Decentralized Attestation of Conceptual Models Using the Ethereum Blockchain

Felix Härer
Digitalization and Information Systems Group
University of Fribourg
Fribourg, Switzerland
felix.haerer@unifr.ch

Hans-Georg Fill
Digitalization and Information Systems Group
University of Fribourg
Fribourg, Switzerland
hans-georg.fill@unifr.ch

***Abstract*—Decentralized attestation methods for blockchains are currently being discussed and standardized for use cases such as certification, identity and existence proofs. In a blockchain-based attestation, a claim made about the existence of information can be cryptographically verified publicly and transparently. In this paper we explore the attestation of models through globally unique identifiers as a first step towards decentralized applications based on models. As a proof-of-concept we describe a prototypical implementation of a software connector for the ADOxx metamodeling platform. The connector allows for (a.) the creation of claims bound to the identity of an Ethereum account and (b.) their verification on the blockchain by anyone at a later point in time. For evaluating the practical applicability, we demonstrate the application on the Ethereum network and measure and evaluate limiting factors related to transaction cost and confirmation times.**

***Index Terms*—Conceptual Modeling, Ethereum, Blockchain, Decentralized Identifiers**

I. Introduction

Distributed ledger and blockchain systems have the unique property of providing a trusted globally consistent view on all information stored, since they can achieve consensus among untrusted nodes of a distributed system [1]–[4]. One of the very fundamental applications of blockchains, which is mostly oblivious the scalability problem [5], is identity. In essence, values for the global and unique identification of objects, persons, documents, or any data, may be used to link or reference digital identities, for sharing data among them or for transferring information attached to them.

In combination with distributed ledgers, these applications can be realized based on the properties of integrity-secured and non-repudiable transactions bound to identities of users [4]. Examples utilizing identity go back to the first application of blockchains for the transfer of money [1] and span to asset transfer, ERP and SCM business process transactions [6]–[8], as well as decentralized registries for documents, records or digital rights shared among organizations [9]. These instances rely on relatively small identifier data stored on a distributed ledger. Identity is involved whenever interactions extend beyond an individual entity, e.g. in business-to-business relationships, customer-facing scenarios or recruiting processes. In a broader sense, this concept has recently been recognized for decentralized applications in light of a general standard for decentralized identifiers (DID) [10].

When implementing the idea of decentralized identification together with verifiable attestations about the identity and existence of information [11], decentralized applications beyond the ones built into distributed ledger systems can be realized. Many applications can be designed in such a way that information does not need to be held available in cases where an assertion about the existence of the information can be made instead. For the attestation of the existence of documents, the concept has been implemented in the form of attestation services based on blockchains [11]–[14]. In this realization, the representation used for attestation by these services is on the level of data. Existing approaches provide a way of attesting to the integrity of data and identity, but do not address how new identity applications can be designed with regard to domain requirements. The importance of requirements and alignment are well established [15]. However, especially in sociotechnical information systems which are decentralized, the system's complexity due to the high amount of components and their autonomy requires careful conceptual design and technological alignment.

In the spirit of the current discussions on decentralized identity, this exploratory research paper constructs and prototypically implements a model-based attestation method independent of centralized services, attesting to the structured information and knowledge represented in conceptual models. Conceptual models capture semantic concepts of a particular domain by their representation expressed in terms of the domain [16], [17]. In a business context, conceptual models commonly represent enterprise architectures, IT infrastructures, processes, or organizational knowledge [18]. When combining the global view of distributed ledger systems with globally unique identity, individual domain concepts from these models can be the subject of an attestation. For example, intellectual property (IP) represented by a model can be attested to, assigned to a global identifier, bound to an identity of a blockchain system and used to prove the existence of the IP at its creation time. The validity of such an attestation and its binding to an identity are then verifiable at a later point in time, by anyone with access to the distributed ledger system. It has been shown before that it is possible to apply blockchain properties such as immutability to the information and knowledge represented by models [19], [20] or to the

design of contracts [21]. However, no model-based attestation methods and implementations in the decentralized setting of a permissionless blockchain could be found. The contribution of this paper is an attestation architecture directed towards model-based identity solutions and a proof-of-concept implementation based on the Ethereum blockchain [22]–[24] and the ADOxx metamodeling platform [25]. Here, an attestation is conducted by (1.) constructing a claim about the existence of a model bound to an identity and (2.) by the validation of the claim through a smart contract.

Ethereum has been chosen due to the standardized support of smart contracts regarding development and tool availability. For modeling, the choice of ADOxx is due to the customization capabilities of metamodels for developers and end users. However, the architecture is independent of the specific choices. By reverting to metamodeling, conceptual models specific to a particular attestation scenario may be designed. With a model as the primary artifact, transformation and code generation operate from a trusted source. In this way, a conceptual model can be used as a single point of truth, making its information dependable and secure, e.g. for the purposes of collaboration in decentralized environments [26] or for the binding use as part of contractual agreements.

The remainder of this paper is structured as follows. Section II introduces foundations on globally unique identification and blockchain systems. Section III outlines the architecture of the approach and its technical implementation. Section IV evaluates the practical applicability of the approach on the Ethereum network and measures limiting factors. Section V discusses the results and its implications. Section VI concludes.

II. Foundations

A. Globally Unique Identification

The concept of globally unique identification of data objects, documents or resources manifests itself in widely used standards such as Universally Unique Identifier (UUID), also called Globally Unique Identifier (GUID) [27], Digital Object Identifiers (DOI) [28] and URIs, URLs and URNs as uniform resource identifiers, locators or names [29]. These identifiers have mostly been thought of and used as references, e.g. in linked data approaches using URL, or for providing uniqueness in order to avoid conflicts within namespaces in software or in hardware, e.g. UUID and MAC addresses. With the consistent global state of a distributed ledger, identifiers stored in it can provide a globally unique binding to data objects. Depending on whether an object's data should determine the identifier, content-dependent or content-independent forms of identification can be used.

1) Content-independent Identification: Generally, two principles for the content- or data-independent assignment of globally unique identifiers can be realized.

- Registration-based approaches, e.g. as found in resource records registered with the domain name system, require an assignment by a registry. By carrying out the assignment of each identifier, the registry is able to control and guarantee global uniqueness. However, without sufficient redundancy, registries can represent single points of failure.
- Generation-based approaches randomly generate identifiers. The generation is performed locally by a secure (pseudo) random number generator. Examples are UUID version 4 identifiers [27] and addresses in blockchain systems [1], [24]. For assuming a unique output, the generator must (a.) be seeded with sufficient entropy and (b.) the resulting key space, or identifier space, must be sufficiently large such that collisions do not occur except for a negligible probability.

2) Content-based Identification: Content-based identifiers are another form of identity, which is created solely based on the represented content. Cryptographic hash functions, such as SHA-2 [30], SHA-3 or Keccak-256 [24], [31] functions compute a fixed-length output v, often 256 bit, from a variable length input message ms. Hash functions are required to be collision resistant [32], i.e., while it must be feasible to compute the value of a hash function $H(ms) = v$, it must be infeasible to find a pair of arbitrary messages ms, ms' resulting in the same v. This property can not be assumed unconditionally for SHA-1 anymore [33]. In addition, it must be infeasible to find an ms given v (preimage resistance) and to derive an ms' from ms resulting in the same v (preimage resistance 2nd). In effect, the function value unpredictably changes in a pseudo-random fashion when the input changes. v, sometimes called message digest, identifies ms and may be used for integrity checking. I.e., if v does not change in subsequent calculations of the function, integrity is assumed. Content-based identifiers are prominently used in blockchains [1], [24], in the Git versioning system [34] and for data retrieval in peer-to-peer networks using the protocols of torrent, IPFS and Swarm for addressing files by their content [35].

B. Blockchain Systems

A blockchain is a data structure of linked blocks, where each block is linked to one predecessor by the value of a hash function for providing integrity across the chain. In a blockchain system, the data structure is used in combination with consensus algorithms for the verifiably consistent storage of non-repudiable transactions, signed and broadcasted by the identities of distributed participants on the infrastructure of a peer-to-peer network [4], [36]. In the generalized form of distributed ledger technology, the data structure of such a system may not necessarily be based on a blockchain.

In this paper, the widely-used permissionless blockchain system Ethereum is utilized [37]. For being permissionless, access rights are not restricted to an a priori defined group of users. The system is outlined in the sections following.

1) Accounts and Transactions: Ethereum is an account-based blockchain which allows for the execution of permanently stored smart contracts [24]. An account is identified by a globally unique address for sending and receiving transactions. It is either an *externally owned account* (EOA), owned

by an identity of a participant, or a *contract account* (CA), storing a smart contract.

- *EOA* rely on elliptic curve public key cryptography for access control. Each identity locally generates a public-private key pair and derives an address of an externally owned account from the public key. Similar to other systems, the private key is used to sign transactions, while the public key is used to validate signatures.
- *CA* are outside the control of external identities and execute smart contracts autonomously on fully validating nodes of the network when invoked by a transaction, calling a contract at a particular storage location.

Each transaction (a.) transfers an amount of the Ether cryptocurrency between accounts, (b.) creates a contract account by storing the code of a smart contract or (c.) invokes a smart contract stored in a contract account.

2) Structure of Blocks: In addition to a hash value of the previous block and transactions, the Ethereum blockchain stores the state of all accounts and so-called receipts for transactions in blocks [24], [38]. The structure of a block includes separate *Merkle Patricia Tree* hash-based tree data structures for (a.) transactions, (b.) state, and (c.) receipts. Similar to Merkle trees, the integrity of the data stored in the tree can be verified by re-calculating hash functions. In contrast to Merkle trees, data is stored inside the tree and not only represented by leaf nodes.

3) Consensus Algorithm: Ethereum uses a proof-of-resource algorithm which requires the expenditure of resources in order to solve computationally hard problems, sometimes called cryptographic puzzles [32]. Whenever a solution is found, a block may be appended by a miner to the blockchain. At this point, Ethereum uses the proof-of-work consensus algorithm Ethash, based on the DaggerHashimoto algorithm for constructing a directed acyclic graph optimized for computation in graphics processing units (GPUs) [38]. The inter-block time, i.e. the minimum amount of time it takes to record a transaction in a block, averages between 14 and 15 seconds with deviations depending on the load [39], [40].

4) Smart Contracts: Smart contracts written in a high-level language such as Solidity are compiled to bytecode before the creation of a contract [38]. After the creation in a CA by a transaction, bytecode is stored together with the execution state of the contract, defined by state variables. By invoking the contract through a transaction transferring a function call with input data, the executed code operates on the state. While the Ethereum instruction set is said to be Turing complete, the execution time of a smart contract invocation is bound by the expenditure of a transaction fee in the unit "gas", set by the sender. The Ethereum Virtual Machine (EVM) is the execution environment for the bytecode of smart contracts, running on every fully validating node of the network. It is a stack-based virtual machine which executes the code stored in a contract account.

III. Attestation of Conceptual Models

The concept of remote attestation concerns claims made about the properties of a target and their attestation by means of evidence transferred over a network to another party [41]. Recently, applications of attestation concepts are being discussed for the web, and in the context of blockchain systems [10], [12], [42]–[44].

A. Attestation Concept

In the context of conceptual modeling, these ideas can be applied for the remote attestation of model artifacts, bound to the identity of a user. Here, a model-based attestation can be conducted by (a.) the creation of a claim about the existence of a model, bound to an identity, and (b.) the validation of the claim by any other identity at a later point in time. An architecture realizing this concept can be constructed by meeting three requirements:

- *Globally Unique Identification* of conceptual models and users when constructing a claim. Note that although models need to be globally identifiable, they may not necessarily need to be stored globally.
- *Trusted Global State* for storing the identifiers in a globally accessible and consistent manner in order to record the claim.
- *Content-based Validation* by a method establishing the correctness of the content of a model as it existed at the time when the claim was created.

In such a system, an attestation can be conducted by the issuance of a claim through a user, referred to as *Claim Issuer*, who records the claim for it to become part of the trusted global state, and by validating the claim through a user, referred to as *Claim Validator*.

B. Attestation Architecture

For realizing an attestation based on this concept, the following architecture of a blockchain-based attestation system for conceptual models can be implemented:

- *Globally Unique Identification* for models is based on content-independent UUID for identifying and permanently addressing models changing over time, as well as for revocation. For identifying users, the blockchain system's addresses are used.
- *Trusted Global State* is provided through the blockchain system. Here, Ethereum is used with state being recorded in state variables of a smart contract. The smart contract contains at least two functions, for recording claims and for verifying claims. For state variables, appropriate data structures must record claims per user and per model.
- *Content-based Validation* is conducted by the application of content-based identifiers in the form of hash functions H to a model m subject to attestation, such that $H(m) = v$ can be used as evidence of the existence of the model. For multiple models, the value v is created from the set of models M by calculating $H(m)$ for each $m \in M$ and by

the construction of a Merkle tree $MerkleTree(M) = v$ as described in the following sections. For the validation, the root hash value v is then used as evidence.

Figure 1 on the next page describes the attestation architecture. The participants and the processes for (a.) the issuance of claims and (b.) their validation are outlined in the following sections. By these processes, a claim regarding a set of models and a user identity bound to it lead to a *valid* or *invalid attestation result*. In the case of *valid*, the identity of claim issuer and a timestamp of the claim's record in a block can be retrieved additionally.

1) System Participants: In the attestation, two users represented by EOA on the Ethereum blockchain are involved, with the CA on the Ethereum blockchain acting as a trusted third party. The process is controlled from the perspective of a user through a locally running modeling tool. For (a.) the issuance of a claim about the existence of globally identifiable models, an *Issuance Component* of the modeling tool is invoked by a user referred to as *Claim Issuer*. For (b.) the validation of a claim in another remotely running instance of the modeling tool, a *Validator Component* of the tool is invoked by a user referred to as *Claim Validator*.

2) Issuance of Claims: Starting from a set of locally available models M present in the modeling tool of claim issuer, the issuance of a claim for M is carried out by this party through the following process in accordance to Figure 1. Conceptually, M represents a set of one or more models belonging together for the purposes of the claim, possibly with inter-model references between the members of M. For issuing claims on a per-model basis, the process may be carried out with $|M| = 1$.

2.1) Serialization of Models: For the models to be transferred to claim validator and for the generation of content-based identifiers, a serialization is generated by the modeling tool. The serialization is stored locally and may be transferred to claim validator over an insecure channel independent of the attestation process.

2.2) Generation of Content-independent Model Identifiers: Models are identified by a content-independent identifier, here described as UUID. That is, one UUID identifies one or more models in M, which are subject to the claim. This allows for flexibility, since it is up to the claim issuer to create multiple claims with $|M| = 1$, resulting in one UUID per model, or to combine models belonging together and identify them by a single UUID. The *Issuance Component* generates and assigns the UUID randomly, independent of individual models. The UUID may be added as a model attribute to individual models or stored separately.

2.3) Generation of Content-based Model Identifiers: Evidence of the existence of models is created for the claim as a content-based model identifier, calculated as the value of a hash function applied to the model serialization through the *Issuance Component*. The hash function must be a secure cryptographic hash function, e.g. of the SHA-2 or SHA-3 family [30], [31].

- In the case of $|M| = 1$ for one individual model $m \in M$, $H(m) = v$ is calculated and is the content-based identifier for M.
- In the case of $|M| > 1$, the Merkle tree's root hash value $MerkleTree(M) = v$ identifies the models of M. After calculating $H(m)$ for each $m \in M$, the tree is constructed from these values, which represent its leaf nodes. In each iteration of the calculation, one level of the resulting tree is created: For any given level, starting at the leaf nodes, all hash values of the level are partitioned into pairs [45] (v_1, v_2), which are concatenated and hashed as in $H(v_1||v_2) = v_{1||2}$, such that the number of values making up the next level halves. In the case of an odd number of values, the remaining value forms a pair with $v_2 = 0$. This process is repeated until the root hash value v is reached for identifying M.

2.4) Generation of User Identifier: Users are identified by the addresses of EOA on the Ethereum blockchain. The EOA is generated locally by the *Issuance Component* from a randomly generated public-private key pair, from which the claim issuer's address CIA is derived as the user identifier. Alternatively, a previously generated EOA may be retrieved locally.

2.5) Creation of the Claim: The claim is generated by the *Issuance Component* in the form of the tuple $(v, UUID, CIA, Merkle\_Tree_M)$. It contains (a.) the content-based model identifier v, (b.) the content-independent identifier $UUID$ of M, (c.) CIA as the user identifier in the form of the Ethereum EOA address of claim issuer, and (d.) the Merkle tree of hash values representing, but not containing, the members of M.

2.6) Recording of Evidence from the Claim: For recording evidence with the global state of the smart contract stored in CA, the *Issuance Component* calls the function *recordClaim(UUID,v)*. The function records the claim's UUID, v, and CIA in state variables of the smart contract. For consistently identifying changing models over time, $UUID$ is a key for identifying v and CIA, e.g. implemented by Solidity "mapping" data structures. After a defined number of transaction confirmations, the record is acknowledged (ack).

2.7) Transfer of Claims and Models: Models and according claims are created locally, such that the transfer from claim issuer to claim validator may be carried out over an insecure channel in separate messages, independent of the attestation process. Subject to the transfer are (a.) the model message with models $M_V \subseteq M$ and (b.) the claim message consisting of $(UUID, CIA, MLN)$, where MLN are the Merkle tree's leaf nodes in a totally ordered set. It is up to claim issuer to decide which members of M are sent to claim validator in the context of the domain, e.g. for the creation of multiple certificates in the form of individual models in a scenario where only certificates intended for a specific claim validator

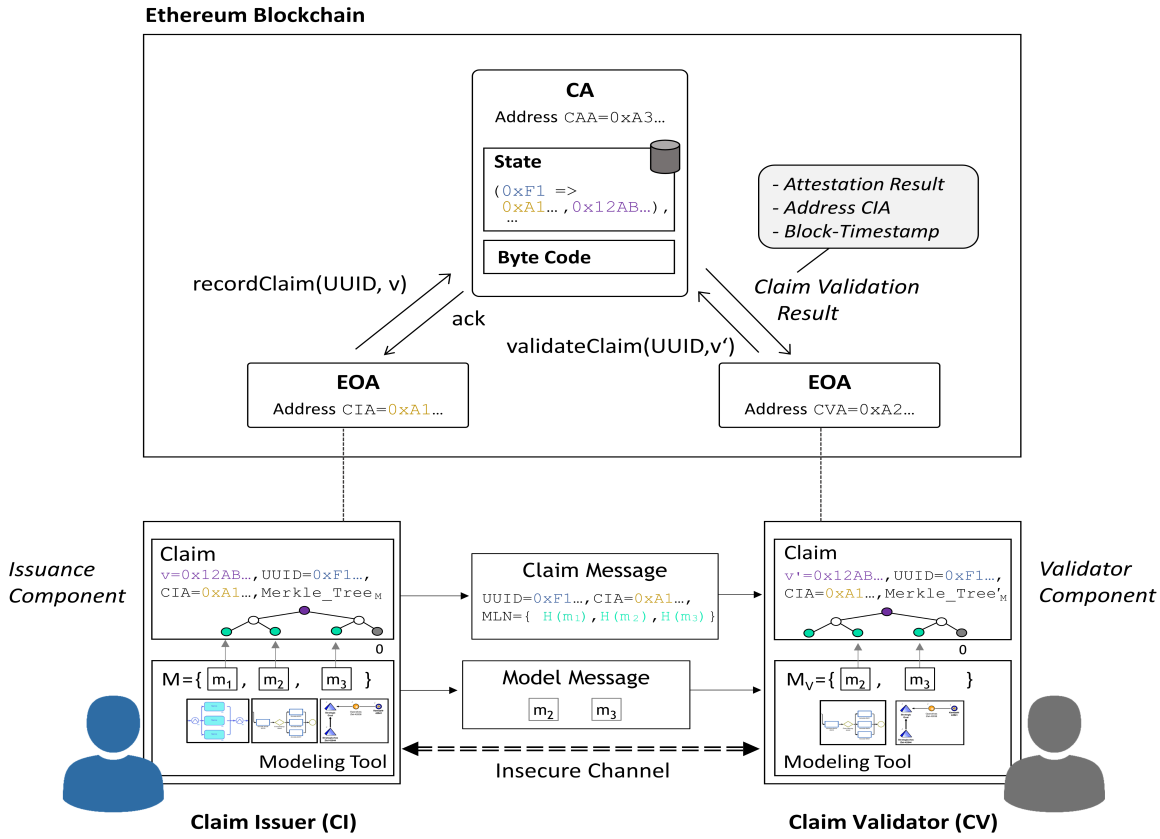

Fig. 1. Attestation Architecture

might be sent to them. By the Merkle tree, any member of M_V may be validated by claim validator without revealing M.

3) Validation of Claims: Through claim validator, the following process is carried out by invoking the *Validator Component* in the modeling tool, in accordance to Figure 1.

3.1) Import of Models: The models of the set M_V are imported by their serialization into the modeling tool. As a prerequisite, an according metamodel must be present. The metamodel might be validated through a prior attestation.

3.2) Import of the Claim: In order to validate the claim received in a message, the tuple $(UUID, CIA, MLN)$ is locally imported into the modeling tool. From the claim message, the identifiers of (a.) the model $UUID$, (b.) the EOA address of claim issuer, and (c.) the Merkle tree leaf nodes are present locally for validation.

3.3) Generation of EOA Address as User Identifier: For interacting with the CA smart contract, claim validator generates an EOA locally with a randomly generated address similar to 2.4. Alternatively, a previously generated EOA may be retrieved locally.

3.4) Validation of the Claim: The validation of the claim is performed by re-constructing the Merkle tree (3.4.1), by retrieving and validating it with the CA smart contract (3.4.2), followed by additional local validations (3.4.3). Note that in the special case of $M_V = \emptyset$, only the Merkle tree is validated, without taking models into account.

3.4.1) Re-construction of the Merkle tree: The validation of the claim establishes the existence of the models in M_V at a prior point in time. From the members of M_V and the received MLN, a content-based identifier for M is re-calculated as v'. While in the case of $|MLN| = 1$, the hash value is calculated from the element of MLN, the Merkle tree must be re-constructed in the case of $|MLN| > 1$ as $Merkle\_Tree'_M$. This is achieved by (1.) calculating the hash value for each $m_V \in M_V$, (2.) by validating that each hash value is a leaf node, i.e. $H(m_V) \in MLN$, and (3.) by re-calculating the Merkle tree from all leaf nodes by the process given in step 2.3, resulting in the root hash value v'. The attestation fails here if $\exists m_V \in M_V$ such that $H(m_V) \notin MLN$.

3.4.2) Validation with the CA smart contract: By invoking the function *validateClaim(UUID,v')*, the smart contract retrieves the claim for the given set of models identified by $UUID$ and validates it using the recorded v. In case of $v' = v$, the *attestation result* is *valid*; or else *invalid*. The function returns a *claim validation result* with (a.) the *attestation result* and, in the *valid* case, (b.) the claim issuer's identity as EOA address and (c.) a timestamp of the block containing the claim.

3.4.3) Additional validations: Depending on the use case, additional validations beyond the proof-of-existence paradigm

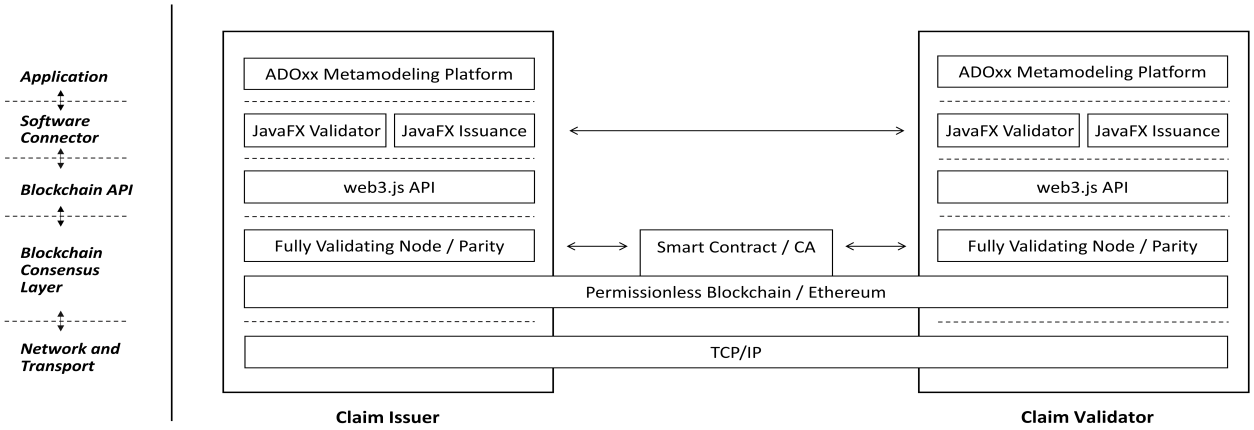

Fig. 2. Layered System Architecture Based on a Permissionless Blockchain for Communication

may be performed. From (b.), the user identity of claim issuer may be validated in cases where it is known, for example if this user represents a public certificate authority. The public-private key pair used to generate the claim issuer's address, and the private key used to send the transaction, bind the claim issuer's address to the claim. For this identity to be valid, the claim issuer must match the address returned by the smart contract. From (c.), the date and time of the block containing the transaction of the claim may be utilized for validation. Here, Ethereum acts as a trusted distributed timestamping service, such that the prior existence of information can be assumed for the time when the claim was recorded.

C. Prototypical Implementation

For evaluating the feasibility of this approach, the architecture has been implemented as a prototype model attestation system using Ethereum and ADOxx as a metamodeling platform. The software realization is based on the layers of the blockchain-based system architecture given in Figure 2.

On the basis of the network and transport layer, realized by internet protocols, a blockchain consensus layer is realized by Ethereum with Parity nodes in a fully validating configuration and the autonomous CA smart contract. This contract is deployed on Ethereum at address 0x048e82278A597c1e977ED78c4Ba20FB5caECb73A[1]. The blockchain consensus layer is accessed by the Web3.js API for providing a standardized method of interaction with the blockchain. The software connector [46] in a separate layer interacts through this API for (1.) the generation of Ethereum addresses, (2.) for the broadcasting of transactions from an EOA to the CA smart contract and (3.) for receiving transactions and event handling. Based on these features, the software connector layer hosts the Issuance and Validator components written in JavaFX. By these components, the ADOxx metamodeling platform is connected to the blockchain API, allowing for all user interactions to be handled through the platform. The user interfaces of the JavaFX implementation (see Figures 3, 5) are invoked through and controlled by AdoScript customizations of the platform.

[1]C.f. https://etherscan.io/address/<address>

IV. Evaluation

The practical applicability of the approach is evaluated on the Ethereum network through the proof-of-concept implementation. For identifying limiting factors and to demonstrate the application of the approach, the following sections outline an assumed use case, describe the generation of data in the form of models, and discuss the measurements, the setup, and the results.

A. Evaluation Use Case

One of the use cases discussed for blockchains is certification, since organizations often rely on third parties for the assurance of information bound to an identity. In many business processes, certificates are provided by external entities and are required to be verifiable with a third party. For instance, one of the first steps of a recruiting process is the validation of the applicant's certifications with third party institutions. For university degrees and courses, this approach has been discussed and tested as a blockchain application [47], [48].

Certification in recruiting processes represents a typical attestation use case since (a.) a certificate is created in the form of a document or a model by an authority such as a university, which acts as claim issuer, and (b.) the validation of the certificate may be carried out at a later point in time by anyone, e.g. an HR department acting as claim validator. After the issuance of the claim on the existence of the certificate model, no centralized server is required for validation in this decentralized attestation use case.

The ADOxx modeltype *Certificate*, implemented for this use case, is defined by a metamodel of the four classes `Institution`, `Course`, `Student` and `Grade` with according relations. A 1-to-n relation between Institution and Course indicates offered courses, another one between Course and Student represents the inscribed students of a course. 1-to-1 relations between Student and Grade as well as between Course and Grade represent the completion of a course by

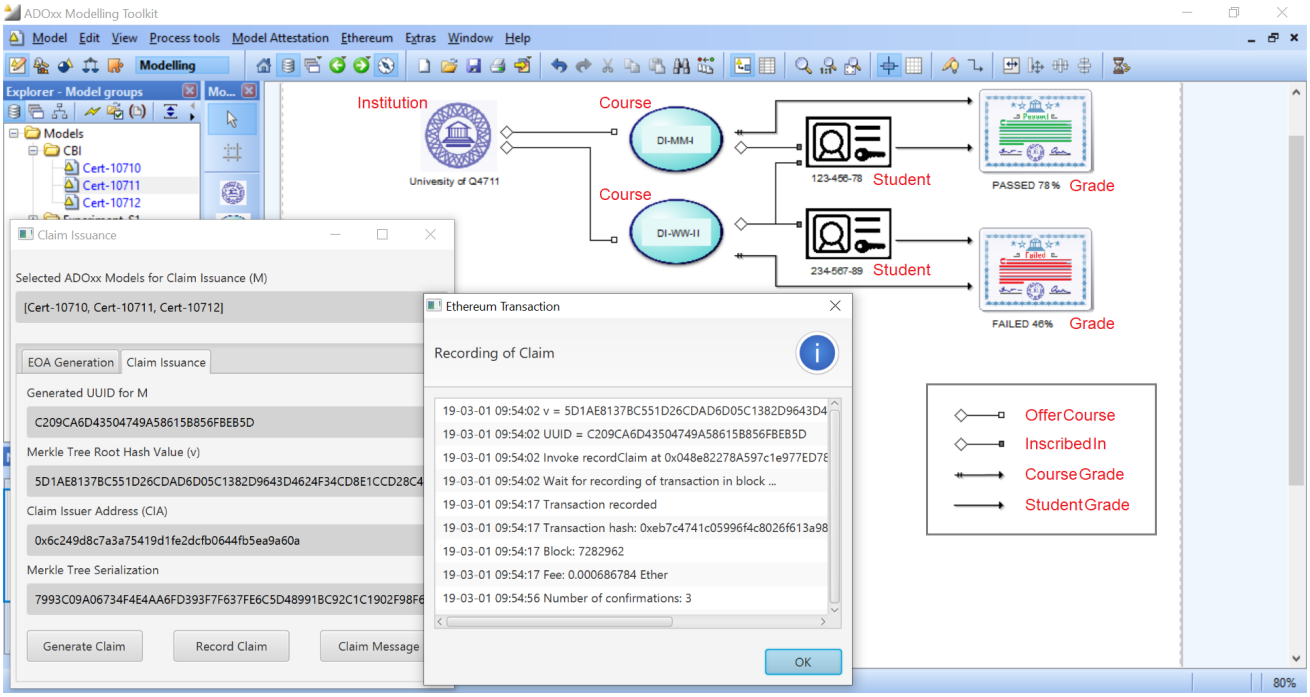

Fig. 3. Certificate Model and Issuance Component in ADOxx with Annotations (Red Font)

a student, such that a grade and the final result *passed* or *failed* are shown. Figure 3 gives an example of such a model in ADOxx. Any certificate model represents the structured information of one or more certifications, processible by the ADOxx platform for model transformation, code or document generation. For example, the model-based generation of according data models, database records or certificate pdf documents directly relies on trusted information from the model.

B. Generation of Conceptual Models

For working with conceptual models, the contents of the models need to be taken into account at the level of information instead of data. Therefore, two sets of data (D) were created. (D1) n=100 models representing individual certificates, each attesting the completion of an institution's course by one student. (D2) One single model for attesting the same information of the 100 certificates. Each model of (D1) consists of 4 elements, while the model of (D2) contains 206 elements with 3 identifying and describing attributes per element in all cases. Note that the total number of elements for (D1) is higher compared to (D2), since the models of (D1) contain redundant information about the institution, courses and students. The generation was carried out by a Perl program generating random content for all model elements.

In order to measure the overhead of multiple attestations over one individual attestation with the same information, three scenarios (S) are accounted for. In each scenario, an institution issuing certificates acts as claim issuer.

- (S1) assumes the issuance of one claim for all models from (D1), such that the institution can transfer each certificate to a student or another party acting as claim validator. 1 claim for $|M| = 100$ models is recorded, and 100 validations with $|M_V| = 1$ model are carried out.
- (S2) assumes the issuance of one claim from the model of (D2) for the purposes of the institution, e.g. for archiving and integrity checking by the institution's stakeholders acting as claim validators. 1 claim with $|M| = |M_V| = 1$ model is recorded and validated.
- (S3) assumes the issuance of one claim per certificate from each model of (D1), such that the institution can transfer each certificate to a student or another party acting as claim validator. 100 claims are recorded with $|M| = 1$ model, and 100 validations with $|M_V| = 1$ model are carried out.

(S3) is added for comparison to (S1), since it also can be carried out easily using existing file-based attestation services with individual certificate files.

C. Measurements

The goal of this evaluation is the measurement of factors which could potentially limit applicability. Especially the time and cost of recording and validating claims need to be taken into account here, due to the scalability limitations and comparatively high transaction cost of today's blockchain systems [5].

For recording a claim in accordance to the architecture, the claim issuer generates it and records it on Ethereum by broadcasting a transaction to the smart contract. The transaction contains metadata, such as the EOA address of the sender, and data for the function invocation $recordClaim(UUID, v)$ with a 128 bit UUID and a 256 bit hash value.

For validating a claim in accordance to the architecture, the claim validator retrieves data of the claim from Ethereum by invoking the function $validateClaim(UUID, v')$ of the smart contract. Since this so-called $view$ function only has read access to the locally available replication of the blockchain data, there is no transaction broadcast to the network.

Therefore, the following measurements on the Ethereum network were taken for the recording of claims in each scenario:

- Transaction input data size, i.e. the amount of data in the input data field of a transaction.
- Transaction cost, i.e. the amount of Ether currency payed, calculated through the consumed "gas", to a miner for including the transaction in a block.
- Time to record the transaction, i.e. the time it takes for the Ethereum network to include the transaction in a block and issue a transaction receipt.

In addition to the cost per claim issuance, a one-time cost for the deployment of the smart contract incurs.

D. Setup

The measurements were performed on the Ethereum main network in its so-called "Byzantium" stage [49]. The node software used was Parity Ethereum, version 2.30 beta, in a configuration for fully replicating, validating ("tracing") and indexing ("fat-db") all transactions locally. This configuration archives all transactions, however, only recent state representations are kept in a cache. With these parameters, the blockchain node was set up after 225 hours of synchronization with the network, carried out over 4.5 weeks, resulting in 201 GB of data.

E. Results

The experiments were carried out separately per scenario and are recorded with claim data in the transactions to the smart contract[2] in the blocks 7267828 (S1), 7267830 (S2), and [7267833, 7268181] (S3). Table I lists the measurements.

TABLE I
RESULTS FOR SCENARIO S1, S2, S3

	S1	S2	S3
Number of models	100	1	100
Number of claims issued	1	1	100
Number of validations	100	1	100
Size of transactions	544 bit	544 bit	54.4 kbit
Cost of transactions in Ether	6.8678E-4	6.8678E-4	6.8670E-2
Time to record transactions	47 s	60 s	114.44 min

1) Transaction Data Size: S1 and S2 required one transaction, each recording one claim in 544 bit per transaction. The 100 claims of S3 required 100 transactions using the same amount of data per transaction as S1 and S2. The size of 544 bit per transaction far exceeds the data for UUID and Hash Value, since the UUID only utilized half of the word length of 256 bit in Ethereum. With each transaction, 128 bit of superfluous zero values were transmitted.

2) Transaction Cost: The per-transaction cost for S1, S2 and S3 was 6.8678E-4 Ether or 0.095 USD[3]. For S1 and S2, this is the total cost, while S3 resulted in 6.8670E-2 Ether or 9.53 USD total cost. The transaction cost is dependent on the gas price set by the claim issuer and the size of the transaction. In order to prevent delays in recording transactions and to minimize the impact of network load on the measurement of time, the gas price was set to about 133% of the median gas price of the last 500 blocks, i.e. 8E-9 Ether. The one-time cost to deploy the contract was 1.9782E-3 Ether or 0.27 USD[4].

3) Time to Record Transactions: The load of the network and the chosen transaction cost influence the time to include a transaction in a block. For S3, the median time was 46.2 s as indicated by the box plot in Figure 4, however, the maximum time was 360 seconds. In total, the time to confirm the 100 transactions of S3 was 1.9 hours. These measurements are specific to the situation in the network at the time of the test and might deviate further when load on the network increases. In addition, local processing time was taken into account for generating UUIDs, hash values and Merkle trees, however, the time for S1, S2, and, S3 was below 0.1 s.

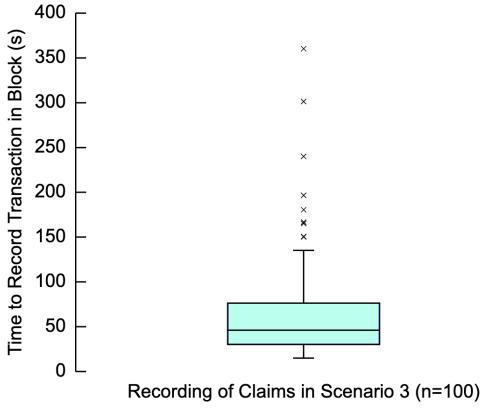

Fig. 4. Time to Record Transactions

V. DISCUSSION

The model-based attestation use case demonstrates the application of domain-specific models for the domain of certification. Here, an attestation was carried out for individual models representing certificates in S1 and S3, and for one model in S2. While S2 may be thought of as a special case for attesting to all the information from S1 or S3 internally by a single claim, it demonstrates the impact of modeling decisions in contrast to file-based attestations. In the metamodeling platform, elements of this model can be transformed into the individual models of S1 and S3 easily, e.g. using copy and paste, by specifying functional expressions, or by procedural programs, possibly with transformations. For the typical use case of handing out certificates, S1 shows the advantage of

[2]C.f. https://etherscan.io/txs?a=0x048e82278a597c1e977ed78c4ba20fb5caecb73a

[3]Given the exchange rate of 138.81 Ether/USD indicated by Etherscan.io

[4]Given the exchange rate of 136.48 Ether/USD indicated by Etherscan.io

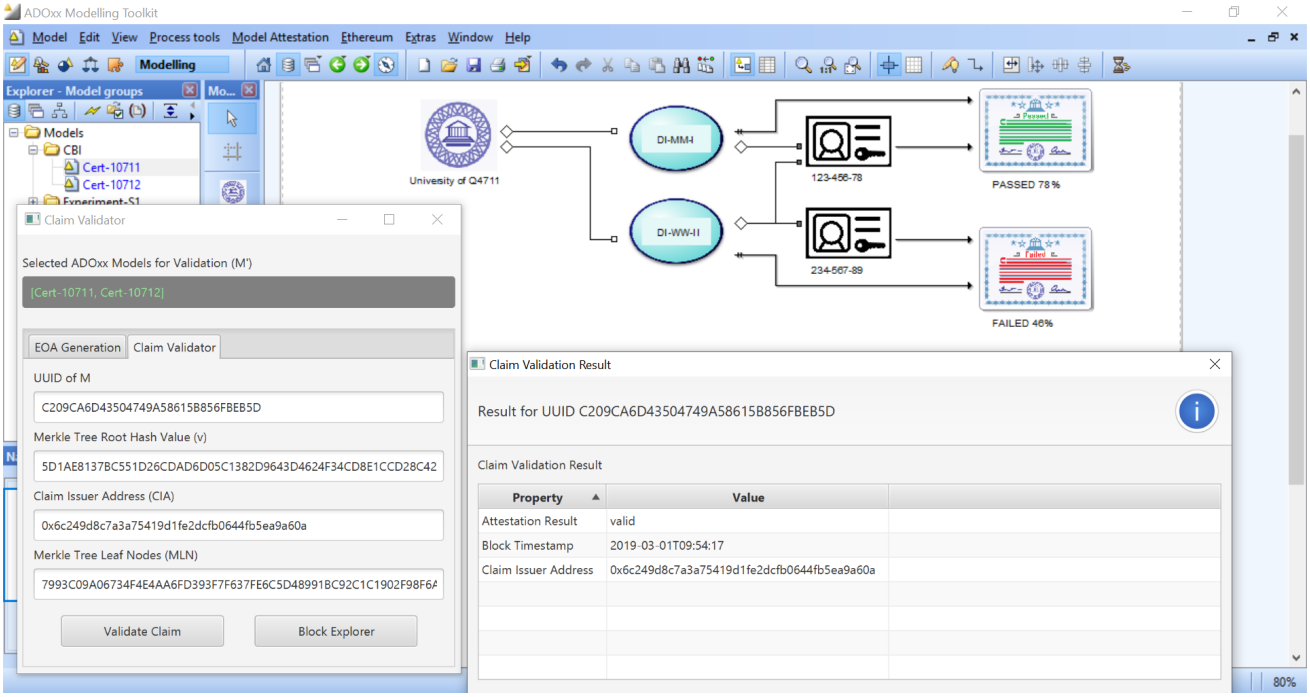

Fig. 5. User Interface of the Validator Component in ADOxx

using hash-based tree data structures in comparison to the issuance of individual claims per model.

While the results demonstrate the applicability of the approach, they also highlight several practical limitations. Firstly, the lengthy and resource intensive setup process for a fully validating Ethereum node is a one-time expenditure which makes the approach hard to deploy and impractical for performance or throughput constrained computing environments. Secondly, as a function of the number of claims issued, the transaction cost and the time to record transactions limit the applicability to use cases where (a.) a cost on the order of tens of USD cents per claim is acceptable and (b.) a delay on the order of tens of seconds to minutes per claim is acceptable. In the scenario of issuing course certificates this might not be an issue, however, the variability in confirmation time does not allow for guarantees which might be necessary for other cases.

VI. Conclusion and Outlook

An attestation approach based on a metamodeling platform allows the modeling of attestations in the context of a specific domain, and the recording and validation of claims based on it. With appropriate attestation methods and data structures, identity applications beyond the ones build into today's blockchain systems can be implemented. While this has been demonstrated for the issuance of certificates here, it might be extended to any domain. However, with the practical applicability being limited by today's blockchain systems, further research on limiting technical and sociotechnical factors and their influence on identity use cases is necessary to evaluate the promising idea of decentralized identity. Arguably, the most interesting property of this fundamental concept of identity is its broad applicability. In combination with capturing arbitrary domains in domain-specific conceptual models, decentralized and self-sovereign identity solutions might be specified by models in order to realize model-based identity solutions.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," http://www.bitcoin.org/bitcoin.pdf, 2008, accessed on 2019-02-28.

[2] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling Blockchain: A Data Processing View of Blockchain Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[3] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," in *34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015)*, Sofia, Bulgaria, 2015.

[4] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A Taxonomy of Blockchain-Based Systems for Architecture Design," in *2017 IEEE International Conference on Software Architecture (ICSA)*, Gothenburg, Sweden, 2017, pp. 243–252.

[5] S. Kim, Y. Kwon, and S. Cho, "A Survey of Scalability Solutions on Blockchain," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 1204–1207.

[6] S. A. Abeyratne and R. P. Monfared, "Blockchain Ready Manufacturing Supply Chain Using Distributed Ledger," *International Journal of Research in Engineering and Technology*, vol. 05, no. 09, pp. 1–10, 2016.

[7] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in *Proceedings of the 50st Hawaii International Conference on System Sciences (HICSS-50)*, Big Island, Hawaii, USA, 2017.

[8] D. Linke and S. Strahringer, "Integration einer Blockchain in ein ERP-System für den Procure-to-Pay-Prozess: Prototypische Realisierung mit SAP S/4HANA und Hyperledger Fabric am Beispiel der Daimler AG," *HMD Praxis der Wirtschaftsinformatik*, vol. 55, no. 6, pp. 1341–1359, 2018.

[9] V. L. Lemieux, "Trusting records: Is Blockchain technology the answer?" *Records Management Journal*, vol. 26, no. 2, pp. 110–139, 2016.

[10] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, and M. Sabadello, "W3C Draft: Decentralized Identifiers (DIDs) v0.11." https://w3c-ccg.github.io/did-spec/, 2019, accessed on 2019-02-28.

[11] M. Swan, *Blockchain: Blueprint for a New Economy*, first edition ed. Beijing : Sebastopol, CA: O'Reilly, 2015.

[12] OpenTimestamps, "OpenTimestamps: A timestamping proof standard," https://opentimestamps.org/, 2019, accessed on 2019-02-28.

[13] J. Kirk, "Could the Bitcoin network be used as an ultrasecure notary service?" https://www.computerworld.com/article/2498077/could-the-bitcoin-network-be-used-as-an-ultrasecure-notary-service-.html, 2013-05-23, accessed on 2019-02-28.

[14] M. Crosby, "BlockChain Technology: Beyond Bitcoin," *Applied Innovation Review*, no. 2, p. 16, 2016.

[15] A. Fayoumi and P. Loucopoulos, "Conceptual modeling for the design of intelligent and emergent information systems," *Expert Systems with Applications*, vol. 59, pp. 174–194, 2016.

[16] K. Sandkuhl, H.-G. Fill, S. Hoppenbrouwers, J. Krogstie, F. Matthes, A. Opdahl, G. Schwabe, Ö. Uludag, and R. Winter, "From expert discipline to common practice: A vision and research agenda for extending the reach of enterprise modeling," *Business & Information Systems Engineering*, vol. 60, no. 1, pp. 69–80, 2018.

[17] E. J. Sinz, "On the Evolution of Methods for Conceptual Information Systems Modeling," in *The Art of Structuring: Bridging the Gap Between Information Systems Research and Practice*, K. Bergener, M. Räckers, and A. Stein, Eds. Cham: Springer International Publishing, 2019, pp. 137–144.

[18] D. Karagiannis, H. C. Mayr, and J. Mylopoulos, Eds., *Domain-Specific Conceptual Modeling*. Springer Berlin Heidelberg, 2016.

[19] H.-G. Fill and F. Härer, "Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling," in *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS-51)*, Waikoloa Village, Hawaii, USA, 2018, pp. 4045–4054.

[20] H.-G. Fill, "Applying the Concept of Knowledge Blockchains to Ontologies," in *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019)*, Stanford University, Palo Alto, California, USA, 2019.

[21] G. Gal and W. McCarthy, "Implementation of rea contracts as blockchain smart contracts," in *Proceedings of the 12th International Workshop on Value Modeling and Business Ontologies 2018*, Amsterdam, Netherlands, 2018.

[22] V. Buterin, "Ethereum: The Ultimate Smart Contract and Decentralized Application Platform," http://web.archive.org/web/20131228111141/http://vbuterin.com/ethereum.html, 2013, accessed on 2019-02-28.

[23] V. Buterin, G. Wood, and J. Wilcke, "Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform," https://github.com/ethereum/wiki/wiki/White-Paper, 2014, accessed on 2019-02-28.

[24] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," https://gavwood.com/paper.pdf, 2014, accessed on 2019-02-28.

[25] H.-G. Fill and D. Karagiannis, "On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform," *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 8, no. 1, pp. 4–25, 2013.

[26] F. Härer, "Decentralized Business Process Modeling and Instance Tracking Secured By a Blockchain," in *Proceedings of the 26th European Conference on Information Systems (ECIS)*, Portsmouth, UK, 2018.

[27] P. J. Leach, M. Mealling, and R. Salz, "Rfc4122 a Universally Unique IDentifier (UUID) URN Namespace," https://tools.ietf.org/html/rfc4122, 2005, accessed on 2019-02-28.

[28] N. Paskin, "Digital Object Identifiers for scientific data," *Data Science Journal*, vol. 4, pp. 12–20, 2006.

[29] W3C, "URIs, URLs, and URNs: Clarifications and Recommendations 1.0," https://www.w3.org/TR/uri-clarification/, 2001, accessed on 2019-02-28.

[30] Q. H. Dang, *Secure Hash Standard. NIST FIPS 180-4*. National Institute of Standards and Technology, 2015, accessed on 2019-02-28.

[31] M. J. Dworkin, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. NIST FIPS 202*. National Institute of Standards and Technology, 2015.

[32] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*. Princeton University Press, 2016.

[33] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The First Collision for Full SHA-1," in *Advances in Cryptology – CRYPTO 2017*, J. Katz and H. Shacham, Eds., vol. 10401. Springer International Publishing, 2017, pp. 570–596.

[34] S. Chacon and B. Straub, *Pro Git*, second edition ed. Apress, 2014.

[35] A. Tenorio-Fornés, S. Hassan, and J. Pavón, "Open Peer-to-Peer Systems over Blockchain and IPFS: An Agent Oriented Framework," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems - CryBlock'18*. Munich, Germany: ACM Press, 2018, pp. 19–24.

[36] F. Härer and H.-G. Fill, "A Comparison of Approaches for Visualizing Blockchains and Smart Contracts," in *22nd International Legal Informatics Symposium (IRIS 2019)*. Salzburg, Austria: Editions Weblaw, 2019.

[37] Diar, "Cryptocurrency Exchanges Mark Record Year in Bear Market," *Diar*, vol. 3, no. 1, 2019, https://diar.co/volume-3-issue-1/, accessed on 2019-02-28.

[38] A. M. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and Dapps*. S.l.: O'Reilly Media, 2018.

[39] Etherchain, "Average block time of the Ethereum Network," https://www.etherchain.org/charts/blockTime, 2019, accessed on 2019-02-28.

[40] Ethstats, "Ethereum Network Status," https://ethstats.net/, 2019, accessed on 2019-02-28.

[41] G. Coker, J. Guttman, P. Loscocco, J. Sheehy, and B. Sniffen, "Attestation: Evidence and Trust," in *Information and Communications Security*, ser. Lecture Notes in Computer Science, L. Chen, M. D. Ryan, and G. Wang, Eds. Springer Berlin Heidelberg, 2008, pp. 1–18.

[42] P. Dunphy and F. Petitcolas, "A First Look at Identity Management Schemes on the Blockchain," *IEEE Security & Privacy*, vol. 16, 2018.

[43] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, "uport a platform for self-sovereign identity," http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf, 2016, accessed on 2019-02-28.

[44] J. Andrieu, L. Sunny, and O. Nate, "Verifiable Claims Use Cases W3C Working Group Note 08 June 2017," https://www.w3.org/TR/verifiable-claims-use-cases/, 2017, accessed on 2019-02-28.

[45] A. M. Antonopoulos, *Mastering Bitcoin: Programming the Open Blockchain*, second edition ed. Sebastopol, CA: O'Reilly, 2017.

[46] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The Blockchain as a Software Connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. Venice, Italy: IEEE, 2016, pp. 182–191.

[47] I. Konstantinidis, G. Siaminos, C. Timplalexis, P. Zervas, V. Peristeras, and S. Decker, "Blockchain for Business Applications: A Systematic Literature Review," in *Business Information Systems*. Springer International Publishing, 2018, pp. 384–399.

[48] H. Sun, X. Wang, and X. Wang, "Application of Blockchain Technology in Online Education," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 13, no. 10, pp. 252–259, 2018.

[49] J. Hudson, "Ethereum Constantinople Upgrade Announcement," https://blog.ethereum.org/2019/01/11/ethereum-constantinople-upgrade-announcement/, 2019, accessed on 2019-02-28.