radicalement divergents. Ces trois sources de blocages sont (du moins partiellement) atténués grâce à des outils de gouvernance bureaucratique. Je propose de passer ces trois catégories en revue.

1. Un mode d'interaction favorisant l'opposition plutôt que la discussion. La lecture de la liste de diffusion du groupe de travail OpenPGP permet de se rendre compte de l'hétérogénéité du groupe en question²¹⁸. Aux côtés des personnes en charge des grandes implémentations (GnuPG, OpenPGP.js, Sequoia, RNP), participent également des personnes qui ont été très actives dans la création d'OpenPGP sans pour autant continuer de l'implémenter. D'autres encore sont des personnes externes qui, par goût du travail bien fait ou par simple curiosité, sont inscrites à la liste de diffusion. Il n'est pas si rare de voir des emails qui commencent avec une telle mention :

(Sorry, I'm a lurker in this group, but wanted to chime in...)

(Membre du groupe de travail OpenPGP, liste de diffusion du groupe de travail, 2 avril 2019)²¹⁹

Cette hétérogénéité est une richesse et certainement un atout dans certains cas. Elle montre par exemple l'inclusivité de ce groupe de travail qui n'est pas composé que d'ingénieur·e·s d'une ou deux entreprises. Toutefois, c'est aussi un problème car cela cause parfois des blocages, notamment à cause d'attentes qui divergent fortement.

Par ailleurs, le mode d'intervention dans cette liste (ainsi que la plupart des autres listes de diffusion de l'IETF que j'ai consultées) favorise la confrontation. En effet, une

²¹⁸ Je parle ici d'une certaine hétérogénéité, notamment celle qui concerne l'engagement de ces intervenant·e·s envers le protocole. Mais on pourrait également remarquer certaines ressemblances : la quasi-totalité des emails sont signés avec des prénoms masculins. De plus, la plupart de ces personnes sont basées en Europe (dont beaucoup en Allemagne) et en Amérique du Nord.

²¹⁹ https://mailarchive.ietf.org/arch/msg/openpgp/P2Y1pkz70_n76o2PjaPMLxXY8K8/.

des caractéristiques des interactions de cette liste sont des emails avec des opinions souvent très tranchées, parfois d'une longueur considérable, et le tout pimenté de temps en temps d'un ton véhément ; une sorte de performance d'une certaine masculinité²²⁰. En outre, la manière d'engager une discussion sur la liste de diffusion favorise le désaccord plutôt que la recherche de compromis. J'ai notamment souvent été frappé de constater à quel point certaines discussions s'ouvraient sans se conclure. L'exemple des négociations autour de la *chunk size* de l'AEAD montre à quel point un sujet peut prendre du temps avant de trouver une forme de conclusion – qui n'est jamais que provisoire. Cela est peut-être dû au fait que la grande majorité des emails sont écrits pour infirmer ce qui a été dit et que rares sont les emails qui disent simplement « oui, je suis d'accord ». Peut-être à cause de mon manque de compréhension technique, j'ai souvent eu l'impression que les discussions s'achevaient sans qu'une véritable décision soit formulée.

2. La compatibilité avec les systèmes hérités. La mission du groupe de travail est aussi compliquée par le fait qu'OpenPGP est développé depuis des décennies et qu'il « traîne » avec lui de nombreux paramètres qu'il n'est pas aisé de changer, comme je l'ai déjà détaillé dans la première partie de cette thèse. Ce n'est toutefois pas une impasse définitive et l'un des présidents du nouveau groupe de travail a précisé que cette situation n'était absolument pas particulière à OpenPGP. Je lui avais demandé s'il était inquiet que les délais fixés lors de la première séance du groupe de travail en mars 2021 avaient tous été dépassés et il m'a répondu qu'il n'était pas du tout inquiet et considérait que les chances de succès de cette nouvelle norme étaient toujours aussi bonne (environ 50 % selon lui). Il a ensuite comparé le groupe de travail OpenPGP à d'autres et m'a donné les explications suivantes :

Il y a plusieurs classes de groupes de travail. Si tu arrives à réunir un groupe de personnes voulant créer une nouvelle chose, et qui est assez sûr de ce que vous voulez faire, il y aura un groupe de personnes plus petit et ça ira bien. Il peut aussi y avoir des groupes de travail avec des discordes importantes sur comment y arriver. Par exemple le groupe de travail WEBRTC a passé 2 ans en étant

220 Certains emails cumulent les trois aspects, par exemple l'email de Jon Callas datant du 29 mars 2019 : <https://mailarchive.ietf.org/arch/msg/openpgp/8Lk5H36wXZmnm-f1hvBJ49aB46A/>.

bloqué pour savoir comment faire vidéo codec. Il y avait toujours quelque chose qui n'allait pas. Il y avait un désaccord fondamental. [...] Il y a d'autres groupes qui sont un peu comme OpenPGP, où tu travailles avec une technologie qui est assez vieille, mail : 30 ans ; PGP : 20 ans [sic]²²¹. Quand c'est le cas, il y a un gros bagage, non seulement en termes de personnes mais aussi en termes de « Oh tu ne peux pas faire ça parce que il y a 15 ans, une personne a utilisé cette chose pour X et à cause de ça, tu ne peux pas facilement la changer ». Ce genre de blocages, c'est vraiment quelque chose qu'on a avec ce genre de groupe, par exemple les mails, parce que c'est vieux. Ça a été utilisé de tellement de manières différentes depuis des décennies.

(Un des présidents du groupe de travail d'OpenPGP, entretien du 1^{er} juillet 2021)

Au-delà des enjeux techniques, cette citation souligne une autre dimension : celle de l'expertise historico-technique accumulée par les membres du groupe de travail ainsi que les négociations nécessaires pour aboutir au consensus recherché. Les personnes qui ont ce bagage de connaissances historiques à propos du protocole sont effectivement celles qui semblent apporter le plus souvent des objections à l'égard de nouvelles propositions. Leur souci est légitime mais les solutions proposées sont toujours des compromis entre la meilleure solution pour un certain usage « actuel » du protocole et un moindre mal pour d'ancienne version du protocole. Il serait toutefois naïf de croire qu'il s'agit simplement d'un refus de se « moderniser » : certaines personnes m'ont par exemple confié leur crainte que les mêmes erreurs du passé soient faites. Il ne serait donc pas correct de n'attribuer la source de tous ces blocages qu'à une opposition « vieux Vs moderne ».

3. La cohabitation de plusieurs projets-sécurité. Les négociations sur la liste de diffusion cherchent à trouver un dénominateur commun à des programmes et des programmeur-se-s qui utilisent OpenPGP. Or, comme je l'ai déjà mentionné à de nombreuses reprises, l'un des défis d'OpenPGP est sa versatilité que certain-e-s qualifient d'ailleurs de couteau suisse. Un exemple frappant est celui de Peter Gutmann qui affirme, dans un des emails du fil « AEAD Chunk Size » :

Au cours des cinq dernières années environ, je n'ai reçu à peu près aucun courriel crypté par PGP, et je suis l'une des personnes qui a contribué à l'écriture de ce logiciel. Par ailleurs, j'ai probablement chiffré des gigaoctets de données avec PGP, presque toujours avec une clé

²²¹ Il faudrait ajouter une décennie à ces chiffres : le *Simple Mail Transfer Protocol* (SMTP) a été implémenté dans l'ARPANET en 1983 et PGP a été publié en 1991 par Zimmermann.

symétrique communiquée de manière externe et confidentielle. Dans aucun de ces cas, la protection par authentification des blocs de chiffrement [*blocked auth protection*] n'aurait été utile. (Peter Gutmann, email sur la liste de diffusion, 29 mars 2019)²²²

Cette citation illustre très bien que l'usage d'OpenPGP est très divers. Dans ce contexte, trouver le plus petit dénominateur commun pour avoir un protocole inter-opérable n'est pas aisé²²³. Ici, le problème concerne justement les cas d'usage qui varient fortement, de même que les modèles de risque. Ksenia Ermoshina et Francesca Musiani ont très bien montré comment l'identification des menaces contre lesquelles se protéger était une étape à part entière dans la production de technologies de chiffrement (Ermoshina et Musiani 2022, chap. 1). Cette étape est normalement réalisée par les utilisatrices qui évaluent les risques selon leur propre environnement et contexte ; toutefois, les codeur-se-s ont également un rôle en décidant contre quels types d'attaques leur application devrait protéger ses utilisatrices. Or, dans le cas d'OpenPGP, les usages sont si variés qu'une telle projection ne peut fonctionner, et c'est pourquoi certaines personnes insistent pour ne pas complexifier davantage le protocole avec des critères supplémentaires (comme une *chunk size* minimum) alors que d'autres insistent pour qu'un maximum de risques soient évités en incorporant des menaces qui ne concernent que certains usages de la technologie en question. Il ne s'agit pas de définir en amont le projet-sécurité des utilisatrices, mais de choisir quels sont les *nudges* que les développeuses veulent proposer pour servir au mieux un maximum de cas d'utilisation.

Il est intéressant de constater qu'ici, à défaut de pouvoir agir spécifiquement sur les pratiques des utilisatrices, les développeuses peuvent tout de même agir afin d'aller dans le sens du projet-sécurité qu'elles se sont fixées. Par exemple, un des

222 « Re: [openpgp] AEAD Chunk Size » : <https://mailarchive.ietf.org/arch/msg/openpgp/MQ-sPn0De-NYZFYcFKbvmJrkpTKQ/>.

223 Cette citation me questionne également à un autre niveau : en tant qu'anthropologue, je ne suis pas équipé pour saisir le statut de l'affirmation : « In none of those cases would blocked auth protection have been useful. » S'agit-il d'une affirmation scientifique ou de la formulation d'une opinion ? Bien entendu, je ne cherche pas à établir la vérité quand à l'utilité ou pas de l'usage de l'AEAD : ce qui m'intéresse concerne les modalités de la négociation et, n'étant pas cryptographe moi-même, je ne sais pas comment traiter certaines affirmations comme celle-ci, qui paraissent à la fois péremptoires et tout à fait contestées.

objectifs affichés de l'implémentation de Sequoia est d'être une implémentation difficile à mal utiliser [« *hard to misuse* »]. Lors de plusieurs sessions de travail, cet objectif se traduisait en débats parfois intenses sur la manière concrète de procéder. Jusqu'où pouvait-on, devait-on, prendre des décisions pour les utilisatrices ? Devait-on rendre impossibles certains usages, ou au contraire tout permettre, quitte à ce que certains usages soient potentiellement dangereux ? Faire cohabiter différents projets-sécurité – que ce soit au sein du groupe de travail OpenPGP ou au sein d'une équipe de développement – nécessite souvent d'explicitier quelles sont les responsabilités morales que veulent avoir, doivent avoir, ou ont les développeuses à l'égard des utilisatrices de leur technologie. Cela implique nécessairement des négociations à propos de ce qu'est la sécurité, mais aussi de quels sont les usages légitimes et de la responsabilités des uns, des unes et des autres. Comme toutes négociations, rien ne garantit que celles-ci aboutissent et c'est donc une source de blocages majeurs pour faire avancer les négociations du groupe de travail en vue d'obtenir des résultats consensuels et constructifs.

7.2.4. La bureaucratie au service des normes techniques

Au final, ce qui ressort de l'analyse de ces interactions sur la liste de diffusion concerne le fait que la supposée « objectivité » technique n'est qu'une affaire de constants compromis. L'exemple des échanges liés à l'AEAD et sa *chunk size* soulignent que la négociation et la recherche de compromis fait partie intégrante de la fabrique d'un protocole de sécurité. Ici encore, l'idée de bricolage fait largement ressortir le côté artisanal et profondément humain de la sécurité informatique, aspect qui se cache souvent derrière la figure d'un gros cadenas bleu, objectif, rationnel, indéniable. Cela ressort notamment d'un extrait d'email de Paul Wouters. À l'époque, il n'était pas encore co-éditeur du groupe de travail et contribuait aux discussions en tant que participant non engagé dans la maintenance d'une implémentation d'OpenPGP. Dans cet email daté du 19 avril 2020 en réponse à la demande de Vincent Breitmoser, il suggérait justement d'inclure un co-éditeur moins lié à une implémentation en ces termes :

Il est possible qu'il soit bénéfique qu'un éditeur soit ajouté afin d'améliorer le temps de réponse aux différentes modifications du document. Peut-être même quelqu'un de moins impliqué dans openpgp afin qu'il-elle puisse garder son rôle aussi administratif que possible sans biais techniques ?

(Membre du groupe de travail OpenPGP, email sur la liste de diffusion, 19 avril 2020)²²⁴

Cette citation parle du rôle de l'administration contre les « biais techniques », que j'interprète comme étant induits par le travail de soin d'une implémentation particulière qui donnerait au protocole telle ou telle orientation. Cette perspective renverse la vision habituelle où la technique est considérée comme objective et neutre. Pour lui, c'est l'administration de l'IETF qui aide à maintenir l'objectivité des discussions techniques. Cette remarque est loin d'être anodine et montre que le mode de production des normes de l'IETF repose sur le postulat que la bureaucratie administrative (*rough consensus*, établissement de chartes, procédures de création et dissolution des groupes de travail) permet de gérer au mieux les différends non seulement interpersonnels, mais surtout liés à une potentielle subjectivité technique. Cette approche de la bureaucratie est très proche de ce qu'entendaient Nyqvist et ses collègues :

Comprendre la bureaucratie en tant que phénomène social, c'est en grande partie comprendre comment les conventions ou la rationalité, le formalisme et l'indifférence entrent en jeu, mais c'est aussi comprendre la façon dont les gens se rapportent à la bureaucratie en tant qu'expression de sentiments et de conventions façonnés collectivement. (Nyqvist, Garsten, et Garsten 2013, 8, ma traduction).

La bureaucratie est en ce sens une « technologie de pouvoir » puissante dans le mode de « gouvernementalité » de l'IETF. Michel Foucault (1994a, 635ss) définit la gouvernementalité comme un mode spécifique d'exercice du pouvoir. Dans un texte justement consacré à la notion foucaldienne, Pierre Lascoume (2004, 6) résume ainsi la gouvernementalité :

Michel Foucault [...] formule ainsi son programme sur l'étude de la gouvernementalité : cette approche « impliquait que l'on place au centre de l'analyse non le principe général de la loi, ni le mythe du pouvoir, mais les pratiques com-

224 « Re: [openpgp] Requesting the editor to step down » : <https://mailarchive.ietf.org/arch/msg/openpgp/HnBZAcdePT6MimeQTRPHARiM6E/>.

plexes et multiples de gouvernementalité qui suppose d'un côté des formes rationnelles, des procédures techniques, des instrumentations à travers lesquelles elle s'exerce et, d'autre part, des enjeux stratégiques qui rendent instables et réversibles les relations de pouvoir qu'elles doivent assurer » [(Foucault 1994b, 584)]. Foucault souligne ainsi l'importance des « procédures techniques », de « l'instrumentation » en tant qu'activité centrale dans « l'art de gouverner ».

En organisant le travail de normalisation de protocoles autour d'outils tels le consensus approximatif, un archivage des discussions plénières et différentes procédures bureaucratiques, l'IETF propose un modèle de gouvernance qui se veut ouvert et inclusif. Toutefois, ce modèle n'est pas exempt de relations de pouvoir (Belli 2015). L'exemple de la constitution d'une *design team* est révélateur d'une importante tension entre la nécessité d'un processus de normalisation ouvert et un besoin de flexibilité au sein de l'infrastructure d'Internet (Hanseth, Monteiro, et Hatling 1996). Les procédures bureaucratiques sont certes des freins importants dans la mise à niveau des protocoles de l'IETF²²⁵, mais elles sont aussi le moyen mis en place par l'IETF pour tâcher de garantir un maximum d'ouverture et d'inclusivité.

7.3. La persistance d'OpenPGP

Au vu des complications que je viens de décrire, il peut paraître étonnant que le protocole OpenPGP reste une norme reconnue et maintenue. Maintenir cette norme à jour semble nécessiter non seulement de très grandes compétences techniques, mais aussi de solides attitudes sociales ainsi qu'une patience exemplaire. Le processus de normalisation de cette norme technique exige en effet un travail minutieux sur le long terme, travail qui est fait de compromis entre différents projets-sécurité, et de médiation entre des opinions souvent tranchées. Et tout ce travail aurait pour but, selon John Oliver, de faire en sorte que « rien ne se passe ».

Paradoxalement, pour qu'il ne se passe rien, ou pas grand-chose, de nombreuses personnes s'activent et le fruit de leur travail nous fait souvent soupirer : « Ah ! encore une mise à jour. » Étalaé dans le temps, mais aussi fragmenté en micro-événements et

225 L'exemple du protocole IPv6 est en ce sens particulièrement flagrant (DeNardis 2006) : après des décennies de négociations, cette norme n'a toujours pas été totalement adoptée.

parfois exposé au grand jour lors d'une spectaculaire divulgation, le travail de la sécurité informatique est facilement contesté et rarement valorisé. Cela fait partie du paradigme fondamental dont j'ai déjà parlé : la sécurité tient tant qu'il n'y a pas d'insécurité. Or, des insécurités, il y en a une infinité.

Dans ce chapitre, j'ai notamment souligné la manière dont certains problèmes persistent et comment le groupe de travail de l'IETF tâchait de pallier ces problèmes durant des années, voire des décennies. Face à cette infinité de problèmes dans les spécifications et dans le code des implémentations d'OpenPGP, la simple évocation de l'âge d'OpenPGP devrait susciter une certaine attention : durant plus de trois décennies, des gens ont transformé OpenPGP, par essais et erreurs, patchant et repatchant les bogues après parfois de longues et pénibles conversations. À l'instar des marins qui ont changé chacune des planches du bateau d'Ulysse dans l'Odyssée d'Homère, les mainteneuses d'OpenPGP sont prêtes à modifier toutes les parties d'OpenPGP pourvu que la norme reste utilisable et fiable. Ce sont des années d'usage, d'usure, de maintenance, de « crises » multiples, de vulnérabilités innombrables. Malgré toutes les critiques, les pressions, les découvertes de vulnérabilités, les primitives cryptographiques cassées et tout le reste, OpenPGP a survécu. Cela n'a rien d'anodin et démontre l'incroyable persistance de cette technologie – et donc la motivation, l'endurance et la résignation des gens qui s'en occupent. Une des pistes pour expliquer cette persistance est l'ancrage infrastructurel fort et profond d'OpenPGP dans de multiples milieux : non seulement celui des emails chiffrés, mais aussi et surtout celui des communautés open source qui utilisent la signature OpenPGP pour authentifier les contributions, que ce soit des paquets linux, des commits sur GitHub ou encore des contributions à Wikipédia.

J'aimerais clore ce chapitre par deux métaphores qui éclairent sous un autre angle les enjeux du présent et de l'avenir dans le soin des technologies numériques. Dans le premier cas, il s'agit de décrire l'état permanent de crise par la métaphore de la marche déjà évoquée ; la deuxième métaphore évoque quant à elle le point commun entre la belle au bois dormant et la sécurité informatique.

7.3.1. Un état de panne compensé

Pour conclure cette partie, donc, revenons à la divulgation publique d'EFail et la controverse qu'elle a suscité et décrite dans le chapitre précédent. Ce processus d'insécurité majeur et – il faut le dire – spectaculaire, a été amorcé par une équipe d'universitaires. Des centaines de publications sur les réseaux sociaux, forums, blogs et sites d'informations ont suivi l'annonce conjointe des chercheurs et de l'EFF. De nombreuses personnes ont formulé des inquiétudes, des opinions et des conseils. L'incertitude qui en découlait a jeté le doute à la fois sur la capacité d'OpenPGP de garantir la confidentialité, l'intégrité et l'accessibilité des messages chiffrés et sur le sérieux et l'éthique des chercheurs accusés d'avoir largement amplifié leur découverte afin de favoriser leur propre carrière. Cet accident a d'ailleurs donné de la voix aux nombreuses personnes qui critiquaient déjà ce protocole et qui considéraient que le monde irait mieux sans lui. Par exemple, dans une chronologie partielle de PGP publiée en janvier 2020, David Wong a affirmé qu'« EFail est la goutte d'eau qui a fait déborder le vase. PGP est officiellement mort » (Wong 2020, ma traduction). Je ne nie pas la rationalité qui préside à ce genre d'affirmation ; au contraire, elles montrent clairement que pour certaines personnes, OpenPGP ne répond plus à leur projet-sécurité et qu'il faut l'abandonner.

D'un autre côté, une foule hétérogène et inquiète d'utilisatrices a demandé des comptes et exigé des réponses rapides, claires et, surtout, rassurantes – réponses qui n'ont été délivrées que partiellement et de manière confuse. Dans cette situation pour le moins chaotique, il y a eu plusieurs figures « héroïques » : celle, controversée, des chercheurs qui ont trouvé une série de vulnérabilités critiques et qui ont tenté de les divulguer de la manière la plus sûre possible, celle des codeuses qui ont proposé des correctifs, celle de Robert J. Hansen qui n'a pas dormi de la nuit pour tenter de répondre à toutes les questions qu'on lui posait. Puis, avec le temps, les correctifs ont été implémentés et la controverse s'est essoufflée, faisant croire que les codes avaient été réparés définitivement. « Faisant croire », car le code n'est jamais vrai-

ment réparé, c'est-à-dire restitué à l'état neuf, mais plutôt affiné avec l'ajout (ou parfois, plus rarement, la suppression) de lignes de code.

Du côté du groupe de travail OpenPGP de l'IETF, la situation était différente. La réaction n'a pour le moins pas été immédiate et les discussions ont pris un temps considérablement plus long que la réparation du code. EFAIL n'a pas été un « début » et il n'y a jamais eu de « fin » définitive aux problèmes identifiés. La mise en contexte des problèmes évoqués permet de remonter du début des années 2000 sans qu'au printemps 2023 alors que j'écris ces lignes, une solution à long terme ait été actée au sein du groupe de travail. Cette vision encourage à contempler l'effort du groupe de travail sur la durée et non dans l'immédiateté des événements. Le temps de la maintenance est un temps long ; il ne me semble toutefois pas « non-événementiel » dans le sens qu'il consiste justement en un nombre impressionnant de pratiques et un engagement tout aussi impressionnant de la part des personnes qui prennent soin des technologies numériques. Ainsi, plutôt que de considérer que le travail de maintenance consiste à faire que « rien ne se passe », j'aimerais suggérer qu'il consiste à compenser l'instabilité causée par l'émergence permanente d'anciens et nouveaux problèmes. Il s'agit d'éviter les catastrophes annoncées et de faire en sorte que les choses continuent à fonctionner aussi bien que possible.

Ainsi, patcher le code, c'est corriger une erreur une fois, mais cela n'a que peu à voir avec le travail sur le long terme de la sécurité informatique. Ce travail consiste à patcher ce même problème décliné en différents bogues durant six mois, trois ans, dix ans (ou plus encore). De même, modifier les spécifications d'OpenPGP ne consiste pas à prendre une seule décision mais à proposer des changements au sein du groupe de travail et discuter ces propositions, et les rediscuter quelques semaines/mois/années plus tard, et les rediscuter encore un peu plus tard. Le but de ces discussions n'est pas d'obtenir un résultat définitif mais de parvenir à un compromis qui – on l'espère – tiendra le coup quelques années. Cela exige à la fois des dispositions techniques, mais aussi des compétences en négociation et de la patience, beaucoup de patience. D'un côté, il y a un sprint où le jeu consiste à patcher le code avant qu'il

ne soit attaqué ; de l'autre, il y a un marathon qui consiste à avoir l'endurance de répéter ce sprint encore et encore – et cela sans avoir la moindre idée de où et quand aboutira la course. L'« infini recommencement » semble effectivement être le cheval de bataille du soin de la sécurité à condition de considérer le fait que ce ne sont jamais les mêmes tâches qui sont recommencées. Le soin des choses informatiques nécessite de bricoler toujours et encore de nouvelles solutions aux problèmes anciens et nouveaux qui ne cessent jamais d'arriver.

7.3.2. Le temps long des vulnérabilités : à propos du point commun entre la Belle au bois dormant et les vulnérabilités

Je l'ai déjà affirmé à plusieurs reprises, les vulnérabilités informatiques sont les objets centraux du paradigme dominant de la sécurité informatique. Cela pose une question légitime : quand naît et quand meurt une vulnérabilité ? Une façon de répondre à cette question est d'étudier le cycle de vie de ces vulnérabilités, ce qui a été fait à plusieurs reprises (Arbaugh, Fithen, et McHugh 2000; Frei, Tellenbach, et Plattner 2008; Vyas 2013). Toutefois, le cas d'EFAIL semble problématique, une fois de plus. La naissance de la vulnérabilité remonte à la conception même du protocole et des algorithmes de chiffrement utilisés qui ne permettent pas le chiffrement authentifié. Or, en 2000, cette vulnérabilité a été identifiée théoriquement et l'on pourrait considéré qu'il s'agit ici d'une première découverte. Ainsi, une même vulnérabilité peut être découverte et divulguée plusieurs fois. À l'époque, Werner Koch avait proposé le MDC comme patch et selon l'email envoyé à la suite d'EFAIL, il semblerait qu'il soit toujours efficace contre les vulnérabilités d'EFAIL. Toutefois, la découverte et la divulgation des vulnérabilités d'EFAIL montrent que ce patch n'a pas fonctionné, à l'instar de ceux qu'avait proposés Enigmail en 2018 suite à EFAIL. Avec le recul, on pourrait considérer que la fenêtre d'exposition a duré près d'une vingtaine d'années, voire encore plus vu que les spécifications d'OpenPGP n'ont toujours pas été officiellement mises à jour à l'IETF en été 2023. EFAIL n'est donc jamais morte – du moins de ce point de vue.

De plus, l'article d'EFAIL pour la conférence Usenix (Poddebniak et al. 2018) n'a été que le premier d'une plus longue série (Müller, Brinkmann, Poddebniak, Böck, et al. 2019; Müller, Brinkmann, Poddebniak, Schinzel, et al. 2019; Müller et al. 2020; Schwenk et al. 2020). EFAIL est bien plus qu'une série de vulnérabilités, et EFAIL a également persisté dans le temps et dans l'écosystème OpenPGP. Cela souligne que l'existence et l'implémentation de correctifs ne sont pas suffisantes pour rendre tous les utilisatrices sûres. Arbaugh et ses collègues (2000) affirment qu'une vulnérabilité peut « naître » dans la phase de conception d'un projet et qu'elle « meurt » au moment où le nombre de systèmes qu'elle peut affecter se réduit à un niveau insignifiant. Ils affirment qu'« en pratique, les administratrices ne peuvent jamais patcher tous les systèmes vulnérables. Ils finissent inévitablement par retirer ces systèmes, et les attaquant·e·s et les médias perdent fréquemment leur intérêt pour la vulnérabilité – mais parfois seulement de manière temporaire, créant ainsi une expérience proche de la mort » (Arbaugh, Fithen, et McHugh 2000, 54s ma traduction).

Ainsi, la mort n'est jamais complètement effective. Ou du moins, ce n'est pas la vulnérabilité qui meurt mais le système qui a appris à se défendre. Or, comme l'ont montré différentes attaques, et notamment le rançongiciel WannaCry, ce sont souvent des vulnérabilités « mortes » ou du moins déjà patchées dans les systèmes mis à jour que les attaques informatiques exploitent (Aminot 2020). Ainsi, à l'instar de la Belle au bois dormant, la fin d'un cycle d'une vulnérabilité endormirait (plus que ne tuerait) la vulnérabilité ; entourée d'une épaisse forêt de ronces bâtie par d'ingénieuses codeuses, la faille reposerait, passive, endormie, mais encore vivante, dans l'attente qu'un sinistre baiser vienne la réveiller au grand dam des utilisatrices – et surtout celles et ceux qui n'auraient pas mis à jour leur système [souponner].

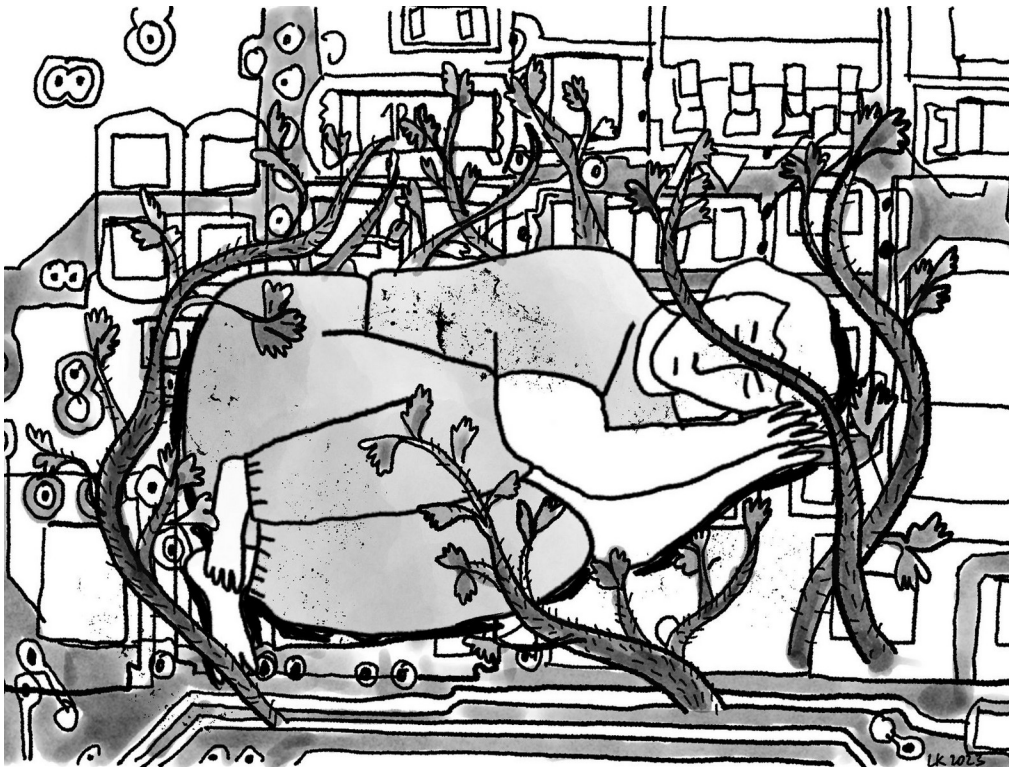


Illustration 27: Bogue au bois dormant, dans l'attente qu'Ève (ou Mallory ou peut-être Johnny) vienne la réveiller. Source: dessin de Léa Kissling (@lea_kissling_officielle).

Quoi qu'il en soit, on peut supposer que ce « baiser » constituera un nouvel événement qui déclenchera un nouveau « cycle » de maintenance. C'est pourquoi il semble qu'une approche de la sécurité informatique par les événements (Bensa et Fassin 2002) se justifie pour comprendre la manière dont les technologies numériques se transforment petit à petit, et transforment l'infrastructure autour d'elles. Cela veut dire que chaque vulnérabilité, chaque panne, chaque bogue peut impliquer certains changements importants dans la « vie » des technologies et, partant, des infrastructures qui les soutiennent. S'inspirant du sens courant d'une route se divisant en deux, Daniel Florentin (2018, 244) définit une « bifurcation infrastructurelle » dans un environnement urbain comme « un changement radical d'orientation dans le fonctionnement des systèmes techniques urbains ». Les technologies du numérique ne sont pas des routes et peuvent se développer dans différentes directions – OpenPGP en est bien la preuve. Toutefois, certaines dispositions – ou indispositions – peuvent avoir d'importants impacts sur la manière dont la technologie ou l'infrastructure évoluent. Ainsi, même si ce ne sont peut-être pas des bifurcations au sens propre mais

plutôt de nouvelles combinaisons de branchements qui aiguillent l'écosystème dans certaines directions plutôt que d'autres, ces bogues, pannes et vulnérabilités ont des conséquences importantes sur les technologies et sur leur développement. D'une part, ces problèmes ont une incidence sur les usages de ces technologies et la manière dont on les perçoit. D'autre part, les nombreux patches qui sont appliqués au code et les multiples ajouts qui sont appendus aux spécifications de la norme complexifient fortement la technologie et en contraignent les développements futurs. Ce sont d'ailleurs ces développements que je propose d'aborder dans la dernière partie de cette thèse.

7.3.3. La direction du soin : vers la résilience prospective

La distinction que j'ai faite entre l'urgence du patch et les lentes négociations à l'IETF entrent dans la ligne des travaux sur les *maintenance and repair studies*. Dans leur ouvrage *Le soin des choses* (2022), Jérôme Denis et David Pontille ont notamment résumé la distinction généralement admise entre la réparation et la maintenance. Ils affirment que la notion de réparation « est étroitement liée aux figures de la panne et de l'accident » (p. 37) et qu'elle « pointe vers un horizon dégagé, tout en désignant un impératif moral » (p. 38) : la réparation est un retour à l'état neuf (ou presque neuf). C'est un acte unique et définitif dans le sens qu'une fois réparé, l'objet ne demande plus aucune attention (jusqu'à sa prochaine casse, évidemment). La réparation relève ainsi du champ sémantique de l'exceptionnel et de l'héroïsme car la personne qui répare est une sauveuse. Dans le cas de la divulgation d'EFAIL, Robert J. Hansen a par exemple particulièrement endossé ce rôle de « héros » qui sacrifie son confort et sa paix pour secourir des victimes innocentes.

D'un autre côté, la maintenance est une activité routinière et étalée sur le long terme qui vise à faire « durer les choses » *avant* qu'elles ne se cassent. Jérôme Denis et David Pontille (2022) insistent particulièrement sur le temps que prend la maintenance et son travail ordinaire et non-événementiel qui consiste en un « infini recommencement » des mêmes tâches de soin. C'est pourquoi les personnes qui en ont la charge ne sont que peu visibles par leurs actes :

À la différence de la réparation, la maintenance oblige d'une part à se préoccuper de ce qui ne fait pas événement et d'autre part à prendre en considération toutes celles et tous ceux qui y participent, qui ne sont ni héroïnes ni héros. (Denis et Pontille 2022, 45)

En considérant cette opposition entre réparation et maintenance, il serait tentant de distinguer deux logiques séparées dans le soin d'OpenPGP. D'une part, la logique de la casse et de la réparation, exemplifiée par la divulgation publique d'EFail, et d'autre part, la logique de la maintenance sur le long terme au niveau de l'IETF dont nous avons vu l'illustration dans le présent chapitre. Dans certains cas, par exemple une *issue* sur GitLab, la logique de la casse prédomine : le bogue est réparé et le logiciel fonctionne à nouveau comme il était censé le faire avant. Dans d'autres cas que j'aborde plus en détail dans le chapitre suivant (« Penser les fragilités à venir pour panser l'avenir », p. 300), la logique de la maintenance est cruciale pour faire durer la technologie jour après jour, mois après mois, années après années. Toutefois, dans de nombreux cas, ces deux logiques s'entremêlent largement dans le soin des technologies numériques : ce qui se casse est aussi en rapport avec ce qui se négocie difficilement depuis des années, par exemple un mode de chiffrement authentifié (l'AEAD et sa *chunk size*). Cela est crucial pour comprendre le soin porté aux technologies de sécurité informatique, car souvent, les grosses pannes qui font les grands titres et font trembler les utilisatrices sont dues à des problèmes identifiés depuis longtemps, ou à des logiciels non mis à jour²²⁶. Cette partie sur les rythmes de la sécurité a montré justement les différentes logiques qui permettent d'aller bien au-delà de la simple constatation de l'oscillation permanente entre sécurité et insécurité que décrivent les théories de la sécurisation. Il y a un lien profond entre la lente évolution des spécifications des technologies et le travail de réparation immédiat et souvent urgent des mainteneuses : ces deux rythmes s'enchevêtrent dans ce que j'appelle la (p)réparation de l'avenir. J'aborde dans le détail cette nouvelle logique du soin des choses (informatiques) dans la dernière partie de cette thèse intitulée « La durée de l'avenir ».

226 C'était notamment le cas du *ransomware* WannaCry (Aminot 2020).

Troisième partie : La durée de l'avenir

Nous voici presque au bout du cheminement à travers les temps du soin de la sécurité informatique. La première partie a montré comment les technologies numériques vieillissaient d'une part en s'enlisant dans l'infrastructure et d'autre part en se complexifiant irrémédiablement. En analysant la manière dont le passé représentait à la fois un poids contraignant et gênant pour les développeuses (l'enlissement infrastructurel) et une base solide qui offrait les conditions mêmes de la « survie » d'OpenPGP (l'ancrage infrastructurel), j'ai souligné la dimension du mouvement de cet état de déséquilibre compensé. Prendre soin d'une technologie, la faire vieillir, c'est la maintenir en mouvement pour qu'elle demeure fonctionnelle dans un écosystème qui, lui, ne cesse jamais d'évoluer. Car si elle s'arrête, elle meurt.

La deuxième partie a quant à elle précisé en quoi consistait concrètement le travail de soin de la sécurité informatique. À travers l'analyse de la trajectoire d'EFAIL, c'est le quotidien des technologies associées à OpenPGP qui a été examiné. Différents rythmes, de l'urgence du patch aux lentes négociations, coexistent au quotidien et mon analyse a montré les multiples façons de vivre et faire ce travail de soin de la sécurité informatique. Ces deux parties ont permis de comprendre quelque chose de fondamental dans le travail des développeuses : il ne se terminera jamais parce qu'il y aura toujours de nouvelles tâches à accomplir. En d'autres termes, le travail de

soin de la sécurité informatique n'atteindra jamais un stade « final » – ou alors ce sera parce qu'il aura été abandonné, ce qui, parfois, est aussi une bonne solution²²⁷.

Comme dans le jeu du pont Lego que l'enfant doit maintenir jusqu'à sa mort²²⁸, le soin d'une technologie du numérique n'a pas de date de péremption. Toutefois, une différence fondamentale existe entre un pont Lego et une technologie numérique. Le pont est fait non seulement de calculs et de connaissances en génie civil, mais aussi de matière qui le constitue et qui est directement menacée par les effets du temps. Au contraire, une technologie telle qu'OpenPGP est de nature informationnelle et n'est de ce fait pas soumise au même régime d'usure mécanique. Elle est en quelques sortes physiquement protégée contre les affres du temps. Bien entendu, cette information dépend des supports matériels qui, eux, s'abîment (Denis 2012), mais l'information en tant que telle peut être aisément copiée à l'identique sur un autre support sans qu'elle ne subisse de détérioration significative. Toutefois, si l'information n'est pas directement menacée par l'usure causée par le temps qui passe, elle l'est par ce que l'avenir prépare. L'enjeu est donc radicalement différent que pour le soin des choses matérielles : prendre soin d'OpenPGP implique nécessairement à la fois des tâches pour s'accommoder du passé et des tâches d'anticipation de l'avenir, pour le rendre résilient face aux futures failles.

Dans la dernière partie de ma thèse, j'analyse la troisième dimension des temps de la sécurité : la durée. Étudier la durée de la sécurité, c'est nécessairement se projeter dans l'avenir, non seulement pour imaginer combien de temps la sécurité durera, mais également quelles durées importeront à l'avenir. Dans le premier cas, il s'agit de questionner jusqu'à quand une primitive cryptographique permettra de protéger une information. Comme je l'ai déjà mentionné, on estime par exemple le temps nécessaire pour essayer toutes les combinaisons possibles d'une clé AES 256 bits avec les

227 Par exemple, la version 5 du protocole IP n'a jamais été officialisée. Les discussions ont eu lieu dès les années 1990 pour répondre aux limites connues de l'IPv4 (déjà opérationnel à l'époque et toujours utilisé actuellement) et de l'IPv5 (qui n'a jamais été implémenté). IPv6 a commencé à être déployé dès 2006 et fonctionne actuellement en parallèle et de manière incompatible à l'IPv4.

228 Rappelons-nous l'exemple du Lego : « Il est livré construit. Tu le maintiens, et si tu le fais correctement, rien ne se passe. Et à la fin, tu meurs. » (John Oliver cité par Denis et Pontille 2022, 46).

moyens actuels à plusieurs milliards d'années. Or, l'arrivée de nouveaux moyens de calculs, et notamment les ordinateurs quantiques, pourrait à terme réduire ce chiffre astronomique comme une peau de chagrin. On estime en effet qu'il est probable que ces mêmes opérations ne prennent plus que quelques semaines, voire quelques jours ou même peut-être quelques heures. Cela représenterait un danger important pour de nombreuses personnes.

Dans le deuxième cas, en se demandant quelles durées importeront à l'avenir, il s'agit de questionner d'autres durées qui sont beaucoup plus abstraites. On sait par exemple que certaines techniques de cryptanalyse permettent de calculer une clé privée en mesurant le temps que durait une opération²²⁹. Ces temps se mesurent en millisecondes et du fait que les moyens de mesure de ces durées ne cessent d'augmenter, il y aura toujours une incertitude de ce côté également.

Les durées sont donc un problème majeur de la sécurité, problème qui est d'ailleurs renforcé par les incertitudes de l'avenir. Dans cette partie, je présente la manière de penser les fragilités à venir afin de panser l'avenir. Je propose de passer d'une vision du travail du soin qui préserve à un travail de soin qui (se) projette. D'ordinaire, les activités de soin sont décrites comme passéistes ou ancrées dans le présent. La réparation prend comme point de référence l'état neuf (ou du moins non cassé), situé dans le passé. Il s'agit typiquement de préserver l'authenticité d'une œuvre d'art (Rubio 2020), d'une voiture de collection (Hummel et Desaleux 2018) ou – moins typiquement – de préserver le corps de Lénine tel qu'il était à sa mort (Denis et Pontille 2022). La maintenance consiste quant à elle à prolonger l'état actuel des choses – ce qu'on appelle souvent « entretien ». Les exemples typiques consistent en l'entretien de la propreté du sol (voir les travaux de l'artiste Mierle Laderman Ukeles analysé dans Denis et Pontille 2022), de la réparation des objets du quotidien (Greg-

229 Il s'agit des attaques temporelles. J'en ai notamment parlé à propos de la librairie RustCrypto qui ne propose pas des opérations en temps constants et qui est donc vulnérable à certaines attaques par canal auxiliaire (cf. section 5.1.3 p. 202). Oui, je sais, cette note est un peu complexe à comprendre pour les personnes qui ne sont pas initiées et c'est pourquoi je suggère la lecture de l'article de Wikipédia qui décrit ces attaques avec bien plus de détails : https://fr.wikipedia.org/wiki/Attaque_temporelle.

son, Metcalfe, et Crewe 2009) ou de respecter la norme des panneaux de signalisation du métro parisien (Denis et Pontille 2010)²³⁰. Or, le travail de soin d'OpenPGP présente un autre régime de temporalité dirigé vers l'avenir et le point de référence, l'objectif visé, ne se situe ni dans le passé ni dans le présent mais dans l'avenir, c'est-à-dire dans ce que l'on souhaite que la technologie soit capable de faire. Cela implique non seulement d'apprendre à penser l'avenir malgré toutes les incertitudes qu'il présente, mais aussi d'imaginer des manières de panser les failles à venir. C'est à ce renversement du sens du soin que convie cette partie : un soin futuriste ayant pour objectif une résilience prospective.

Cette partie interroge des pratiques d'innovation et d'anticipation qui semblent consubstantielles du soin des choses numériques. Quoique fort éloignée de la rhétorique disruptive et spectaculaire généralement liée à son évocation, une certaine forme d'innovation participe grandement aux activités de soin des choses informatiques. Dans le chapitre 8 « Penser les fragilités à venir pour panser l'avenir » (p. 300), je montre comment la maintenance et l'innovation – loin d'être opposées – fonctionnent ensemble dans une logique de soin. Deux exemples permettent de comprendre comment la préparation du futur permet de formaliser des connaissances et en générer de nouvelles : la documentation du code d'une part, et des élucubrations pour implémenter une nouvelle fonctionnalité d'autre part. Le chapitre 9 « Équiper OpenPGP face aux ordinateurs quantiques » (p. 326) est quant à lui dirigé vers la manière de construire des connaissances ou du moins des suppositions sur ce vers quoi l'on se dirige, avec l'exemple de la cryptographie résistante aux ordinateurs quantiques.

230 Ces différents exemples ont tous été présentés dans « Le soin des choses » (Denis et Pontille 2022).

8. Penser les fragilités à venir pour panser l'avenir

L'avenir est souvent vu, imaginé et même décrit à travers le prisme de l'innovation. Cette innovation est par ailleurs souvent fantasmée, surtout quand il s'agit des technologies du numériques qui sont si souvent perçues comme « la solution à tous les problèmes » (Morozov 2014). En effet, les « nouvelles » technologies de l'information sont généralement décrites en utilisant un vocabulaire révolutionnaire qui brosse un tableau où le monde sera – au futur – radicalement meilleur après la venue de telle ou telle technologie. L'innovation se doit d'être spectaculaire, géniale et disruptive²³¹. En ce sens, elle incarne l'idéologie du progrès généralement représentée par une ligne du temps parfaitement linéaire et entièrement orientée vers un monde à venir supposé toujours meilleur :

[L'innovation] est même un idéal à atteindre : il faut innover et être innovant ! Les réactions d'une classe d'élèves-ingénieurs à l'évocation du terme « innovation » drapent cet impératif catégorique d'un imaginaire radieux : « progrès », « avenir », « créativité », « technologies », « améliorer le quotidien », « bénéfices » sont fréquemment cités quand l'exercice est réalisé. (Gaglio 2011, 3)

Un merveilleux exemple de la rhétorique de l'innovation révolutionnaire et de toutes ses promesses concerne la communication quantique. Il s'agit d'utiliser cer-

231 Dans la rhétorique néo-libérale typique de l'innovation actuelle, « disruptif » est un anglicisme relativement nouveau qui tend à remplacer le mot « révolutionnaire », peut-être par trop ancré dans une rhétorique communiste.

taines propriétés des particules quantiques, notamment l'intrication, pour communiquer de manière confidentielle sur des grandes distances. Stéphane Bortzmeyer, informaticien et commentateur avisé et critique de l'actualité d'Internet, a écrit un court billet de blog où il dénonce la manière dont les médias traitent les informations en lien avec ce champ de recherche²³². Il critique notamment un article de l'Express qui vulgarise en exagérant généreusement le trait les résultats d'une publication scientifique à propos de la communication quantique²³³. Bortzmeyer souligne notamment certaines tournures qui montrent un engouement certain pour l'innovation spectaculaire, comme par exemple un « Internet inviolable » et « basé à l'avenir sur les principes de la physique quantique », des « ordinateurs dotés d'une puissance de calcul sans commune mesure avec les plus puissantes machines actuelles », et même de la « téléportation ».

Penser l'avenir à travers le prisme principal du soin plutôt que celui de la solution fait prendre conscience de la fragilité de cet avenir, de son incertitude et des problèmes qu'il risque de nous apporter, qu'il nous apportera sûrement, voire qu'il nous apporte déjà. Jérôme Denis et David Pontille ont subtilement décrit comment une attention soutenue aux activités de maintenance des choses matérielles (en l'occurrence des panneaux du métro parisien) leur ont donné à voir « un monde radicalement nouveau » où « tout ne fonctionnait pas aussi simplement [qu'ils l'avaient] imaginé » (Denis et Pontille 2022, 68) ; dans mon cas, l'attention que j'ai portée aux activités de soin d'OpenPGP a transformé ma vision des technologies numériques en me faisant voir un monde nouveau dans lequel c'était l'avenir même qui ne fonctionnait plus aussi bien. En effet, il n'était plus synonyme de promesses mais de périls, de fragilités et de soucis.

Comme l'a illustré Edgerton dans son livre *The Shock of the Old* (2008), la caractéristique principale des technologies n'est pas l'innovation mais leurs dysfonctionnements, leurs pannes et la maintenance qu'elles engendrent. La casse est la norme,

232 <https://www.bortzmeyer.org/communication-quantique.html>.

233 https://www.lexpress.fr/sciences-sante/sciences/teleportation-quantique-tout-comprendre-a-la-prouesse-realisee-par-la-chine_1918657.html.

même si les tâches de réparation et de maintenance ne sont souvent pas valorisés. Par ailleurs, Steven Jackson souligne l'importance de sortir d'une rhétorique centrée sur la nouveauté :

[...] Les conceptions productivistes dominantes de la technologie placent l'innovation, avec son statut inattaquable, son cachet culturel et sa valeur économique valorisée, au sommet d'un changement ou d'un processus, tandis que la réparation se trouve quelque part ailleurs : plus bas, plus tard ou après l'innovation en termes de processus et de valeur. Mais il s'agit là d'une représentation fausse et partielle de la manière dont les outils technologiques fonctionnent réellement, lorsqu'ils fonctionnent. (Jackson 2014, 227, ma traduction)

Ce chapitre vise à contribuer à cet appel à développer une « pensée du monde cassé » lancé par Jackson. En interrogeant l'importance de l'innovation dans notre monde que l'on dit volontiers gouverné par l'innovation technologique, je montre comment le souci de l'avenir peut aussi s'observer dans des pratiques quotidiennes de soin. Pour cela, je commence par faire une plongée dans le passé en revenant aux premiers temps d'OpenPGP. Le logiciel de Zimmermann a été une innovation capitale dans le domaine de la protection des données de la population civile et est d'ailleurs souvent citée comme une révolution en la matière (Singh 1999; Levy 2001; Coleman et Golub 2008). Toutefois, dès ses débuts, cette innovation, cette révolution que l'on qualifierait aujourd'hui volontiers de disruptive, a reposé sur des pratiques intenses et capitales de soin. Autrement dit, l'innovation repose simultanément sur la nouveauté et sur les pratiques incessamment renouvelées de correction, d'affinage et d'anticipation. La première section permet de nuancer le rôle de l'innovation et de souligner l'importance de la maintenance dans le succès (ou non) des technologies. Après cette plongée dans le passé, je me tourne vers l'avenir et analyse comment ce dernier est préparé au quotidien par des tâches de soin qui ont comme ligne de mire non par l'innovation à venir, ni la réparation du passé, mais ce que je nomme la « résilience prospective » de la technologie.

8.1. L'innovation est capitale... de même que la maintenance

Peut-être faut-il commencer par donner une définition un peu plus précise de ce que j'entends pas « innovation ». Pour Gérard Gaglio, « une innovation est une invention qui s'est répandue » (Gaglio 2011, 4). Il reprend en ce sens la distinction faite par Norbert Alter (2002) entre invention et innovation pour insister sur le fait que c'est l'adoption et l'usage d'une technologie qui transforme une invention en innovation. L'innovation dépend donc largement du contexte et du moment. Madeleine Akrich avait d'ailleurs affirmé la même chose en insistant sur le rôle clé des utilisatrices dans les processus d'innovation (Akrich 2006b). J'aimerais ici apporter un autre pièce du puzzle : la métamorphose d'une invention en innovation dépend aussi largement de la manière dont la technologie en question est maintenue dès ces premiers moments d'existence. En effet, une innovation n'est pas seulement une invention qui s'est répandue (Gaglio 2011), mais aussi et peut-être surtout une invention qui dure. L'exemple que j'aimerais donner pour illustrer nous replonge plus de trente années en arrière, en 1991, alors que Philip Zimmermann venait d'accoucher de Pretty Good Privacy.

8.1.1. L'exemple de la création de PGP pour nuancer l'importance de l'innovation

L'exemple de la première année de Pretty Good Privacy est riche d'enseignements. On retient souvent la nuit du 5 juin 1991 comme le moment où PGP est né. Mais cette version 1.0 du programme a été publiée à la hâte – Zimmermann avait même oublié de mettre à jour la date du copyright (Garfinkel 1995, 99). Elle fonctionnait, certes, et plusieurs personnes se sont approprié cette nouvelle technologie. Elle avait toutefois de nombreux défauts. En août 1991, quelques semaines à peine après cette publication, Zimmermann a rencontré Eli Biham lors de la conférence Crypto à Santa Barbara en Californie. Voici le récit que Simson Garfinkel en fait :

Après que Biham et Zimmermann ont récupéré leur nourriture et se sont assis, Zimmermann a sorti quelques pages de fichiers informatiques. En quelques minutes, Biham a trouvé des failles fondamentales dans Bass-O-Matic. Certains de ces défauts étaient des faiblesses subtiles qui rendaient l'algorithme vulnérable à

la cryptanalyse différentielle, qui était la spécialité de Biham. D'autres étaient plus embarrassantes, comme une erreur conceptuelle dans l'algorithme de Zimmermann qui empêchait le dernier bit de chaque octet d'être correctement crypté. Après dix minutes d'assauts de Biham, Zimmermann a compris que Bass-O-Matic était une cause perdue. Il s'est également rendu compte d'une chose qu'il savait depuis longtemps, mais qu'il n'avait jamais vraiment admise : seuls les cryptographes de renommée mondiale étaient en mesure de créer des algorithmes de chiffrement fiables. Les autres personnes qui s'y essaient sont soit des génies méconnus, soit des imbéciles. Zimmermann avait appris sa leçon. (Garfinkel 1995, 102, ma traduction)

Cette première version de PGP était certes une révolution sous plein d'aspects ; elle n'en était pas moins pleine de défauts. Et si PGP est effectivement devenue une invention qui a fait date en matière de protection de la vie privée en ligne, ce n'est certainement pas uniquement dû à ce premier coup du 5 juin 1991. Au contraire, dès le début, il a fallu s'en occuper, et cela impliquait entre autres de s'entourer de personnes compétentes et motivées. Ce n'est donc qu'avec la version 2.0 publiée en automne 1991 que PGP est vraiment devenu « *The real ThingPGP* » (S. Garfinkel 1995). En inventant Pretty Good Privacy, la contribution de Philip Zimmermann a été essentielle, audacieuse et probablement géniale ; mais Pretty Good Privacy n'est devenu une innovation majeure que parce que Zimmermann a su s'entourer de personnes qui, à un moment ou à un autre, l'ont aidé à prendre soin de son « bébé ».

En effet, le grand paradoxe de cette histoire est que PGP aurait pu disparaître quelques mois après sa publication : en plus des poursuites judiciaires par le FBI qui auraient certainement découragé de nombreuses personnes, beaucoup d'erreurs techniques étaient présentes dans la première version de PGP et auraient pu être fatales au programme. Si le travail s'est poursuivi, si PGP a survécu, c'est parce que de nombreuses personnes ont trouvé un sens dans l'intention de Zimmermann et ont rejoint son projet en l'utilisant d'une part et en y contribuant d'autre part. À mes yeux, la prouesse de Zimmermann n'est pas d'avoir offert de la sécurité comme s'il s'agissait d'une matière ou d'une solution magique : « tiens, voici Pretty Good Privacy, tu seras en sécurité. » Son véritable exploit tient à mon avis beaucoup plus au fait d'avoir tenu bon face aux critiques, aux difficultés techniques et aux poursuites du

FBI, et d'avoir fait en sorte que PGP soit réparé, maintenu et préparé à ce que l'avenir préparait.

Suivre les dysfonctionnements et la manière d'y pallier est donc une piste majeure non seulement pour comprendre comment les technologies continuent d'exister à travers le temps, mais aussi, et peut-être surtout, comment ces technologies émergent et passent du statut d'invention à celui d'innovation. Dans les deux premières parties de cette thèse, j'ai montré qu'une technologie continue d'exister, il faut qu'un groupe de personnes prennent soin d'elle. Cette condition de survie a été identifiée depuis longtemps par les études sur la maintenance (Henke 2000; Graham et Thrift 2007; Denis, Mongili, et Pontille 2015). L'analyse que je viens de faire permet toutefois d'affirmer plus radicalement que pour les technologies numériques, la maintenance est une condition première à l'innovation. En effet, l'identification d'un moment d'innovation comme le fameuse nuit du 5 juin 1991 ne peut advenir que si et seulement si des personnes s'affairent déjà à réparer, modifier, transformer ladite technologie dès son apparition au monde. Sans cette quasi immédiate série d'actions, l'invention ne peut devenir innovation car elle ne sera déjà plus « à jour ». Ce que l'histoire retient est généralement l'acte fondateur, avec notamment la persistance tenace du mythe des « pères fondateurs » déjà décrié par Callon (1994)²³⁴. Toutefois, cet acte fondateur s'inscrit dans une chaîne d'actions beaucoup plus longue, des premières recherches en amont de la « date de naissance » à la série infinie d'interventions en aval.

Ces dysfonctionnements peuvent être techniques, telles les faiblesses de l'algorithme Bass-O-Matic de Zimmermann, mais aussi juridiques, comme l'illustre l'exemple des poursuites contre Zimmermann accusé d'exporter du matériel de guerre ou les récurrents problèmes de licences dans le monde de l'Open Source²³⁵. On peut aussi facilement imaginer des dysfonctionnements sociaux, par exemple au sein des équipes de développement, ou psychologiques : il est probable que peu de

234 Et qui seraient les mères, d'ailleurs ?

235 Voir par exemple le court billet de blog d'un membre de l'équipe de Sequoia sur l'arithmétique des licences Open Source : <https://ideas.liw.fi/licence-arithmetic.html>.

monde aurait supporté les pressions exercées sur Zimmermann durant plusieurs années, ou les incessantes critiques qu'a subies Werner Koch durant des années, alors qu'il développait (presque) seul GnuPG, la seule implémentation libre et gratuite d'OpenPGP durant des années.

Cet exemple des débuts de PGP montre que l'innovation appelle nécessairement à des pratiques de soin qui – on la vu dans la première partie de cette thèse – complexifie inévitablement la technologie. Ainsi, dès son émergence, une technologie s'enlise. Malgré tout leur nouveauté, les technologies numériques s'enfoncent irrémédiablement dans les sables mouvants de la maintenance puisque les maintenir signifie les rendre de plus en plus complexes, de plus en plus épaisses, de plus en plus ancrées dans l'infrastructure.

Ces pratiques de soin présentent deux faces. D'une part, il s'agit de réparer les erreurs commises dans le passé, ce qui a été vu dans la deuxième partie de cette thèse. D'autre part, il s'agit de préparer la technologie à ce qui va arriver. Réparer le passé et préparer l'avenir sont les deux directions du soin des choses informatiques. L'exemple suivant permet de se faire une idée des modestes et néanmoins complexes innovations qui permettent de préparer l'avenir et contribuent dès lors au soin d'OpenPGP.

8.1.2. « Être à l'avant-garde du progrès »... en changeant un paramètre par défaut

Le 15 janvier 2019, Sunny Rajan, qui travaillait pour ProtonMail, a envoyé un email sur la liste de diffusion du groupe de travail OpenPGP pour signaler que son entreprise avait décidé de favoriser les algorithmes de chiffrement à courbes elliptiques Ed25519/Curve25519 par défaut pour la création des nouvelles clés²³⁶. Dans ce court email, Sunny Rajan ajoutait que ProtonMail était conscient que cela pourrait causer des problèmes d'interopérabilité avec les implémentations qui ne supportaient pas encore ces deux algorithmes de chiffrements et d'authentification et qu'il ne fallait pas hésiter à les contacter en cas de problèmes. Cet email n'a reçu que trois ré-

²³⁶ La discussion peut être consultée sur la liste de diffusion du groupe de travail : https://mailarchive.ietf.org/arch/msg/openpgp/_GCmrofKP8kkBOQeUyKgCo5KPCA/.

ponses, toutes allant plus ou moins dans le même sens : très bien, on enverra un signe sur cette liste en cas de problèmes. Une des réponses se terminait ainsi :

Je suppose qu'il y a d'autres implémentations (moi y compris) qui pensent faire le pas et changer cette configuration par défaut.

Donc félicitations pour votre courage d'être à l'avant-garde du progrès. :-)

(Membre du groupe de travail OpenPGP, liste de diffusion du groupe de travail, 15 janvier 2019)²³⁷

Avoir « le courage d'être à l'avant-garde du progrès » : voici donc ce qu'avait fait ProtonMail avec cette décision. Arrêtons-nous un instant sur la formulation : en premier lieu, je note encore une fois une rhétorique masculiniste (avant-garde, courage) et l'idée d'un progrès linéaire, mais ce n'est pas tout. Il y a des félicitations, ce qui, dans cette liste de diffusion et venant d'une implémentation concurrente, n'est pas anodin. C'était, visiblement, la bonne décision à prendre et il fallait avoir le courage de le faire. Je pense que « courage » peut se comprendre ici selon deux éléments : le courage de devancer l'état actuel de « status quo » d'une part²³⁸, et, d'autre part, avoir le courage de potentiellement devoir affronter de multiples demandes d'utilisatrices mécontentes car elles n'arriveraient pas à communiquer avec certain·e·s de leur correspondant·e·s. Dans ce cas, ProtonMail était peut-être l'implémentation la plus « armée » (pour suivre la métaphore guerrière) pour affronter les potentielles « hordes » de plaintes puisque c'était la plus grande équipe de développement de l'écosystème.

Cette décision de ProtonMail d'implémenter cet algorithme par défaut a été une décision qui a, semble-t-il, fait l'unanimité et a été suivie dans les prochaines années par les autres implémentations. Par ailleurs, à ma connaissance, aucune plainte n'a été formulée par rapport à cela²³⁹. Il ne s'agit donc pas d'un « tour de force » qui a contraint d'autres implémentations, comme cela est parfois dénoncé quand de

237 « Re: [openpgp] ProtonMail moving to default X25519 keys » : https://mailarchive.ietf.org/arch/msg/openpgp/hgIKarGcfQulE9V_syX1kBctE5k/.

238 Le groupe de travail n'était pas encore reformé à cette époque.

239 Ces plaintes auraient probablement été faites *via* des canaux privés et je ne peux pas affirmer que ce changement de paramètres par défauts s'est fait sans aucun heurt. Toutefois, le fait que les autres implémentations ont eu tendance à suivre cette mesure souligne un certain consensus et une certaine assurance en la matière.

grandes entreprises comme Google ou Microsoft « imposent » leurs normes à l'IETF. La décision qu'a prise ProtonMail était, dans le cas présent, « logique » et relativement consensuelle, du moins officieusement : les deux algorithmes étaient décrits dans le RFC 6637²⁴⁰ et déjà largement implémentés par les grandes implémentations d'OpenPGP. De plus, ces algorithmes étaient plus rapides, plus puissants, et le matériel des clés plus légers. Ils étaient par ailleurs largement reconnus comme fiables et sécurisés par la communauté scientifique qui les étudiaient depuis plusieurs années, ainsi que « testés et approuvés » par d'autres logiciels utilisés « dans le vrai monde » avant eux.

Dans un billet de blog qui a accompagné la sortie de la nouvelle version du logiciel de ProtonMail²⁴¹, Ben Wolford a particulièrement insisté sur la dimension innovante de la cryptographie sur courbe elliptique et a expliqué en quoi ces nouveaux algorithmes seraient plus sûrs, plus rapides et moins gourmands en énergie, ce qui devait probablement préserver la batterie des ordinateurs portables²⁴². Le raisonnement repose sur le fait que l'algorithme RSA a été durant des années le protocole le plus utilisé pour la cryptographie à clé publique mais qu'il commençaient à montrer ses limites du fait de la montée en puissance des ordinateurs actuels. Pour compenser, il fallait augmenter la taille des clés RSA, ce qui rendaient les opérations plus lentes et plus gourmandes en ressources. Ainsi, en optant pour ces nouveaux algorithmes plus efficaces, ProtonMail se préparait aux menaces qui planeraient bientôt sur les clés RSA, tout en palliant des problèmes actuels, comme l'économie des batteries des ordinateurs portables. Notons encore que ce billet de blog ne mentionnait pas les détails « ennuyants » à propos du processus de normalisation des spécifications d'OpenPGP et il est relativement improbable que la plupart des utilisateurices connaissent et s'intéressent à ces enjeux.

240 <https://datatracker.ietf.org/doc/rfc6637/>.

241 <https://proton.me/blog/elliptic-curve-cryptography>.

242 Ce dernier argument me semble toutefois relativement anecdotique, car la création de clé, le chiffrement et déchiffrement d'email me semblent nettement moins exigeants en ressources que les vidéos conférences ou le streaming, par exemple. Toutefois, au vu de la catastrophe climatique en cours, toute économie, même modeste, me semble bonne à prendre.

J'ai choisi cet exemple pour illustrer comment « être à l'avant-garde du progrès » peut parfois consister à prendre une décision qui peut sembler anodine. Elle ne l'était pourtant pas. D'une part, il s'agit d'une illustration qui permet d'observer comment ProtonMail peut stratégiquement user de son influence sur l'écosystème d'OpenPGP pour accélérer la prise de cette décision qui semblait déjà implicitement faire consensus. D'autre part, cet exemple montre que l'innovation n'est pas toujours spectaculaire et « la marche du progrès » ne fonctionnent pas toujours pas bonds, d'innovations en innovations, mais au rythme de pratiques de soin souvent discrète et peu valorisées dans les discours et les encouragements à l'innovation. J'aimerais insister sur la modestie de ce changement en soulignant qu'il ne s'agissait que d'un changement de paramètres par défaut qui ne concernait du coup que les nouvelles clés créées. Ce n'était donc que les nouvelles utilisatrices qui étaient concernées. Le risque était donc limité.

Cet exemple nous enseigne surtout comment la préparation de l'avenir implique ce que j'appelle la maintenance par l'innovation et l'innovation par la maintenance. L'une appelle l'autre, irrémédiablement. On ne peut concevoir une innovation dans les technologies du numérique qui se passe de maintenance ; et pour maintenir cette innovation, il est souvent nécessaire d'innover, par exemple en changeant un paramètre par défaut ou en implémentant de nouveaux algorithmes²⁴³. De l'extérieur, OpenPGP ne changeait pas et les *managers*, PDG ou bailleurs de fonds ne verraient sans doute rien d'extraordinaire dans ce changement de paramètre somme toute relativement mineur si on le considère comme un cas isolé. Toutefois, ce sont ces petits pas qui donnent de la valeur à OpenPGP en tant que normes interopérable, fédérée et open source.

Du point de vue de la gouvernance, cet exemple souligne surtout la valeur de l'interopérabilité dans ce processus puisqu'il serait probablement moins coûteux et certainement plus facile d'implémenter sa propre architecture de chiffrement, comme le propose Tutanota ou Signal. Si ProtonMail utilise OpenPGP, c'est probablement parce

243 Voir le chapitre suivant (p. 326) pour une analyse de différents enjeux lors de l'implémentation de la cryptographie résistante aux ordinateurs quantiques dans OpenPGP.

qu'OpenPGP est une norme interopérable dont la fiabilité fait consensus – au moins au sein de certains groupes. Une fois de plus, on voit que le projet-sécurité n'est pas uniquement lié à des caractéristiques techniques mais dépend aussi d'autres facteurs, et notamment l'adoption de la norme par d'autres actrices.

Cet exemple du changement de paramètre par défaut est aussi intéressant car il montre que les pratiques d'innovation aussi peuvent être invisibilisées, surtout quand elles ne sont pas spectaculaires. D'ordinaire, ce sont les pratiques de soin qui sont décrites comme peu visibles et peu regardées, et cet exemple montre que dans certains cas, c'est aussi le cas de l'innovation.

Je viens de donner deux exemples où pratiques d'innovation et pratiques de soin sont intimement liées. Dans le premier cas, la maintenance de la première version de PGP a été une étape décisive pour faire en sorte que cette invention devienne innovation. Dans le deuxième cas, un changement de paramètres par défaut a renforcé une innovation (les courbes elliptiques) en leur donnant plus de poids dans OpenPGP. Dans les deux cas, l'innovation n'est pas révolutionnaire, ni spectaculaire, ni même extraordinaire. Au contraire, il s'agissait surtout de faire des petits pas pour que la technologie « avance dans la bonne direction ». Ces deux exemples ont montré que l'innovation, en tant qu'« invention qui s'est répandue » (Gaglio 2011), nécessite de telles pratiques de soin. J'aimerais maintenant explorer comment ces pratiques de soin ne sont pas uniquement orientées vers le passé – vers ce qui avait été mal conçu ou mal pensé – mais aussi vers l'avenir. Cela implique de créer et formaliser des connaissances sur la technologie dans le souci de rendre la technologie *soignable* à l'avenir.

8.2. OpenPGP en route vers l'avenir

Dans leur ouvrage « Le soin des choses », Jérôme Denis et David Pontille (2022) insistent sur l'ancrage dans le présent des pratiques qu'ils analysent. « L'infini recommencement » qu'ils décrivent est le fruit d'un travail ordinaire, non-événementiel et fondamentalement ancré dans le présent :

Le soin des choses est une pulsation quotidienne, une ritournelle qui ne s'arrête jamais. Il tisse à même le quotidien de la modernité des lignes de temps qui échappent aussi bien à la mobilisation aveuglante de la flèche du progrès qu'à celle de la catastrophe annoncée. La maintenance déplie le présent et l'épaissit d'une durée toujours en train de se faire. Elle ouvre les possibles, non par la rupture ou la projection dans le futur, mais dans l'invention pratique de la continuité. (2022, 352)

Un détail de cette citation m'a fait prendre conscience d'une particularité du soin porté à une technologie du numérique : la nécessaire et continue projection dans l'avenir des codeuses et développeuses d'OpenPGP. Denis et Pontille (et bien d'autres) ancrent la maintenance des choses matérielles dans un présentisme où la chose maintenue fait l'objet de soin dans « l'invention pratique de la continuité » sans que cela passe nécessairement par des projections dans l'avenir. Or, ma recherche parmi des équipes de développement m'a fait comprendre à quel point cette projection dans l'avenir était centrale dans le travail de soin de la sécurité informatique. Toutes les développeuses interviewées m'ont fait part de différents projets-sécurité qu'elles nourrissaient à long terme pour OpenPGP. Le développement de fonctionnalités de confidentialité persistante et l'ajout de primitives cryptographiques résistantes aux ordinateurs quantiques ne sont que deux exemples qui sont respectivement abordé dans la suite de ce chapitre et le chapitre suivant. Ce travail de soin va donc plus loin qu'inventer des solutions pour garantir la continuité de la technologie. Il s'agit d'anticiper ce qui va arriver, de préparer la technologie au quotidien d'un avenir qui n'est jamais acquis et toujours flou. Prendre soin d'OpenPGP, c'est aussi chercher à assurer la continuité de cette technologie dans l'avenir.

Jusqu'à présent, j'ai montré – sans grande innovation – comment l'innovation à elle seule n'était pas suffisante pour fabriquer les technologies de l'avenir. En plus des nouvelles idées, des nouveaux outils et des nouvelles possibilités de ces outils, il s'agit aussi de bâtir un écosystème (en langage émique) ou une infrastructure (en langage STS) qui soutiendra ces nouvelles technologies, ainsi que de constituer et consolider des communautés qui utilisent, soignent et étudient ces innovations. Bien que l'avenir soit par définition incertain, sa préparation génère et mobilise de nombreuses connaissances, que je propose d'examiner dans la prochaine section. J'aime-

rais ici présenter deux exemples qui permettent de comprendre comment le soin des technologies numérique est prospectif, c'est-à-dire orienté vers le futur, et pour le futur.

Le premier est sans doute le plus surprenant d'un point de vue des études sur la réparation et la maintenance puisque je propose de considérer le travail de documentation comme une pratique de maintenance qui génère elle-même de la maintenance. En effet, l'effort de documentation consiste notamment à accompagner le code vers ces futures transformations en faisant en sorte qu'il soit le plus résilient possible, c'est-à-dire compréhensible, testable, et modifiable pour les codeuses qui en prendront soin à l'avenir. Le deuxième exemple concerne une stratégie d'anticipation consistant à développer des connaissances théoriques et empiriques grâce à des prototypes. L'exemple des tâtonnements en direction de la confidentialité persistante permet de comprendre les contraintes qui pèsent sur l'implémentation d'une fonctionnalité qui est en train de devenir une norme en matière de messagerie sécurisée et qui ne semble cependant pas compatible avec le fonctionnement actuel d'OpenPGP.

8.2.1. Documenter le code pour préparer l'avenir

Le 16 décembre 2020, l'équipe de Sequoia a organisé une fête (en ligne, à cause de la pandémie mondiale de coronavirus) pour célébrer la publication de la version 1.0 de sa librairie. Mon écran d'ordinateur s'est parsemé de rectangles présentant le portrait souriant des nombreuses personnes « présentes » et connectées à cet événement. Une grippe et un féroce mal de crâne m'ont tenu écarté de l'esprit festif qui se diffusait à travers les pixels de mon écran mais j'ai quand même tenu à participer quelques heures à cette fête qui était l'aboutissement de plus de trois années de travail. Un événement majeur pour l'équipe, mais aussi pour la communauté OpenPGP en général qui comptait désormais une nouvelle librairie écrite en RUST. J'étais d'autant plus heureux de partager la joie de l'équipe de développement que j'étais conscient que cette publication de la v1.0 était attendue depuis près d'un an et que l'équipe avait fait beaucoup d'efforts pour y parvenir.

En quoi ont donc consisté ces efforts ? Outre le grand travail de codage d'un programme informatique entier basé sur un protocole âgé et complexe, et la traversée d'une période de confinement et de pandémie, l'équipe de Sequoia a été confrontée à un autre défi : celui de la documentation du code. Comme en atteste une publication sur Mastodon datant du Noël précédent (illustration 28), tout le monde pensait que c'était une mince affaire :

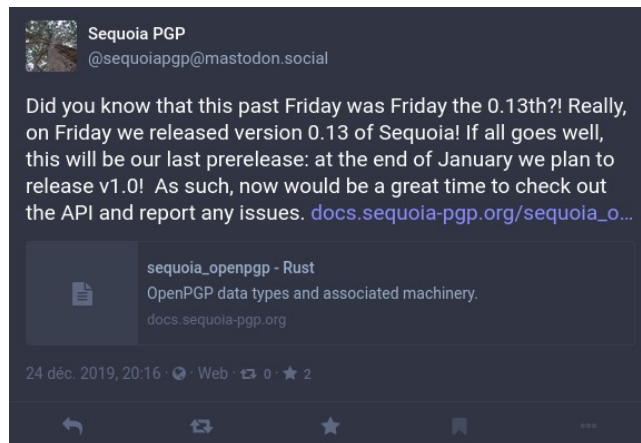


Illustration 28: Capture d'écran d'une publication de Sequoia sur Mastodon.

Source : <https://mastodon.social/@sequoiapgp/103364362621954545>

Durant des mois, j'ai cru – et l'équipe de Sequoia le croyait aussi – que cette publication était « imminente ». Lors d'entretiens que j'ai menés en juin 2020 avec plusieurs membres de l'équipe, j'avais compris que le code était prêt, qu'il ne manquait « que quelques détails » dans la documentation qui l'accompagnait pour pouvoir enfin publier la v1.0. Mais cette publication était sans cesse repoussée, semaine après semaine, mois après mois. Que se passait-il dans cette équipe de développement qui légitimait le report durant des mois de cette première publication officielle ? Qu'est-ce qui prenait autant de temps à cette équipe ? Le billet de blog qui a accompagné la publication de la v1.0 donne au moins trois éléments de réponse. Le premier concerne la raison fondamentale : améliorer la documentation prend du temps, nécessairement.

En réalité, nous avons presque publié la version 1.0 il y a environ un an. Toutes les fonctionnalités que nous avons prévues pour la version 1.0 étaient implé-

mentées, nous avons une bonne couverture de test, et nous avons même quelques utilisatrices. Mais nous avons décidé d'attendre. Nous avons décidé d'attendre non pas parce que nous avons pensé à une autre fonctionnalité, ou parce que nous avons pris conscience d'un bogue important, mais parce que nous avons décidé de prendre le temps d'améliorer notre documentation.²⁴⁴

Quelques lignes plus bas, un deuxième élément est mentionné : cette documentation a permis de repérer de nouveaux bogues et d'améliorer largement le code dans son ensemble :

Deuxièmement, au cours du processus de documentation de notre API publique et de rédaction d'exemples, nous avons découvert de nombreux désagréments mineurs, quelques incohérences et plus que quelques bogues. Comme nous n'avions pas encore adopté une API stable, nous pouvions les corriger et nous l'avons fait.

Et finalement, un troisième point est expliqué. Il concerne le fait qu'à mesure que ce travail avançait, des gens ont commencé à utiliser Sequoia, et leurs commentaires sont alors entrés dans la boucle du travail de développement, générant de nouvelles pratiques de soin.

Enfin, le taux de changement de l'API ayant diminué, davantage de projets étaient prêts à essayer Sequoia. Ils ont apporté des commentaires supplémentaires utiles, que nous avons intégrés.

Documenter, réparer, ajouter, voici ce qui a occupé l'équipe de Sequoia durant la préparation de cette publication. Présentés de la sorte, ces éléments semblent relativement anodins. Pourtant, ils racontent une histoire importante de la vie des technologies informatiques : ces dernières ne se réduisent pas à produire du code, il faut aussi créer la connaissance qui accompagne ce code, la diffuser, et l'améliorer, encore et toujours. Ce que nous apprend cet effort de documentation concerne la ferme volonté d'ancrer ce code dans la durée, et la documentation est essentielle dans cette tâche.

Voyons plus précisément ce que « documenter » signifie dans ce contexte. Dans le chapitre 5 (« Hériter, faire vieillir et transmettre le code », p. 186), j'ai décrit comment

²⁴⁴ Cette citation ainsi que les deux suivantes sont tirées du billet de blog de Sequoia : <https://sequoia-pgp.org/blog/2020/12/16/202012-1.0/> (ma traduction).

était construit le code et j'ai parlé des commentaires dans le code lui-même (pour commenter une ligne, une fonction ou un module entier). Ces commentaires consistent en une partie de la documentation dont parle Neal Walfield dans ce billet de blog. En tâchant de décrire ce que le code faisait, les codeuses ont commencé à formaliser un savoir à propos de l'utilisation de ce code qui s'est avéré beaucoup plus riche et complexe que ce à quoi elles s'étaient attendues. Il y avait des connaissances purement techniques qu'il fallait expliquer : telle ligne de code fait cela dans tel but et est activée dans telle situation ; telle fonction a tel rôle ; tel segment du code a été écrit pour telles raisons et les réglages peuvent être ajuster de telles et telles façons ; tel patch a été ajouté pour telle raison. Ces commentaires sont inscrits directement dans le code source et permettent de comprendre comment le code est construit. Cette documentation au sein même du code est primordiale pour la maintenance du code : sans cette documentation, il devient en effet très dur de lire, comprendre et modifier le code. Sachant qu'il est beaucoup plus difficile – et plus fréquent – de lire le code que d'autres personnes ont écrit que d'écrire le sien, on comprend pourquoi un code bien documenté est plus facile à maintenir, modifier et corriger. Ainsi, je propose de considérer ce travail de documentation au sein du code comme une activité de soin tournée vers l'avenir qui participe pleinement à la prolonger la durée de la sécurité que cherche à offrir le code.

Mais ce n'est pas tout : Sequoia a aussi publié des instructions sur la manière d'utiliser la librairie Sequoia. Cette documentation, rédigée sous forme de texte autonome en anglais, est une sorte de mode d'emploi de la librairie, avec des explications techniques, des exemples pratiques et des conseils²⁴⁵. Cette documentation est adressée directement aux utilisatrices de Sequoia, qui ne sont pas des utilisatrices finales comme moi, mais de développeuses qui veulent utiliser la librairie dans leur propre programme. Pour des personnes avec mon niveau de connaissances techniques et ma maîtrise du jargon – toutes deux médiocres malgré les cinq dernières années passées auprès de développeuses –, la lecture de ces nombreuses

245 https://docs.sequoia-pgp.org/sequoia_openpgp/.

pages est un exercice globalement assez barbant et démoralisant. En effet, j'avoue en toute honnêteté ne pas y comprendre grand-chose. Pourtant, ces pages regorgent de conseils utiles et pratiques pour les personnes qui en maîtrisent le langage : tout ce que la librairie permet de faire y est décrit, et des exemples pédagogiques sont proposés. L'importance d'une bonne documentation est explicitée par cette citation tirée d'une conversation informelle sur Signal avec une codeuse de Sequoia :

La documentation pour les utilisatrices doit décrire la sémantique des fonctions et, idéalement, fournir des exemples réalistes qui peuvent être directement copiés et collés

De mauvais exemples, par exemple, des exemples avec des raccourcis, mènent à la propagation de ces raccourcis

Écrire les exemples signifie penser à comment les utilisatrices vont véritablement utiliser les fonctions

Et cela mène parfois à découvrir que l'API [Application Programming Interface] de la fonction n'est pas optimale

Elle encourage peut-être les utilisatrices à l'utiliser de la mauvaise manière

Ou plutôt d'une manière dangereuse

(Codeuse de Sequoia, discussion informelle sur Signal, 24 janvier 2023)

Il s'agit non seulement de transmettre les informations utiles pour savoir utiliser cette librairie, mais également de s'assurer que celle-ci n'est pas dangereuse ou – comme aiment le dire les codeuses de Sequoia – qu'elle soit difficile à mal utiliser [« *hard to misuse* »].

Écrire cette documentation a donc été un catalyseur de création de connaissances à propos de la librairie que les codeuses avaient pourtant développer durant des années. Prenant le rôle de leur propre public récuratif (Kelty 2008), les codeuses ont appris à manier les outils qu'ils et elles avaient conçus, à explorer tout ce qu'ils permettaient de faire de manière voulue et non-voulue, et de proposer des ajustements. Étant donné que le code n'avait pas encore été publié officiellement, cela signifiait que l'équipe pouvait encore modifier des éléments de fond, notamment en changeant l'API et en ajoutant des fonctionnalités. Le fruit de cette préparation qui a duré quasiment une année est la formalisation de connaissances riches et denses sur

le fonctionnement de la librairie. Ces connaissances ont permis aux codeuses d'améliorer leur produit en identifiant et réparant certains problèmes tout en fournissant une documentation très détaillée aux futures utilisatrices. Cet exemple souligne l'importance du travail de documentation qui est un travail de préparation. En effet, une bonne documentation est essentielle à une bonne compréhension de la technologie, et cette compréhension est à son tour essentielle la maintenance future. Cette documentation est donc un outil offert aux générations futures pour qu'elles puissent mieux comprendre Sequoia et donc mieux en prendre soin. De manière peut-être plus anecdotique, cet exemple montre aussi comment le développement d'une librairie cryptographique est éloignée de la pureté mathématique avec laquelle on imagine parfois la science du chiffrement.

8.2.2. Prototypes et tâtonnements

Le deuxième exemple que j'aimerais donner présente une autre façon de préparer l'avenir, à savoir l'élaboration de prototypes pour développer des connaissances pratiques et empiriques à propos d'un défaut connu depuis longtemps dans OpenPGP. Ce vieux problème concerne l'usage d'une même clé privée sur le (très) long terme. En effet, la conception d'OpenPGP permet, voire encourage avec le Web of Trust, l'usage d'une même clé privée principale durant des années, voire des décennies. Cette durée est un problème car si cette clé est découverte, alors ce sont toutes les informations chiffrées avec cette clé qui deviennent vulnérables. Celles chiffrées par le passé, bien entendu, mais aussi toutes celles qui le seront à l'avenir. En somme, c'est à peu près la même chose qui se passe quand le pauvre Johnny utilise le même mot de passe pour tous ces comptes en ligne, et que ce mot de passe est découvert et utilisé sans que Johnny le sache et ne pense à le changer.

La norme *de facto* de la confidentialité persistante

Le concept de « confidentialité persistante » [*forward secrecy*] a été développé par les créateurs (tous des hommes) du protocole Off-The-Record (OTR) en 2004. Ce protocole, rappelons-le, a été développé pour servir d'alternative à OpenPGP, que les concepteurs d'OTR considéraient comme insécure justement à cause de ce problème (Borisov, Goldberg, et Brewer 2004). La confidentialité persistante est une invention astucieuse qui permet de s'assurer que même si une attaquante, appelons-la Ève, parvient à déchiffrer un message, elle ne parviendra pas à déchiffrer les messages futurs car ils seront chiffrés avec d'autres clés. Ainsi, Ève doit chercher une clé pour chacun des messages chiffrés, ce qui ralentit considérablement ses chances d'obtenir des informations pertinentes.

Comme l'ont montré Francesca Musiani et Ksenia Ermoshina (2019), le protocole Signal a développé ce mécanisme avec son protocole à « *double ratchet* ». Suite aux révélations de Snowden et le succès du protocole Signal, la confidentialité persistante est devenue une sorte de norme pratique *de facto* de ce que doit proposer impérativement une messagerie sécurisée dite « moderne ». Autrement dit, cette fonctionnalité est devenue une exigence pour les autres services concurrents sans qu'elle n'ait été formellement normalisée, ni que les utilisateurices sachent exactement à quoi elle fait référence. C'est plutôt une case à cocher dans une liste de fonctionnalité à avoir si l'on désire être compétitive.

En près de vingt ans, ce problème n'a toujours pas été réglé pour OpenPGP – et il ne le sera probablement pas de si tôt car l'infrastructure décentralisée des emails ne permet pas les mêmes choses que les applications de messagerie instantanées. Toutefois, cela n'empêche pas d'explorer différentes possibilités. Lors d'un entretien en juin 2021 avec une développeuse de ProtonMail, j'ai posé la question de quels étaient selon lui les enjeux importants pour l'avenir du protocole. Le premier point de sa réponse a été de mentionner le remplacement de tous les algorithmes insécures (comme SHA-1) par de meilleurs algorithmes, ce qui ne m'a pas étonné. Le deuxième point m'a en revanche plus étonné : il s'agissait justement de « résoudre le problème

de la confidentialité persistante ». Lorsque la développeuse m'a parlé de cette idée, j'ai en effet été étonné de savoir que ce projet était si important pour ProtonMail car ce problème n'avait pour le moment été abordé que de manière très floue dans les différents cercles que je fréquentais. Certes, il s'agissait d'une faiblesse connue et importante d'OpenPGP, mais elle était structurelle et y pallier nécessitait d'apporter des changements considérables dans le fonctionnement infrastructurel d'OpenPGP. Ainsi, hormis quelques présentations²⁴⁶ et discussions informelles parmi les développeuses, rien de concret ne semblait être sur la table. Et voici que tout à coup, cette codeuse plaçait ce point en deuxième position dans la liste des enjeux à résoudre au sein du groupe de travail. Peut-être s'agissait-il d'une question de *timing*, que notre entretien s'était déroulé juste après une matinée où ce problème avait été évoqué à l'interne et qu'il était apparu si tôt dans notre conversation car mon interlocutrice l'avait en tête à ce moment. Dans tous les cas, l'élaboration de sa réponse indique clairement un souci à ce sujet. Je cite ci-dessous un large extrait d'entretien duquel j'ai supprimé une partie d'explications techniques ainsi qu'une question de relance de ma part.

Oui, c'est un problème assez difficile à résoudre. Je ne blâme pas tellement les spécifications du protocole de ne pas encore avoir trouvé de solution à ce problème. Nous [ProtonMail] avons des idées, nous avons des plans. À un certain point, nous voulons, au minimum, commencer par faire une itération automatique, pour que les clés aient une moins longue durée de vie, de sorte que si tu casses une clé, tu n'auras la possibilité de lire que quelques messages. Et puis, nous aimerions commencer à faire un re-chiffrement symétrique. Cela fonctionnerait ainsi : après que tu reçois des messages avec l'une de ces clés, tu les déchiffres et les re-chiffres de manière symétriques plutôt qu'asymétrique, de sorte que tu puisses après jeter la clé RSA par exemple. Et dans le cas où les ordinateurs quantiques sont davantage développés, ce serait bien de ne pas avoir une pile géante de millions de messages chiffrés avec RSA tranquillement installés dans nos bases de données, attendant d'être cassés. Donc nous avons des idées. Je veux dire, ce n'est pas une solution complète, mais ça devrait au moins être un pas dans la bonne direction. Mais c'est en quelque sorte hors du cadre de la *crypto refresh*²⁴⁷, n'est-ce pas ? Donc je pourrais aussi proposer quelque

246 Par exemple, la présentation « Moving forward: Forward Secrecy in OpenPGP » que Justus Winter a donnée en 2018: <https://www.youtube.com/watch?v=an6oYjikAPY>.

247 La révision actuelle du standard, débutée officiellement en mars 2021.

chose comme ça pour d'autres spécifications séparées, plus tard, mais je ne sais pas si d'autres personnes seront aussi enthousiastes à ce propos.

[...] Cela dit, évidemment, nous pourrions au moins commencer à faire des choses à l'interne sans suivre le protocole. Et le re-chiffrement symétrique ferait certainement partie de ce premier pas dans cette direction. [...] Nous voyons cela comme une extension d'OpenPGP et aimerions au final standardiser, mais évidemment, il n'y a aucune garantie. Donc ce ne serait pas très différent que de juste le faire quelque chose de différent d'OpenPGP. Mais comme nous maintenons OpenPGP.js et GolanPGP, nous l'ajouterions probablement dans ces deux bibliothèques, évidemment désactivé par défaut, parce que ce n'est pas standard, mais quand même, si des personnes voulaient les utiliser, elles le pourraient. [...] Et après, à un certain point, [si les personnes sont intéressées], tu peux simplement l'ajouter aux spécifications du protocole.

(Codeureuse de ProtonMail, entretien du 6 juin 2021)

Cet long extrait souligne d'un côté l'inquiétude face aux incertitudes de l'avenir et de l'autre la nécessaire expérimentation de solutions qui ne vont jamais de soi. L'extrait ci-dessus illustre en outre le fait que les solutions de l'avenir ne sont souvent que le fruit de tâtonnements à partir de l'état actuel de l'infrastructure, qui n'est jamais idéal (sauf éventuellement quand l'on part de zéro)²⁴⁸. Dans le cas d'OpenPGP, le fait que la confidentialité persistante ne soit pas implémentée est une faiblesse pour plusieurs personnes car il s'agit d'une norme *de facto* qui manque par rapport aux autres protocoles de chiffrement (Ermoshina et Musiani 2019). Pour maintenir OpenPGP à flot et en concurrence avec les autres services, tâcher de trouver des solutions pour implémenter de manière aussi simple que possible ce système est une façon supplémentaire de prolonger la vie d'OpenPGP. Dans ce cas, la force d'innovation ne repose pas dans la capacité à créer quelque chose d'entièrement neuf, mais dans la volonté de créer de nouvelles solutions pour maintenir une infrastructure déjà existante : l'innovation pour la maintenance. Plus précisément, l'innovation consiste en la recherche par tâtonnement de nouvelles solutions pour favoriser la « survie » d'une technologie vieillissante en réduisant la surface d'attaque. Cet exemple me paraît très évocateur des difficultés que pose l'innovation qui n'est pas *ex nihilo* mais *ab intra structuram*, de l'intérieur de la structure : il ne s'agit plus d'inventer un modèle abstrait et de le déposer dans le monde mais de bricoler sur le modèle existant pour

²⁴⁸ Dans ce cas, il ne faut généralement pas attendre très longtemps pour voir les premiers problèmes arriver.

l'amener au plus près de la projection de ce qui paraît à la fois réaliste et souhaitable : le projet-sécurité.

8.3. Le régime futuriste de la maintenance informatique

À la lumière des différents exemples parcourus durant ce chapitre, il paraît évident que le futur tient une place prépondérante dans le travail de soin d'OpenPGP. À côté des tâches de résolution de failles dont il a été question dans la deuxième partie, une grande partie des activités de soin concerne en effet des projections dans un futur plus ou moins proche où l'état actuel du protocole pourrait poser problème. De ce point de vue, et en dépit de l'enlisement infrastructurel hérité du passé, le travail de maintenance informatique que j'ai observé s'inscrit dans un régime de temporalité prospectif, tourné vers l'avenir, dans lequel l'anticipation et l'interprétation de l'avenir sont centrales dans le sens donné aux activités effectuées. Il ne s'agit plus seulement de garantir la sécurité du seul présent en éliminant tous les bogues connus et hérités du passé, mais de prolonger la durée de la confidentialité, de l'intégrité et de l'accessibilité des données dans l'avenir. Cela pose deux problèmes de taille : le premier est qu'on ne sait pas de quoi l'avenir sera fait et on doit pourtant s'y préparer ; le second est que l'avenir ne cessera jamais d'être à venir.

Quand j'écris que les activités de soin des choses informatiques que j'ai pu observer sur mon terrain s'inscrivent dans un modèle résolument futuriste, je ne parle certainement pas du même futur que si j'avais fait ma recherche auprès d'ingénieur·e·s en charge de développer de nouveaux logiciels (par exemple autour de la communication quantique). Pour les mainteneureuses d'OpenPGP qui nous intéressent ici, le futur n'est pas porteur de solutions telles que des ordinateurs à la fois plus puissants et plus rapides. Le modèle prospectif du soin des choses informatiques d'OpenPGP n'est pas celui de la révolution heureuse vers une vie plus simple et plus sûre. C'est une autre vision de l'avenir qui prédomine, dans laquelle l'avenir regorge de périls qu'il s'agit d'écarter : des algorithmes bientôt décrépis, de nouvelles failles qui émergeront, ou encore des ordinateurs quantiques qui pourront craquer toutes les clés

privées actuelles dans un temps supposé être relativement bref. Autrement dit, l'avenir est un terrain dangereux dans lequel il faut apprendre à s'orienter.

Dans le modèle futuriste du soin des choses informatiques, l'horizon vers lequel on tend est rarement celui d'une rupture, car préparer l'avenir ne doit s'opposer à la maintenance de la compatibilité avec les systèmes hérités. Les pratiques de soin que je décris sont prospectives en ce qu'elles ont pour but d'assurer la continuité de cet héritage dans l'avenir. L'exemple du changement d'algorithme par défaut pour la création de nouvelles clés décidé par ProtonMail en 2019 était une étape parmi d'autres pour tenter d'optimiser les ressources en utilisant des algorithmes plus performants, rapides et économes. Cependant, ProtonMail n'a pas cessé de produire des clés RSA et continuera de supporter cette fonctionnalité encore probablement longtemps. Ainsi, et contrairement aux discours *mainstream* sur l'innovation, cette forme d'innovation pour la maintenance se fait non pas dans la rupture mais dans la continuité. En d'autres termes, l'objectif des développeuses que j'ai côtoyées durant mon travail de recherche n'étaient pas d'apporter une solution miracle unique mais de faire en sorte que ce qu'elles ont fait par le passé et ce qu'elles font actuellement continuent d'exister à l'avenir. Il s'agit de maintenir la durée de la sécurité d'OpenPGP en faisant en sorte que le protocole et ses implémentations répondent aux exigences des utilisatrices de demain.

Il y a une double dynamique qui consiste d'un côté à produire de l'innovation pour la maintenance et de l'autre à produire de la maintenance pour l'innovation. L'exemple de la première année de PGP est parlant : l'innovation n'aurait pas tenu le coup si directement après l'étape cruciale de la publication de PGP v1.0 Zimmerman n'avait pas cherché à continuer d'améliorer son produit. De même, l'innovation est nécessaire à la maintenance informatique comme je l'ai montré avec l'exemple de recherche de solution pour implémenter la confidentialité persistante dans OpenPGP. Partant, la maintenance pour l'innovation nécessite de transformer le code aussi intensément que nécessaire pour le doter de tout ce dont il aura besoin pour continuer de fonctionner dans un contexte toujours changeant, y compris une

bonne documentation ; et l'innovation pour la maintenance vise à poursuivre cet effort de création non pas pour créer quelque chose de nouveau, mais de cultiver la continuité de la technologie. Dans les deux cas, il s'agit de tisser les liens de la continuité entre passé, présent et futur, ainsi que de se projeter dans l'avenir pour en repérer les dangers et pour proposer des solutions dont on ne peut pas savoir en avance leur efficacité.

Cela me fait penser à la manière dont certaines navigateurices tâchent de planifier leur route en mer. Lors de ma formation amateur de voile en Bretagne, deux écoles m'ont été présentées pour mener mon bateau d'un port à un autre sans le fracasser contre les surnois récifs des côtes bretonnes. L'approche optimiste consiste à tracer sur la carte la route que je veux faire en identifiant où je veux mener mon bateau : passer à l'Est du phare, prendre tel cap pendant 5 milles, puis repérer tel amer et faire cap jusqu'à voir l'entrée du port. Elle est optimiste car malgré sa précision, la route tracée sur la carte ne correspond pas toujours à ce qu'il est possible de faire en fonction de l'état actuel de la mer et du vent. L'autre approche, que je qualifie de pragmatique, consiste à repérer sur la carte tous les endroits où je ne veux pas aller parce que ce sont des zones dangereuses, tout en gardant une idée générale du cap à faire : je ne veux pas passer à l'ouest de telle balise, je ne veux pas franchir cette ligne de fond, je ne veux pas passer dans cette zone de courant avec la marée descendante. Je pense que hormis la compagnie de dauphins, ces deux approches se retrouvent dans les pratiques des développeuses que j'ai rencontrées : certaines implémentations ont suivi un *road map* précis pour que leur code soit aussi fidèle que possible avec, par exemple, la suite de test d'interopérabilité conçue par l'équipe de Sequoia²⁴⁹. Dans d'autres cas, il s'agit d'identifier quels sont les dangers que réserve l'avenir et de tâcher de faire en sorte que le code ou le protocole soient les moins vulnérables possibles. L'analogie est plaisante, mais un détail « déroutera » peut-être les lecteurices : les navigateurices ont des cartes pour s'orienter et la destination est généralement connue. Or, rien de tel n'existe avec OpenPGP. Aucune carte ne permet de s'orienter,

249 <https://tests.sequoia-pgp.org/>.

ni d'éviter les écueils. Toutefois, les projets-sécurité peuvent aider les développeuses à prioriser certains développements et faire ainsi office de cap à suivre, que ce soit sous forme de *wish list*, de *milestones* ou de mandat (par projet, par contrat). C'est pourquoi il est primordial de comprendre ce qui pose problème dans l'avenir. Dans le chapitre suivant, un des gros problèmes de l'avenir est justement questionné : le développement des ordinateurs quantiques.

Penser les fragilités à venir pour panser l'avenir

9. Équiper OpenPGP face aux ordinateurs quantiques

Il y a ici beaucoup de discussions sur la cryptographie où vous comprendrez que la cryptographie a des problèmes. La cryptographie a des énormes problèmes d'utilisation, des problèmes de performance, des écueils pour les développeuses, une complexité folle dans l'implémentation, des normes stupides, des millions de lignes de code non auditées, et puis tous ces problèmes sont combinés dans un grand amas de merde [*a grand unified clusterfuck*] appelé la sécurité de la couche de transport [*transport layer security* (TLS)]. Mais en fait, la situation est bien pire. À cause de ce qui se prépare, c'est-à-dire les ordinateurs quantiques.

(Daniel J. Bernstein, communication au 32^e CCC, 2015²⁵⁰)

Le dernier chapitre de cette thèse s'ouvre ainsi avec ce merveilleux exemple de discours d'insécurisation si typique du champ de la sécurité de l'information : « tout va mal, et ça ira bientôt encore pire. Écoutez-nous. Merci. » L'arrivée « imminente » – ou du moins que certaines personnes qualifient comme telle – des ordinateurs quantiques fait peser de lourdes inquiétudes quant à l'avenir de la cryptographie, et particulièrement la cryptographie à clé publique. Capable de faire certains calculs beaucoup plus rapidement que les ordinateurs actuels, ce nouveau type d'ordinateurs devraient permettre, à terme, de casser l'immense majorité des algorithmes de chiffrement asymétrique utilisés actuellement (DeNardis 2022). Cette possibilité est connue depuis 1994, c'est-à-dire depuis que Peter Shor (1997) a conçu un algorithme

250 Présentation du 27 décembre 2015 par Daniel J. Bernstein et Tanja Lange intitulée « *PQC Hacks. A gentle introduction to post-quantum cryptography* » : <https://media.ccc.de/v/32c3-7210-pqchacks>.

quantique pour factoriser des nombres entiers naturels. Mais les récents développements en informatique quantique semblent avoir sorti les ordinateurs quantiques d'un futur de science fiction pour les installer durablement dans notre monde bien réel, ne serait-ce qu'à travers des processus d'(in)sécurisation et en particulier des discours et l'instauration de normes *de facto* promouvant les algorithmes dits « post-quantiques ».

Plus concrètement, l'Institut européen des normes de télécommunications (ETSI) parle par exemple d'« une menace significative pour notre infrastructure d'information mondiale »²⁵¹. Dans une publication de 2015, l'ETSI résume la situation en affirmant que certains protocoles sont supposés être résistants aux ordinateurs quantiques, alors que d'autres sont connus pour être vulnérables. La nuance est de taille : certains sont bons à être jetés, alors que d'autres méritent d'être examinés plus avant sans qu'il n'y ait aucune garantie de sécurité. La publication de l'ETSI montre également le télescopage de trois niveaux potentiellement vulnérables :

Les mesures de sécurité qui sont connues pour être très vulnérables aux attaques quantiques et qui peuvent être facilement brisées par un ordinateur quantique comprennent les suivantes :

1. Tout système de cryptographie reposant sur les complexités mathématiques de la factorisation des nombres entiers et des logarithmes discrets. Cela inclut RSA, DSA, DH, ECDH, ECDSA et d'autres variantes de ces algorithmes de chiffrement. Il est important de souligner que la quasi-totalité de la cryptographie à clé publique utilisée aujourd'hui dans les produits et protocoles de sécurité utilise ces types de chiffrement.
2. Tout protocole de sécurité qui tire sa sécurité des algorithmes de chiffrement à clé publique susmentionnés.
3. Tout produit ou système de sécurité qui tire sa sécurité des protocoles susmentionnés.²⁵²

Ce sont donc les niveaux 2 et 3 qui concernent la communauté OpenPGP, qui n'est pas en charge de la conception des primitives cryptographiques mais de la conception du protocole et de ses implémentations : le protocole OpenPGP lui-même tire sa sécurité des algorithmes de chiffrement asymétrique comme RSA ; et ProtonMail,

251 <https://www.etsi.org/technologies/quantum-safe-cryptography>, ma traduction.

252 <https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf>, ma traduction.

Thunderbird, GnuPG ou encore Sequoia tirent leur sécurité d'OpenPGP. Ce chapitre vise à comprendre comment la communauté OpenPGP considère cette menace et comment elle se prépare à cette arrivée probable bien qu'incertaine des ordinateurs quantiques. Je commence par expliquer en quoi les ordinateurs quantiques représentent un problème, puis aborde trois exemples qui montrent comment la communauté OpenPGP se saisit de ce problème.

9.1. Une bombe à retardement hors du commun ?

Ainsi, depuis quelques années, une nouvelle bête féroce a rejoint le bestiaire des innovations informatiques qui procurent la chair de poule : l'ordinateur quantique. Ces ordinateurs sont effrayants – du moins sont-ils dépeints comme tels. L'illustration 29 ci-dessous montre quelques représentations de ces ordinateurs quantiques, qui paraissent parfois directement sortie de la science fiction. Ces ordinateurs, qui d'ailleurs ont besoin d'un ordinateur « normal » pour fonctionner, sont basés sur les principes de la mécanique quantique et traitent l'information grâce à des concepts qui sont très difficiles à se représenter, comme la superposition quantique. Le champ de recherche en question s'appelle communément la « cryptographie post-quantique » (souvent abrégé PQC en anglais)²⁵³.

Au lieu des *bits* binaires (donc des 0 et 1) qui étaient déjà intimidants (rappelons-nous des représentations stéréotypées de ce code binaire, cf. illustration 13 p. 117), les ordinateurs quantiques utilisent des *qubits* (pour *quantum bits*), qui sont des unités de traitement qui peuvent exister dans un état superposé, représentant à la fois un 0 et un 1 simultanément, ainsi que dans un état d'interférence et d'entrelacement

²⁵³ Un-e cryptographe que j'ai interviewé-e m'a rendu attentif à distinguer deux champs de recherches qui sont souvent confondus : d'un côté, il y a la cryptographie résistante aux ordinateurs quantiques et de l'autre, la communication quantique, ou la cryptographie avec des ordinateurs quantiques. Cette dernière est encore au stade expérimental, alors que la première était à un stade déjà avancé. Dans ce qui suit, j'ai gardé l'expression « post-quantique » pour parler de la cryptographie résistante aux ordinateurs quantiques. Notons toutefois que dans le cas d'OpenPGP, la cryptographie sur ordinateur quantique n'est pas encore à l'agenda, alors que la cryptographie résistante aux ordinateurs quantiques a déjà commencé à occuper les ingénieur-e-s. Ainsi, dans ce chapitre, seule la cryptographie résistante aux ordinateurs quantiques est analysée, car elle porte une menace directe (quoique hypothétique) sur OpenPGP.

quantiques. Grâce à ces *qubits*, les ordinateurs quantiques peuvent théoriquement réaliser certaines opérations de manière beaucoup plus rapide que les ordinateurs classiques. Par exemple, alors que la factorisation est un problème difficile à réaliser avec nos ordinateurs actuels, les ordinateurs quantiques devraient pouvoir réaliser ces tâches en relativement peu de temps, même s'il est encore très difficile de savoir exactement combien de temps sera nécessaire pour déchiffrer par exemple des clés RSA de 2048 *bits*. Cette capacité théorique en font une menace directe pour la cryptographie à clé publique largement utilisée sur Internet (DeNardis 2022).

On ne peut que très difficilement se représenter ce que sont ces *qubits*, ce qu'ils font, et ce dont ils sont capables, et il est d'ailleurs très intéressant de constater qu'un imaginaire stéréotypé n'a pas encore sa place dans les moteurs de recherche. L'illustration 29 ci-dessous présente des résultats relativement hétérogènes, allant de photographie d'ordinateurs quantiques à des schémas explicatifs, en passant par le fameux chat de Schrödinger.

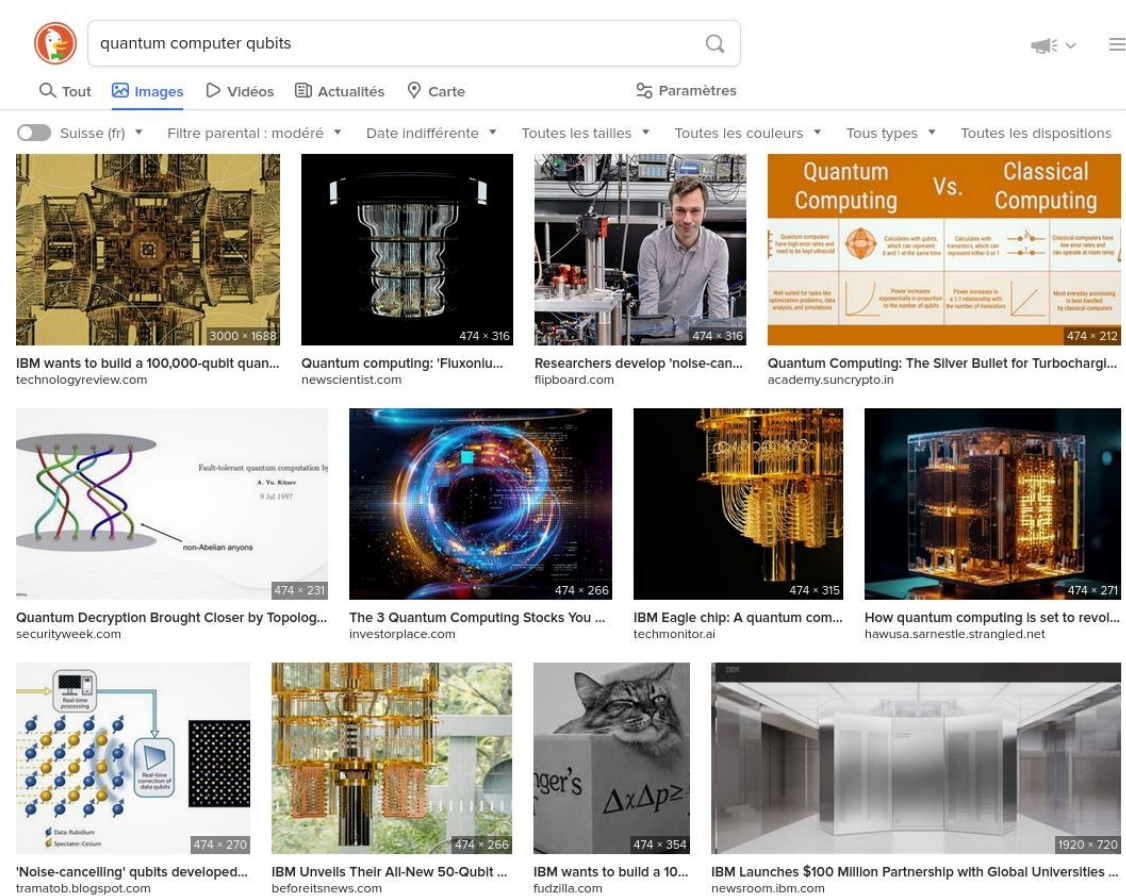


Illustration 29: Résultat de recherche avec les mots clés "quantum computer qubits". On distingue à la fois des schémas explicatifs, des photographies d'ordinateurs quantiques (dans un jeu de lumière fantaisiste) et même le fameux chat de Schrödinger. Source : le 27 mai 2023, <https://duckduckgo.com/>.

9.1.1. La catastrophe annoncée de la cryptographie post-quantique

Les discours à son sujet sont donc formels : la cryptographie post-quantique représente un véritable danger, mais quel est-il ? Il est en cela intéressant de creuser davantage ces discours pour comprendre comment ce danger est présenté. En 2015, l'IETF publiait l'affirmation suivante :

L'informatique quantique est une technologie qui n'a pas encore fait ses preuves, et il se peut qu'elle ne soit pas opérationnelle avant des décennies, mais comme elle représente une menace existentielle pour la cryptographie, nous devons commencer à nous préparer dès maintenant à la possibilité qu'un jour l'informatique quantique devienne une réalité.²⁵⁴

Cette affirmation est intéressante en ce qu'elle montre deux nuances presque paradoxales : d'une part, il ressort de la première partie que les ordinateurs quantiques

²⁵⁴ <https://www.ietf.org/blog/internet-security-vs-quantum-computing/>, ma traduction.

sont encore à un stade de développement relativement peu avancé. D'autre part, ils font peser sur la cryptographie une menace « existentielle ». La probabilité d'avoir des ordinateurs quantiques fonctionnels capables de briser des algorithmes cryptographiques était à l'époque encore considérée comme basse et peu imminente, mais l'impact de cette probabilité était jugé très critique. Aujourd'hui, près de dix ans plus tard, cette menace est devenue plus prégnante et les gouvernements ont commencé à prendre certaines mesures à la fois offensives et défensives. Au niveau offensif, plusieurs gouvernements investissent beaucoup dans la recherche dans ce domaine et la course à la « suprématie quantique » a d'ores et déjà commencé. La suprématie quantique est un terme critiqué même par la personne qui l'a formulé, John Preskill²⁵⁵. Il fait référence à la capacité d'un ordinateur quantique de résoudre certains problèmes de manière plus rapide ou plus efficace que les ordinateurs classiques les plus avancés. En 2019, la NASA et Google ont conjointement déclaré avoir atteint la suprématie quantique en réalisant en quelques secondes des opérations qui auraient duré des années²⁵⁶. Même si cette affirmation a tout de suite fait l'objet d'une controverse (DeNardis 2022), un point semble faire consensus : Google et la NASA ont « démontré qu'il est désormais possible de construire une machine quantique suffisamment grande et précise pour résoudre un problème que nous ne pouvions pas résoudre auparavant »²⁵⁷.

De plus, il semble que plusieurs gouvernements aient déjà commencé à collecter des données chiffrées pour pouvoir un jour les déchiffrer grâce aux ordinateurs quantiques²⁵⁸. Laura DeNardis l'a exprimé clairement :

255 <https://www.quantamagazine.org/john-preskill-explains-quantum-supremacy-20191002/>. John Preskill affirme d'ailleurs dans cet article que le terme était sans doute mal choisi. Toutefois, il est désormais adopté assez largement et il semble trop tard pour revenir en arrière.

256 <https://www.nasa.gov/feature/ames/quantum-supremacy>.

257 <https://www.quantamagazine.org/john-preskill-explains-quantum-supremacy-20191002/>, ma traduction.

258 Il est intéressant de constater qu'une autre menace plane sur nos vies humaines : la catastrophe climatique et la sixième extinction de masse. La cryptographie étant très gourmande en ressources informatiques, elle ne fait que renforcer la consommation croissante d'énergie électrique. Toutefois, cette menace ne semble pas être une préoccupation centrale pour les développeuses qui sont plus préoccupées de trouver des solutions plus fiables et rapides (même si, indi-

Une préoccupation claire en matière de sécurité nationale et de sécurité humaine est que les données et les communications chiffrées qui sont piratées, interceptées et stockées aujourd'hui - alors qu'elles sont indéchiffrables aujourd'hui - pourraient être déchiffrées à l'avenir et devraient donc être considérées comme compromises dès l'arrivée d'un ordinateur quantique suffisamment puissant. En d'autres termes, la crainte est que toutes les données collectées aujourd'hui par les gouvernements, les groupes terroristes ou les intermédiaires d'information commerciale puissent être déchiffrées rétrospectivement. (2022, 10, ma traduction)

Cette menace est d'autant plus lourde qu'une fois de plus, de nombreuses incertitudes existent. On ne sait pas quand cette technologie quantique sera suffisamment maîtrisée pour représenter un véritable danger pour les protocoles de chiffrement actuels et on ne sait pas quand le public sera mis au courant. Il est d'ailleurs fort probable que les gouvernements tâcheront de garder le plus longtemps possible secrètes leurs véritables avancées en la matière, comme le souligne Laura DeNardis (2022).

Parallèlement, plusieurs mesures défensives sont recherchées et la plus importante d'entre elles consiste à développer de nouveaux algorithmes résistants aux ordinateurs quantiques. En s'appuyant sur la théorie quantique, on peut savoir avec une relativement grande précision quelles primitives cryptographiques seront les plus rapidement vulnérables aux ordinateurs quantique, et à quels critères doivent répondre les nouveaux algorithmes de chiffrement pour être résistants aux ordinateurs quantiques. Par exemple, un concours pour choisir la prochaine norme de cryptographie résistante aux ordinateurs quantiques a été lancé par l'agence de standardisation états-unienne NIST en 2016. En été 2022, l'agence a annoncé la nomination des quatre algorithmes finalistes²⁵⁹. Toutefois, comme l'a par ailleurs affirmé très clairement Laura DeNardis, « la normalisation ne garantit pas la mise en œuvre ou l'utilisation » (DeNardis 2022, 20 ma traduction). En effet, contrairement au fonctionnement de l'IETF, l'agence NIST n'est pas basée sur une approche *bottom-up*, mais *top-down*. Cela signifie que les normes sont imposées d'en haut, sans l'avis des

rectement, la l'économie de temps fait aussi économiser de l'énergie). La sécurisation suit donc bien une logique de problématisation et d'intéressement, et non de pure rationalité.

259 <https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms>.

implémenteurices. Ainsi, même si de nouveaux protocoles semblent être de bons candidats, il faudra encore des années avant qu'ils ne soient utilisés quotidiennement.

Les États-Unis ne sont pas seuls dans la course et plusieurs initiatives cherchent à développer leurs connaissances et compétences en la matière. C'est notamment le cas de l'Allemagne : l'Office fédéral de la sécurité des technologies de l'information (*Bundesamt für Sicherheit in der Informationstechnik*, abrégé BSI) a lancé un appel à projet afin de créer un prototype d'implémentation d'un algorithme de chiffrement résistant aux ordinateurs quantiques dans Thunderbird. En visant l'une des applications open source les plus utilisées pour les emails, Thunderbird, le BSI s'appuyait sur une technologie open source, soutenue par une vaste communauté et une solide organisation (Mozilla), habituées à expérimenter²⁶⁰.

Ainsi, les choses avancent sur plusieurs plans : du côté de la conception des ordinateurs quantiques eux-mêmes, mais aussi du côté de la conception des algorithmes censés leur résister, et du côté des connaissances scientifiques à leur sujet. Il ne me semble pas important d'aller beaucoup plus loin dans les explications quant à l'actualité de ces protocoles et c'est pourquoi je propose de prendre du recul par rapport à ces développements en tâchant de comprendre en quoi ces ordinateurs représentent une menace d'un nouvel ordre.

9.1.2. Une menace si exceptionnelle ?

Commençons par répéter une évidence historique : les problèmes que pose l'arrivée des ordinateurs quantiques ne doivent pas surprendre, car l'histoire nous a en effet appris que même les meilleurs algorithmes finissent toujours par être cassés un jour :

« [...] l'expérience passée nous montre que chaque code réputé inviolable a, tôt ou tard, succombé à la sagacité des cryptanalystes. Le code de Vigenère fut appelé "le chiffre indéchiffrable", jusqu'à ce que Babbage en vienne à bout, et Enigma a été considéré comme invulnérable, avant que les Polonais n'en révèlent les faiblesses. Il est toujours très risqué de prédire les développements futurs de

260 Ce projet est analysé plus en détails dans la section la section 9.2.2 « Bricoler et tâtonner en attendant mieux » (p. 339).

n'importe quelle technologie, mais dans le cas des chiffrements, c'est particulièrement aventureux. Non seulement il nous faut deviner quelles découvertes interviendront dans le futur, mais aussi lesquelles ont déjà été faites, et sont restées cachées » (Singh 1999, 398).

La cryptographie numérique actuelle repose sur des problèmes qui ont longtemps eu la réputation d'être insolubles – ou du moins si difficiles que les résoudre demanderait des milliards d'années. Cela ressort très bien d'une discussion que Julian Assange, Jacob Appelbaum, Andy Müller-Maguhn et Jérémie Zimmermann, quatre figures majeures du hacking, ont eu en mars 2012 :

Jacob [Appelbaum]: [...] Il faut admettre qu'en matière de cryptographie, aucune violence ne résoudra jamais un problème mathématique. [...] Si nous pouvions résoudre tous ces problèmes, la situation serait différente et, bien entendu, le gouvernement serait en mesure de résoudre ces problèmes mathématiques si quelqu'un le pouvait.

Julian [Assange]: Mais il se trouve que c'est un fait de la réalité – comme le fait que l'on puisse fabriquer des bombes atomiques – qu'il y a des problèmes mathématiques que l'on peut créer et que même l'état le plus fort ne peut pas résoudre. (Assange et al. 2016, 59, ma traduction)

Ce que les quatre hackers ont oublié de mentionner dans leur conversation est que cette supériorité des mathématiques sur la violence physique ne dure qu'un temps. La taille des clés de chiffrement symétrique et asymétrique n'a pas attendu l'arrivée des ordinateurs quantiques pour être revue drastiquement à la hausse. Les clés de l'ancienne norme DES faisaient 56 bits, alors que les clés AES font désormais jusqu'à 256 bits, soit $1,6 \times 10^{60}$ fois plus grandes. Toutefois, bien qu'incertaine et lente, l'apparition des ordinateurs quantiques a fait se rapprocher considérablement l'horizon à partir duquel les anciens problèmes mathématiques supposés insolubles ne le seraient plus. La conséquence sera que certaines actrices, et probablement les gouvernements des grandes puissances avant les autres, seront capables de craquer presque tous les algorithmes cryptographiques à clé publique utilisés jusqu'à aujourd'hui (DeNardis 2022). Pour ce qui concerne OpenPGP, tous les algorithmes de chiffrement asymétrique (RSA, courbes elliptiques, etc.) utilisés dans le protocole OpenPGP seront vulnérables face aux ordinateurs quantiques. On comprend donc les

propos alarmistes de Daniel J. Bernstein cités en début de chapitre : la cryptographie a effectivement un problème et OpenPGP n’y échappe pas.

Bien entendu, face à cette sombre perspective, de nouveaux problèmes « insolubles » sont en train d’être imaginés pour résister aux ordinateurs quantiques. Toutefois, un important travail reste à faire avant que les développeuses d’OpenPGP ajoutent au moins un algorithme aux spécifications du protocole, que celui-ci deviennent un paramètre par défaut et que toutes les utilisatrices actuelles et à venir recréent une paire de clés pour s’adapter à cette nouvelle norme.

Finalement, avant de poursuivre avec une analyse de comment la communauté se prépare à l’arrivée des ordinateurs quantiques, j’aimerais encore rappeler un fait important. En effet, comme me l’a dit une organisatrice de *cryptoparty*, « la cryptographie n’a d’autre but que de te faire gagner du temps » (discussion informelle non enregistrée), dans le sens qu’elle ne vise pas à protéger les données *éternellement*, mais suffisamment longtemps pour que leur déchiffrement illégitime ne soit plus problématique. Dans le même esprit, Jean-Philippe Aumasson comparait la cryptographie post-quantique à une assurance :

Je présente souvent [la cryptographie post-quantum] comme une assurance. Comme vous achetez une assurance pour protéger votre maison contre les inondations ou – je ne sais pas – des attaques d’aliens. Même si vous savez que c’est très improbable que ça arrive vraiment, peut-être que vous vous sentez mieux si vous avez ce genre de protection. Donc c’est une question de gestion du risque.²⁶¹

Il est intéressant de souligner un certain paradoxe : d’un côté la cryptographie nous ferait « gagner », dans l’autre, les opérations cryptographiques ralentissent nos ordinateurs. C’est d’ailleurs d’autant plus le cas que les protocoles résistants aux ordinateurs quantiques semblent particulièrement lent. En effet, comme me l’a expliqué une codeuse d’OpenPGP, le problème des protocoles post-quantique connus à ce jour est qu’ils demanderont beaucoup plus de ressources pour le chiffrement et le déchiffrement. De plus, comme il ne sont pas encore utilisés dans la pratique, ils

261 <https://youtu.be/mrLLWW7rrls?t=1010>, ma traduction.

risquent très probablement d'être couplés à un algorithme connu mais non-post-quantiques, comme me l'expliquait une codeuse lors d'un entretien.

Tous ces algorithmes sont très récents, à une notable exception près : le schéma classique de McEliece, qui est presque aussi vieux que RSA et qui résiste à la technologie post-quantique. Je pense qu'il fonctionne bien - même si, je veux dire, je ne suis pas un expert - mais il est lent et les clés sont grandes. Les gens essaient donc de progresser à partir de ce modèle. Dans un avenir proche, nous allons probablement adopter une approche hybride. Tu prends un algorithme classique, comme les courbes elliptiques, et les nouveaux algorithmes, puis tu combines les entrées, avec la propriété intéressante que si le nouvel algorithme s'avère vulnérable, le schéma combiné sera toujours aussi fort que le schéma classique.

(Codeuse d'une implémentation d'OpenPGP, entretien du 12 janvier 2023)

Je trouve cette citation particulièrement intéressante car elle montre la méfiance des cryptographes à l'égard des nouveaux protocoles. La réponse proposée ici, cumuler deux algorithmes, est certes prudente d'un point de vue de la confidentialité, mais s'avère beaucoup plus exigeante en ressources, puisqu'il s'agit de chiffrer deux fois l'information en question.

Ce rapide parcours m'a permis de relever quelques uns des potentiels risques que font courir les ordinateurs quantiques sur la confidentialité des communications en ligne, et de soulever de nombreuses incertitudes. Sans m'attarder davantage sur des considérations générales quant à ces ordinateurs, j'aimerais maintenant me pencher sur la manière dont la communauté OpenPGP se saisit de ces problèmes et comment les ordinateurs quantiques la mettent au travail.

9.2. Des préparations sérieuses, dans l'incertitude et les tâtonnements

J'aimerais donner trois exemples qui permettent de comprendre comment la communauté OpenPGP s'est mise au travail pour préparer l'arrivée des ordinateurs quantiques. Ces exemples montrent à la fois des enjeux de gestion des risques quand il s'agit de s'accorder sur les priorités à suivre pour les futurs développements, mais aussi de trouver une réponse appropriée aux appels d'offre et de négocier des détails concrets quant à l'implémentation de certaines mesures. Bien qu'ils figurent dans

l'ordre chronologique, ces trois exemples ne racontent pas une histoire cohérente, mais ouvrent trois fenêtres sur un travail de longue haleine qui vient à peine de commencer et qui durera certainement des années.

9.2.1. S'accorder sur les priorités

Le premier exemple date de l'automne 2020, alors que le groupe de travail OpenPGP n'avait pas encore été officiellement reconstitué. Il s'agit d'un échange d'email relativement anodin sur la liste de diffusion du groupe de travail de l'IETF. Le contexte plus général de cet échange concerne une discussion à propos du changement d'un paramètre du certificat des clés OpenPGP.

[...]

- > Le fait que les clés post-quantiques soient très éloignées ne signifie pas
- > que c'est une bonne idée de proposer aujourd'hui des fonctionnalités qui
- > risquent de ne plus fonctionner lorsque les clés post-quantiques
- > arriveront.

Le soleil va exploser. Mais je fais encore des plans pour l'avenir.

Chaque mesure est une solution palliative [*a stopgap solution*]. Et je pense que si nous continuons à faire avancer l'écosystème, nous trouverons d'autres solutions qui, même si elles ne remplacent pas celle-ci, seront au moins complémentaires. Je pense donc qu'il vaut la peine de réfléchir à cette solution, même si elle n'est pas aussi générale que nous le souhaiterions.

(Neal H. Walfield, email à la liste de diffusion OpenPGP de l'IETF, 2020²⁶²)

Dans cet extrait, Neal Walfield répondait à son collègue Justus Winter qui s'opposait à l'idée d'implémenter un paramètre qui cesserait peut-être de fonctionner avec les nouvelles clés post-quantiques. Je souligne le « peut-être » de la phrase précédente : ne sachant pas à quoi allaient ressembler ces nouvelles clés post-quantiques, les deux développeurs envisageaient deux solutions assez éloignées l'une de l'autre : la première consistait à implémenter un paramètre qui sera probablement amené à être modifié avec l'apparition des nouvelles clés post-quantiques, et la seconde consistait à faire preuve de prudence quant à ce changement et d'attendre de pouvoir être sûr de sa pertinence avant d'agir. Les deux développeurs se rendaient compte que l'arrivée des ordinateurs quantiques allait se produire « demain ». Mais

262 Extrait d'un email envoyé sur la liste de diffusion du groupe de travail OpenPGP de l'IETF : https://mailarchive.ietf.org/arch/msg/openpgp/POPpzK_arXxkq661H6dR9G2T2Wk/.

des problèmes d'« hier » demandaient d'être résolus « aujourd'hui » et la solution envisagée ne serait certainement plus valable « demain » justement à cause des problèmes que « demain » posait. Dans cette situation, comment agir au mieux pour réparer le passé tout en préparant l'avenir ? Cette citation me semble particulièrement révélatrice des problèmes que pose l'avenir : fallait-il tout de même agir immédiatement, alors que l'incertitude ne pouvaient être résorbée immédiatement ?

Cette question n'est pas nouvelle et je suis convaincu que la majorité des développeuses se sont posées cette question à propos de nombreux bogues sans que ceux-ci soient nécessairement liés à l'arrivée des ordinateurs quantiques. L'incertitude fait partie intégrante du travail de codage et du soin des technologies du numérique, comme on l'a vu dans la deuxième partie de cette thèse. Toutefois, dans le cas présent, au vu des bouleversements probables que représente l'arrivée des ordinateurs quantiques et des algorithmes de chiffrement résistants à ces ordinateurs, la situation peut être qualifiée de limite. Comme souvent dans des cas limites, des moments d'« élicitation » surviennent. Ces moments, lors desquels « l'ensemble du possible relatif à une scène donnée est énuméré » (Dousset 2019, 29s), sont des moments épistémiques intéressants pour comprendre comment une communauté se saisit d'un problème nouveau.

Dans le cas présent, il est intéressant de constater que la réponse de Neal H. Walfield décrit une situation « normale » où chaque mesure est considérée comme une solution palliative [« *stopgap* »]. Selon sa vision, il s'agit de faire autant de petits pas que nécessaire dans une direction que l'on suppose être la « bonne ». Ces solutions ne sont souvent ni idéales, ni générales, mais elles permettent à l'écosystème de se maintenir à flot. Toutefois, quelque chose dans son propos me frappe : tacitement, il émet l'hypothèse qu'OpenPGP continuerait d'exister avec l'apparition des ordinateurs quantiques et que de nouveaux algorithmes seraient intégrés au protocole. Autrement dit, selon cet ingénieur, l'apparition des ordinateurs quantiques ne signerait pas la mort d'OpenPGP et au contraire ne représenterait qu'un nouveau problème contre lequel il faudrait prendre de nouvelles mesures palliatives en temps vous. Les

solutions à court terme qu'il proposait ne représentaient que des tâtonnements afin de dépasser l'incertitude sans pouvoir la résorber, comme l'ont été d'autres mesures tel le MDC dont il a été question plus tôt.

Ce que suggère les propos de Walfield, c'est la conscience des développeuses des compromis qu'elles doivent continuellement faire : accepter de bricoler des solutions que l'on sait ni générales, ni pérennes et encore moins miraculeuses, afin de tenter de pousser l'ensemble de l'écosystème dans la direction d'un certain projet-sécurité (« *to move forward* »). Autrement dit, c'est la continuité entre l'état passé et l'état futur souhaité qui est cultivée pour envisager l'avenir. Le projet-sécurité peut ici effectivement servir de carte pour la navigation, selon la métaphore qui a clôt le dernier chapitre, mais il ne permet pas de faire l'économie sur l'attention aux problèmes actuels qui, tels de surnois récifs, interdisent l'usage du pilote automatique.

9.2.2. Bricoler et tâtonner en attendant mieux

Près d'une année plus tard, un autre email²⁶³ est envoyé sur la liste de diffusion par Kai Engert, un développeur en charge l'implémentation native d'OpenPGP dans Thunderbird, le client email de la fondation Mozilla. Cet email communique un appel à projet lancé par l'Office fédéral de la sécurité des technologies de l'information (BSI). Cet appel avait pour but d'implémenter un support pour cryptographie post-quantum dans Thunderbird²⁶⁴. Voici un extrait de l'email envoyé par Kai Engert, en charge de l'implémentation native d'OpenPGP dans Thunderbird :

Comme je le comprends, ils ont l'intention de payer un mandataire pour un large ensemble de tâches pour implémenter la cryptographie post-quantique dans Thunderbird. Cela comprend le travail pour standardiser l'usage de la cryptographie post-quantique avec OpenPGP, y compris les implémentations RNP, Botan, GnuPG et Libgcrypt.

Il semble que le BSI a déjà fait des suggestions qu'il aimerait exiger l'usage [des protocoles post-quantiques] CRYSTALS-Kyber and -Dilithium.

Est-ce un choix raisonnable ?

Est-il judicieux de définir une restriction à ces méthodes à ce stade ?

263 https://mailarchive.ietf.org/arch/msg/openpgp/i2qnaxF0RNhcVnobj0C_HE211wk/.

264 <https://www.evergabe-online.de/tenderdetails.html?0&id=397181&cookieCheck>.

(Kai Engert, liste de diffusion du groupe de travail OpenPGP, 24 juin 2021)²⁶⁵

Deux éléments me paraissent particulièrement intéressants ici. Le premier concerne l'émergence d'un acteur inattendu – une agence gouvernementale qui plus est – à la table des négociations des spécifications d'OpenPGP. En suggérant deux protocoles en particuliers, le BSI court-circuite en quelques sortes le processus de négociation ordinaire de l'IETF basé sur une gouvernance transparente fondée sur le consensus approximatif, l'expertise technique et des codes fonctionnels. Par ailleurs, il donne un poids considérable à Thunderbird dans la course. Le deuxième élément concerne l'invitation de Kai Engert à réfléchir à la réponse à donner à cet appel à projet, et ce point sera traité dans la sous-section suivante.

Pour le moment, revenons à la question que pose l'intervention du BSI dans la négociation des spécifications d'OpenPGP : s'agit-il d'une ingérence dans le travail de l'IETF ? Ce point a été relevé dans plusieurs réponses : il semblait en effet trop tôt pour arrêter un choix de deux primitives cryptographiques post-quantique alors que la compétition de la NIST n'avait pas encore donné ses résultats. J'aimerais ici apporter deux éléments de réponses. Le premier consiste à rappeler l'un des mots de l'IETF : « Nous croyons dans le consensus approximatif et le code fonctionnel »²⁶⁶. Ce mot souligne l'un des principes de gouvernance de l'IETF qui n'est pas une organisation de normalisation qui impose aveuglément ses critères, mais qui promeut des critères qui ont fait leur preuve sur le terrain. Ainsi, OpenPGP était d'abord un programme informatique fonctionnel (bien qu'avec de nombreux problèmes) avant d'être une norme. L'expérimentation fait donc partie de l'ADN de l'IETF qui organise d'ailleurs des Hacktahons à cet égard, et une norme n'est en principe pas adoptée si il n'y a pas eu assez d'expérimentation et s'il n'y a pas au moins deux implémentations qui l'utilisent. Ainsi, du point de vue du BSI, qui utilise et soutient OpenPGP, il s'agit d'encourager l'expérimentation.

²⁶⁵ <https://mailarchive.ietf.org/arch/msg/openpgp/EHFAwFKz4YxW5yOE9oZRk7ld1Zc/>.

²⁶⁶ [We believe in rough consensus and running code] ; voir notamment <https://www.ietf.org/running-code/>.

Par ailleurs, comme le rappelle Engert dans un autre email, l'appel du BSI suggérait l'idée de « crypto-agility », l'agilité cryptographique²⁶⁷. Cette idée suggère une potentielle flexibilisation du processus de normalisation d'OpenPGP, inspirée par les récents développements dans la gestion du code informatique²⁶⁸. Serait-il envisageable de choisir un algorithme, le tester, puis décider de changer si l'expérience n'était pas concluante ? Cette suggestion paraît plutôt improbable dans l'écosystème d'OpenPGP qui évolue si lentement. Elle a toutefois trouvé un écho positif chez ProtonMail qui a répondu à l'email d'Engert en affirmant que l'entreprise avait un projet similaire et serait heureuse de collaborer à ce sujet :

[...] Pour ce que cela vaut, nous (ProtonMail) avons également prévu de créer une implémentation expérimentale de la cryptographie post-quantique dans OpenPGP, à l'automne – sans avoir l'intention de "créer une nouvelle norme", notre raisonnement était plutôt qu'une implémentation expérimentale pourrait nous apprendre quelque chose sur une normalisation potentielle à venir.

Ceci dit, si le groupe de travail est prêt pour ce travail d'ici là, nous sommes bien sûr heureux·euses d'y contribuer, et de collaborer en vue d'une potentielle normalisation tout en travaillant sur une implémentation expérimentale.

(Codeureuse de ProtonMail, liste de diffusion du groupe de travail OpenPGP, 28 juin 2021)²⁶⁹

Ainsi, plutôt que d'être un problème, il semble que l'appel à projet du BSI ait été une source de motivation et d'explicitation de différents projets au sein de la communauté OpenPGP. En fait, la cryptographie post-quantique a l'avantage de faire partie de ce qu'un·e des personnes interviewées appelaient des « *shiny things* », qui motivent certaines personnes, et souvent les détournent du travail fastidieux et relativement ennuyant de mettre à jour les propriétés moins nouvelles du protocole. C'est peut-être d'ailleurs dans ce sens qu'il faut lire la réponse d'un des présidents du groupe de travail au mail précédemment cité. Dans son message, Daniel Kahn Gillmor semble en effet vouloir jouer sur l'attrait de ces questions pour renforcer la cohésion du groupe de travail :

267 <https://mailarchive.ietf.org/arch/msg/openpgp/D0iodrIPQm98N7H3-BUOD92VMVU/>.

268 L'« agilité » est depuis quelques temps devenu le mot d'ordre de la gestion du code informatique (Gurses et Van Hoboken 2017; Poller et al. 2017).

269 <https://mailarchive.ietf.org/arch/msg/openpgp/ifm3dqQdyh8kOTZF7yONAsDZb3Q/>.

Si nous parvenons à mettre à jour le document "crypto refresh" [le brouillon de la nouvelle version du protocole à l'IETF] assez rapidement, ce serait un bon moyen de montrer que nous sommes prêts, en tant que communauté, à trouver un moyen d'intégrer les mécanismes de la cryptographie post-quantum à OpenPGP. Et [...] l'une des façons probables de faire serait d'avoir des combinaisons de clés multiples – de sorte à ne pas introduire un algorithme relativement nouveau qui rendrait le protocole plus fragile que les algorithmes asymétriques traditionnels. Cela nécessite une certaine nuance et une planification qui ne font pas partie du simple mandat de "rafraîchissement de la cryptographie" dont nous disposons à l'heure actuelle.

(Co-président du groupe de travail OpenPGP, liste de diffusion du groupe de travail, 24 juin 2021)²⁷⁰

J'aimerais maintenant passer à l'analyse du deuxième point mentionné plus haut, concernant la réponse à donner à cet appel à projet. Le fil de discussion cité ci-dessus a donné deux résultats : le premier a été d'explicitier les intentions des différentes implémentations, en l'occurrence celles de Thunderbird et celles de ProtonMail. Le deuxième concernait un début de réflexion sur la manière de procéder avec ces nouveaux protocoles post-quantiques. Dans le fil de discussion ci-dessus, l'idée a été émise d'avoir des doubles clés et donc de chiffrer deux fois la même information. Cette solution, temporaire et imparfaite (car exigeant le double de ressource informatique pour chiffrer deux fois l'information), était considérée comme suffisamment acceptable par le groupe de travail. Dans cette discussion déjà, différentes possibilités étaient étudiées, et les choses se sont précisées. Une année plus tard, le 5 juillet 2022, la NIST annonce les gagnants du concours lancé, parmi lesquels se trouvent CRYSTALS-Kyber comme protocole de chiffrement, et CRYSTALS-Dilithium comme protocole de signature numérique²⁷¹. Peu après cette annonce, le 21 juillet 2022, Falko Strenzke envoie un email sur la liste de diffusion d'OpenPGP avec deux propositions concrètes pour introduire ces protocoles dans OpenPGP. Je passe les détails techniques et vient à l'objectif de l'email :

Ce que nous voudrions apprendre de la communauté est de savoir s'il y a une préférence claire pour l'une des deux options. Savoir qu'il y aurait une claire préférence sur la question à ce stade nous aiderait, car nous pourrions potentiellement nous focaliser seulement sur l'option favorite et oublier la deuxième.

270 <https://mailarchive.ietf.org/arch/msg/openpgp/EHFAwFKz4YxW5yOE9oZRk7ld1Zc/>.

271 <https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms>.

(Falko Strenzke, liste de diffusion du groupe de travail, 21 juillet 2022)²⁷²

Une fois de plus, les choses sont encore très floues. Ce mail a reçu seulement deux réponses, l’une favorisant l’option A, l’autre l’option B. Ce qui m’importe ici, c’est l’argument de la deuxième réponse qui préfère l’option B :

Je sais que l’option A est plus compatible avec les spécifications actuelles, mais l’un des avantages que je vois avec l’option B est que nous pourrions nous débarrasser de certaines bizarreries de ECDH-over-Curve25519 en faveur de la norme X25519, par exemple. À l’avenir, si toutes les implémentations prennent en charge ces nouveaux algorithmes, une nouvelle implémentation pourrait se débarrasser de cette bizarrerie. Si nous intégrons l’ECDH actuel dans les algorithmes post-quantiques, nous resterons coincés avec ça pendant bien plus longtemps.

(Membre du groupe de travail OpenPGP, liste de diffusion du groupe de travail, 23 juillet 2022).²⁷³

Ici, c’est un nouveau cas de figure qui se présente : les nouveaux algorithmes post-quantum viendraient au secours des spécifications vieillissantes d’OpenPGP en aidant les « bizarreries » de l’ECDH à être mise au rebus. Cette suggestion est intéressante car elle pose frontalement la question de l’interopérabilité de la nouvelle norme post-quantique. Un des soucis, c’est justement que ces innovations ne soient pas aussi compatibles avec l’infrastructure actuelle, ce qui correspond à l’un des points importants soulignés par Laura DeNardis :

La réalisation de cette compatibilité rétrospective et de cette intégration nécessaires nécessite une coordination/collaboration entre les innovateurs émergents et les communautés d’experts déjà impliquées dans la conception et la coordination de l’architecture de base de l’internet, comme l’IETF. Toute approche de l’internet quantique qui n’est pas intégratif et rétrocompatible a peu de chances d’être mise en œuvre et utilisée. (DeNardis 2022, 19, ma traduction)

Dans le cas présent, c’est plutôt l’inverse qui se passe. En effet, la cryptographie post-quantique est décrite comme une opportunité de décharger l’avenir d’une partie d’un héritage dont on sait déjà qu’il ne sera pas souhaité. C’est un exemple intéressant où la maintenance future est explicitement en jeu. Ceci est crucial pour comprendre les différentes dynamiques de la maintenance de choses numériques. Jérôme Denis et David Pontille (2022) l’ont affirmé à plusieurs reprises : la maintenance est « l’art de faire durer les choses ». Ils se sont principalement occupés des

272 <https://mailarchive.ietf.org/arch/msg/openpgp/ll4oss6CiOVuKwnVPJ7G-mf3CC8/>.

273 https://mailarchive.ietf.org/arch/msg/openpgp/RjaVT-5p84wlJydn5I_9NQQolso/.

choses matérielles qui sont ontologiquement sensibles au passage du temps. Le temps use la matière (dégradation, déformation, salissure volontaire ou non, rouille, etc.). Le problème des choses numériques n'est pas le même. D'une part, ce n'est pas leur ontologie qui est en jeu, mais la pertinence des informations en lesquelles elles consistent. Maintenir OpenPGP, c'est tenter de maintenir aujourd'hui et pour demain la pertinence que cette technologie avait hier. D'autre part, pour maintenir cette pertinence, il faut irrémédiablement ajouter de nouvelles fonctionnalités. Ainsi, la maintenance des choses numériques implique un épaississement desdites choses. C'est comme si, en plus de nettoyer et maintenir les rouages et la tuyauterie, on maintenait un barrage hydroélectrique en ajoutant régulièrement une nouvelle couche par dessus le barrage. Très vite, nous aurions affaire à un monstre de béton, un dédale architectural d'une complexité folle. En tâchant d'anticiper et réduire ce dédale avant qu'il ne soit implémenté, l'équipe de ProtonMail veut faire le ménage avant que de potentielles galeries s'enlisent dans l'infrastructure. Dans la section suivante, je développe le principe de résilience prospective pour décrire justement cette volonté de produire une technologie aussi résiliente que possible *dans l'avenir*, et pas uniquement dans le présent.

Ces trois exemples montrent que malgré les incertitudes, et à l'instar du passé, l'avenir *agit* sur les technologies du numérique. Plus précisément, au nom de l'avenir, des décisions importantes sont débattues et prises (même « ne rien faire » ou « ne pas encore décider » sont déjà des choix). Ces décisions influencent grandement l'état actuel des technologies numériques avant même que l'on sache à quoi ressembleront vraiment les protocoles de chiffrement post-quantiques. Malgré une certaine concurrence et malgré le manque d'avancée sur d'autres dossiers, le groupe de travail OpenPGP affiche ici une claire volonté d'avancer ensemble – ce qui implique de définir un projet-sécurité commun. L'expérimentation d'implémentation de nouveaux protocoles de chiffrement est en ce sens un pilier central de la gouvernance de l'IETF, basée notamment sur l'idée de codes fonctionnels et interopérables.

9.3. La résilience prospective comme horizon

Comme je l'ai déjà suggéré à plusieurs reprises dans ce qui précède, un des défis les plus importants des développeuses interviewées est de s'adapter aux normes du passé tout en orientant leur produits vers l'avenir dans une recherche constante de fiabilité et de sécurité. En d'autre terme, il s'agit de maintenir dans la durée un haut niveau de sécurité et de fiabilité en dépit de l'héritage du passé et de l'incertitude de l'avenir. J'appelle ceci le principe de « résilience prospective » et l'utilise pour tâcher de comprendre l'entremêlement de différentes temporalités du soin de la sécurité informatique. Avec cette notion, je veux parler de la volonté de planifier la capacité d'une technologie numérique à faire preuve de résilience quand émergeront de nouveaux problèmes, quand bien même on ne peut que supposer la nature de ces problèmes.

La résilience prospective pour penser la sécurité

Le terme de résilience désigne originellement la résistance d'un métal au choc. Il existe de nombreuses interprétations de ce terme, dans différentes disciplines. En informatique, il désigne en général la capacité d'un système à se remettre d'une faille ou la capacité d'une organisation à assurer la continuité du fonctionnement de son parc informatique même en cas d'incident (panne, surcharge, piratage). Ces interpénétrations suivent la métaphore du choc dont on se remet « automatiquement » de par les propriétés matériels du métal (pour l'interprétation physique) ou du parc informatique (pour l'analogie des sciences informatiques). En psychologie, le terme résilience est aussi utilisé et désigne la capacité du sujet à se remettre d'un traumatisme. Toutefois, la résilience implique ici une véritable transformation : pour pouvoir survivre, la victime d'un accident traumatique doit devenir un être différent, survivant et non plus victime ; la transformation est cruciale et permet la guérison. En m'inspirant de ces approches, je définis la « résilience prospective » comme l'anticipation de ce qui est nécessaire pour permettre à une technolo-

gie numérique de se transformer afin de pallier une menace future que le présent ne permet pas d'identifier clairement.

Dans la maintenance informatique, la modélisation des cycles de vie des vulnérabilités ou de la gestion des bogues (cf. chapitre 5 « Hériter, faire vieillir et transmettre le code », p. 186) reproduisent l'idée d'une temporalité cyclique où l'objectif est d'être dans le même état qu'avant le choc. Or, le souci de l'avenir que j'ai présenté dans ce chapitre souligne une conscience aiguë d'une incertitude fondamentale dans la marche de l'avenir qui rend l'idée de cycle problématique. Des évolutions majeures (comme l'arrivée des ordinateurs quantiques) aussi bien que mineures (comme le choix d'un nouvel algorithme comme paramètre par défaut) interviennent dans les vies du code et des protocoles en le transformant constamment. L'historique des transformations du code sur GitLab découle d'une logique temporelle linéaire, ou du moins cumulative. En ce sens, la résilience prospective représente la nécessaire préparation pour tâcher de donner à la technologie la capacité à se remettre de futurs incidents.

9.3.1. Connaître pour être sûr-e ? La préparation de l'avenir

Ce que veulent éviter les codeuses, ce sont des catastrophes au sens où Lakoff et Collier (2015, 20, ma traduction) l'entendent, c'est-à-dire « des événements soudains et imprévisibles qui perturbent les systèmes essentiels à la vie économique et sociale ». Toutefois, contrairement aux risques couverts par l'article de Lakoff et Collier, les bogues et vulnérabilités qui affectent OpenPGP ne sont pas considérés comme un problème d'ordre public et les gouvernements ne sont pas intéressés par ce genre de vulnérabilités – sauf quand il s'agit de les acheter pour les utiliser à des fins offensives (Goerzen et Coleman 2022). Ainsi, les utilisatrices d'OpenPGP ne peuvent compter sur un gouvernement pour réagir en cas de problème et ce sont les développeuses qui portent toute la responsabilité de préparer le système à faire preuve de résilience lors des prochaines catastrophes. Il s'agit donc de questionner comment se déroule cette préparation et ce qu'elle implique.

Comme l'a écrit Frédéric Keck au sujet de la surveillance et la prédiction de crises sanitaires dues à des pandémies affectant oiseaux et humains, « [l']annonce d'une catastrophe à venir s'appuie sur la mémoire des catastrophes passées » (2010, 28). Même si le contexte d'étude est différent, ce chapitre, ainsi que le précédent, ont montré que la préparation de l'avenir est centrale dans les pratiques de soin de la sécurité informatique, et que cette préparation ne peut se faire qu'en s'appuyant sur l'expérience passée. Il s'agit de se projeter dans l'avenir afin de tâcher de deviner de quoi l'on aura sans doute besoin. Dans un projet de sécurité informatique, cela implique de comprendre quelles seront les prochaines menaces et les risques les plus probants, et, sur la base des connaissances actuelles issues des expériences passées, de déterminer les possibilités de développement, ainsi que les besoins pour réaliser ce travail.

Cela fait écho à l'intérêt relativement récent de l'anthropologie autour de la question de l'avenir. En effet, le futur est devenu une source majeure de questionnements anthropologiques (Valentine et Hassoun 2019). Andrew Lakoff (2017) a notamment décrit comment il y aura toujours plus de crises mondiales et comment, à chaque crise, on semble pris de cours. Une des raisons évoquées concerne le fait qu'en plus d'une menace relativement identifiée (par exemple, une crise sanitaire ou une catastrophe naturelle), il y a toujours d'autres facteurs géopolitiques, démographiques ou liés aux échanges internationaux de biens et services qui viennent s'ajouter à l'équation de base. Or, cette multitude de facteurs ne se laisse que difficilement anticiper. Dans les technologies de la sécurité de l'information, ces différents facteurs peuvent paraître infinis : menace sur la vie privée, sur la sécurité nationale, sur le commerce en ligne, sur la stabilité des parcs informatiques des entreprises et institutions, sans parler des défis pour assurer une information de qualité. Dans le cas particulier d'OpenPGP, les menaces sont à la fois externes (l'arrivée des ordinateurs quantiques, par exemple) et internes (une exigence de compatibilité avec les systèmes hérités du passé qui contraindrait l'évolution de la technologie, ou l'implosion de la communauté à force de ne pas arriver à des spécifications consensuelles).

En revanche, ce qui se laisse beaucoup plus aisément anticiper, ce sont les besoins des personnes qui seront en charge de prendre soin des technologies en question. En s'appuyant sur l'expérience bien connue de leur propre quotidien, les codeuses savent très bien ce qui leur pose le plus problème et ce qui facilite leur travail. Face à un code qu'il faut patcher, une bonne documentation, une explicitation des fonctions ou des *hacks* du code (notamment tous les patches qui ont été implémentés), ainsi que la présence d'outils pour faciliter la mise à jour du code (par exemple, des *unit test*), sont autant d'éléments qui rendent les technologies numériques plus maintenables. Ainsi, même si on ne peut être sûr de la nature des crises de demain, il est certain qu'une bonne documentation sera un précieux atout. Préparer l'avenir, c'est donc aussi passer du temps à expliquer ce qui a été fait, et ce qui continue d'être fait, pour faire fonctionner cette technologie.

9.3.2. Vers une nouvelle norme dans les attentes et les besoins des utilisatrices

La cryptographie post-quantique n'est certes pas le seul défi, ni peut-être le plus urgent, que les ingénieurs·es d'OpenPGP devront relever à l'avenir, mais il s'agit sans doute de l'un des problèmes les plus incertains. Comme le relevait un·e ingénieur·e interviewé·e et cité·e en début de ce chapitre, tous ces algorithmes sont très récents et nous ne savons que peu de choses à leur sujet. Nous ne savons pas combien de temps sera nécessaire pour ces ordinateurs pour briser les algorithmes actuels, tels RSA ou les courbes elliptiques. Nous ne savons pas quand seront découvertes les vulnérabilités des algorithmes post-quantum proposés dans la nouvelle norme, et nous ne savons pas comment les ordinateurs actuels pourront gérer ces nouveaux algorithmes, notamment avec le matériel cryptographique plus imposant (par exemple des clés plus longues) et informatique (les calculs plus complexes à réaliser, et donc plus de ressources à consommer). Finalement, nous ne savons même pas quand nous aurons besoin car il est très difficile de savoir quand les redoutables ordinateurs quantiques seront suffisamment stables et fiables pour être utilisés pour briser la cryptographie à clé publique. La seule certitude, à ce stade, est que les ordinateurs

quantiques sont perçus comme une menace par de nombreux acteurs (gouvernements, cryptographes, ingénieur·e·s, codeuses, utilisatrices) et que cette inquiétude demande une réponse appropriée.

À l'instar de la confidentialité persistante qui est devenue une espèce de norme pour les nouvelles applications de messageries chiffrées (Ermoshina et Musiani 2019), la cryptographie post-quantique se répand assez rapidement en tant que norme vers laquelle tendre. Même si les utilisatrices ne savent pas forcément ce que cela implique exactement, ni ce que cela signifie, ces critères sont souvent signalés dans les comparatifs entre applications, et deviennent donc des critères « boîtes noires » qui orientent le choix des utilisatrices. Par exemple, l'entreprise Tutanota, qui propose sa propre solution de messagerie électronique chiffrée qui ne suit pas la norme OpenPGP, justifie ainsi ce choix :

Les normes de chiffrement actuelles telles que PGP et S/MIME ont certains problèmes auxquels Tutanota a l'intention de répondre. Ces normes ne sont pas compatibles avec la confidentialité persistante et ne sont pas résistantes aux attaques en provenance d'ordinateurs quantiques.²⁷⁴

Nous sommes à nouveau face à un discours relativement classique de sécurisation qui semble efficace même si la personne qui lit ne sait pas ce qu'est la confidentialité persistante ou les attaques en provenance d'ordinateurs quantiques. Schématisant ce discours, on se trouve face au schéma suivant : « sans [boîte noire technique A] ni [boîte noire technique B], la cryptographie n'est pas efficace ; or nous sommes la seule implémentation à proposer cela pour les emails ; achetez notre produit. » C'est d'autant plus frappant qu'à ma connaissance, à l'heure où j'ai copié cet extrait, Tutanota n'utilisait pas encore d'algorithmes post-quantiques mais mentionner qu'OpenPGP était vulnérable face aux attaques provenant d'ordinateurs quantiques et que Tutanota était conscient du danger et voulait protéger les emails de ses clients par des algorithmes plus puissants a certainement un effet performatif sur le sentiment de sécurité des utilisatrices.

²⁷⁴ <https://tutanota.com/fr/faq#pgp>, ma traduction.

Par ailleurs, il s'agit aussi un acte de positionnement dans un marché de plus en plus compétitif. D'ailleurs, comme l'ont montré Ksenia Ermoshina et Francesca Musiani (2022), les utilisatrices à haut-risques (par exemple des activistes en régime autoritaire ou des journalistes) ne cherchent pas forcément une technologie gratuite, open source et fédérée : l'important est un service de qualité qui les protègent effectivement. Ainsi, un tel acte de positionnement et de justification de la part de Tutanota permet d'établir leur produit comme une solution pragmatique, certes payante, mais efficace et surtout plus fiable que les services concurrents.

Dans un même mouvement, cela contribue aussi à instaurer une nouvelle norme en matière de garantie de sécurité, ce que Ermoshina et Musiani (2019), en suivant l'exemple de Signal, ont nommé « la normalisation *de facto* grâce à un code fonctionnel ». Cette normalisation n'est pas seulement l'affaire des spécifications techniques, mais aussi des attentes des utilisatrices. Cela peut donc amener des entreprises concurrentes à répondre à ce nouveau besoin en proposant également leur solutions post-quantique.

Le projet-sécurité de chacune et chacun se construit autour de ses propres connaissances et ses propres projections de l'avenir. Il est très probable que dans un avenir relativement proche, pour choisir une solution de chiffrement asymétrique, il sera certainement conseiller de pouvoir cocher la case « cryptographie post-quantique ». Or, une fois de plus, ces algorithmes sont relativement nouveaux et nous ne savons pas comment ils se comporteront à moyen et long terme car nous n'avons que très peu d'expérience à leur sujet. Cela fait écho la situation discutée dans la section 5.1.3 (« Le choix d'une librairie cryptographique », p. 202) où il était question des critères pour choisir une librairie cryptographique. Pour certaines personnes, il était beaucoup plus rassurant d'avoir l'entier du code en RUST – même si cela impliquait d'utiliser une implémentation que l'on supposait fortement de ne pas être résistante à certaines attaques sophistiquées – alors que pour d'autres, il était préférable d'utiliser une librairie pour les primitives cryptographiques qui soit fiable, réactive et réputée, bien que codée en C. Dans les deux cas, la « sécurité » n'est pas uniquement une

question de caractéristiques techniques, mais de compromis, de projections, et certainement aussi souvent de persuasion, voire d'auto-persuasion. C'est d'ailleurs ce dernier point que j'aimerais aborder ici, en analysant le lien entre sécurité, connaissance et avenir.

9.3.3. Surfer sur la vague de l'innovation

Prendre soin d'une technologie numérique serait comparable à ce que fait une surfeuse pour se *maintenir* sur son surf ; tant qu'elle arrive à maintenir son équilibre et adopter sa vitesse à celle de la vague, tout va bien. Mais si elle se fait dépasser par la vague, ou au contraire, si elle va trop vite et se retrouve sans plus d'élan en bas de la vague, ou encore si la vague se casse, elle tombe. Il faut alors qu'elle fasse attention à ne pas se noyer. Il faut aussi qu'elle puisse garder suffisamment de force pour remonter dans la mer et recommencer. Et continuer ainsi à faire face à la puissance de la mer jusqu'à ce qu'elle se lasse, qu'elle ait trop froid, ou que la mer se calme. Cet exercice exige des compétences techniques, mais aussi d'analyse de la situation actuelle ainsi que de l'anticipation pour voir les vagues venir.

Maintenir une technologie numérique, c'est non seulement maintenir son équilibre sur la vague et garder une vitesse appropriée, mais c'est aussi gérer l'anticipation des vagues, choisir la bonne et tenter de les suivre jusqu'au bout. Bien entendu, la métaphore a des limites. L'une des premières qui vient à l'esprit est que contrairement au surf, le soin porté à ces technologies n'est pas un jeu mais un travail conséquent, généralement peu valorisé, chronophage et incertain. Une autre limite est que dans la mer, les vagues vont généralement dans la même direction et qu'il suffit de remonter au large pour attraper une nouvelle vague qui vous emmène vers la côte. Dans la vie des codeuses, la direction de l'avenir n'est jamais certaine et les vagues vont dans plusieurs directions en même temps. Choisir la « bonne » vague n'est pas évident, d'autant plus qu'il s'agit parfois d'en prendre plusieurs en même temps. Finalement, alors que la surfeuse se met elle-même en danger dans son activité sportive, la maintenance informatique n'engage pas la sécurité des seules personnes responsables mais de toutes celles qui utilisent leur programme. Je pense

pourtant que cette métaphore est utile pour se rendre compte du type d'effort qu'engage la maintenance informatique : il s'agit de maintenir une technologie à la surface des vagues et d'avancer *avec* le monde vers une direction qui n'est pas toujours prévisible mais que l'on apprend à regarder.



Illustration 30: Apothéose d'un cadenas en plein surf. Image créée par un générateur d'images par texte <https://www.craiyon.com/>.

Ce chapitre contribue à comprendre ce que l'avenir a de problématique dans le soin d'une technologie de sécurité informatique. Cette question paraît évidente et pourtant, sa réponse permet de comprendre quelque chose de fondamental dans la notion de soin des technologies informatiques : ce qui compromet les technologies numériques ne les touche pas et ne les transforme pas. Ce n'est pas le temps qui altère les primitives cryptographiques, les spécifications d'OpenPGP ou le code des implémentations, c'est tout le reste qui change avec le temps. Ce qui va se détériorer, c'est l'usage qu'on peut faire de ces technologies si personne n'en prend soin, et leur pertinence si personne ne veille à les adapter aux nouvelles conditions. Prendre soin de technologies numériques ne signifie donc pas faire en sorte qu'elles ne s'abîment pas, mais qu'elles restent pertinentes. Contrairement à un cadenas IRL, le cyber-cadenas ne va pas rouiller ni se gripper, il va simplement cesser d'être pertinent dans le monde car le monde aura changé.

Ce que cela nous apprend sur la gouvernance de la sécurité informatique concerne la gestion de la durée de la sécurité. S'il était possible de figer la sécurité au sommet de la vague afin que celle-ci y demeure éternellement, les choses seraient bien plus simples. Toutefois, la vague avance et se brise, irrémédiablement. Saisir la durée de la vague pour savoir quand se préparer à y sortir sans trop de casse, tel est l'enjeu.

Les pratiques de soin des choses informatiques sont dirigées vers la fonction plutôt que la forme, le contenu plutôt que le contenant. Ce qui compte n'est pas d'assurer une cohérence esthétique et encore moins une quelconque authenticité avec le passé, mais de préserver les fonctions. Or, l'avenir est plein d'incertitudes et de menaces qui représentent d'importantes contraintes qui pèsent sur le travail de soin. L'anticipation de ce qui va arriver devient ainsi cruciale pour concevoir un projet sécurité et tenter de s'en rapprocher. L'anticipation implique non seulement de comprendre ce qui va arriver, mais aussi de rendre la technologie aussi résiliente que possible dans l'avenir, c'est-à-dire faciliter le travail de soin des personnes qui s'en occuperont.

Finalement, cette partie sur l'avenir permet d'analyser concrètement la notion de projet-sécurité. La conception même de la sécurité que cherchent à produire les développeuses d'OpenPGP est fondamentalement liée à leur perception de l'avenir. Je n'ai rencontré personne qui ait suggéré qu'OpenPGP allait demeurer fiable et solide dans le temps, et encore moins qu'il en existait ne serait-ce qu'une implémentation infaillible. Au contraire, la majorité des personnes interviewées m'a dit plus ou moins explicitement être toujours dans l'attente voire dans l'anticipation du prochain problème à régler.

Pour résumer, la résilience prospective ne doit toutefois pas être considérée comme une propriété mais bien comme un projet, ou même une projection. Ce n'est pas une chose, un paramètre, une fonction que la technologie aurait, ni même une idée qu'elle soutiendrait. Il s'agit de la volonté de préparer au mieux l'avenir en tâchant de rendre la technologie en question aussi fiable et stable que possible, mais

également flexible pour qu'elle puisse, à terme, s'adapter aux exigences de l'avenir dont on ne sait encore très peu de choses.

Au terme de ce chapitre, j'aimerais souligner à quel point l'avenir, quoique incertain et vertigineux, joue un rôle crucial dans le travail de soin de l'écosystème OpenPGP. Comme je l'ai montré, cette anticipation de l'avenir est cruciale pour comprendre l'entrelacement des pratiques d'innovation et de maintenance et le souci de l'avenir se conjugue ainsi avec celui de la continuité.

10. Conclusion : penser le compromis, panser la sécurité

*De tout, il restera trois choses :
la certitude que tout était en train de commencer,
la certitude qu'il fallait continuer,
la certitude que cela serait interrompu avant que d'être terminé.
Faire de l'interruption, un nouveau chemin,
faire de la chute, un pas de danse,
faire de la peur, un escalier,
du rêve, un pont,
de la recherche...
une rencontre.*

(Fernando Sabino, O encontro marcado, 1956)

Ces trois certitudes décrivent subtilement ce que l'anthropologue Marc Augé (2011, 9) nomme la prise de conscience du fini et de l'infini : « [l]e premier paradoxe du temps est inhérent à la conscience que prend l'individu d'exister dans un temps qui a précédé sa naissance et qui continuera après sa mort ». Nous vieillissons dans un monde qui ne cesse de se transformer, ne serait-ce que parce que les choses vivantes et non-vivantes qui composent ce monde naissent, se transforment et disparaissent autour de nous, et que nous faisons de même. En énonçant ces trois certitudes, le poète brésilien évoque – probablement sans chercher à le faire – l'impératif moral ressenti par la plupart des personnes qui prennent soin d'OpenPGP. En effet, « l'art de faire durer les choses » (Denis et Pontille 2022) consiste à prendre soin des choses qui sont héritées du passé et à tâcher de les faire continuer – dans un « in-

fini recommencement » qui ne se termine jamais. En appliquant l'évocation du poète au cas d'étude de la maintenance des choses numériques, c'est exactement cela qu'affirme la dernière certitude : il n'est pas possible de *terminer* le soin car cela serait toujours trop tôt. Il est certes possible d'achever différentes tâches, mais le soin demande un travail continu de correction, de mise à jour et de projection qui maintient la technologie en mouvement. Si ce travail cesse, la technologie s'arrête également et meure.

Dans le cas d'OpenPGP, la deuxième certitude – « il fallait continuer » – semble pourtant controversée, et ce depuis des décennies : tant de personnes clament depuis des années qu'OpenPGP devrait mourir et qu'il vaudrait mieux passer à autre chose (Off-The-Record en 2004, Signal en 2013, et encore beaucoup d'autres solutions à venir). Cependant, « faire mourir » OpenPGP n'est pas si facile. La mort du protocole n'est pas une option envisageable pour beaucoup de personnes qui comptent sur lui pour protéger des données d'une part et pour authentifier leurs *commits* et paquets logiciels open source d'autre part. La panique suscitée par la divulgation d'EFAIL le montre bien : OpenPGP ne peut mourir sans que des milliers de personnes s'en inquiètent. De plus, OpenPGP est enlisé dans une infrastructure que le protocole a lui-même contribué à bâtir. L'infrastructure dépend de lui tout autant qu'il dépend d'elle.

Renforçant cette dépendance réciproque, le programme PGP, devenu le protocole OpenPGP, a traversé trois décennies, ce qui peut sembler à la fois peu et beaucoup de temps. D'un côté, cette trentaine d'années peut sembler beaucoup pour une technologie qui a été au cœur de moult controverses dès sa publication. Le rythme fulgurant du développement des technologies du numérique donne l'impression que tout change très vite et qu'il n'y a pas de place pour les « anciennes » technologies. Les vieux logiciels sont régulièrement remplacés par des nouveaux, supposés être plus rapides, plus performants et, parfois, plus sûrs. Cependant, alors que les logiciels changent, certains anciens protocoles demeurent en toile de fond de cette infrastructure : nous utilisons toujours le protocole TCP/IP (1981) pour nous connecter à Inter-

net, HTTP (1996) assure encore nos connexions aux sites web que nous consultons et les protocoles SMTP (1982) et IMAP (1986) continuent d'assurer l'envoi et la réception de nos emails. OpenPGP fait partie de ces « dinosaures » qui, même vétustes, persistent et continuent d'être des outils importants parmi les technologies d'Internet.

D'un autre côté, au vu des millénaires d'expériences cryptographiques, la cryptographie numérique à clé publique mise au point à la fin des années septante fait encore figure de toute nouvelle venue, à peine adulte, et OpenPGP est, de ce point de vue, à l'aube de l'adolescence. En replaçant le chiffrement numérique à clé publique en général et OpenPGP en particulier dans le temps long de l'histoire de la cryptographie qui dure depuis des années, on prend conscience qu'il n'est absolument pas anormal de voir émerger de nouvelles vulnérabilités. Si, comme dans bien d'autres domaines d'ailleurs, l'informatique a en effet constitué un bon en avant majeur dans le domaine de la cryptographie, les techniques de chiffrement n'ont pas cessé d'évoluer depuis des millénaires et il ne fait pas de doute que ce mouvement se poursuive encore à l'avenir.

Les questions auxquelles j'ai tâché de répondre dans cette thèse concernaient le vieillissement des technologies de la sécurité informatique : quelles sont les logiques des activités de maintenance, d'innovation et de préparation de l'avenir ? Qu'est-ce que ces logiques peuvent nous apprendre sur la gouvernance de la sécurité de l'information ? Et comment ces technologies sont-elles marquées, voire contraintes, par des pratiques et des manières de faire héritées d'un temps que les digital natives ne peuvent pas connaître ?²⁷⁵ J'ai apporté plusieurs réponses qui montrent comment le soin de ces technologies est astreint à différents rythmes qui façonnent l'écosystème, à l'image de l'urgence ressentie lors de la crise de la divulgation publique d'EFAI et de la lenteur des négociations de l'IETF. Les différents rythmes étudiés font partie intégrante de la manière dont la sécurité informatique est perçue. J'ai aussi présenté comment l'avenir agit sur le présent de ces technologies, et sur le quotidien des développeuses qui en prennent soin par le biais des projets-sécurité. Dans cette

²⁷⁵ Néroum, Aznavour.

conclusion, je propose une synthèse de mes différentes contributions en démêlant les trois dimensions de la temporalité de la sécurité informatique que j'ai explorées : le mouvement, le rythme et la durée. Ce que j'appelle le « triangle des temps de la sécurité » permet de comprendre les conditions temporelles de la sécurité informatique. Finalement, à la fin de cette conclusion, je souligne encore une fois l'essence fondamentalement collective de la sécurité informatique et propose une dernière contribution à la compréhension des infrastructures numériques.

10.1. Les trois dimensions temporelles de la sécurité informatique

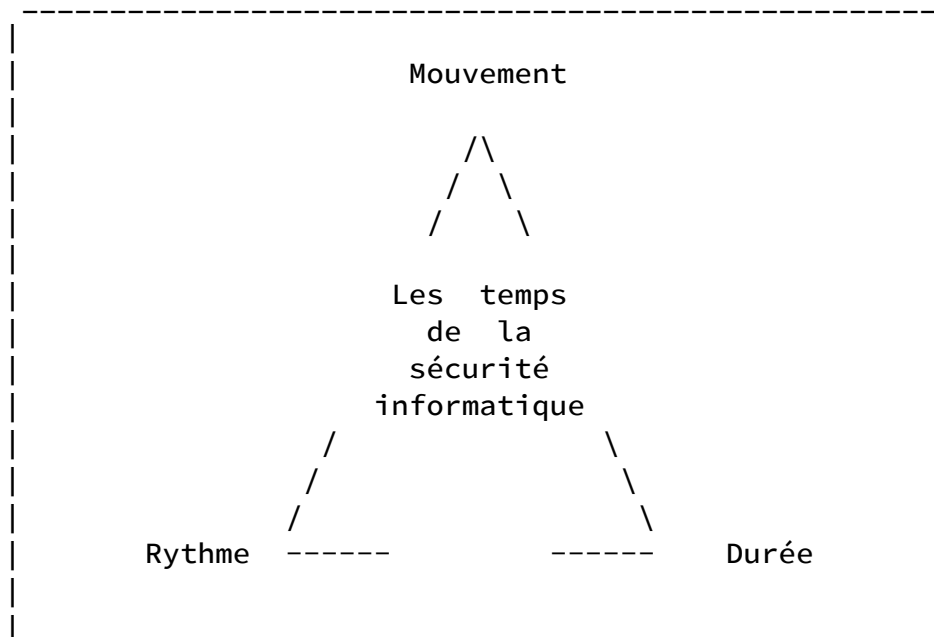
Les observations que j'ai faites et les entretiens que j'ai conduits durant cette recherche contribuent à enrichir la manière de penser la sécurité informatique en suggérant une forte interconnexion entre la sécurité et le temps – ou plutôt les temps puisque le temps s'est justement décliné en différentes durées et en différents rythmes. Mes analyses ont aussi montré une incessante transformation à partir de tâches répétitives, comme la correction de bogues.

J'aimerais ici une dernière fois revenir à la métaphore du cadenas pour représenter la sécurité informatique. C'est une métaphore utile pour appréhender la sécurité de manière moins pure que la façon dont elle est souvent présentée. En effet, la sécurité est pleine de cambouis et celles et ceux qui y mettent les mains le savent bien. Plutôt qu'une série de caractéristiques techniques claires et limpides que l'on peut cocher sur une liste, la sécurité informatique apparaît comme une chose non seulement continuellement bricolée (Kocksch et al. 2018) mais aussi sans cesse compromise, dans le double sens du terme : à la fois toujours susceptible d'être remise en question, et objet d'un compromis. Contrairement au cadenas IRL dont les caractéristiques cessent d'évoluer une fois sorti de l'usine, le cyber-cadenas bleu électrique est un véritable *patchwork*, littéralement une collection de patches, raccommodages de toutes sortes collés ensemble au fil des ans.

Dès l'introduction, j'ai souligné l'une des différences entre les deux cadenas, l'un d'une nature matérielle et l'autre constitué d'informations, et donc non matériel. Le

cadenas IRL se dégrade à cause de sa condition matérielle. Face à cette usure, le soin prodigué cherche à prolonger la durée de vie. En revanche, le cyber-cadenas n'est quant à lui pas soumis aux mêmes conditions matérielles car il est avant tout informations et non matière²⁷⁶. En analysant comment la sécurité informatique était produite, recherchée et compromise, j'ai souligné les conditions temporelles de cette sécurité. En particulier, la notion de projet-sécurité permet d'appréhender la sécurité informatique d'un point de vue diachronique, c'est-à-dire en questionnant son évolution à travers le temps (Bosa 2011), mettant ainsi spécifiquement en valeur son caractère dynamique.

Mon analyse a montré que ce dynamisme était causé par différentes appréhensions du temps sous forme de durées, de rythmes et de possibilités de mouvement des technologies numériques. Ces trois dimensions du temps sont interdépendantes et s'articulent entre elles. Le schéma encadré ci-dessous présente ce « triangle des temps de la sécurité » :



276 Une fois de plus, je précise que la fragilité des choses matérielles de l'informatique n'est absolument pas remise en question : les ordinateurs se cassent, les fils se coupent, les serveurs brûlent. Il s'agit néanmoins d'un autre problème qui ne concerne ni le code ni les spécifications des protocoles, mais des supports sur lesquels l'informatique fonctionne.

10.2. Mouvement : vieillir n'est pas mourir, c'est survivre

Dans la première partie « L'héritage du passé en mouvement » (p. 146), nous avons navigué dans les eaux troubles du vieillissement des technologies numériques en nous laissant guider par un premier paradoxe : vieillir, pour un protocole ou un code informatique, ne signifie pas se rapprocher de sa fin, mais, au contraire, s'éloigner de sa mort. À l'inverse, rester pur et inchangé, c'est se noyer dans l'ébullition constante des technologies numériques, et disparaître. Les deux chapitres de la première partie, respectivement centrés sur le protocole et le code, montrent que sans l'intervention continue de nombreuses personnes, OpenPGP n'aurait jamais survécu si longtemps. Ainsi, pour une technologie numérique, le vieillissement est une bonne chose qui n'est possible que si des personnes s'affairent à la faire durer, c'est-à-dire à maintenir des fonctionnalités du passé pour rester interopérable, tout en incorporant les nouvelles fonctionnalités pour rester d'actualité avec le présent. C'est en cela que cette partie parle du mouvement : comment la technologie est en mouvement, entre le passé et l'avenir, mais aussi entre l'enlissement et la (p)réparation de l'avenir vue dans la dernière partie de la thèse. Cela nécessite un travail d'attention et de soin de la part des humains, mais également un ancrage infrastructurel fort.

Retournons un peu en arrière. Dans le quatrième chapitre, j'ai particulièrement insisté sur l'interdépendance entre le protocole et l'infrastructure en montrant comment ces entités s'alimentaient réciproquement, pour le meilleur et pour le pire. Cet ancrage est souvent perçu comme un enlissement, un frein à l'innovation, ou même un non-mouvement. En tant que tel, c'est un problème contre lequel il faut agir. L'exemple du Web of Trust permet de montrer l'inertie infrastructurelle d'une idée qui est critiquée par beaucoup. Les obstacles que cause cet enlissement infrastructurel sont tels que même les « petits » changements qui paraissent évidents et inéluctables à certaines personnes demandent beaucoup de négociations avant d'être effectués. J'ai pourtant également montré quelques contre-exemples d'innovations qui semblent fonctionner : le protocole Autocrypt en est une illustration. Dans ce cas, le poids de l'infrastructure, sa stabilité, devient un argument en faveur de la technolo-

gie en question ; l'enlissement devient ancrage, une garantie de stabilité qui permet d'avancer, à condition d'avoir l'expertise nécessaire pour développer un tel protocole malgré les contraintes de cette infrastructure.

Le cinquième chapitre a quant à lui analysé comment s'accommoder du passé nécessite de raccommode le code continuellement. Le code peut être perçu comme le trait d'union entre les propositions théoriques du protocole et les attentes pratiques des utilisatrices. L'exemple de la fonction `parse_iso8601` a permis de montrer comment le code est construit autour d'étapes très simples qui peuvent prendre des proportions importantes tant les ordinateurs sont « pointilleux ».

Cet exemple a pu mettre en lumière la syntaxe du code, entre fonctions, commentaires et tests, ainsi que l'architecture du code et ses liens de dépendance avec d'autres bibliothèques. Maintenir un code, ce n'est rien d'autre que le transformer continuellement pour maintenir non pas son esthétique mais sa fonctionnalité : le code est transformé, amputé de certaines lignes, augmenté de nouvelles fonctions, et cela toujours de manière incrémentale à chaque fois qu'un bogue est détecté et corrigé ou qu'une amélioration est ajoutée. C'est en ce sens qu'il est inévitable pour un code de demeurer en mouvement si l'on veut qu'il demeure fonctionnel. Mais les changements opérés laissent de nombreuses traces comme autant de cicatrices du passé qui augmentent la complexité du code. Par conséquent, pour le code, vieillir, c'est aussi devenir de plus en plus complexe, épais, et donc difficile à mettre en mouvement. En effet, quand arrive le moment de le remanier, ce *patchwork* sera d'autant plus difficile à comprendre et à corriger. Autrement dit, prendre soin du code, le faire vieillir, nécessite de ne jamais cesser de le transformer et de le maintenir en mouvement ; or, plus le code est transformé, corrigé, augmenté, plus sa complexité et son enlissement s'accroissent et plus il est difficile de le maintenir à flot. La tâche des mainteneuses est donc toujours plus complexe.

Dans cette partie, j'ai montré comment le passé d'OpenPGP a souvent été critiqué par de nombreuses personnes pour de multiples raisons. Or, il me semble important de souligner un aspect fondamental de cette technologie qui est devenu clair au fil

des chapitres de cette thèse : malgré toutes les controverses, malgré toutes les vulnérabilités déjà trouvées et documentées, et malgré la difficulté d'usage régulièrement critiquée, OpenPGP persiste et de nombreuses personnes s'entêtent à maintenir cette technologie : non seulement pour prolonger la mission que s'était fixée Philip Zimmermann de protéger les emails, mais aussi – et peut-être surtout – parce qu'OpenPGP est d'ores et déjà enlisé dans l'écosystème du développement open source, où la signature de *commits* et de paquets logiciels s'effectuent avec ce protocole. Ainsi, après plus de trente ans d'existence tumultueuse, je pense qu'il est aussi possible de voir OpenPGP comme une *success story*.

Le portrait de la sécurité informatique que j'ai brossé fait appel à une figure toujours mouvante et incomplète. Mouvante car les menaces évoluent sans cesse, de même que les techniques pour y répondre ; et incomplète car la sécurité est un projet que l'on sait inatteignable car toujours compromis. Autrement dit, la sécurité nécessite toujours plus de soin, c'est-à-dire de réparation, de maintenance et d'innovation ; si l'on s'arrête, la sécurité fait de même et disparaît. Ainsi, vieillir, être en mouvement, n'est pas mourir un peu mais, tout au contraire, tenter de survivre.

Une des contributions de cette thèse est de montrer comment la sécurité informatique est construite et déconstruite au quotidien. Ici encore, le préfixe *dé-* est trompeur : déconstruire n'est pas toujours synonyme de détruire. Dans le cas des pratiques de soin d'OpenPGP que j'ai observées, déconstruire signifie aussi trouver des vulnérabilités, comprendre ce qui les cause, et tâcher de transformer cette partie de la technologie. Déconstruire peut également signifier défaire pour reconstruire, ou construire encore, mais différemment. À l'instar de la marche, c'est donc dans un mouvement paradoxal de « déséquilibre compensé » (Laplantine 2023, 39) que la sécurité et l'insécurité sont produites simultanément. Les différentes pratiques de réparation, de maintenance et d'anticipation que j'ai analysées ont mis à jour un bricolage perpétuel fait d'essais et d'erreurs. Les personnes qui en ont la charge sont guidées non pas par la certitude de savoir quoi faire, mais par « la certitude qu'il fallait continuer », autrement dit, qu'il fallait avancer.

Il devient alors intéressant de questionner la relation entre sécurité et mouvement, même si ce mouvement est parfois paradoxal. D'un côté, j'ai exposé comment la poursuite de la sécurité ne se construit pas dans l'événement, dans la création, dans l'innovation, c'est-à-dire dans une rupture, dans un mouvement entre l'avant et l'après, mais dans la continuité, dans la poursuite d'un projet, dans le soin routinier qui est porté à des technologies. En d'autres termes, la recherche de la sécurité semble vouloir instaurer un temps linéaire, monotone, infini, et la sécurité se voudrait un état fixe, figé. De ce point de vue, la sécurité correspond métaphoriquement à une absence de mouvement. Mais d'un autre côté, mon analyse a montré que le sentiment d'insécurité survient généralement dans des moments de crise, de *clash*, ou qui sont du moins souvent présentés comme tels. J'ai illustré ce point avec plusieurs exemples : EFAIL, *SHA-1 is a Shamble*, les attaques sur les serveurs des clés ou encore, de manière plus ordinaire, la découverte de bogues. Ces événements ébranlent le fonctionnement « normal » de l'infrastructure en place, la force à bouger, à s'ébranler, à se remettre en question, causant ainsi de l'insécurité effective et affective : effective en montrant comment exploiter des vulnérabilités déjà présentes, et affective en suscitant différentes émotions chez différentes personnes (de la peur, de la colère, de l'exaspération, peut-être aussi de la curiosité et de l'excitation). Le mouvement, dans ce cas, c'est l'avancée de l'insécurité ; or, comme je l'ai montré à de multiples reprises, cette insécurité est nécessaire à la sécurité, comme l'est l'état de « déséquilibre compensé » lors de la marche (Laplantine 2023).

Ainsi, d'un côté, il y aurait donc le temps de la continuité et de l'autre, celui de la crise. Or, comme Janet Roitman (2014) l'a démontré : les crises aussi ont tendance à s'éterniser et à se fondre dans le moule du quotidien. Ainsi, ce contraste entre la stabilité désirée de la sécurité et son intempestive impermanence que l'on nomme insécurité ne se vérifie que partiellement dans la pratique. En effet, la sécurité et l'insécurité sont souvent beaucoup plus ambiguës que cela. L'exemple d'EFAIL est parlant : la recherche des scientifiques avait pour but de trouver des vulnérabilités dans le protocole à la fois pour le sécuriser et l'insécuriser. Les patchs qui ont été ap-

pliqués aux codes ont parfois permis de sécuriser certains aspects, mais ont été en même temps sources d'ennuis et de bogues ultérieurs. La sécurité et l'insécurité se côtoient dans une instabilité sans cesse ajustée, dans un temps de crise qui dure et dans une routine sans cesse réinventée. Cela crée des remous dans la manière de vivre le présent, engageant alors d'examiner la deuxième dimension des temps de la sécurité : le rythme.

10.3. Rythme : une construction fragile prête à s'enflammer

Dans sa forme la plus routinière et la plus ordinaire, la sécurité est ce long travail de soin qui s'observe au quotidien, notamment sur les plateformes de partage de code GIT ou les listes de diffusion. Elle prend alors la forme de dialogues, de négociations, d'incompréhensions parfois, d'explications souvent, de tentatives, de bricolages. En ce sens, la sécurité est une attention et une intention : l'attention à la fragilité ordinaire des choses, et l'intention d'en faire quelque chose. Cela a été particulièrement analysé dans la deuxième partie intitulée « les rythmes du présent », qui entre en résonance avec l'affirmation de Steven Jackson :

« [...] le monde est toujours cassé ; c'est dans sa nature de se casser. Cette casse est générative et productive [...] » (Jackson 2014, 223).

Cette deuxième partie de la thèse s'est focalisée sur différents rythmes de travail, de l'urgence du patch à la lenteur des négociations de l'IETF. Elle a analysé les répercussions de la divulgation publique d'EFAIL durant les quelques jours de la controverse médiatique suscitée par cette divulgation, et une période beaucoup plus longue et distendue où ces vulnérabilités ont fait travailler les codeuses et ingénieurs de l'IETF durant des années. En se focalisant sur le temps présent, cette partie illustre à quel point la notion même de sécurité ne fait absolument pas consensus, donnant ainsi raison, du moins en partie, aux théories constructives de la sécurité. En effet, alors qu'on aurait pu attendre d'un domaine aussi technique et binaire que les sciences informatiques en général et la cryptographie en particulier une certaine objectivité, force est de constater que les choses ne sont pas aussi simples que cela.

Le cas d'EFAIL est parlant. Plutôt que d'avoir un problème précis et des solutions claires, la communauté OpenPGP s'est retrouvée dans le flou sans parvenir à trouver un consensus, ne serait-ce que pour s'accorder sur la gravité de ces vulnérabilités. Les découvertes des chercheurs ont été intensément débattues. Dans son billet de blog à propos de la divulgation d'EFAIL, le cryptographe et scientifiques Matthew Green (2018, ma traduction) a souligné le sérieux de ces vulnérabilités : « Si l'on se réfère aux vulnérabilités cryptographiques, on peut dire qu'il s'agit de la plus mauvaise situation possible. » Les scientifiques ont fait une bonne recherche qui a fait avancer les connaissances scientifiques sur OpenPGP, et leurs résultats semblent catégoriques – du moins du point de vue de la communauté scientifique. Pourtant, la communauté OpenPGP n'a pas réussi à s'accorder sur le sens à donner à propos de ce qui était en train de se passer. Certaines personnes issues des implémentations ont affirmé l'inverse : « Cet article est mal nommé. [...] Cette attaque vise des clients email bogueés. [...] Les auteurs ont fait une liste de clients email bogueés » (R. J. Hansen 2018). Finalement, certains médias ont clamé la mort d'OpenPGP (Katwala 2018).

L'incertitude régnait et chaque contre-argument augmentait la controverse. Cette controverse a permis de comprendre un aspect peu spectaculaire et peut-être aussi peu rassurant de la sécurité de l'information. Étrangement, c'est un travail scientifique qui a été la source des nombreuses incertitudes qui ont alimenté la controverse d'EFAIL. Alors que la science devrait apporter des réponses, le cas d'EFAIL nous montre que parfois, c'est elle qui nous plonge dans l'embarras sans que l'on sache quoi faire. En effet, loin de se référer à un état purement technique et observable comme l'ont présenté les scientifiques d'EFAIL, l'insécurité a été perçue à travers les reproches, les critiques et la peur suscitée lors de la révélation publique. L'incertitude sur la manière de traiter ces vulnérabilités est d'ailleurs demeurée bien après que la controverse est retombée, et les longues négociations à propos de l'AEAD et de la *chunk size* qui ont eu lieu sur la liste de diffusion du groupe de travail de l'IETF montrent que les questions techniques n'étaient pas l'unique point d'achoppement sur lequel butait le groupe de travail. Ainsi, à travers l'exemple d'EFAIL, j'ai montré

comment la sécurité était une construction fragile et prête à s'enflammer. Cette construction tient en place grâce à l'improvisation et les bricolages, malgré les hésitations et les incertitudes. De cette partie, on retiendra surtout que ce sont ces autres paramètres « humains » qui donnent à la sécurité des rythmes très différents : la peur, la panique, les hésitations, les incertitudes, etc. Précipitant l'action parfois, la ralentissant drastiquement d'autres fois, ces émotions et sentiments rythment la marche du développement du protocole et des codes.

En mettant l'accent sur le mouvement et le dynamisme des situations en jeu, notamment à travers la notion de « processus d'(in)sécurisation », j'ai affirmé que les processus de sécurisation et d'insécurisation répondaient d'une seule et même logique. Au terme de cette thèse, il semble toutefois se détacher une légère différence dans la représentation qu'elles donnent de ce qui va advenir. La sécurisation est empreinte d'optimisme car elle propose des solutions pour améliorer le monde. En ce sens, elle est en lien avec l'idée moderne de progrès (Latour 1991) qui parfois tourne à la caricature d'un solutionnisme technologique (Morozov 2014). Au contraire, l'insécurisation souligne que ce qui va advenir est porteur de menaces et demande une action, une réaction, dont on ne peut pas toujours être sûr du résultat. La flèche du temps s'inverse : d'un côté, ce sont les problèmes du passé qui sont résolus dans le futur ; de l'autre, ce sont les problèmes de l'avenir qui viennent à nous. Cette distinction n'est pas purement philosophique et a des conséquences bien précises puisqu'elle permet de distinguer deux mouvements opposés mais non contradictoires (si on ne les caricature pas trop) : d'un côté, il y a une fuite vers l'avant, caractérisée par des pratiques de création et d'innovation, de l'autre, une tentative de ralentissement de la dégradation de l'état actuel des choses, et le soin que certaines personnes leur portent. Cette partie a montré toute l'ambiguïté des pratiques de soin des technologies numériques et de la difficulté de s'entendre sur ce que « sécurité » signifie.

Dans son ensemble, cette deuxième partie sur les rythmes de la sécurité contribue à mieux cerner la manière dont les codeuses et ingénieur·e·s travaillent en dépit des nombreuses incertitudes qui s'offrent à elles. Sans savoir exactement où aller, ils

et elles sont guidées par leur projet-sécurité, sans cesse négocié par les contingences de la vie. Leur équipement comprend un solide bagage de connaissances techniques. En outre, ils et elles peuvent compter sur leur expérience et leur connaissance du code, la certitude qu'aucun code n'est parfait et leurs convictions qu'il est nécessaire d'avoir un code aussi bon que possible. Enfin, j'ai également observé de nombreuses croyances, comme celles que les ordinateurs quantiques représentent une menace existentielle pour OpenPGP, et surtout des valeurs telles le droit à la vie privée en ligne, l'interopérabilité, le consensus approximatif.

Bien que suffisamment concrète pour être saisie, travaillée, et affinée par les personnes qui s'en occupent au quotidien, la sécurité ne se limite pas aux objets qui la constituent en partie (bogues, vulnérabilités, *commits*, patches, rapports d'audit, etc.). Elle échappe aux réifications et se déploie au contraire dans de nombreuses dimensions sociotechniques, mais aussi affectives et de gouvernance. Elle apparaît par exemple en tant que discours et pratiques d'(in)sécurisation. Elle n'est plus seulement un réglage technique, mais elle devient une action, voire une réaction palliative. Elle fait parfois controverse, dans des situations d'incertitudes par exemple, ou quand des vulnérabilités sont découvertes et divulguées.

10.4. Vers la résilience prospective

La troisième partie, « La durée de l'avenir », repose sur un troisième constat : alors que l'on a généralement tendance à considérer la maintenance et l'innovation comme deux phénomènes distincts, il semble que, dans certains cas, elles participent d'un seul et même mouvement où le passé, le présent et l'avenir s'entremêlent inévitablement. La maintenance d'OpenPGP nécessite en effet d'anticiper les menaces à venir et l'innovation tente de réparer des problèmes du passé. Ainsi, prendre soin d'un protocole ou d'un code informatique, c'est entreprendre un mouvement d'anticipation afin de préparer la chose maintenue à faire face à des failles dont on ne connaît encore rien d'autre que la probabilité : non seulement il s'agit d'anticiper les chocs futurs, mais surtout de faire en sorte que le code et le protocole

puissent se montrer suffisamment flexibles et résilients quand cela sera nécessaire. L'horizon des codeuses n'est pas celui d'aujourd'hui mais celui de demain, et leur succès (ou échecs) ne se mesurent pas uniquement au présent, mais dans la durée. Leur inquiétude sincère à propos de la sécurité des données des utilisatrices et les pratiques d'anticipation de bogues futurs montrent qu'ils et elles en sont conscient·e·s.

Cependant, contrairement aux *success stories* d'une innovation fantasmée où le futur a une place radieuse, les pratiques d'innovation et d'anticipation de l'avenir autour d'OpenPGP envisagent un avenir effrayant, plein de menaces et d'incertitudes. Il en découle une situation intéressante et relativement paradoxale : le passé est plein de promesses (que les développeuses de l'époque ont faites) et l'avenir est fait de compromis (que font les codeuses actuel·le·s pour tâcher de tenir ces promesses). En s'appuyant sur les écrits d'Hanna Arendt sur la volonté comme « organe du futur », Shoshana Zuboff a écrit, dans *L'âge du capitalisme de surveillance* que « [f]aire une promesse, c'est prédire l'avenir. Honorer une promesse au moyen de l'exercice de la volonté, c'est transformer cette prédiction en fait » (2020, 445, ma traduction). Dès lors, on pourrait émettre l'hypothèse que les développeuses de l'époque ont prédit que des gens – peut-être elles·eux-mêmes, peut-être d'autres – allaient prendre soin de faire que les vieilles données chiffrées seraient toujours protégées contre des tentatives illégitimes de déchiffrement tout en assurant le bon fonctionnement du programme pour les jours à venir. Les développeuses du présent tâchent quant à elles·eux de tenir ces promesses. À cheval entre l'héritage du passé et les perspectives futures, le présent est fait de compromis.

Se préparer à l'arrivée des ordinateurs quantiques est une tâche qui demande bien plus que « simplement » ajouter une ou plusieurs primitives cryptographiques à la liste des algorithmes déjà supportés. Il s'agit surtout d'une tâche qui entre en dialogue, voire en concurrence, avec d'autres tâches tout aussi nécessaires et qui demandent l'attention des codeuses, comme c'était le cas dans les négociations du format des nouvelles paires de clé de 5^e génération. Une fois de plus, les considéra-

tions technoscientifiques du champ de la sécurité de l'information se heurtent à la subjectivité des développeuses sous la forme de la pluralité des projets-sécurité de chacune et chacun. L'avenir continue d'être menaçant – ou construit comme tel – et le passé continue d'être un fardeau que certaines personnes s'obstinent à traîner et que d'autres appellent à abandonner. Sans pouvoir savoir vers quoi ils et elles se dirigent, les codeuses travaillent du mieux qu'ils et elles peuvent pour obtenir la version d'OpenPGP la plus proche possible de leur propre projet-sécurité.

À l'instar des crises économiques qui font partie du système néolibéral (Giraud 2001), il semble que les vulnérabilités fassent intrinsèquement partie des technologies numériques, ainsi que les crises qu'elles génèrent. On ne sera jamais complètement prêt·e car il y aura toujours une conjonction de facteurs additionnels et imprévus qui donneront une saveur particulière à cette crise (Keck 2010), mais on sait qu'elles auront lieu. Ce que j'ai montré dans cette partie, pourtant, c'est la manière dont les ingénieur·e·s et codeuses tâchent de se tenir aussi prêtes que possible à réagir lors d'une situation d'urgence. Cela implique non seulement de tâcher d'anticiper les vulnérabilités avant qu'elles n'arrivent, par exemple en se préparant déjà aujourd'hui à l'arrivée des ordinateurs quantiques. Cela implique également de préparer la technologie elle-même à faire face au choc – notamment en documentant son fonctionnement aussi bien que possible, de sorte à ce que les codeuses qui devront la corriger à l'avenir puissent s'y repérer le mieux possible. C'est ce que j'ai appelé la résilience prospective.

Cette troisième partie a montré comment la sécurité s'inscrit aussi dans la durée. Il y a la question du temps de la protection : combien de temps OpenPGP permet-il de protéger les données chiffrées ? Dans ce cas, c'est un temps hypothétique et toujours remis en question par l'émergence de nouvelles vulnérabilités. Cette question demeure néanmoins au centre des préoccupations des utilisatrices et beaucoup voient dans les technologies de chiffrement qu'une façon de « faire gagner du temps » à Alice et Bob, ou en faire perdre à la maudite Ève. Mais il y a aussi le temps de l'usure à la différence que dans ce cas-là, ce ne sont pas les technologies numé-

riques elles-mêmes qui s’usent, mais le monde autour d’elles qui évolue et qui les rend obsolètes. Face à cette usure, il y a alors le temps du patch : combien de temps sera-t-il nécessaire pour corriger telle vulnérabilité, ou tel bogue ? C’est alors la durée de la fenêtre de vulnérabilité qui s’ouvre.

La cryptographie est particulièrement affectée par ce mélange de temporalités avec des temps extrêmement longs quand il s’agit de calculer le temps nécessaire à tester toutes les possibilités d’une clé, et extrêmement courts quand il s’agit d’améliorer les performances des primitives qui se mesurent parfois en centièmes de seconde. La cryptographie est un édifice bâti sur des fondations profondément ancrées dans la rigueur des mathématiques et voué à tenir tête aux attaques les plus obstinées durant des centaines voire des milliers (millions, ou milliards?) d’années ; elle est un maillon essentiel à la sécurité de l’information ; pourtant, à y regarder de près, elle semble aussi être une construction fragile que détériorent constamment des problèmes de toutes sortes.

En résumé, cette thèse fait remarquer que la sécurité informatique n’est ni éternelle comme dans les histoires d’Alice et Bob, ni événementielle comme dans les histoires de sécurisation, ni non plus uniquement rattachée à des pratiques de maintenance et de réparation. De même que les utilisatrices des technologies doivent d’abord concevoir leur modèle de menaces avant de choisir leurs stratégies (Kazansky 2021), les développeuses des technologies de sécurité informatique doivent également être au clair avec leur projet-sécurité. Le modèle parfait n’existe pas et des choix doivent nécessairement être faits, notamment au niveau des architectures réseaux (Ermoshina et Musiani 2022). La notion de projet-sécurité permet de placer la sécurité dans le temps et dans l’espace, fût-il cyber.

10.5. Communauté, Compatibilité, Customisation, Care et Compromis

Pour finir, j’aimerais revenir sur cinq valeurs centrales des infrastructures fédérées. Dans un récent ouvrage, Ksenia Ermoshina et Francesca Musiani (2022) ont identifié « les quatre C des fédérations » que sont la communauté, la compatibilité, la customi-

sation, et le care. Les deux sociologues ont notamment insisté sur les modes de gouvernance des fédérations comme l'IETF. Ces modes sont basés sur le consensus, les solutions fonctionnelles et pragmatiques [*running code*], et la gestion de la communication entre les parties prenantes. Elles reconnaissent qu'« il est probable que la transition vers les protocoles de chiffrement de nouvelles générations au sein d'écosystèmes fédérés soit lente et difficile » (Ermoshina et Musiani 2022, 178, ma traduction). Mais elles soulignent également les opportunités de tels écosystèmes, à la fois en termes d'interopérabilité (le deuxième C) et de flexibilité des technologies (le troisième C). Avec l'analyse d'exemples comme Autocrypt, Briar ou Conversation, elles ont montré que des possibilités de développement existaient au sein de ces fédérations très souvent critiquées pour leur lenteur et leur manque de flexibilité (Marlinspike 2016). Elles ont aussi souligné les modalités de customisation qu'offraient les écosystèmes fédérés et ont affirmé que « les fédérations offraient un espace pour la créativité » (Ermoshina et Musiani 2022, 180, ma traduction).

Le soin aux technologies numériques (le quatrième C) est également central – et je pense que cette thèse aura permis de s'en rendre compte. De tels écosystèmes ne se maintiennent pas tout seuls et de nombreuses personnes s'activent pour prendre soin d'eux. Des personnes cherchent (ou rencontrent) des vulnérabilités et les reportent à d'autres personnes, qui tâchent de comprendre le(s) problème(s) et de proposer des solutions. Ces solutions sont parfois partielles et incomplètes, et toujours susceptibles de causer d'autres problèmes. Avec le temps, les technologies vieillissent et deviennent de plus en plus complètes.

À ces quatre C, j'aimerais en ajouter un cinquième dont j'ai beaucoup parlé dans cette thèse : le compromis. Le compromis est central dans la vie des technologies fédérées comme OpenPGP. Comme les autres C, il représente à la fois des opportunités et des contraintes. Les contraintes sont évidentes : alors que les systèmes fermés, comme Signal ou Tutanota, peuvent suivre le projet-sécurité de leur conceptrices et appliquer les décisions le jour-même où elles sont prises, les fédérations doivent négocier chaque petit pas. D'ailleurs, il arrive régulièrement que la définition même

de la sécurité fasse l'objet de négociations. Faut-il utiliser SHA-1 ? Faut-il limiter la taille de la *chunk size* de l'AEAD ? Faut-il ajouter tel algorithme cryptographique post-quantique ? Finalement – et en tant que citoyen suisse, je suis bien placé pour le savoir – les solutions tirées de compromis ont l'art de ne satisfaire personne entièrement tout en mettant tout le monde plus ou moins d'accord.

Cela peut paraître une démarche bien étrange pour une chose aussi sérieuse et grave que la sécurité de l'information – de laquelle dépendent des vies humaines. Or, le compromis est fondamental pour concilier différents projets-sécurité. Sans compromis, il serait difficilement envisageable d'avoir des solutions interopérables dans une infrastructure décentralisée qui offre elle aussi de nombreux avantages. En affirmant qu'une technologie de sécurité informatique comme OpenPGP est constituée de compromis, je ne dis pas qu'elle se compromet, mais qu'elle com-promet, qu'elle promet avec d'autres. Les différentes implémentations d'OpenPGP qui, par la force des choses, sont aussi en concurrence les unes avec les autres, travaillent ensemble pour faire aboutir des négociations qui, jour après jour, bogue après bogue, patch après patch, font vieillir OpenPGP. Elles montrent que les normes techniques sont aussi « des questions politiques parce qu'elles représentent une forme de contrôle sur la technologie » (Abbate 2000, 147, ma traduction).

La plupart des études en sciences sociales sur la sécurité de l'information questionnent comment des programmes informatiques (in)sécurisent les utilisatrices, la société ou l'État. Par exemple, plusieurs études ont été publiées sur la manière dont les programmes de chiffrement inspiraient (ou pas) la confiance (Ruh 2018; Dechand et al. 2019; Bruun, Andersen, et Mannov 2020) ou sur des programmes malveillants et d'autres virus (Balzacq et Dunn Cavelty 2016; Stevens 2020). Ma démarche a consisté à faire le mouvement inverse, à savoir analyser ce que la sécurité faisait aux technologies. En montrant comment les projets-sécurité « entraînent dans la danse » et façonnaient les technologies du numérique, j'ai insisté non seulement sur le soin porté auxdites technologies ainsi qu'aux communautés qui s'en occupaient, mais aussi et surtout à l'attachement et l'engagement que manifestaient de nom-

breuses personnes à l'égard d'une technologie vieillissante. Vieillissante grâce à ces personnes.

*Tout était en train de commencer,
il fallait continuer,
et cela serait interrompu avant que d'être terminé.*

Conclusion : penser le compromis, panser la sécurité

Bibliographie

- Abbate, Janet. 2012. *Recoding Gender: Women's Changing Participation in Computing*. Cambridge Mass., London: The MIT Press.
- Akrich, Madeleine. 2006a. « La description des objets techniques ». In *Sociologie de la traduction: Textes fondateurs*, édité par Michel Callon et Bruno Latour, 159-78. Sciences sociales. Paris: Presses des Mines.
<https://books.openedition.org/pressesmines/1197>.
- . 2006b. « Les utilisateurs, acteurs de l'innovation ». In *Sociologie de la traduction: Textes fondateurs*, édité par Michel Callon et Bruno Latour, 253-65. Sciences sociales. Paris: Presses des Mines.
<https://doi.org/10.4000/books.pressesmines.1200>.
- . 2010. « From communities of practice to epistemic communities: health mobilizations on the internet ». *Sociological Research Online* 15 (2): 116-32.
- Akrich, Madeleine, Michel Callon, et Bruno Latour. 2006. *Sociologie de la traduction. Textes fondateurs*. Paris: Ecole des mines de Paris.
- Alcaras, Gabriel, et Antoine Larribeau. 2022. « Codes. L'informatique comme elle s'écrit ». *RESET. Recherches en sciences sociales sur Internet*, n° 11 (mars).
<https://doi.org/10.4000/reset.3914>.
- Alter, Norbert. 2002. *Les logiques de l'innovation: : approche pluridisciplinaire*. Recherches. Paris: Découverte.
- Aminot, Jean-Luc. 2020. « WannaCry, une frayeur à l'échelle planétaire ». *Annales des Mines - Responsabilité et environnement* 98 (2): 53-56.
<https://doi.org/10.3917/re1.098.0053>.

- Andersen, Jeffrey Thomas. 2016. « The Quest to Secure Email: A Usability Analysis of Key Management Alternatives ». M.Sc., Ann Arbor, United States.
<https://www.proquest.com/docview/2445353391/abstract/C23F153B77D64652PQ/1>.
- Angeli Aguiton, Sara, Lydie Cabane, et Lise Cornilleau. 2019. « Anti-Crisis : penser avec et contre les crises ? Entretien avec Janet Roitman ». *Critique internationale* 85 (4): 107-21. <https://doi.org/10.3917/crii.085.0107>.
- Arbaugh, William A., William L. Fithen, et John McHugh. 2000. « Windows of vulnerability: a case study analysis ». *Computer* 33 (12): 52-59.
<https://doi.org/10.1109/2.889093>.
- Assange, Julian, Jacob Appelbaum, Andy Muller-Maguhn, et Jeremie Zimmermann. 2016. *Cypherpunks: Freedom and the Future of the Internet*. London; New York: OR Books.
- Atwater, Erinn, Cecylia Bocovich, Urs Hengartner, Ed Lank, et Ian Goldberg. 2015. « Leading Johnny to Water: Designing for Usability and Trust ». In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, 69-88.
<https://www.usenix.org/conference/soups2015/proceedings/presentation/atwater>.
- Augé, Marc. 2011. *Où est passé l'avenir?* Paris: Éditions du Seuil.
- Aumasson, Jean-Philippe. 2018. *Serious Cryptography. A Practical Introduction to Modern Cryptography*. San Francisco: No Starch Press.
- Austin, John Langshaw. 1970. *Quand dire c'est faire*. L'ordre philosophique. Paris: Ed. du Seuil.
- Aviram, Nimrod, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, et al. 2016. « {DROWN}: Breaking {TLS} Using SSLv2 ». In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 689-706.
<https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/aviram>.
- Balzacq, Thierry, éd. 2011. *Securitisation Theory: How Security Problems Emerge and Dissolve*. London, New York: Routledge.
- . 2015. « The 'Essence' of Securitization: Theory, Ideal Type, and a Sociological Science of Security ». *International Relations* 29 (1): 103-13.
<https://doi.org/10.1177/0047117814526606b>.
- Balzacq, Thierry, et Myriam Dunn Cavelty. 2016. « A Theory of Actor-Network for Cyber-Security ». *European Journal of International Security* 1 (2): 176-98.

<https://doi.org/10.1017/eis.2016.8>.

Barenghi, Alessandro, Nicholas Mainardi, et Gerardo Pelosi. 2017. « A Security Audit of the OpenPGP Format ». In *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks 2017 11th International Conference on Frontier of Computer Science and Technology 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*, 336-43.

<https://doi.org/10.1109/ISPAN-FCST-ISCC.2017.35>.

Beaud, Stéphane. 1996. « L'usage de l'entretien en sciences sociales. Plaidoyer pour l'«entretien ethnographique» ». *Politix. Revue des sciences sociales du politique* 9 (35): 226-57. <https://doi.org/10.3406/polix.1996.1966>.

Belli, Luca. 2015. *De la gouvernance à la régulation de l'internet*.

<https://bibliotecadigital.fgv.br:80/dspace/handle/10438/33380>.

Benenson, Zinaida, Gabriele Lenzini, Daniela Oliveira, Simon Parkin, et Sven Uebelacker. 2015. « Maybe Poor Johnny Really Cannot Encrypt: The Case for a Complexity Theory for Usable Security ». In *Proceedings of the 2015 New Security Paradigms Workshop*, 85-99. NSPW '15. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/2841113.2841120>.

Bensa, Alban, et Eric Fassin. 2002. « Les sciences sociales face à l'événement ». *Terrain. Anthropologie & sciences humaines*, n° 38 (mars): 5-20.

<https://doi.org/10.4000/terrain.1888>.

Besençon, Sylvain. 2022. « (Dé)coder la sécurité de l'information entre maintenance et innovation : le cas d'OpenPGP ». *RESET. Recherches en sciences sociales sur Internet*, n° 11 (mars). <https://doi.org/10.4000/reset.3539>.

Besençon, Sylvain, et David Bozzini. 2020. « The Ethnography of a Digital Object. An Example From Computer Security ». *TSANTSA – Journal of the Swiss Anthropological Association*, n° 25: 153-60. <https://doi.org/10.36950/tsantsa.2020.025.15>.

Bialski, Paula. 2019. « Code Review as Communication: The Case of Corporate Software Developers ». In *Communication*, par Paula Bialski, Finn Brunton, et Mercedes Bunz, 93-111. Lüneburg: University of Minnesota Press.

<https://mediarep.org/handle/doc/12915>.

———. 2020. « Speeding up, slowing down, breaking down: an ethnography of software-driven mobility ». *Mobilities* 15 (5): 740-55.

<https://doi.org/10.1080/17450101.2020.1816035>.

Bijker, Wiebe E., Thomas Parke Hughes, et T. J. Pinch. 1987. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*.

- Cambridge, Massachusetts: MIT Press.
- Blanc, Soline, Abdelkader Lahmadi, Kévin Le Gouguec, Marine Minier, et Lama Sleem. 2022. « Benchmarking of Lightweight Cryptographic Algorithms for Wireless IoT Networks ». *Wireless Networks* 28 (8): 3453-76.
<https://doi.org/10.1007/s11276-022-03046-1>.
- Blanchette, Jean-François. 2012. *Burdens of Proof. Cryptographic Culture and Evidence Law of Electronic Documents*. Cambridge, Mass., London: MIT Press.
- Böck, Hanno. 2021. « Das Ende der alten PGP-Keyserver - ». News. *Golem* (blog). 21 juin 2021.
<https://www.golem.de/news/sks-das-ende-der-alten-pgp-keyserver-2106-157613.html>.
- Boltanski, Luc, et Laurent Thévenot. 1991. *De la justification. Les économies de la grandeur*. Paris: Gallimard.
- Borisov, Nikita, Ian Goldberg, et Eric Brewer. 2004. « Off-the-record communication, or, why not to use PGP ». In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, 77-84. WPES '04. Washington DC, USA: Association for Computing Machinery. <https://doi.org/10.1145/1029179.1029200>.
- Borradaile, Glencora, Kelsy Kretschmer, Michele Gretes, et Alexandria LeClerc. 2021. « The Motivated Can Encrypt (Even with PGP) ». *arXiv:2104.04478 [cs]*, avril.
<http://arxiv.org/abs/2104.04478>.
- Bosa, Bastien. 2011. « Ce qui change et le déjà fait ». *Revue européenne des sciences sociales. European Journal of Social Sciences*, n° 49-2 (décembre): 169-96.
<https://doi.org/10.4000/ress.1032>.
- Bourdieu, Pierre. 2001. *Langage et pouvoir symbolique*. Points. Paris.
- Bourrier, Mathilde, et Nicolas Nova. 2019. « (En)quêtes de pannes ». *Techniques Culture* n° 72 (2): 12-29.
- Bowker, Geoffrey, Karen Baker, Florence Millerand, et David Ribes. 2009. « Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment ». In *International Handbook of Internet Research*, édité par Jeremy Hunsinger, Lisbeth Klastrup, et Matthew Allen, 97-117. Dordrecht: Springer Netherlands.
https://doi.org/10.1007/978-1-4020-9789-8_5.
- Bowker, Geoffrey, et Susan Leigh Star. 2000. *Sorting Things Out: Classification and Its Consequences*. MIT Press.

- Boyd, Danah. 2008. « How can qualitative internet researchers define the boundaries of their projects: A response to Christine Hine ». *Internet inquiry: Conversations about method*, 26-32.
- Bozzini, David. 2016. « Cryptographie et surveillance digitale ». In *Opacités du patrimoine culturel immatériel*, par Bernard Knodel, Marc-Olivier Gonseth, Yann Laville, Grégoire Mayor, et Olivier Schinz, 276-87. Neuchâtel: Musée d'ethnographie.
- . 2017. « Penser les questions de sécurité ». *Schweizer Volkskunde* 107 (2): 40-44.
- . 2023. « How Vulnerabilities Became Commodities. The Political Economy of Ethical Hacking (1990-2020) », 2023. <https://hal.science/hal-04068476>.
- Brandom, Russell. 2014. « New Documents Reveal Which Encryption Tools the NSA Couldn't Crack ». *The Verge* (blog). 28 décembre 2014. <https://www.theverge.com/2014/12/28/7458159/encryption-standards-the-nsa-cant-crack-pgp-tor-otr-snowden>.
- Broca, Sébastien. 2018. *Utopie du logiciel libre : du bricolage informatique à la réinvention sociale*. Neuilly-en-Champagne: Le Passager clandestin.
- Brugère, Fabienne. 2017. *L'éthique du « care »*. Que sais-je? : le point des connaissances actuelles. Paris: Presses universitaires de France.
- Bruun, Maja Hojer, Astrid Oberborbeck Andersen, et Adrienne Mannov. 2020. « Infrastructures of Trust and Distrust: The Politics and Ethics of Emerging Cryptographic Technologies ». *Anthropology Today* 36 (2): 13-17. <https://doi.org/10.1111/1467-8322.12562>.
- Buier, Natalia. 2023. « The Anthropology of Infrastructure: The Boom and the Bubble? » *Focaal* 2023 (95): 46-60. <https://doi.org/10.3167/fcl.2022.012401>.
- Buzan, Barry, Jaap De Wilde, et Ole Waever. 1998. *Security: A New Framework for Analysis*. Boulder CO etc.: Lynne Rienner.
- Callon, Michel. 1986. « Éléments pour une sociologie de la traduction. La domestication des coquilles Saint-Jacques et des marins-pêcheurs dans la baie de Saint-Brieuc ». *L'Année sociologique (1940/1948-)* 36: 169-208.
- . 1994. « L'innovation technologique et ses mythes ». *Gérer & Comprendre*, n° 34: 5-17.
- . 2006. « Pour une sociologie des controverses technologiques ». In *Sociologie de la traduction: Textes fondateurs*, édité par Madeleine Akrich, Bruno Latour, et Michel Callon, 135-57. Sciences sociales. Paris: Presses des Mines. <http://books.openedition.org/pressesmines/1196>.

- Callon, Michel, Pierre Lascoumes, et Yannick Barthe. 2001. *Agir dans un monde incertain. Essai sur la démocratie technique*. La couleur des idées. Paris: Seuil.
http://data.rero.ch/01-R003095961/html?view=FR_V1.
- Cardon, Dominique. 2019. *Culture numérique*. Paris: Presses de Sciences Po.
<https://www.cairn.info/culture-numerique--9782724623659.htm?WT.tsrc=cairn-SearchAutocomplete>.
- Casilli, Antonio. 2005. « Les avatars bleus ». *Communications* 77 (1): 183-209.
<https://doi.org/10.3406/comm.2005.2269>.
- Castells, Manuel. 2004. *The Internet Galaxy: Reflections on the Internet, Business, and Society*. [Reprint.]. Oxford: Oxford Univ. Press.
- Chateauraynaud, Francis, et Patrick Trabal. 2007. « Des vigiles invisibles: les administrateurs réseaux et la sécurité informatique ». *Annales Des Télécommunications* 62 (11): 1293-1311. <https://doi.org/10.1007/BF03253319>.
- Chauveau, Jean-Pierre. 1985. « Mise en valeur coloniale et développement ». In *Payans, experts et chercheurs en Afrique noire. Sciences sociales et développement rural*, par Pierre Boiral, Jean-François Lanteri, et Jean-Pierre Olivier de Sardan, 143-66. Paris: Karthala.
- Cogburn, Derrick L. 2016. « The Multiple Logics of Post-Snowden Restructuring of Internet Governance ». In *The Turn to Infrastructure in Internet Governance*, par Francesca Musiani, Derrick L. Cogburn, Laura DeNardis, et Nanette S. Levinson, 25-45. Houndmills: Palgrave Macmillan.
- Coleman, Gabriella. 2009. « CODE IS SPEECH: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers ». *Cultural Anthropology* 24 (3): 420-54. <https://doi.org/10.1111/j.1548-1360.2009.01036.x>.
- . 2010. « The Hacker Conference: A Ritual Condensation and Celebration of a Lifeworld ». *Anthropological Quarterly* 83 (1): 47-72.
<https://doi.org/10.1353/anq.0.0112>.
- . 2013. *Coding Freedom: The Ethics and Aesthetics of Hacking*. Princeton: Princeton University Press.
- . 2015. *Hacker, Hoaxer, Whistleblower, Spy: The Many Faces of Anonymous*. First publ. in paperback. London: Verso books.
- Coleman, Gabriella, et Alex Golub. 2008. « Hacker Practice: Moral Genres and the Cultural Articulation of Liberalism ». *Anthropological Theory* 8 (3): 255-77.
<https://doi.org/10.1177/1463499608093814>.

- Collet, Isabelle. 2019. *Les oubliées du numérique : [l'absence des femmes dans le monde digital n'est pas une fatalité]*. Paris: Le Passeur.
- Collet, Isabelle, et Nicole Mosconi. 2010. « Les informaticiennes: de la dominance de classe aux discriminations de sexe ? » *Nouvelles Questions Feministes* Vol. 29 (2): 100-113.
- Collier, Stephen J., et Andrew Lakoff. 2015. « Vital systems security: Reflexive biopolitics and the government of emergency ». *Theory, culture & society* 32 (2): 19-51.
- Collins, Keith. 2016. « How One Programmer Broke the Internet by Deleting a Tiny Piece of Code ». Quartz. 27 mars 2016.
<https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/>.
- Couture, Stéphane. 2012. « L'écriture collective du code source informatique ». *Revue d'anthropologie des connaissances* Vol. 6, n° 1 (1): 21-42.
- Cross, Mai'a K. Davis. 2013. « Rethinking Epistemic Communities Twenty Years Later ». *Review of International Studies* 39 (1): 137-60.
<https://doi.org/10.1017/S0260210512000034>.
- Dakhli, Jocelyne. 1995. « Le terrain de la vérité ». *Enquête. Archives de la revue Enquête*, n° 1 (octobre): 141-52. <https://doi.org/10.4000/enquete.270>.
- De Seta, Gabriele. 2020. « Three Lies of Digital Ethnography ». *Journal of Digital Social Research* 2 (1): 77-97. <https://doi.org/10.33621/jdsr.v2i1.24>.
- Dechand, Sergej, Alena Naiakshina, Anastasia Danilova, et Matthew Smith. 2019. « In Encryption We Don't Trust: The Effect of End-to-End Encryption to the Masses on User Perception ». In *2019 IEEE European Symposium on Security and Privacy (EuroSP)*, 401-15. <https://doi.org/10.1109/EuroSP.2019.00037>.
- Demazière, Didier, François Horn, et Marc Zune. 2011. « Ethnographie de terrain et relation d'enquête. Observer les « communautés » de logiciels libres ». *Sociologie* 2 (2): 165-83.
- DeNardis, Laura. 2006. « IPv6: Politics of the next generation internet ». PhD Thesis, Virginia Tech.
- . 2022. « Quantum Internet Protocols ». SSRN Scholarly Paper. Rochester, NY.
<https://doi.org/10.2139/ssrn.4182865>.
- DeNardis, Laura, et Francesca Musiani. 2016. « Governance by Infrastructure ». In *The Turn to Infrastructure in Internet Governance*, édité par Francesca Musiani, Derrick L. Cogburn, Laura DeNardis, et Nanette S. Levinson, 3-21. Information Technology

- and Global Governance. New York: Palgrave Macmillan US.
https://doi.org/10.1057/9781137483591_1.
- Denis, Jérôme. 2009. « Les ressorts de la sécurité informatique. Des hommes, des machines et des données ». In *L'évolution des cultures numériques, de la mutation du lien social à l'organisation du travail*, par Christian Licoppe, FYP, 190-99. Paris.
- . 2012. « L'informatique et sa sécurité ». *Réseaux* n° 171 (1): 161-87.
- Denis, Jérôme, Alessandro Mongili, et David Pontille. 2015. « Maintenance & Repair in Science and Technology Studies ». *TECNOSCIENZA: Italian Journal of Science & Technology Studies* 6 (2): 5-16.
- Denis, Jérôme, et David Pontille. 2010. *Petite sociologie de la signalétique: les coulisses des panneaux du métro*. Paris: Presses des Mines.
<https://doi.org/10.4000/books.pressesmines.1155>.
- . 2022. *Le soin des choses. Politiques de la maintenance*. Paris: La Découverte.
- Di Salvo, Philip. 2020. *Digital Whistleblowing Platforms in Journalism: Encrypting Leaks*. Cham: Springer. <https://doi.org/10.1007/978-3-030-38505-7>.
- Diffie, Whitfield, et Martin E. Hellman. 1976. « New directions in cryptography ». *IEEE Transactions on Information Theory* 22 (6): 644-54.
<https://doi.org/10.1109/TIT.1976.1055638>.
- Diffie, Whitfield, et Susan Landau. 2010. *Privacy on the Line*. Cambridge: The MIT Press.
- Dousset, Laurent. 2019. « De l'incertitude au risque: un outil heuristique ». *Moussons. Recherche en sciences humaines sur l'Asie du Sud-Est*, n° 34 (novembre): 21-48.
<https://doi.org/10.4000/moussons.5238>.
- Dunn Cavelty, Myriam. 2007. *Cyber-security and threat politics: US efforts to secure the information age*. Routledge.
- . 2014. *Cybersecurity in Switzerland*. SpringerBriefs in Cybersecurity. Springer International Publishing. <https://doi.org/10.1007/978-3-319-10620-5>.
- Edgerton, David. 2008. *The Shock of the Old: Technology and Global History since 1900*. London: Profile.
- Ermoshina, Ksenia. 2023. « “Voices from the Island”: Informational Annexation of Crimea and Transformations of Journalistic Practices ». *Journalism*, janvier, 14648849231152359. <https://doi.org/10.1177/14648849231152359>.
- Ermoshina, Ksenia, et Francesca Musiani. 2019. « “Standardising by running code”: the Signal protocol and de facto standardisation in end-to-end encrypted messa-

- ging ». *Internet Histories* 3 (3-4): 343-63.
- . 2022. *Concealing for Freedom: The Making of Encryption, Secure Messaging and Digital Liberties*. Manchester: Mattering Press.
- Ermoshina, Ksenia, Francesca Musiani, et Harry Halpin. 2016. « End-to-end encrypted messaging protocols: An overview ». In *International Conference on Internet Science*, 244-54. Florence: Springer.
- Fabian, Johannes. 2017. *Le temps des autres. Comment l'anthropologie construit son objet*. Traduit par Estelle Henry-Bossonney et Bernard Müller. Toulouse: Anacharsis.
- Falzon, Mark-Anthony. 2016. « Introduction. Multi-sited ethnography: Theory, praxis and locality in contemporary research ». In *Multi-sited ethnography: Theory, praxis and locality in contemporary research*, édité par Mark-Anthony Falzon, 1-23. New York: Routledge.
- Faubion, James D., et George E. Marcus, éd. 2009. *Fieldwork is not what it used to be: learning anthropology's method in a time of transition*. Cornell paperbacks. Ithaca: Cornell University Press.
- Feathers, Michael. 2005. *Working Effectively with Legacy Code*. Prentice Hall Professional.
- Florentin, Daniel. 2018. « La bifurcation infrastructurelle ». *Revue européenne des sciences sociales. European Journal of Social Sciences*, n° 56-1 (mai): 241-62. <https://doi.org/10.4000/ress.4060>.
- Foucault, Michel. 1994a. *Dits et écrits. Tome 3, 1976-1979*. Paris: Éditions Gallimard.
- . 1994b. *Dits et écrits. Tome 4. 1980-1988*. Paris: Éditions Gallimard.
- Franceschi-Bicchierai, Lorenzo. 2015. « Even the Inventor of PGP Doesn't Use PGP ». *Vice* (blog). 2 septembre 2015. <https://www.vice.com/en/article/vvbw9a/even-the-inventor-of-pgp-doesnt-use-pgp>.
- . 2018. « People Are Freaking Out That PGP Is 'Broken'—But You Shouldn't Be Using It Anyway ». *Vice* (blog). 14 mai 2018. <https://www.vice.com/en/article/3k4nd9/pgp-gpg-efail-vulnerability>.
- Frei, Stefan, Bernhard Tellenbach, et Bernhard Plattner. 2008. « 0-Day Patch Exposing Vendors (In)security Performance ». *Black Hat Europe, London*. <http://www.blackhat.com/presentations/bh-europe-08/Frei/Whitepaper/bh-eu-08-frei-WP.pdf>.

- Gaglio, Gérald. 2011. *Sociologie de l'innovation*. Que sais-je? Paris: Presses Universitaires de France.
- Gallagher, Sean. 2016. « Rage-Quit: Coder Unpublished 17 Lines of JavaScript and “Broke the Internet” ». *Ars Technica*. 25 mars 2016.
<https://arstechnica.com/information-technology/2016/03/rage-quit-coder-unpublished-17-lines-of-javascript-and-broke-the-internet/>.
- Garfinkel, Simson. 1995. *PGP: Pretty Good Privacy. Encryption for Everyone*. Beijing, Cambridge, etc.: O'Reilly Media, Inc.
- Garfinkel, Simson L., et Robert C. Miller. 2005. « Johnny 2: a user test of key continuity management with S/MIME and Outlook Express ». In *Proceedings of the 2005 symposium on Usable privacy and security*, 13-24. SOUPS '05. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/1073001.1073003>.
- Geertz, Clifford. 1973. *The interpretation of cultures: selected essays*. New York: Basic Books.
- Gieter, Jon, réal. 2017. *Why Johnny STILL Can't Encrypt*. San Francisco: RSA Conference 2017. <https://www.youtube.com/watch?v=KuAZFNRCb4E>.
- Gilligan, Carol. 2017. *Une voix différente : pour une éthique du « care »*. Champs. Essais. Paris: Flammarion.
- Giraud, Pierre-Noël. 2001. *Le commerce des promesses. Petit traité sur la finance moderne*. Paris: Seuil.
- Glaser, Barney G., et Anselm L. Strauss. 2010. *La découverte de la théorie ancrée: Stratégies pour la recherche qualitative*. Paris: Armand Colin.
- Goerzen, Matt, et Gabriella Coleman. 2022. « Wearing Many Hats. The Rise of the Professional Security Hacker ». *Data & Society*.
<https://datasociety.net/library/wearing-many-hats-the-rise-of-the-professional-security-hacker/>.
- Goldstein, Daniel M. 2010. « Toward a Critical Anthropology of Security ». *Current Anthropology* 51 (4): 487-517. <https://doi.org/10.1086/655393>.
- Graham, Stephen, et Nigel Thrift. 2007. « Out of order: understanding repair and maintenance. » *Theory, culture & society*. 24 (mai): 1-25.
<https://dx.doi.org/10.1177/0263276407075954>.
- Green, Matthew. 2014. « What's the Matter with PGP? » Personal blog. *A Few Thoughts on Cryptographic Engineering* (blog). 13 août 2014.
<https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/>.

- . 2018. « Was the Efail Disclosure Horribly Screwed Up? » Personal blog. *A Few Thoughts on Cryptographic Engineering* (blog). 17 mai 2018.
<https://blog.cryptographyengineering.com/2018/05/17/was-the-efail-disclosure-horribly-screwed-up/>.
- Greenberg, Andy. 2012. *This Machine Kills Secrets: How WikiLeaks, Cypherpunks, and Hacktivists Aim to Free the World's Information*. New York: Dutton.
- Gregson, Nicky, Alan Metcalfe, et Louise Crewe. 2009. « Practices of Object Maintenance and Repair: How Consumers Attend to Consumer Objects within the Home ». *Journal of Consumer Culture* 9 (2): 248-72.
<https://doi.org/10.1177/1469540509104376>.
- Gürses, Seda, Arun Kundnani, et Joris Van Hoboken. 2016. « Crypto and Empire: The Contradictions of Counter-Surveillance Advocacy ». *Media, Culture & Society*, 576-90. <https://doi.org/10.1177/0163443716643006>.
- Halpin, Harry. 2020. « SoK: Why Johnny Can't Fix PGP Standardization ». *Proceedings of the 15th International Conference on Availability, Reliability and Security*, août, 1-6. <https://doi.org/10.1145/3407023.3407083>.
- Hansen, Lene, et Helen Nissenbaum. 2009. « Digital Disaster, Cyber Security, and the Copenhagen School ». *International Studies Quarterly* 53 (4): 1155-75.
- Hansen, Robert J. 2019. « SKS Keyserver Network Under Attack ». *GitHub Gist* (blog). 2019.
<https://web.archive.org/web/20220605071047/https://gist.github.com/rjhansen/67ab921ffb4084c865b3618d6955275f>.
- Hanseth, Ole, Eric Monteiro, et Morten Hatling. 1996. « Developing Information Infrastructure: The Tension Between Standardization and Flexibility ». *Science, Technology, & Human Values* 21 (4): 407-26. <https://doi.org/10.1177/016224399602100402>.
- Hardy, Paul. 2021. *Refactoring Legacy ABAP Code*.
- Haworth, Jessica. 2018. « Signal Desktop Gatecrashed by Web Vulnerabilities ». Infosec news blog. *The Daily Swig | Cybersecurity News and Views* (blog). 17 mai 2018.
<https://portswigger.net/daily-swig/signal-desktop-gatecrashed-by-web-vulnerabilities>.
- Hellegren, Isadora. 2017. « A history of crypto-discourse: encryption as a site of struggles to define internet freedom ». *Internet Histories* 1 (4): 285-311.
<https://doi.org/10.1080/24701475.2017.1387466>.
- . 2020. « Crypto-Discourse, Internet Freedom, and the State ». Oxford Research Encyclopedia of Communication. 2020.

- <https://doi.org/10.1093/acrefore/9780190228613.013.887>.
- Henke, Christopher R. 2000. « The Mechanics of Workplace Order: Toward a Sociology of Repair ». *Berkeley Journal of Sociology* 44: 55-81.
- Hine, Christine. 2000. *Virtual ethnography*. London: Sage.
- Hintz, Arne, Lina Dencik, et Karin Wahl-Jorgensen. 2017. « Digital Citizenship and Surveillance | Digital Citizenship and Surveillance Society — Introduction ». *International Journal of Communication* 11 (0): 9.
- Hummel, Cornelia, et David Desaleux. 2018. *Mustang. La mécanique de la passion*. Lyon: Fablyo.
- Jackson, Steven J. 2014. « Rethinking Repair ». In *Media technologies: Essays on communication, materiality, and society*, édité par Tarleton Gillespie, Pablo J. Boczkowski, et Kirsten A. Foot, 221-39. Cambridge, MA: MIT Press.
- Joniak-Lüthi, Agnieszka. 2019. « Introduction: Infrastructure as an Asynchronic Timescape ». *Roadsides*, n° 1: 3-10. <https://roadsides.net/joniak-001/>.
- Kahn, David. 1996. *The Codebreakers: The Story of Secret Writing*. [Revised and Updated ed.]. New York: Scribner.
- Katwala, Amit. 2018. « We're Calling It: PGP Is Dead ». *Wired UK*, 17 mai 2018. <https://www.wired.co.uk/article/efail-gpg-vulnerability-outlook-thunderbird-smime>.
- Katz, Jonathan. 2008. « Bridging Game Theory and Cryptography: Recent Results and Future Directions ». In *Theory of Cryptography*, édité par Ran Canetti, 4948:251-72. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78524-8_15.
- Kazansky, Becky. 2021. « 'It Depends on Your Threat Model': The Anticipatory Dimensions of Resistance to Data-Driven Surveillance ». *Big Data & Society* 8 (1): 2053951720985557. <https://doi.org/10.1177/2053951720985557>.
- Keck, Frédéric. 2010. « Une sentinelle sanitaire aux frontières du vivant ». *Terrain. Anthropologie & sciences humaines*, n° 54 (mars): 26-41. <https://doi.org/10.4000/terrain.13928>.
- Kelty, Christopher M. 2008. *Two Bits: The Cultural Significance of Free Software*. Experimental Futures. Durham, N.C.: Duke University Press.
- Khandelwal, Swati. 2018a. « Severe Bug Discovered in Signal Messaging App for Windows and Linux ». Infosec news blog. *The Hacker News* (blog). 11 mai 2018. <https://thehackernews.com/2018/05/signal-messenger-vulnerability.html>.

- . 2018b. « Another Severe Flaw in Signal Desktop App Lets Hackers Steal Your Chats in Plaintext ». Infosec news blog. *The Hacker News* (blog). 16 mai 2018. <https://thehackernews.com/2018/05/signal-desktop-hacking.html>.
- Knorr-Cetina, Karin. 1995. « Laboratory Studies: The Cultural Approach to the Study of Science ». In *Handbook of Science and Technology Studies*, par Sheila Jasanoff, Gerald Markle, James Peterson, et Trevor Pinch, 140-66. Thousand Oaks: Sage. <https://doi.org/10.4135/9781412990127.n7>.
- Knorr-Cetina, Karin, et Michael Mulkay, éd. 1983. *Science Observed. Perspectives on the Social Study of Science*. London: SAGE.
- Kocher, Paul, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, et al. 2019. « Spectre Attacks: Exploiting Speculative Execution ». In *40th IEEE Symposium on Security and Privacy (S&P'19)*.
- Kocksch, Laura, Matthias Korn, Andreas Poller, et Susann Wagenknecht. 2018. « Caring for IT Security: Accountabilities, Moralities, and Oscillations in IT Security Practices ». *Proceedings of the ACM on Human-Computer Interaction 2* (CSCW): 92:1-92:20. <https://doi.org/10.1145/3274361>.
- Kol, Gillat, et Moni Naor. 2008. « Cryptography and Game Theory: Designing Protocols for Exchanging Information ». In *Theory of Cryptography*, édité par Ran Canetti, 4948:320-39. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78524-8_18.
- Korn, Matthias, et Susann Wagenknecht. 2017. « Friction in Arenas of Repair: Hacking, Security Research, and Mobile Phone Infrastructure ». In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2475-88. CSCW '17. Portland, Oregon, USA: Association for Computing Machinery. <https://doi.org/10.1145/2998181.2998308>.
- Kuhn, Christiane, Martin Beck, Stefan Schiffner, Eduard Jorswieck, et Thorsten Strufe. 2019. « On privacy notions in anonymous communication ». *Proceedings on Privacy Enhancing Technologies 2019* (2): 105-25. <https://doi.org/10.48550/arXiv.1812.05638>.
- Lagesen, Vivian Anette. 2008. « A Cyberfeminist Utopia?: Perceptions of Gender and Computer Science among Malaysian Women Computer Science Students and Faculty ». *Science, Technology, & Human Values* 33 (1): 5-27. <https://doi.org/10.1177/0162243907306192>.
- Lakoff, Andrew. 2017. *Unprepared: Global Health in a Time of Emergency*. University of California Press. <https://www.jstor.org/stable/10.1525/j.ctt1rfsrwc>.

- Lallement, Michel. 2015. *L'âge du faire : hacking, travail, anarchie*. Couleur des idées. Paris: Seuil.
- Laplantine, François. 2023. *Chronotopie*. Dépayage.
- Larkin, Brian. 2013. « The Politics and Poetics of Infrastructure ». *Annual Review of Anthropology* 42 (1): 327-43. <https://doi.org/10.1146/annurev-anthro-092412-155522>.
- Lascoumes, Pierre. 2004. « La Gouvernamentalité : de la critique de l'État aux technologies du pouvoir ». *Le Portique. Revue de philosophie et de sciences humaines*, n° 13-14 (septembre). <http://journals.openedition.org/leportique/625>.
- Latacora. 2019. « The PGP Problem ». Latacora's Blog. 16 juillet 2019. <https://latacora.micro.blog/2019/07/16/the-pgp-problem.html>.
- Latour, Bruno. 2007. *Changer de société, refaire de la sociologie*. Paris: La Découverte.
- Latour, Bruno, et Steve Wooglar. 1988. *La vie de laboratoire : la production des faits scientifiques*. Sciences et société. La Découverte. Paris: La Découverte. http://data.rero.ch/01-1040149/html?view=FR_V1.
- Latzko-Toth, Guillaume, et Florence Millerand. 2015. « Objet-frontière ». In *Sciences, technologies et sociétés de A à Z*, par Julien Prud'homme, Pierre Doray, et Frédéric Bouchard, 163-68. Montréal: Presses de l'Université de Montréal.
- Lave, Jean, et Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.
- Law, John, et John Hassard, éd. 1999. *Actor Network Theory and After*. Sociological Review Monograph. Oxford etc.: Blackwell. http://data.rero.ch/01-R242162460/html?view=FR_V1.
- Lazaro, Christophe. 2008. *La liberté logicielle : une ethnographie des pratiques d'échange et de coopération au sein de la communauté Debian*. Anthropologie prospective. Louvain-La-Neuve: Bruylant-Academia.
- Le Breton, David. 2008. *L'interactionnisme symbolique*. Quadrige. Manuels. Paris: Presses universitaires de France.
- Lee, Micah. 2019. « The Death of SKS PGP Keyservers, and How First Look Media is Handling It ». *First Look Media* (blog). 20 août 2019. <https://web.archive.org/web/20220605071047/https://code.firstlook.media/the-death-of-sks-pgp-keyservers-and-how-first-look-media-is-handling-it>.
- Lemieux, Cyril. 2007. « À quoi sert l'analyse des controverses ? » *Mil neuf cent. Revue d'histoire intellectuelle* n° 25 (1): 191-212.

- Leurent, Gaëtan, et Thomas Peyrin. 2020. « SHA-1 is a Shambles. First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust ». In *29th USENIX Security Symposium (USENIX Security 20)*, 1839-56. <https://sha-mbles.github.io/>.
- Levy, Steven. 2001. *Crypto: How the Code Rebels Beat the Government Saving Privacy in the Digital Age*. 1st edition. New York: Penguin Books.
- . 2013. *L'éthique des hackers*. Paris: Globe.
- Lipp, Moritz, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, et al. 2018. « Meltdown: Reading Kernel Memory from User Space ». In *27th USENIX Security Symposium (USENIX Security 18)*.
- Lopian, Eli. 2018. « Defining Legacy Code ». Blog. Dzone.Com. 15 mai 2018. <https://dzone.com/articles/defining-legacy-code>.
- Lübbe, Hermann. 2009. « The Contraction of the Present ». In *High-Speed Society. Social Acceleration, Power, and Modernity*, édité par Hartmut Rosa et William Scheuerman, 159-78. University Park, Pa: Pennsylvania State University Press.
- Lyon, David. 2018. *The culture of surveillance: Watching as a way of life*. Cambridge: John Wiley & Sons.
- Macleod, Alex. 2004. « Les études de sécurité: du constructivisme dominant au constructivisme critique ». *Cultures et Conflits*, n° 54: 13-51.
- Maguire, Mark, Catarina Fróis, et Nils Zurawski. 2014. *The Anthropology of Security: Perspectives from the Frontline of Policing, Counter-Terrorism and Border Control*. Anthropology, Culture and Society. London: Pluto Press.
- Marcus, George E. 1995. « Ethnography in/of the World System: The Emergence of Multi-Sited Ethnography ». *Annual Review of Anthropology* 24: 95-117.
- Marlinspike, Moxie. 2016. « Reflections: The Ecosystem Is Moving ». Software blog. Signal Messenger. 10 mai 2016. <https://signal.org/blog/the-ecosystem-is-moving/>.
- , réal. 2019. *The Ecosystem Is Moving. Challenges for Distributed and Decentralized Technology from the Perspective of Signal Development*. https://media.ccc.de/v/36c3-11086-the_ecosystem_is_moving.
- Massey, Doreen B. 2005. *For Space*. London: Sage.
- Merton, Robert K. 1942. « Science and technology in a democratic order ». *Journal of legal and political sociology* 1 (1): 115-26.
- Michael, Mike. 2017. *Actor-Network Theory: Trials, Trails and Translations*. London, Thousand Oaks, etc.: SAGE.

- Mol, Annemarie. 2009. *Ce que soigner veut dire : repenser le libre choix du patient*. Collection Sciences sociales. Paris: Mines ParisTech : Presses des Mines.
- Mol, Annemarie, Ingunn Moser, et Jeannette Pols. 2010. *Care in Practice: On Tinkering in Clinics, Homes and Farms*. Transcript Verlag. Bielefeld.
- Morozov, Evgeny. 2014. *Pour tout résoudre, cliquez ici : l'aberration du solutionnisme technologique*. Innovation. Limoges: Fyp éd.
- Müller, Jens, Marcus Brinkmann, Damian Poddebniak, Hanno Böck, Sebastian Schinzel, Juraj Somorovsky, et Jörg Schwenk. 2019. « "Johnny, you are fired!" –Spoofing OpenPGP and S/MIME Signatures in Emails ». In *28th USENIX Security Symposium*, 1011-28.
- Müller, Jens, Marcus Brinkmann, Damian Poddebniak, Sebastian Schinzel, et Jörg Schwenk. 2019. « Re: What's Up Johnny? Covert Content Attacks on Email End-to-End Encryption ». In *Applied Cryptography and Network Security*, édité par Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, et Moti Yung, 24-42. Lecture Notes in Computer Science. Cham: Springer International Publishing.
https://doi.org/10.1007/978-3-030-21568-2_2.
- . 2020. « Mailto: Me Your Secrets. On Bugs and Features in Email End-to-End Encryption ». In *2020 IEEE Conference on Communications and Network Security (CNS)*, 1-9. <https://doi.org/10.1109/CNS48642.2020.9162218>.
- Musiani, Francesca. 2012. « Caring about the Plumbing: On the Importance of Architectures in Social Studies of (Peer-to-Peer) Technology ». *The Journal of Peer Production* (blog). 2012. <http://peerproduction.net/issues/issue-1/peer-reviewed-papers/caring-about-the-plumbing/>.
- . 2015. « Practice, Plurality, Performativity, and Plumbing: Internet Governance Research Meets Science and Technology Studies ». *Science, Technology, & Human Values* 40 (2): 272-86. <https://doi.org/10.1177/0162243914553803>.
- . 2018. « L'invisible qui façonne. Études d'infrastructure et gouvernance d'Internet ». *Tracés. Revue de Sciences humaines*, n° 35 (décembre): 161-76.
<https://doi.org/10.4000/traces.8419>.
- Musiani, Francesca, Derrick L. Cogburn, Laura DeNardis, et Nanette S. Levinson. 2016. *The turn to infrastructure in Internet governance*. Houndmills: Palgrave Macmillan.
- Musiani, Francesca, et Ksenia Ermoshina. 2017. « What is a good secure messaging tool? The EFF secure messaging scorecard and the shaping of digital (usable) security ». *Westminster Papers in Communication and Culture* 12 (3): 51-71.

- Narayanan, Arvind. 2013a. « What happened to the crypto dream?, part 1 ». *IEEE security & privacy* 11 (2): 75-76.
- . 2013b. « What happened to the crypto dream?, part 2 ». *IEEE security & privacy* 11 (3): 68-71.
- Nickolls, Ben. 2017. « Sustain. A one day conversation for open source software sustainers. The report. GitHub HQ ». <https://sustainoss.org/media/#reports>.
- Nissenbaum, Helen. 2005. « Where Computer Security Meets National Security ». *Ethics and Information Technology* 7 (2): 61-73.
<https://doi.org/10.1007/s10676-005-4582-3>.
- Nova, Nicolas, et Anaïs Bloch. 2020. *Dr. Smartphones: an ethnography of mobile phone repair shops*. Lausanne: IDP.
- Olivier de Sardan, Jean-Pierre. 2008. *La rigueur du qualitatif: les contraintes empiriques de l'interprétation socio-anthropologique*. Anthropologie prospective, no 3. Louvain-la-Neuve: Academia-Bruylant.
- Olivier de Sardan, Jean-Pierre. 1996. « La violence faite aux données ». *Enquête. Archives de la revue Enquête*, n° 3 (novembre): 31-59.
<https://doi.org/10.4000/enquete.363>.
- Paquin, Christian, Douglas Stebila, et Goutam Tamvada. 2020. « Benchmarking Post-Quantum Cryptography in TLS ». In *Post-Quantum Cryptography*, édité par Jintai Ding et Jean-Pierre Tillich, 72-91. Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-44223-1_5.
- Pastinelli, Madeleine. 2011. « Pour en finir avec l'ethnographie du virtuel ! Des enjeux méthodologiques de l'enquête de terrain en ligne ». *Anthropologie et Sociétés* 35 (1-2): 35-52. <https://doi.org/10.7202/1006367ar>.
- Perrin, Julie, Nolwenn Bühler, Marc-Antoine Berthod, Jérémie Forney, Sabine Kradolfer Morales, et Laurence Ossipow-Wuest. 2020. « En quête d'éthique: Dispositions légales et enjeux empiriques pour l'anthropologie ». *TSANTSA – Journal of the Swiss Anthropological Association* 25 (septembre): 225-67.
<https://doi.org/10.36950/tsantsa.2020.025.30>.
- Pink, Sarah. 2015. *Doing Sensory Ethnography*. SAGE.
- Poddebniak, Damian, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, et Jörg Schwenk. 2018. « Efail: Breaking S/MIME and OpenPGP Email Encryption Using Exfiltration Channels ». In *27th USENIX Security Symposium*, 549-66. Baltimore. <https://www.usenix.org/node/217635>.

- Prud'homme, Julien, Pierre Doray, et Frédéric Bouchard, éd. 2015. *Sciences, technologies et sociétés de A à Z*. Montréal CA: Les Presses de l'Université de Montréal.
https://www.pum.umontreal.ca/fichiers/livres_fichiers/9782760634961.pdf.
- Ptacek, Thomas H. 2018. « A unified timeline of Efail PGP disclosure events ». 2018.
<https://web.archive.org/web/20210423014441/https://flaked.sockpuppet.org/a-unified-timeline/>.
- Puig de la Bellacasa, Maria. 2010. « Matters of Care in Technoscience: Assembling Neglected Things »: *Social Studies of Science*, décembre.
<https://doi.org/10.1177/0306312710380301>.
- Raymond, Eric S. 2001. *The Cathedral and the Bazaar Musings on Linux and Open Source by an Accidental Revolutionary*. [Rev. ed.]. Beijing: O'Reilly.
- Raynaud, Dominique. 2018. *Sociologie des controverses scientifiques. De la philosophie des sciences*. Éditions Matériologiques.
- Rivest, R. L., A. Shamir, et L. Adleman. 1978. « A method for obtaining digital signatures and public-key cryptosystems ». *Communications of the ACM* 21 (2): 120-26.
<https://doi.org/10.1145/359340.359342>.
- Roitman, Janet Lee. 2014. *Anti-Crisis*. Durham: Duke University Press.
- Rubio, Fernando Domínguez. 2020. *Still Life: Ecologies of the Modern Imagination at the Art Museum*. Chicago, IL: University of Chicago Press.
<https://press.uchicago.edu/ucp/books/book/chicago/S/bo51203276.html>.
- Ruh, Nicolai. 2018. « Trusting the Math and Mistrusting Humans. How Politically Sensitized Engineers of Cryptographic Systems Cope with Ontological Insecurity in the Digitally Augmented Life-World ». In *Mistrust. Ethnographic Approximations*, par Florian Mühlfried, 23-48. Bielefeld: transcript Verlag.
- Ruoti, Scott, Jeff Andersen, Daniel Zappala, et Kent Seamons. 2016. « Why Johnny Still, Still Can't Encrypt: Evaluating the Usability of a Modern PGP Client ». arXiv.
<https://doi.org/10.48550/arXiv.1510.08555>.
- Schiller, Katharina, et Florian Adamsky. 2022. « Work in Progress: Can Johnny Encrypt E-Mails on Smartphones? » In *Socio-Technical Aspects in Security*, édité par Simon Parkin et Luca Viganò, 13176:182-93. Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-10183-0_9.
- Schinzel, Sebastian. 2018. « Attacking end-to-end email encryption ». Conference website. Media.ccc.de. 28 décembre 2018.
https://media.ccc.de/v/35c3-9463-attacking_end-to-end_email_encryption.

- Schneier, Bruce. 1997. « Why cryptography is harder than it looks ». In *EDI FORUM-OAK PARK-*, 10:87-90. THE EDI GROUP, LTD.
- . 2007. « Full Disclosure of Security Vulnerabilities a “Damned Good Idea” - Schneier on Security ». Personal blog. Schneier on Security. janvier 2007.
https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html.
- . 2018. « Essays: What “Efail” Tells Us About Email Vulnerabilities and Disclosure - Schneier on Security ». Personal blog. Schneier on Security. 2018.
https://www.schneier.com/essays/archives/2018/05/what_efail_tells_us_.html.
- Schwenk, Jörg, Marcus Brinkmann, Damian Poddebniak, Jens Müller, Juraj Somorovsky, et Sebastian Schinzel. 2020. « Mitigation of Attacks on Email End-to-End Encryption ». In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 1647-64. CCS '20. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3372297.3417878>.
- Serres, Michel. 2012. *Temps des crises*. Paris: Le pommier.
- Sheng, Steve, Levi Broderick, Colleen Alison Koranda, et Jeremy J. Hyland. 2006. « Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software ». *Symposium On Usable Privacy and Security*, 2.
- Shor, Peter W. 1997. « Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer ». *SIAM Journal on Computing* 26 (5): 1484-1509. <https://doi.org/10.1137/S0097539795293172>.
- Singh, Simon. 1999. *Histoire des codes secrets : de l'Égypte des Pharaons à l'ordinateur quantique*. Paris: J.-C. Lattès.
- Sousedek, Jan. 2008. « Why Johnny Can't Encrypt: A Usability Study of PGP ». Berlin: Technische Universität Berlin, Germany.
https://www.net.t-labs.tu-berlin.de/teaching/ss08/IS_seminar/PDF/C.1.pdf.
- Spolsky, Joel. 2000. « Things You Should Never Do, Part I ». Blog. *Joel on Software* (blog). 6 avril 2000.
<https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/>.
- Spring, Tom. 2019. « PGP Ecosystem Targeted in ‘Poisoning’ Attacks ». Specialized Security News Website. 5 juillet 2019.
<https://threatpost.com/pgp-ecosystem-targeted-in-poisoning-attacks/146240/>.
- Star, Susan Leigh. 1999. « The Ethnography of Infrastructure ». *American Behavioral Scientist* 43 (3): 377-91. <https://doi.org/10.1177/00027649921955326>.

- Star, Susan Leigh, et James R. Griesemer. 1989. « Institutional Ecology, `Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39 ». *Social Studies of Science* 19 (3): 387-420.
<https://doi.org/10.1177/030631289019003001>.
- Star, Susan Leigh, et Karen Ruhleder. 2010. « Vers une écologie de l'infrastructure ». *Revue d'anthropologie des connaissances* Vol 4, n° 1 (1): 114-61.
- Stevens, Clare. 2020. « Assembling cybersecurity: The politics and materiality of technical malware reports and the case of Stuxnet ». *Contemporary Security Policy* 41 (1): 129-52. <https://doi.org/10.1080/13523260.2019.1675258>.
- Stritzel, Holger. 2016. « Towards a Theory of Securitization: Copenhagen and Beyond ». *European Journal of International Relations*, juillet.
<https://doi.org/10.1177/1354066107080128>.
- Tong, Wenley, Sebastian Gold, Samuel Gichohi, Mihai Roman, et Jonathan Frankle. 2014. « Why King George III Can Encrypt ». *Freedom to Tinker*, 13.
- Tréguer, Félix. 2015. « Hackers vs States: Subversion, Repression and Resistance in the Online Public Sphere ». *Droit et société* N° 91 (3): 639-52.
- Tronto, Joan C. 2009. *Un monde vulnérable: pour une politique du care*. Textes à l'appui. Philosophie pratique. Paris: La Découverte.
- Valentine, David, et Amelia Hassoun. 2019. « Uncommon Futures ». *Annual Review of Anthropology* 48 (1): 243-60.
<https://doi.org/10.1146/annurev-anthro-102218-011435>.
- Valsorda, Filippo. 2016. « Op-Ed: I'm Throwing in the Towel on PGP, and I Work in Security ». *Ars Technica* (blog). 10 décembre 2016.
<https://arstechnica.com/information-technology/2016/12/op-ed-im-giving-up-on-pgp/>.
- . 2021. « I'm giving up on PGP ». Personal blog. *words.filippo.io* (blog). 6 décembre 2021. <https://words.filippo.io/giving-up-on-long-term-pgp/>.
- Venturini, Tommaso. 2010. « Building on Faults: How to Represent Controversies with Digital Methods ». *Public Understanding of Science* 21 (7): 796-812.
<https://doi.org/10.1177/0963662510387558>.
- Vyas, Dhaval. 2013. « Ethnography of software maintenance activities in an industrial engineering setup ». In *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, 242-46. APCHI '13. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/2525194.2525269>.

- Walfield, Neal. 2019. « Hagrid: A New Verifying Key Server Built on Sequoia ». *Sequoia PGP* (blog). 14 juin 2019.
<https://sequoia-pgp.org/blog/2019/06/14/20190614-hagrid/>.
- . 2021. « Hey Signal! Great Encryption Needs Great Authentication ». *Sequoia PGP* (blog). 16 juin 2021.
<https://preview.sequoia-pgp.org/blog/2021/06/16/202106-hey-signal-great-encryption-needs-great-authentication/>.
- West, Sarah Myers. 2018. « Cryptographic imaginaries and the networked public ». *Internet Policy Review* 7 (2).
<https://policyreview.info/articles/analysis/cryptographic-imaginaries-and-networked-public>.
- Whitten, Alma, et J. D. Tygar. 1999. « Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0 ». In *8th USENIX Security Symposium (USENIX Security 99)*.
<https://www.usenix.org/conference/8th-usenix-security-symposium/why-johnny-cant-encrypt-usability-evaluation-pgp-50>.
- Wong, David. 2020. « A history of end-to-end encryption and the death of PGP ». *Cryptologie.net* (blog). janvier 2020.
<https://www.cryptologie.net/article/487/a-history-of-end-to-end-encryption-and-the-death-of-pgp/>.
- Yen, Andy. 2018. « PGP is not broken, not even with the Efail vulnerabilities ». Commercial blog. *Proton* (blog). 15 mai 2018.
<https://proton.me/blog/pgp-vulnerability-efail>.
- Zimmermann, Philip. 1999. « Why I Wrote PGP ». *philzimmermann.com* (blog). 1999.
<https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html>.
- Zuboff, Shoshana. 2020. *L'âge du capitalisme de surveillance le combat pour un avenir humain face aux nouvelles frontières du pouvoir*. Zulma essais. Paris: Zulma.