An Architecture for Attesting to the Provenance of Ontologies Using Blockchain Technologies

Simon Curty[1 [0000−0002−2868−9001]]✉, Hans-Georg Fill[1 [0000−0001−5076−5341]], Rafael S. Gonçalves[2 [0000−0003−1255−0125]], Mark A. Musen[3 [0000−0003−3325−793X]]

[1] University of Fribourg, Digitalization and Information Systems Group
{simon.curty,hans-georg.fill}@unifr.ch
[2] Harvard Medical School, Center for Computational Biomedicine
rafael_goncalves@hms.harvard.edu
[3] Stanford University, Stanford Center for Biomedical Informatics
musen@stanford.edu

Abstract. When applying ontologies in practice, human and machine agents need to ensure that their provenance is trustworthy and it can be relied upon the contained concepts. This is particularly crucial for sensitive tasks such as in medical diagnostics or for safety-criticial applications. In this paper, we propose an architecture for the decentralized attestation and verification of the integrity and validity of ontologies using blockchain technologies. Blockchains are an immutable, tamper-resistant and decentralized storage where all transactions are digitally signed. Thus, they permit tracing the provenance of concepts and identify responsible actors. For a proof-of-concept we extended the WebProtégé editor so that domain experts can attest to the provenance of ontologies via their Ethereum blockchain account, subsequently permitting other actors to reason about the validity and integrity of ontologies. For evaluating the applicability of this approach, we explore a use case in the biomedical domain and perform a cost analysis for the public Ethereum blockchain. It is shown that the attestation procedure is technically feasible and offers a new strategy for placing trust in ontologies.

Keywords: Ontology · Attestation · Blockchain · WebProtégé

1 Introduction

Since the first conceptions of a semantic web, trust has been a central issue due to the decentralized and inconsistent nature of the web itself [4,19]. With the recent integration of linked data, knowledge representations, inferencing mechanisms and machine learning, it has become essential both for human and machine agents to know about the provenance of data and derived information [23,17,31]. Thereby, ontologies play a central role as a formal knowledge resource. Besides *policy-based trust* that regulates the origin and access to information, e.g. through authentication, *reputation-based trust* has played a major role in decentralized settings, where past interactions and/or ratings by users determine the

level of trust [7]. According to O'Hara et al. [39], it can be distinguished between five strategies for ensuring trust: *optimism* where trustful information is regarded as default; *pessimism*: where trust is restricted unless a reason for trust is given; *centralized*: where trust is achieved through centralized institutions; *investigation*: where trustworthiness is achieved through active self-evaluation; and *transitivity*: where it is being relied on other agents.

With the recent popularization of blockchain technologies, another technique for ensuring trust according to the transitive and investigative strategy has been added. Through blockchains as append-only, immutable, decentralized and distributed data stores, trust is achieved through full transparency of the recorded, digitally-signed transactions that are verified through peer-to-peer consensus protocols. These properties qualify blockchain technology for applications where trust in the correctness and integrity of information is essential and shall be publicly verifiable without a central party. Previously proposed techniques for facilitating trust in ontologies, typically reverted to canonical representations that are digitally signed, e.g. [11]. However, this trust information must be shared to be of any use. Through its fundamental properties, blockchain technologies provide means to incorporate these aspects in one decentralized trusted system.

Extending upon previous work [13], where we showcased a first demonstration, we therefore explore in which way blockchain technologies can contribute to the integrity and trust in ontologies through so-called attestations, i.e. verifiable, transparent proofs of the existence of information and derive an architecture for this purpose. Reverting to blockchains yields multiple benefits. First, for domains where the quality of information is of utmost importance, the attestation of ontologies by qualified parties enables the transparent and decentralized guarantee of the correctness of information upon *human judgment*. For example, ontologies in bio-medical domains may be attested by a board of specialists independently of a central organization. A machine learning algorithm is then able to verify via the attestation that this ontology has been approved and is safe to be used for diagnosis without a third-party or central platform. Second, appended records in a blockchain cannot be re-ordered, enabling creation of decentralized immutable *timestamps* for information. Third, the *evolution* of ontologies may be tracked transparently, such that it is evident who committed what change and at what point in time, as information in a blockchain is traceable. Fourth, ontologies may contain sensitive information not suitable to be shared among all parties. It may, however, be necessary to prove the presence of information, for example, to fulfill compliance requirements. In such a scenario, *zero-knowledge proofs* may be used as no information needs to be disclosed.

Furthermore, the collaborative nature of authoring ontologies requires toolsets supporting multiple users. We thus describe a prototypical implementation of an attestation approach using the Stanford WebProtégé editor that has been extended with a plug-in for the Ethereum blockchain. For evaluating the applicability of the attestation procedure, we show its application in a use case of the biomedical domain. We conduct a performance evaluation and a cost analysis for the public Ethereum mainnet.

The remainder of this paper is structured as follows: In Section 2 we discuss related work. In Section 3 we introduce fundamental technologies and concepts relevant for our approach. The architecture itself and its implementation are presented in Sections 4 and 5. We evaluate our approach in Section 6. Finally, we discuss the benefits and limitations of the approach in Section 7 and conclude with an outline of future research in Section 8.

2 Related Work

In this section we briefly review previous work on trust in the context of semantic web, digital signatures and the integrity of ontologies, collaborative ontology authoring and ontologies and blockchains.

2.1 Trust in the Context of Semantic Web

Early approaches for assessing trust in semantic web relied on *reputation* or *transitivity* based strategies where trustworthiness is derived by placing trust in other users and their assessments, e.g. [40]. Later work often followed the *investigation strategy* [39], i.e. where trust is placed in knowledge sources upon active self-evaluation. In this direction, Heymans et al. proposed for example a logic programming-based framework for software agents operating on the semantic web [25]. These agents form a *trusted web*, capable of reasoning about the reliability of knowledge sources.

Due to the distributed nature of the semantic web, sources may become unavailable, causing inconsistencies in inferred knowledge. Schenk et al. [43] therefore proposed trust levels, allowing for *caching* mechanisms to offset the unavailability of sources. Another approach for determining trust based on the consistency of knowledge bases involving uncertain information was presented in [19]. In this work, an inconsistency tolerant trust computation model has been described based on Bayesian description logic, which is capable of computing a degree of inconsistency of a knowledge base.

In semantic web environments, documents are typically annotated based on ontologies in a peer-to-peer setting. Thereby, peers may use different vocabularies, requiring the alignment of these ontologies for answering queries truthfully. Atencia et al. thus proposed a probabilistic model for calculating trust in query answers in such a P2P setting [3]. As shown by Nolle et al., inconsistencies can also be a chance to gather further information [36]. For this purpose, they described an approach for calculating a measure of trust in federated knowledge bases relying on statistical conflict assertions.

Another strategy often found is the *centralized* strategy, where trust is placed in institutions that host knowledge bases. This applies for example to platforms such as NCBO BioPortal [37,52], that provides access to a large amount of principled biomedical ontologies and is maintained by Stanford University, or DBPedia as maintained by the DBPedia Association, which provides an open knowledge graph extracted from Wikipedia [30].

2.2 Digital Signatures and Integrity of Ontologies

In today's web, digital signatures are widely used for message authentication and for ensuring data integrity and non-repudiation. In the context of the semantic web, digital signatures may be used for achieving *policy-based trust*. As a foundation for deriving signatures for RDF graphs, Carroll proposed a canonicalization of RDF graphs without changing their semantics, which allows the graphs to be signed in $O(n\log(n))$ as the signing of arbitrary RDF graphs cannot be done in polynomial time [11]. Later work extended the original approach and introduced the ability to sign individual statements of an RDF graph [51].

In a collaborative environment, the ability to sign only parts of a document is a beneficial feature. An approach to compute a digest of RDF graphs for content identifiers without the need for canonicalization was discussed by Sayers and Karp [42]. Based on these previous works, Kasten et al. [29] presented a framework for signing RDF and OWL graphs, with the capability of signing individual sub-graphs. The use of variations of a Merkle tree for hashing RDF graphs has been proposed in [46]. A Merkle tree is a tree of hashes, where leaves are hashes of data blocks and non-leaf nodes are hashes of child nodes [33]. The root of the tree is a hash representation of the underlying data. Thus, any modification of the data will result in a different root hash. This property allows one to verify the integrity of data. However, a drawback of Merkle trees, is that insertions require reconstruction of the entire tree. This poses a disadvantage for the purpose of integrity verification of evolving knowledge, for example, when monitoring ontology evolution and tracking changes. Sutton and Samavi therefore proposed data structures based on Merkle trees for RDF datasets that support insertions [46].

2.3 Collaborative Ontology Authoring

Real-world ontologies, such as the ones from BioPortal [37,52], are typically created in a collaborative fashion. Already in the early days of the semantic web, efforts have been made for providing tools for collaborative ontology development [44,27]. Since then, these tools have evolved and matured. A popular ontology editor is *Protégé* [49,35], which offers solutions to many challenges that come with collaborative workflows. This includes functionalities for engaging in discussions with other authors, the ability to annotate elements and for tracking changes in ontologies. In recent years, a cloud-based editor in the form of *WebProtégé* has been developed [50,28]. It presents an evolution of the Protégé desktop platform, developed to make use of modern web infrastructure.

2.4 Blockchains and Ontologies

Blockchain technologies are currently widely discussed and have led to innovative solutions in various fields [8]. Multiple benefits have been previously identified for applying blockchains in the semantic web [10], e.g., for using RDF as the data storage format on blockchains and thus providing a decentralized, immutable, tamper-proof data storage for RDF graphs [45]. Another approach has

been proposed in [16]. There, the concept of *knowledge blockchains* is applied for the transparent monitoring of ontology evolution and proving the existence of concepts without disclosing them using so-called zero-knowledge proofs. A hybrid approach for storing RDF triplets for use in edge networks was presented in [48], where triplets are stored in a distributed off-chain RDF store but access is controlled by smart contracts. A blockchain-based architecture for the distribution of knowledge graphs was proposed in [1]. There, linked open data is stored on a blockchain where updates are driven by a community-based consensus. Other works discuss the benefits of applying techniques of the semantic web to blockchains, e.g., in [47] the authors propose a mechanism to index transactions of blockchains as linked data conforming to a vocabulary described by the BLONDiE ontology [24].

In conclusion, previous publications have discussed approaches for establishing trust in ontologies using investigative, centralized, transitive and reputation-based strategies, for digitally signing ontologies and for the collaborative authoring of ontologies and as well as the storage of ontologies on blockchains. What is however missing is a decentralized, light-weight approach for the collaborative attestation of ontologies for ensuring their trustful provenance.

3 Foundations

In this section we briefly introduce fundamental concepts and technologies relevant for describing our approach. This includes a formal definition of cryptographic hash functions and the fundamental properties of blockchain technologies.

3.1 Cryprographic Hash Functions

A hash function $h : M \rightarrow D$ maps a message of arbitrary length to a fixed length digest, such that, given the length function l: $\forall m, m' \in M : l(h(m)) = l(h(m'))$ and $\not\forall m, m' \in M : l(m) = l(m')$. That is, the length of the message digest is always the same, no matter the length of the hashed message. However, in a cryptographic system, additional properties are desired [41]:

- *Pre-image resistance*: Calculating the message from the digest is practically infeasible, that is, for a given d the original message m, such that $d = h(m)$ cannot be computed reliably in polynomial time by a probabilistic function.
- *Second pre-image resistance*: It is practically infeasible to find a message m' for a given m such that $h(m) = h(m'), m \neq m'$.
- *Collision resistance*: It is practically infeasible to find two arbitrary messages m and m' such that $h(m) = h(m'), m \neq m'$.

Furthermore, calculating the digest for any given message should be cheap to compute, that is, there exists a polynomial algorithm to calculate the digest for any input. However, to impede pre-image attacks, there should not be a

correlation between the similarity of messages and their digests. In summary, one would want a cryptographic hash function to map any distinct input x to distinct outputs y, but not to be able to calculate x given y. This is for example achieved by the SHA-256 hash function [15] or Keccak-256 [5] as used by Ethereum.

3.2 Digital signatures

Digital signatures are used to verify authenticity, integrity and non-repudiation of documents or messages. Authenticity refers to the origin of the document, that is, a receiving party can verify that the signed document was actually authored by a known party. The integrity of a document is given, when a receiving party can verify that it was not altered en route by a third party. Non-repudiation guarantees that the signing party cannot dispute authorship of the signed data. Digital signatures are based on the *public key cryptosystem*, also employed in asymmetric encryption of messages. In a public key cryptosystem, a pair of *keys* are used for encryption. The pair consists of a public and a private key. The public key is to be distributed and accessible to other parties, while the private key must be kept secret. The way public keys are published and linked to user identities is application-specific. Messages are signed by encrypting a cryptographic hash of the data with the signer's private key and attaching this information to the original message. The recipient of the message retrieves the attached information and the original message. To verify the signature, the signed hash is decrypted using the signer's public key. Finally, the decrypted hash is compared to a hash of the message calculated by the recipient. If they are identical, the digital signature is valid and the message has not been altered.

3.3 Blockchain Technologies

The term *blockchain* refers to a family of technologies and concepts for storing data in a decentralized, immutable and tamper-proof manner. In the following we only outline the major characteristics of blockchains as required for presenting our approach – for further details we refer the reader to the literature [2,18].

A blockchain can be described as an electronic ledger of transactions distributed over multiple locations. Participants in a blockchain operate a node with a replication of the whole blockchain database as part of a peer-to-peer network. Through this network, participants are able to send transactions to each other. Depending on the blockchain system, a transaction can include a transfer of funds, data or even the deployment of executable software code in the form of *smart contracts* [2]. For preventing fraud and establishing trust, participants can read and validate all transactions.

For initiating a transaction, a participant submits the transaction to a node where it is stored in a pool of *unverified* transactions. Thereby, every transaction is digitally signed by the participant, i.e. the cryptographic hash of the transaction data is signed with the participant's private key. The node then propagates the transaction to its peers for validation using specific consensus protocols.

Nodes that validate transactions are called *validators*[4]. Validators choose unverified transactions from the pool and verify the signature of the initiator. In some systems, additional constraints may need to be fulfilled due to the consensus protocols, e.g. for randomly choosing a validator to prevent fraudulent validators in public blockchains such as Bitcoin or Ethereum. Based on the public key of the initiator, it can be verified, that (i) the transaction data has not been altered and (ii) the sender is in fact the initiator and not an impostor. The validated transactions are then added to the blockchain in the form of an append-only block data structure and linked to the previous block by referencing the hash value of its content. Consequently, the consensus protocols and the validation process make it impossible for a fraudulent participant to change any transaction data, for example, the recipient of a funds transfer. On the other hand, the initiator can neither roll back the transaction nor dispute any involvement.

4 Architecture for Blockchain-based Attestation of Ontologies

We now advance to the description of our blockchain-based attestation architecure for ontologies for ensuring their trustful provenance. An *attestation* describes a cryptographically verifiable claim about the existence of information [22]. In blockchain-based settings, these attestations can be recorded on a blockchain in the course of a digitally-signed transaction. In this way, a blockchain participant can issue a claim about the existence of information that can be verified. In contrast to traditional RDF signatures, it is not necessary to share the claim separately. In practice, such claims are conducted by first calculating the hash value of the information using a hash function. This hash value is then included in a blockchain transaction that is digitally signed and recorded on the blockchain. Subsequently, everyone with access to the blockchain can inspect the transaction, verify who has signed it and retrieve the hash value. By re-calculating the hash value for a given information and comparing it to the stored hash value, the attestation claim can be verified.

For applying attestations to ontologies in a collaborative ontology authoring environment, we have designed the architecture shown in Figure 1. The architecture is composed of two main components: (i) on–chain smart contracts and (ii) a client–server-based ontology editor. The choice of blockchain platform greatly affects the level of decentralization, data visibility and costs, due to the specific properties and network configurations of different platforms. The proposed architecture requires an account-centric blockchain capable of executing smart contracts, i.e. Turing-complete programs that are stored and executed via transactions [2]. Deployed on-chain is an attestation smart contract, responsible for recording and verifying attestation claims.

The remote component of the ontology editor includes an ontology store as well as a component for parsing ontologies and calculating digest values.

[4] Validators are sometimes also called *miners*

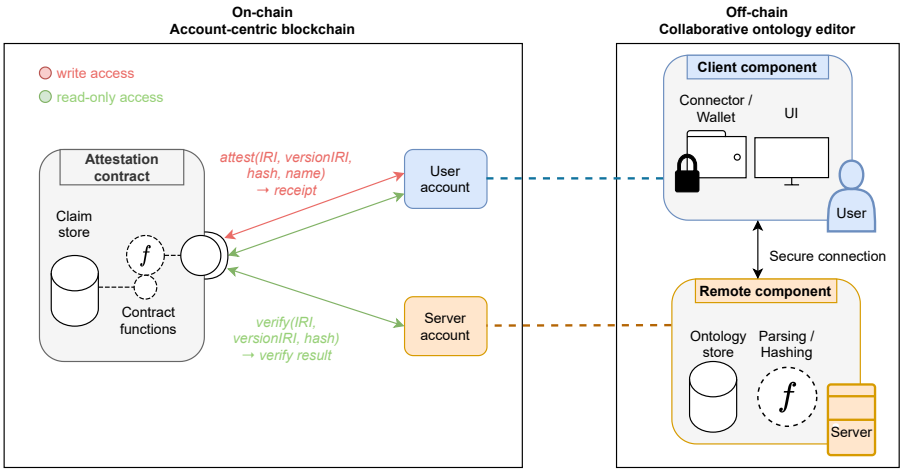

Fig. 1. Architecture of the attestation approach showing a client-server based ontology editor and the on-chain components.

The server further has access to the blockchain and the attestation contract functions via a dedicated account. However, this account is not used to attest to the provenance of ontologies, but for verification only. Different versions of an ontology may have been published such that an ontology's revision is identified by the tuple `(IRI, version IRI)`. Calculation of the ontology's hash value is done by a component on the server so that the client is not required to download the entire ontology for processing each attestation claim. The concrete hashing approach influences additional capabilities of the system, e.g., a merkle-tree–based hashing allows for conducting so-called *zero-knowledge proofs*. This may be used for example to verify that an ontology with a specific IRI containing privacy-sensitive information has been attested and contains specific concepts, without the need to disclose what these are.

Each user holds a personal blockchain account, not linked to the ontology editor. These accounts reside on the blockchain itself and the user alone holds authority over its management. That is, the blockchain accounts are completely independent of the user profiles of the editor. The user accesses this account by connecting to the blockchain network with a wallet application, through which attestation transactions are authorized upon demand.

5 Realization using Ethereum and WebProtégé

For the prototypical realization of the described architecture, we reverted to WebProtégé [28] and the Ethereum blockchain [9]. The choice of Ethereum and related implementation details are further elaborated in 5.2. The two main components of WebProtégé are the web server and the client application, running in the user's browser. Ontologies are stored in a database on the server. When the

user opens a saved ontology in the editor, the server fetches the ontology and transforms it from the database representation into the OWL 2 model. Thereby, only those parts of an ontology instance are transmitted asynchronously to the client that are necessary for the current actions of the user. Instead of transmitting the entire ontology to the client, hash calculation is done on the server, as detailed in 5.1. However, creating, signing and transacting the attestation claim must be done by the client as the authority over personal blockchain accounts should not be transferred away from the user. In order to integrate the attestation approach, some changes had to be made to key classes of WebProtégé in addition to extending it with a plugin in the form of an attestation portlet. Therefore, an open-source fork of the project was created and made publicly available where the attestation approach has been integrated [12] . Further documentation is available in the repository, as well as a pre-built docker image.

5.1 Calculating an Ontology Digest

OWL Ontologies differ from arbitrary text documents in that (i) the order of statements, e.g., triples in an RDF representation, do not change semantics, and (ii) other ontologies may be referenced by use of *imports*. These properties must be considered for hash calculation. There are multiple formats for representing OWL 2 ontologies such as the Turtle syntax or OWL/XML. Some hashing approaches require a specific format, canonical form or pre-processing of the ontology. If specific properties, such as proof of membership, are required, an appropriate hashing approach must be chosen. The OWL API [26] is a Java library for processing OWL 2 ontologies. It provides a mechanism for generating a hash code for OWL elements. The hash code implementation applies the *visitor design pattern* [20] for traversing the OWL 2 element structure. This is accomplished as follows. An OWL 2 ontology consists of *axioms* A_X, *annotations* A_O, as well as a term vocabulary, i.e. the entities. The set of entities is called the *signature* S of an ontology [34]. These sets are merged to a set $U = \bigcup \{S, A_X, A_O\}$ and the hash code of the resulting set is calculated as $hc(U) = \sum_{a \in U} visit(a)$, where *visit* is the concrete visitor function of a structure element provided by the API[5]. The resulting hash code is then converted to a fixed-length hexadecimal hash and used as ontology digest for the attestation claim. This method does not require a canonical form, i.e., representing the ontology as sequence of RDF triples with some deterministic ordering. However, as no cryptographic hash function is currently used, this method is not secure, i.e., not suitable for conducting zero-knowledge proofs. An extension of the hash function using a secure version will be considered for a future version as it will require the adaptation of the OWL-API.

[5] The visitor implementation is found here: http://owlcs.github.io/owlapi/apidocs_4/org/semanticweb/owlapi/util/HashCode.html

5.2 Attestation Contract and Blockchain Integration

A key part of the prototype is a smart contract on the standard Ethereum blockchain. Ethereum was chosen due to its popularity, tooling support and its capability to execute Turing-complete smart contracts. Using the public mainnet has the advantage, that besides the WebProtégé Server, no additional infrastructure must be operated. The proposed approach can be adapted to other blockchain-based systems, as will be discussed in Section 7. The smart contract takes the role of a storage for the ontology hashes and as an API for their verification. When storing a hash on a blockchain, a signed transaction is sent to the address of the smart contract. For verifying that an ontology was attested, only the contract address, the network and the method for hashing must be known. In the prototype implementation, the contract address is known to the server, and we assume the network, including the procedure to connect to it, is known to the user. The users provide their own Ethereum account for interacting with the blockchain. User profiles in WebProtégé are not directly linked to blockchain accounts. This approach requires the users to connect their browser to the blockchain network. For this purpose we integrated a wallet browser plugin in the form of Metamask[6] to handle user login and network interactions.

The process for conducting attestations is shown by the sequence diagram in Figure 2. Thereby it is assumed that some domain expert wants to attest to the provenance of an ontology and triggers the attestation process through the UI in the web browser. The client requests the necessary data from the server and the Metamask wallet plugin prompts the domain expert to login and select the relevant blockchain network. After the connection to the network has been established, the domain expert is prompted to authorize a transaction to the smart contract. With the transaction call, the ontology hash and information about the signer are sent. The smart contract stores the attestation and returns a receipt. Finally, the success of the attestation is reported to the domain expert.

6 Evaluation

Since our implementation is focused on offering a practical solution for an attestation process, we briefly describe an illustrative use case for its practical application. Additionally, we conduct a performance-based evaluation of the hashing and a cost analysis for the Ethereum mainnet. With the prototypical implementation presented in the previous chapter, the technical feasibility of the proposed architecture has been shown.

6.1 Use Case for Biomedical Ontology Authoring

In the biomedical domain ontologies play an important role. As of today, NCBO BioPortal hosts more than 850 ontologies with more than 10 mio. classes. This knowledge repository is highly valuable not only for human users but often serves

[6] A crypto wallet & gateway to blockchain apps - https://metamask.io/

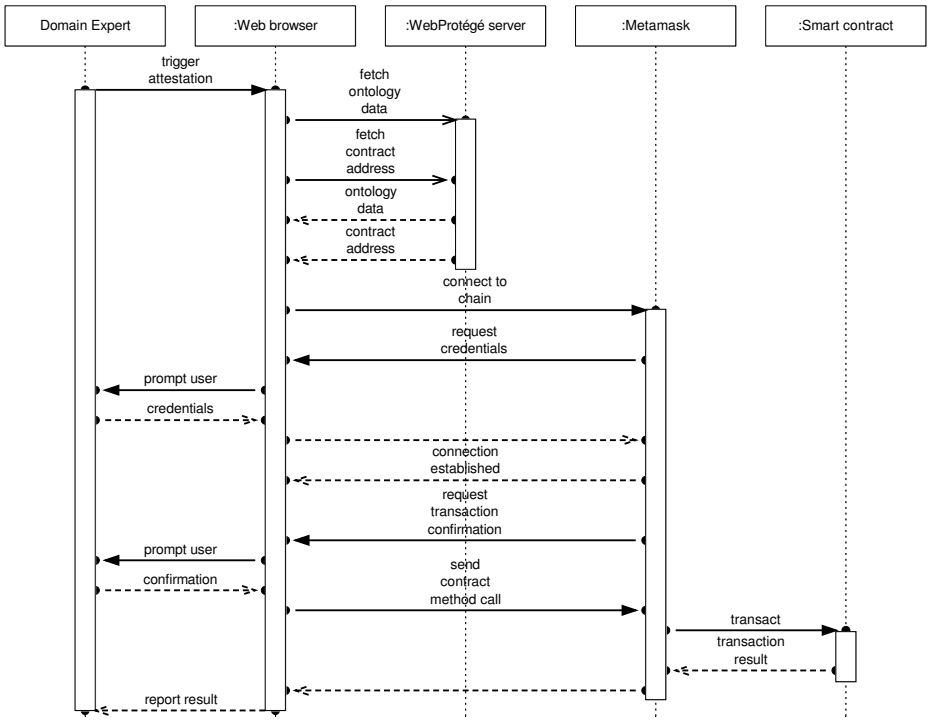

Fig. 2. Sequence diagram of the attestation process.

as a basis for machine learning approaches, e.g. [21]. In addition, various applications, such as specific annotation or recommender services provide easy access to the ontologies [32]. Importing parts of other ontologies, and thereby re-using enclosed concepts, is a common practice when engineering biomedical ontologies. Re-using parts of ontologies has the advantage of reducing maintenance efforts and allows users to focus on concepts specific to their ontology [38]. This highlights the importance of ensuring trust in ontologies. With the proposed attestation approach, we believe that the quality of services operating on biomedical ontologies as well as the trust in the contained concepts can be enhanced by means of a combination of the transitive and investigative trust strategy (see Section 1). By providing an immutable and tamper-proof attestation of an ontology by a domain expert or a board of specialists, the quality of an ontology can be verified both by human as well as machine agents even if it not hosted by an official body, e.g. a gene ontology (GO) version not issued by the GO consortium. Thus, it could be verified, which exact ontology version a machine learning algorithm has used and how the contained information is further propagated, e.g. [17,21]. It would also enable users who wish to re-use existing ontology concepts to verify whether an ontology has been approved by domain experts.

6.2 Performance

The process of attestation involves calculating the ontology digest and transacting the claim. Since the digest is of fixed length (see Sections 3.1 and 5.1), the size of an ontology does not affect the size of an attestation claim. Thus, the validation and confirmation time for a transaction with an attestation claim largely depends on the current performance of the blockchain network, e.g., the Ethereum mainnet, but neither on the file size of an ontology nor the number of contained concepts, e.g., the number of classes. Therefore, the confirmation time for transactions is not part of the evaluation.

Table 1. Selected OWL ontologies from BioPortal. The ten largest in terms of classes as of Sep. 2021 were retrieved.

Acronym	Num. classes	Num. entities	Hash time [ms]
IOBC	126'842	2'123'338	2811
RETO	147'738	2'213'776	2595
REXO	158'239	2'350'454	2622
UPHENO	159'981	1'466'043	2346
NIFSTD	160'818	1'905'381	2490
GEXO	166'254	2'466'421	2991
NCIT	167'138	2'984'569	4609
BIOMODELS	187'520	2'093'464	2684
RH-MESH	305'349	1'959'445	2494
DRON	578'391	3'012'069	4106

In order to evaluate the suitability of the hashing approach for real world ontologies, we conduct performance tests using ontologies from BioPortal, shown in Table 1. The top ten OWL ontologies in terms of the number of classes[7] as of Sep. 2021 were chosen. The tests were run on an Intel i7 Skylake @2.6GHz CPU on Ubuntu 21.04 and the ontologies were loaded into memory before the measurements were taken to minimize the impact of file system delays. The testing script is implemented in Java, using the JUnit5 testing framework. For each ontology, 20 measurements were taken and the trimmed mean [6] calculated – with 5% of upper and lower bounds removed to account for uncontrollable factors, such as OS operations. Figure 3 shows the results of the performance evaluation. While the time needed to calculate a digest increases and correlates with the number of structural elements, i.e. entities, expressions, axioms and annotations, it remains on an acceptable level for this kind of application. The more so as the transaction confirmation time in the Ethereum mainnet is often magnitudes higher.

[7] Filtering by size on BioPortal orders ontologies by number of classes.

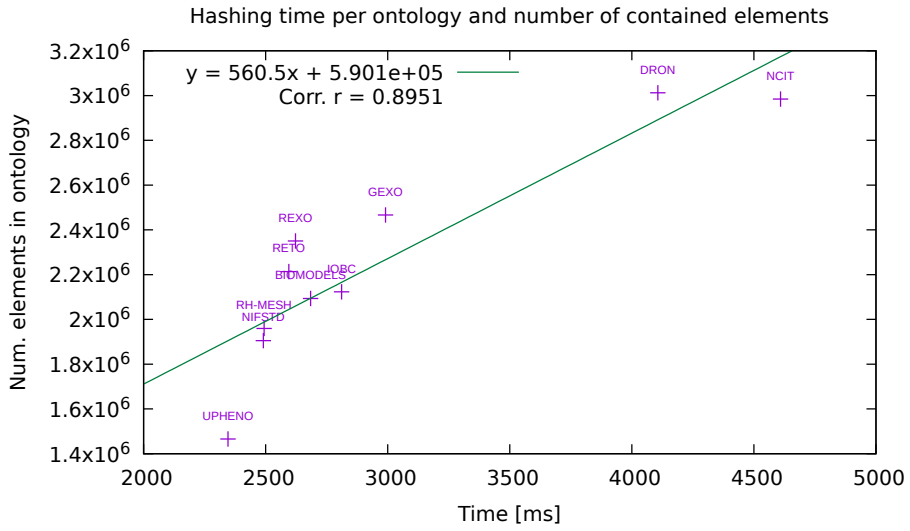

Fig. 3. Time needed to calculate a digest with the OWL API increases linearly with the number of elements in an ontology.

6.3 Cost Analysis

A major drawback of using public blockchains and smart contracts such as found on the Ethereum main network is that execution is slow and expensive compared to traditional, centralized systems, especially when either the size of the transaction data is large or the computations of a contract call are complex. This is mainly due to the decentralized consensus protocols used in public blockchains.

In Ethereum, size and computational complexity of transactions are measured in an energy unit, called *Gas*. This is a fee payed by the originator in the cryptocurrency Ether and depends on the complexity of commands to be executed [2] by Ethereum's virtual machine. The Ether price of a unit of gas is influenced by the transaction volume. Figure 4 shows historical transaction fees in USD for an attestation in comparison to a baseline, a contract storing a 256bit integer value. E.g., on June 1, 2021, an ontology attestation would have cost USD 8.62 (vs. baseline of USD 3.25). Prior to the London Upgrade[8] the transaction cost in USD is calculated as follows: *gas limit*$*$ *gas unit price*$*$ *ether price*, where *gas limit* is the estimated gas cost of an attestation transaction, *gas unit price* the average daily price of a single unit of gas and *ether price* the average daily price of one ether in USD. The historical Ethereum data was obtained from Etherscan[9]. With the London Upgrade, a *priority fee* is added to the *gas unit price* as incentive for the validators to include the transaction in a block – a higher priority fee accelerates transaction confirmation. For this cost analysis we include a

[8] London Upgrade – https://ethereum.org/en/history/#london
[9] Etherscan – https://etherscan.io

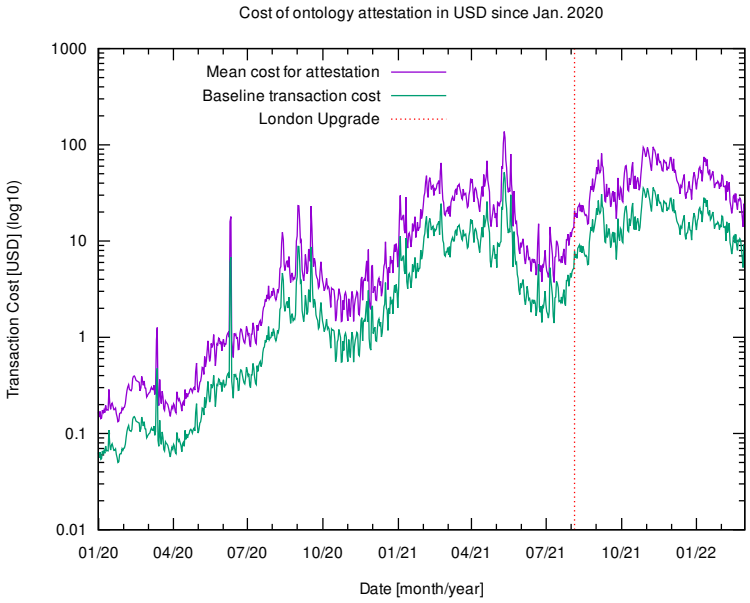

Fig. 4. Transaction costs for attestations based on the price for ETH. The attested ontology only has a minor influence on the incurred costs as the digest length is fixed but the IRI and the signer's name are not. With the London Upgrade the way costs are calculated have been changed.

priority fee of 2 Gwei. Both gas and ether price are volatile. As such, the transaction cost may change significantly in a short period of time. However, Ethereum is in the process of adopting the proof-of-stake consensus mechanism[10], enabling higher transaction throughput and better energy efficiency.

7 Discussion

The approach we described for attesting to the provenance of ontologies describes a novel solution for placing trust in knowledge bases. In contrast to previously described approaches such as in [16,10], we do not store complete ontologies on a blockchain but rather the attestation claims in the form of hash values that are bound to the identities of blockchain users. As such, the approach pursues a *transitive* trust strategy, where trust is placed in resources upon the digitally verifiable certification of peer users of the blockchain. As the identities of users in today's public blockchain systems are pseudonymous, i.e. they are not bound to physical identities, additional measures need to be taken for retrieving the real identity of an attesting user in the sense of an investigative trust strategy. This

[10] See https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/

could for example be achieved by publishing one's public key of the blockchain account on the website of a trusted institution. While this resembles the traditional trust centralization strategy, it does not require the central storage of ontologies and permits the decentralized verification of the attestations.

Although we propose a working approach, it's not without limitations. The currently implemented hashing approach does not yet make use of the full potential of hashing an ontology. In the future the hashing could be adapted towards the use of Merkle-Trees or compression trees, which would permit more fine granular application of zero-knowledge proofs [16,46] as well as the consideration of change-tracking information, which would permit to attest to the nature of changes. The prototype currently does not support zero-knowledge proofs, proof of membership, partial or incremental attestation of ontologies. Further, the smart contract currently does not support the attestation by multiple users but only returns the most recent attestation of an ontology version for verification, i.e. collaborative ontology attestations are not yet supported. However, due to the nature of blockchains, all previous attestations can be inspected in the history of transactions. While we focus on the public Ethereum main network as deployment choice, multiple blockchain-based options exist. The approach can be adapted to other public blockchains, e.g., Avalanche[11], an Ethereum compatible proof-of-stake blockchain promising lower transaction costs compared to Ethereum. Permissioned blockchain networks in the form of a consortium infrastructure may be used as alternative to public networks. In such a consortium infrastructure, access is restricted to authorized and validated participants, sharing the effort to operate the network, e.g., by operating a node. This implies a difference in IT infrastructure and user enrollment [14]. However, instead of building such an infrastructure specific to this purpose, one may revert to existing consortial blockchains such as the Bloxberg infrastructure[12] for decentralized services for the scientific community.

8 Conclusion

In this paper we described an architecture for attesting to the provenance of ontologies using the Ethereum blockchain. The approach has been evaluated through a prototypical implementation and a performance evaluation of the attestation approach by applying it to real-world ontologies in the biomedical domain. Future work will include the investigation of alternative hashing procedures for ontologies for enabling zero-knowledge proofs on a more fine granular level and the extension of the smart contract implementation towards supporting attestations by multiple users.

[11] Avalanche - https://www.avax.network/
[12] Bloxberg infrastructure - https://bloxberg.org/

Acknowledgments

The research on this paper has been partially financed by the Swiss National Science Fund grant number 196889.

References

1. Aebeloe, C., Montoya, G., Hose, K.: Colchain: Collaborative linked data networks. In: WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021. pp. 1385–1396. ACM / IW3C2 (2021). https://doi.org/10.1145/3442381.3450037
2. Antonopoulos, A.M., Wood, G.: Mastering ethereum: building smart contracts and dapps. O'reilly Media (2018)
3. Atencia, M., Euzenat, J., Pirrò, G., Rousset, M.C.: Alignment-based trust for resource finding in semantic p2p networks. In: The Semantic Web – ISWC 2011. pp. 51–66. Springer (2011)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific american **284**(5), 34–43 (2001)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference. Tech. rep., Team Keccak (Jan 2011), https://keccak.team/files/Keccak-reference-3.0.pdf
6. Bolstad, W.M., Curran, J.M.: Displaying and Summarizing Data. In: Introduction to Bayesian Statistics, Third Edition, pp. 31–57. John Wiley & Sons, Ltd (2016)
7. Bonatti, P., Duma, C., Olmedilla, D., Shahmehri, N.: An integration of reputation-based and policy-based trust management. Networks **2**(14), 10 (2007)
8. Braun-Dubler, N., Gier, H.P., Bulatnikova, T., Langhart, M., Merki, M., Roth, F., Burret, A., Perdrisat, S.: Blockchain: Capabilities, Economic Viability, and the Socio-Technical Environment. vdf AG of ETH Zurich (2020)
9. Buterin, V.: A Next-Generation Smart Contract and Decentralized Application Platform (2013), https://ethereum.org/en/whitepaper/
10. Cano-Benito, J., Cimmino, A., García-Castro, R.: Towards blockchain and semantic web. In: Business Information Systems Workshops. pp. 220–231. Springer (2019)
11. Carroll, J.J.: Signing RDF Graphs. In: ISWC. pp. 369–384. Springer (2003)
12. Curty, S.: WebProtégé Attestation: Prototype source code archive (2021). https://doi.org/10.5281/zenodo.5765038
13. Curty, S., Fill, H.G., Gonçalves, R.S., Musen, M.A.: A WebProtégé Plugin for Attesting to the Provenance of Ontologies on the Ethereum Blockchain. In: Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks. CEUR Workshop Proceedings, vol. 2980 (2021), http://ceur-ws.org/Vol-2980/paper329.pdf
14. Curty, S., Härer, F., Fill, H.G.: Towards the Comparison of Blockchain-based Applications Using Enterprise Modeling. In: Lukyanenko, R., Samuel, B.M., Sturm, A. (eds.) Proceedings of the ER Demos and Posters 2021. CEUR Workshop Proceedings, vol. 2958, pp. 31–36 (2021), http://ceur-ws.org/Vol-2958/paper6.pdf
15. Dang, Q.H.: Secure Hash Standard. Tech. Rep. NIST FIPS 180-4, National Institute of Standards and Technology (Jul 2015). https://doi.org/10.6028/NIST.FIPS.180-4
16. Fill, H.G.: Applying the Concept of Knowledge Blockchains to Ontologies. In: AAAI 2019 Spring Symposium. CEUR-WS.org (2019)
17. Fill, H., Härer, F.: Supporting trust in hybrid intelligence systems using blockchains. In: AAAI 2020 Spring Symposium. CEUR-WS.org (2020)

18. Fill, H.G., Meier, A.: Blockchain Kompakt. Springer (2020). https://doi.org/https://doi.org/10.1007/978-3-658-27461-0
19. Fokoue, A., Srivatsa, M., Young, R.: Assessing trust in uncertain information. In: The Semantic Web – ISWC 2010. pp. 209–224. Springer (2010)
20. Gamma, E., Vlissides, J., Helm, R., Johnson, R.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
21. Grigoriu, A., Zaveri, A., Weiss, G., Dumontier, M.: Siena: Semi-automatic semantic enhancement of datasets using concept recognition. Journal of Biomedical Semantics **12**(1), 1–12 (2021)
22. Härer, F., Fill, H.: Decentralized Attestation of Conceptual Models Using the Ethereum Blockchain. IEEE CBI Conference **01**, 104–113 (2019)
23. van Harmelen, F., ten Teije, A.: A boxology of design patterns for hybrid learning and reasoning systems. J. Web Eng. **18**(1-3), 97–124 (2019)
24. Hector, U.R., Boris, C.L.: BLONDiE: Blockchain Ontology with Dynamic Extensibility. arXiv:2008.09518 [cs] (Aug 2020), http://arxiv.org/abs/2008.09518, arXiv: 2008.09518
25. Heymans, S., Van Nieuwenborgh, D., Vermeir, D.: Preferential reasoning on a web of trust. In: The Semantic Web – ISWC 2005. pp. 368–382. Springer (2005)
26. Horridge, M.: OWL API main repository. https://github.com/owlcs/owlapi (2020)
27. Horridge, M., Gonçalves, R.S., Nyulas, C.I., Tudorache, T., Musen, M.A.: Webprotégé 3.0 - collaborative OWL ontology engineering in the cloud. In: ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks. vol. 2180. CEUR-WS.org (2018)
28. Horridge, M., Gonçalves, R.S., Nyulas, C.I., Tudorache, T., Musen, M.A.: WebProtégé: A Cloud-Based Ontology Editor. In: World Wide Web Conference. pp. 686–689. ACM (2019)
29. Kasten, A., Scherp, A., Schauß, P.: A Framework for Iterative Signing of Graph Data on the Web. In: ESCW Conference. pp. 146–160. Springer (2014)
30. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web **6**(2), 167–195 (2015)
31. Martin, A., Hinkelmann, K., Fill, H., Gerber, A., Lenat, D., Stolle, R., van Harmelen, F. (eds.): AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence, Stanford University, USA, March 21-23, 2022, CEUR Workshop Proceedings, vol. 3121 (2022), http://ceur-ws.org/Vol-3121
32. Martínez-Romero, M., Jonquet, C., O'Connor, M.J., Graybeal, J., Pazos, A., Musen, M.A.: NCBO Ontology Recommender 2.0: An enhanced approach for biomedical ontology recommendation. Journal of biomedical semantics **8**(1), 21 (Jun 2017)
33. Merkle, R.: A Digital Signature Based on a Conventional Encryption Function. In: Advances in Cryptology — CRYPTO '87. vol. 293, pp. 369–378. Springer (1987)
34. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., Smith, M.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) (Dec 2012), https://www.w3.org/TR/owl2-syntax/
35. Musen, M.A.: The protégé project: a look back and a look forward. AI matters **1**(4), 4–12 (2015)
36. Nolle, A., Chekol, M.W., Meilicke, C., Nemirovski, G., Stuckenschmidt, H.: Automated fine-grained trust assessment in federated knowledge bases. In: The Semantic Web – ISWC 2017. pp. 490–506. Springer (2017)

37. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G., Musen, M.A.: BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research **37**, W170–W173 (05 2009)
38. Ochs, C., Perl, Y., Geller, J., Arabandi, S., Tudorache, T., Musen, M.A.: An empirical analysis of ontology reuse in BioPortal. Journal of biomedical informatics **71**, 165–177 (Jul 2017)
39. O'Hara, K., Alani, H., Kalfoglou, Y., Shadbolt, N.: Trust strategies for the semantic web. In: Proceedings of the ISWC*04 Workshop on Trust, Security, and Reputation on the Semantic Web. CEUR, vol. 127 (2004)
40. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: International Semantic Web Conference. pp. 351–368. Springer (2003)
41. Rubinstein-Salzedo, S.: The RSA Cryptosystem. In: Cryptography, pp. 113–126. Springer (2018)
42. Sayers, C., Karp, A.: Computing the digest of an RDF graph. Tech. rep., HP Laboratories Palo Alto (2004), https://www.hpl.hp.com/techreports/2003/HPL-2003-235R1.pdf (last access 2021-04-09)
43. Schenk, S.: On the semantics of trust and caching in the semantic web. In: The Semantic Web - ISWC 2008. pp. 533–549. Springer (2008)
44. Simperl, E., Luczak-Rösch, M.: Collaborative ontology engineering: a survey. The Knowledge Engineering Review **29**(1), 101–131 (Jan 2014). https://doi.org/10.1017/S0269888913000192, publisher: Cambridge University Press
45. Sopek, M., et al.: GraphChain: A Distributed Database with Explicit Semantics and Chained RDF Graphs. In: The Web Conference 2018. pp. 1171–1178. ACM (2018)
46. Sutton, A., Samavi, R.: Integrity Proofs for RDF Graphs. Open J. Semantic Web **6**, 1–18 (2019)
47. Third, A., Domingue, J.: Linked Data Indexing of Distributed Ledgers. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 1431–1436. International WWW Conferences Steering Committee (Apr 2017). https://doi.org/10.1145/3041021.3053895, https://doi.org/10.1145/3041021.3053895
48. Tuán, A., Hingu, D., Hauswirth, M., Le-Phuoc, D.: Incorporating Blockchain into RDF Store at the Lightweight Edge Devices. In: Int. Conf. on Semantic Systems. pp. 369–375. Springer (2019)
49. Tudorache, T., Noy, N.F., Tu, S., Musen, M.A.: Supporting Collaborative Ontology Development in Protégé. In: The Semantic Web - ISWC 2008. pp. 17–32. Springer (2008)
50. Tudorache, T., Vendetti, J., Noy, N.: Web-Protege: A Lightweight OWL Ontology Editor for the Web. In: Fifth OWLED Workshop on OWL: Experiences and Directions (Jan 2008)
51. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F.: Signing individual fragments of an RDF graph. In: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web - WWW. pp. 1020–1021. ACM (Jan 2005)
52. Whetzel, P.L., Noy, N.F., Shah, N.H., Alexander, P.R., Nyulas, C., Tudorache, T., Musen, M.A.: BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Res. **39**(Web-Server-Issue), 541–545 (2011). https://doi.org/10.1093/nar/gkr469