Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling

Abstract

Blockchains permit to store information in a tamper-resistant and irrevocable manner by reverting to distributed computing and cryptographic technologies. The primary purpose is to keep track of the ownership of tangible and intangible assets. In the paper at hand we apply these concepts and technologies to the domain of knowledge management. Based on the explication of knowledge in the form of enterprise models this permits the application of so-called knowledge proofs for a. enabling the transparent monitoring of knowledge evolution, b. tracking the provenance, ownership, and relationships of knowledge in an organization, c. establishing delegation schemes for knowledge management, and d. ensuring the existence of patterns in models via zero-knowledge proofs. To validate the technical feasibility of the approach a first technical implementation is described and applied to a fictitious use case.

1. Introduction

In the last years, the increasing adoption of the virtual currency Bitcoin has sparked interest in the underlying technologies that enable the secure exchange of assets in an electronic manner [3]. At the core of these so-called blockchain technologies stand protocols that define the exchange and storage of information using cryptography. These ensure the tamper-resistant, decentralized, and irrevocable storage of transactions between parties in a transparent and entirely virtual, electronic environment [24]. Although the primary area of application is in the financial domain, several proposals have been made to apply blockchain technologies to other fields. Examples include the internet of things for storing the communication between smart devices, the verification of the authenticity of products in e-commerce or the decentralized storage of information about the domain name system (DNS) [26].

When extending the notion of assets in general to the very core of organizations, one immediately recognizes knowledge as one of the most important and valuable resources and a critical factor for remaining competitive [5, 15]. Similar as for financial value, also knowledge as the intellectual capital of an organization needs to be securely stored and, if necessary, shared between parties [25]. In addition, it is vital to track the provenance and ownership of knowledge as well as manage its distributed elicitation and its evolution in order to make it available to all relevant parties.

Based on these similarities between knowledge and electronic assets, we will report about an exploratory research approach for applying the concepts of blockchain technologies to the area of knowledge management. For accomplishing the transition between knowledge products such as know-what, know-why, know-who, and know-when and electronically processable information products we revert to conceptual models [21, 30]. These permit on the one hand to represent explicit, clearly formulated types of knowledge in a machine-processable format [22, 6]. Thereby the major benefit is, that the Blockchain technologies enable us to identify the actors who created or modified the models (know-who) as well as to proof the provenance of their content (know-what) without revealing it in the fashion of zero-knowledge proofs. Thereby, the people behind the models and their knowledge can be accessed as well, e.g. to further investigate the know-why. By using blockchain technologies, the content of models will be stored in a tamper-resistant and irrevocable format. This includes the storage of explicit permissions for accessing and modifying their content as well as time-stamps, thus enabling the recording of the know-when. This corresponds to the considerations made for security in knowledge management regarding confidentiality, integrity, and availability [16]. This type of a blockchain will be denoted as Knowledge Blockchain.

The remainder of the paper is organized as follows. In Section 2 we will give an introduction to blockchain technologies to provide the foundations for Knowledge Blockchains. The approach itself will be described in Section 3. In Section 4, a prototypical implementation will be presented to form the basis for a use case in Section 5. Work related to our approach will be discussed in Section 6, followed by a conclusion in Section 7.

2. Foundations

In this section, we briefly introduce fundamental blockchain concepts and the according terminology as it is commonly used today.

2.1. Blockchain Technologies

At the core of technologies required for realizing blockchains are cryptographic hash functions and public key cryptosystems. A cryptographic hash function H(M) permits summarizing a message M of any length with a fixed length pseudo-random output value V, e.g. with a length of 256 bit. Cryptographic hash functions satisfy the properties of collision and preimage resistance. This means that it is computationally infeasible to a. find two messages M, M' which produce equal hash values as in H(M) = H(M'), b. compute M from V, and c. to derive M' from an existing M so that H(M) = H(M') holds [24].
For summarizing two messages M1 and M2, it is trivial to apply a hash function to the concatenation of their individual hash values V1 and V2 as in $H_{V1||V2}$ = H(H(M1)||H(M2)). In this fashion, values of a set V with |V| mod 2 = 0 can be hashed pairwise to create |V|/2 hash values, which are subjected to the same operation recursively so that a binary tree is created. The leafs of the resulting binary tree are individual values, whereas any node summarizes its two children and the root summarizes all values of the tree. These trees are called Merkle trees based on their inventor [23]. Theoretically, it would be possible to concatenate more than two hash values, however, the binary tree suggested by Merkle has the advantage of allowing a traversal in logarithmic space and time as demonstrated by Szydlo [29]. To prove that an element of V is included in the Merkle tree, at most $\log_2(|V|)*2$ operations need to be carried out. In the fashion of zero-knowledge proofs, this can be leveraged to efficiently show that a message is part of the Merkle tree. To prove membership of a message, other messages do not need to be known, as it is sufficient to know the hash values of the Merkle tree. To prove this for M1, its hash value V1 is re-calculated, concatenated to the existing hash value of its neighbor V2 and hashed as in H(V1||V2). The resulting hash value needs to be equal to the one stored as parent of V1 in the Merkle tree. This operation is applied recursively until the root hash is reached. This means that on every level of the tree, excluding the root, only the hash value in the neighbor node needs to be known to conduct the proof.

Further, by reverting to public key cryptography additional information security features are provided. In a public key cryptosystem, confidentiality of messages can be achieved without a pre-shared secret. For achieving this, a sender and a receiver of a message both have pairs of public and private keys. A crypto-algorithm then allows for a. the encryption of a message using the public key of the receiver, who decrypts it with his corresponding private key, and b. the signing of a message with the private key of the sender so that the receiver can verify the signature using the sender's public key [24].

By putting cryptographic hash functions and public key cryptosystems together, the basic functions of blockchains can be described. In particular, blockchains store data in a way that allows for efficient integrity checks by linking individual blocks – i.e. information parts – using hash values, so that any Block B_i with i > 0 contains a hash value that summarizes the preceding block B_{i-1}. The last block of the resulting chain therefore summarizes the whole chain which is linked back to B_0, commonly referred to as genesis block. A block B_i consists of a block header and data. Data is a Merkle tree which contains individual data values as leaf nodes, summarized by the root hash value MR_i. The block header consists of MR_i and the hash value of the preceding block header BH_{i-1}. Thus, a block B_i can be summarized by hashing its block header BH_i using $H(H(BH_{i-1})||MR_i)$. The whole chain is summarized by the block header of its last block. A modification of any data value in B_i would result in a changed hash value $H(BH_i)$ which breaks the chain, i.e. B_{i+1} is no longer linked to B_i, as BH_{i+1} no longer contains the hash value of the preceding block. Integrity of all data in the chain can be assumed if the hash value of the block header of the last block remains unchanged. This makes blockchains well suited to store data in an immutable and thus tamper-resistant fashion.

Any new block is appended as part of a mining process which enforces the specific rules of a blockchain. Entities carrying out the mining will record new data to be stored in the blockchain and check whether or not the protocol is violated by that data. In case that changes shall be applied to information stored in the blockchain, transactions are specified and sent to the miners. The transactions need to contain

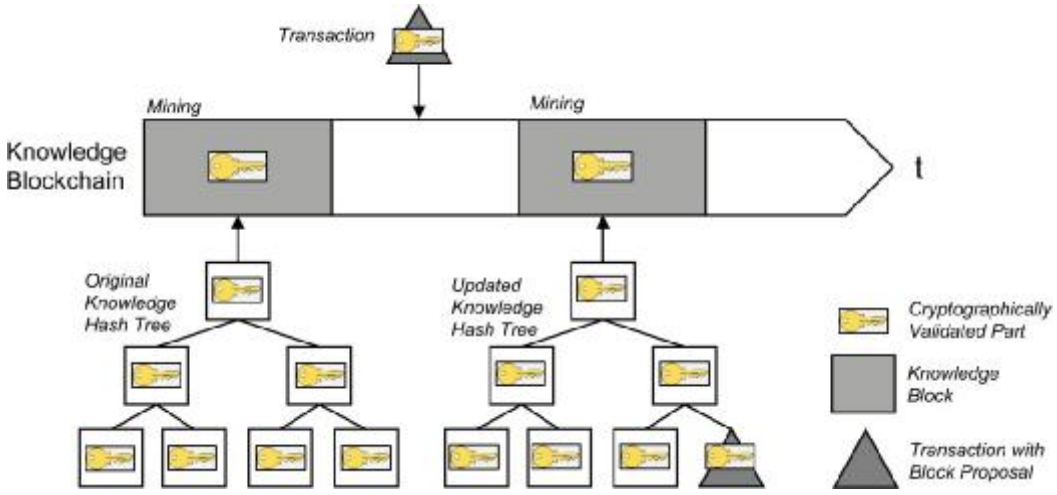

Figure 1: Concept of Knowledge Blockchains

information about the data to be changed (e.g. a transfer of financial assets in bitcoin) and the corresponding identity wishing to conduct that change. The identity is ensured through digital signatures based on the above described public key cryptosystems. In the case of distributed, public blockchains that operate in a so-called permission-less fashion, miners are selected randomly based on the presentation of solutions for cryptographic puzzles. These puzzles are computationally hard deciphering problems that can only be solved by trying a number of possible solutions (proof of work). If a miner finds a solution, it is entitled to add information to the blockchain. Further details on the mining process can be found in [24].

In contrast to public blockchains, private or permissioned blockchains are not necessarily distributed. As a consequence, the access to miners is restricted and transactions need to authenticate themselves against the miner. Furthermore, domain-specific rules for the mining process and for generating new blocks can be defined so that an organization may use its own rules and models when data is added to the blockchain. This includes detailed specification of permissions for all identities interacting with the blockchain.

3. The Concept of Knowledge Blockchains

With the foundations described in the previous section we can now advance to the presentation of Knowledge Blockchains. The concept is based on the assumption that knowledge can - at least partially - be made explicit in the form of conceptual models. This view is common to many approaches in the area of enterprise modeling where knowledge about organizational entities such as business processes, IT systems, data, ontologies, actors and the like is today documented in the form of semi-formal or formal conceptual models [17]. These models can be processed by algorithms and humans alike in the sense of knowledge Information Systems and thus act not only as a basis for communication but also as input for machine-based analyses and simulations [1].

The massive use of enterprise models in organizations requires careful management and appropriate IT support. In cases where a single organization already stores several hundreds or thousands of such models [28], the challenge becomes not only to technically handle the contained information and make it accessible to users but also to provide mechanisms for adequately handling knowledge aspects [14, 21]. In this context, the concepts and technologies behind blockchains offer an extension to the traditional handling of enterprise models. Especially, the core aspect of blockchains to ensure irrevocable, tamper-resistant storage of information in a transparent way without trusted third parties has the potential to fundamentally change the way how knowledge in the form of enterprise models is stored and processed in and across organizations.

As shown in Figure 1, Knowledge Blockchains store information in knowledge blocks that contain cryptographically-validated information parts. In line with the mechanisms in typical blockchains today, these information parts are structured themselves in the form of binary hash trees that permit to efficiently ensure the integrity of the knowledge blocks. If changes to the information shall be made, transactions are sent to the blockchain and processed by miners in the mining process. Thereby, it is checked whether all rules defined for the changes to be conducted are met.

In the following we will describe the required extensions for enterprise modeling languages in order to use them in Knowledge Blockchains. This will be followed by the presentation of the structures of blocks and the mechanisms for permission management and delegation. For processing blocks, a specific mining procedure will be added that ensures the integrity of the Knowledge Blockchain and permits the application of so-called knowledge proofs. These will stand for the concrete application of the Knowledge Blockchain to tasks in knowledge management.

3.1. Required Extensions of Modeling Languages

When realizing a modeling language in a modeling tool, it typically depends on the underlying tool-platform how the language is to be implemented [20]. The extensions of such modeling languages/knowledge representations as required for Knowledge Blockchains abstract from the technical implementation and define generic attributes that need to be available. With the description of a concrete implementation in Section 4 we will show how these attributes can be translated to a technical platform.

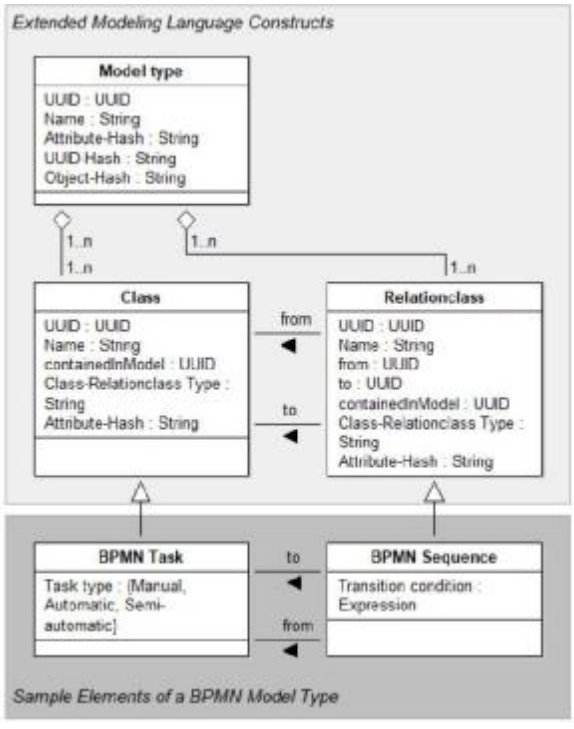

Figure 2: Extensions of modeling language constructs for enabling Knowledge Blockchains and two sample entities of the BPMN modeling language

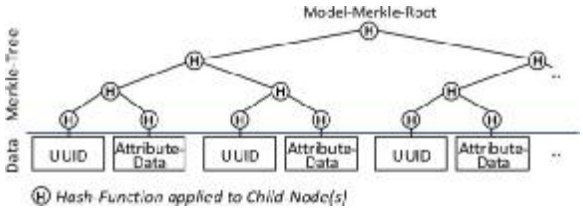

Figure 3: Model Merkle tree

A fundamental requirement for Knowledge Blockchains is the ability to uniquely identify any element in a conceptual model. Therefore, we revert to UUID (universally-unique-identifiers) attributes for model types, classes and relationclasses in the modeling language – see Figure 2. Any instance of these entities then needs to contain a correct UUID value to distinguish it from any other element. This also applies to the description of relations via from and to attributes and the containedInModel attribute to identify the assignment to a concrete model. Furthermore, attributes for containing hash values of attribute names and values (Attribute-Hash) and for all objects contained in a model (Object-Hash) are added. In the process of mining, these hash values will be calculated in the form of a Merkle tree as shown in Figure 3.

3.2. Representation of Blocks

For any blockchain-based application it needs to be decided how information is represented and stored on the chain. The structure derived for blocks in Knowledge Blockchains is shown in Figure 4. At the bottom two Merkle trees are contained in each block: one for representing the hashes of the content of enterprise models and the other one for storing the hashes about permissions. The latter aspect will be discussed in the following Section 3.3. For both trees, also the Merkle root hash is available which becomes part of the Block-Header. In the Block-Header, the hash of the Previous Block, a Timestamp, and the full content of the models and permissions in XML format is further added. For operating in distributed environments under the paradigm of permission-less ledgers, a Nonce (number used only once) is optionally available. This is used to solve cryptographic puzzles for deciding on the next miner to add a block to the chain – for details see [24]. All data in the Block-Header except the Nonce and the Timestamp is signed with the private key of the party submitting a transaction to produce a digital signature. The resulting hash value is stored in the Header-Signature. In this way, the identity of the submitting party is tied to a block. When a miner has solved the cryptographic

puzzle and successfully checked the information contained in the block, it adds the Nonce and the Timestamp values, computes a hash value of the whole block and signs this hash value with its private key. These values are then contained in Block-Hash and Block-Signature.

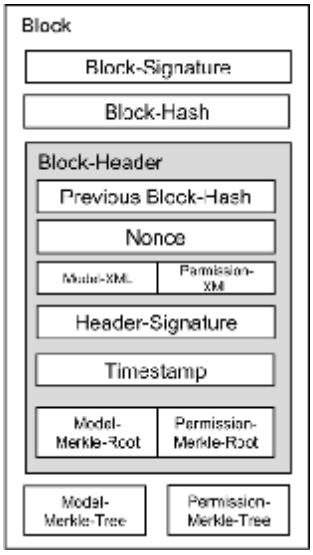

Figure 4: Structure of final knowledge blocks

3.3. Permission Management and Delegation

An essential feature of Knowledge Blockchains is to specify in a very detailed way who is allowed to conduct which changes on the blockchain. These rights are described in permission models which are an inherent part of every block. As shown in Figure 5, three types of permissions are currently available: Create and Delete Permissions that assign an identity to create or delete models, objects, relations, and attribute values and Transfer Permissions that allow an identity to transfer some or all of its permissions to another identity.

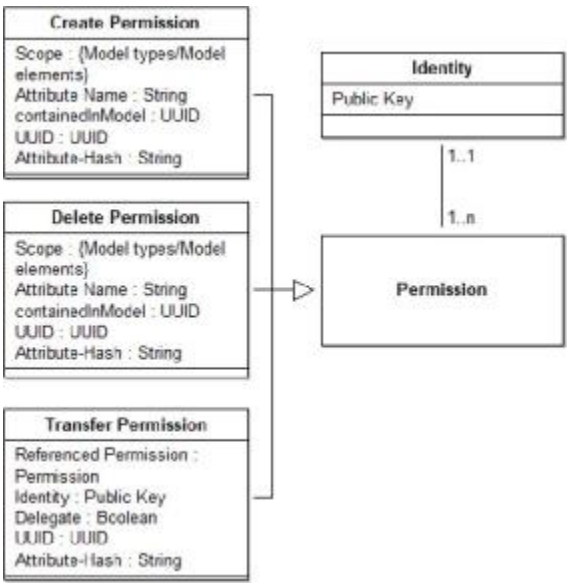

Figure 5: Structure of permission models

Identities in the permission model are represented by their Public Keys. In the genesis (initial) block of Knowledge Blockchains all permissions for all entities are assigned to the creator of the blockchain. He or she can then decide through subsequent transactions if and how these rights are to be delegated to other identities.

3.4. Mining Rules

When new transactions are sent to the blockchain they are checked for their conformance to the rules set by the permission model in the previous block. The mining may either be conducted by authenticated miners in the case of a permissioned Knowledge Blockchain or by randomly selected miners in case of permission-less Knowledge Blockchains.

For conducting the checks during the mining as shown in Figure 6, it is assumed that for each model submitted in the transaction, hash values have been computed a. for the attributes of objects and relations, for all objects and relations in a model, and for all combinations of objects, relations, or models and their corresponding UUIDs, and b. for all combinations of these hash values, thereby forming a Merkle tree up to the Merkle root hash. The same is assumed for the submitted permission models, again up to the Merkle root hash.

The outcome of the mining process is that either the requested changes are conducted and a new block is added to the Knowledge Blockchain. Or, that permissions have been found to be invalid and the transaction is declined.

3.5. Knowledge Proofs

By building upon the description of the properties of Knowledge Blockchains we can now advance to the discussion of their application in knowledge management based on the aforementioned knowledge products [21]. The first application domain is the transparent monitoring of knowledge evolution, respectively the dimensions of know-what and know-when. For tracking the evolution of knowledge as represented in the form of enterprise models, Knowledge Blockchains permit to retrieve in detail how knowledge has evolved over time. This is possible through investigating the blocks containing all changes

in models and conducting proof procedures for checking the integrity of all model contents.

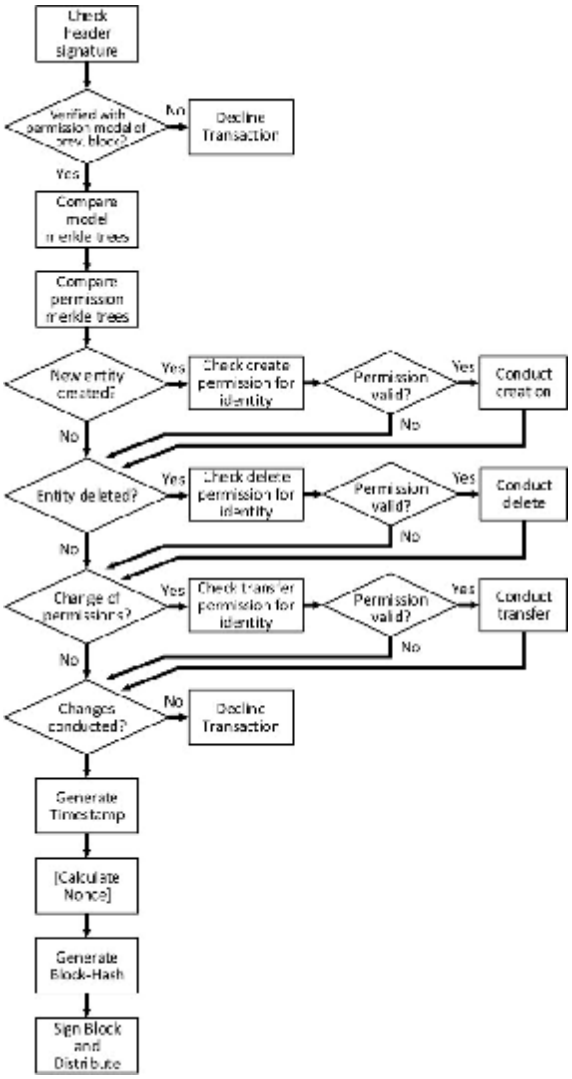

Figure 6: Verification of mining rules in Knowledge Blockchains

This directly leads to the second application domain for tracking the provenance, ownership, and relationships of knowledge in an organization. Not only can the structure of knowledge in the form of enterprise models be analyzed in a reliable manner. Through the availability of digital signatures for the blocks in the Knowledge Blockchain it can also be verified who has conducted which change and who has empowered that person to do so (delegation schemes). This aspect covers the know-who and know-why.

Finally, proofs can be conducted to ensure that certain patterns are contained in existing enterprise models. By computing the hash values for a given model pattern and applying corresponding Merkle proofs, it can be verified whether a model pattern is contained in a model on the Knowledge Blockchain. In this way, it can be proven – e.g. to external authorities – that certain knowledge is present in an organization and that it is used in a specific way, e.g. as part of the compliance checking of business processes through auditors (know-what, know-who, know-when). Such a proof covers the existence of individual model elements only, to support part of a potentially more complex compliance check. In reference to zero-knowledge proofs, the existence of knowledge as part of a model may be proven without revealing the contents of the model. In order to prove the existence of a model element, its attribute data is specified and a hash function is applied to it, possibly by auditors.

To conduct the proof as described in Section 2, knowledge of the Merkle tree which stores the model is sufficient (Figure 3). Thus, attribute data may be removed before the audit. As of now, pattern matching uses an exact match approach.

4. Prototypical Implementation

To demonstrate the applicability of a Knowledge Blockchain and to show that such a system can be implemented in practice, we created a prototypical implementation. The prototype does not cover all mentioned functionalities and primarily serves as a testbed and simulation environment for experimenting with the concept of Knowledge Blockchains. We discuss the components of the implementation, its capabilities and limitations as well as further development.

As a basis for the implementation we chose the ADOxx meta modeling platform [12]. This choice was due to the prior successful application in many industry and research projects. The core component is an ADOxx library consisting of three model-types to specify blockchains, business process models in BPMN, and permission models. For illustrative purposes, all model types have been assigned a graphical notation. The library has been extended with algorithms for a. sending block proposals in the form of transactions, b. mining new blocks, and c. verifying existing blocks. A custom-developed dynamic link library (DLL) is used to generate UUIDs, calculate SHA-256 hash values[1] and as an interface to the OpenSSL library for elliptic curve cryptography (ECC) for public key cryptosystems [18].

[1] NIST: Secure hash standard (shs). Retrieved 31-05-2017. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

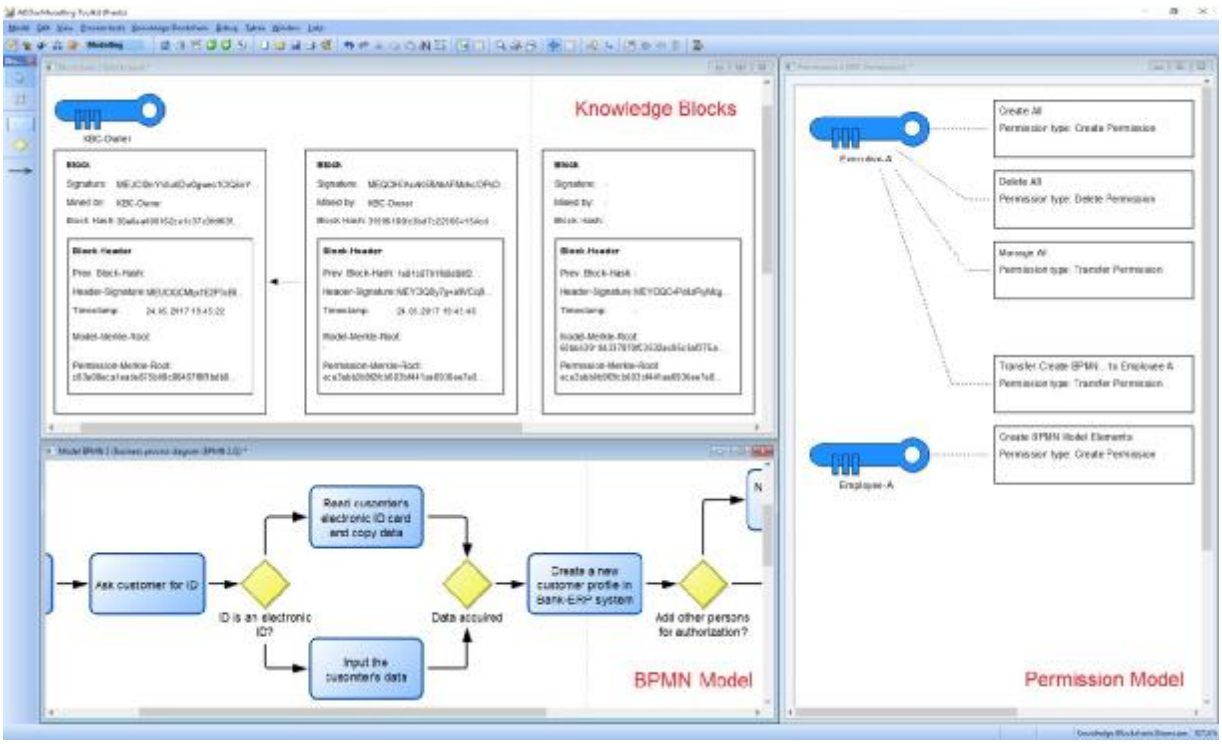

Figure 7: Screenshot of the prototype in ADOxx showing instances of a blockchain model type (upper left), a permission model type (far right), and a sample business process as a BPMN model (lower left)

By using the ADOxx library, a new Knowledge Blockchain can be created by specifying a blockchain owner with a public-private-key-pair. Basic business process models can currently be constructed using BPMN Task and Gateway elements, connected by sequence flows. Permissions are specified and assigned to identities, for which public- and private-keys are created. When sending a block proposal, business processes and a permission model are specified, stored in Merkle trees and signed. Mining is initiated by a designated miner; in this process, the platform enforces permissions concerning the creation of process elements and permission delegation. Further permissions will be added in the future. As a conceptual demonstration, we implemented a permissioned blockchain in a local environment. For the future, it is planned to evaluate also a distributed version in a permission-less fashion.

5. Description of a Fictitious Use Case and Discussion

For illustrating the application of the concept of Knowledge Blockchains we will revert in the following to a fictitious use case.

It builds upon a scenario in the domain of banks that has been used for scientific research before [11]. In this scenario, the steps and decisions for opening an account at a bank are described. This information represents the knowledge about this process that can be made explicit and that shall be documented in the form of conceptual enterprise models.

In the context of Knowledge Blockchains, the first step is to initiate the blockchain and decide which modeling language to use. In our case this is accomplished by the CEO of the respective bank who acts as the blockchain owner and who decides to collect knowledge about the business processes in her company using the BPMN modeling language. For the purpose of simplicity, she chooses to use a permissioned blockchain that is only available within the company and managed by one central miner. Upon creation of the blockchain, the CEO with her identity in the form of her public key is assigned all create, delete, and transfer rights in a permission model - see Figure 8. This information is stored in the genesis block upon calculating the permission model Merkle tree. The genesis block is thereby created and signed by the miner. Subsequently, the CEO decides to delegate the right to create BPMN models to an employee.

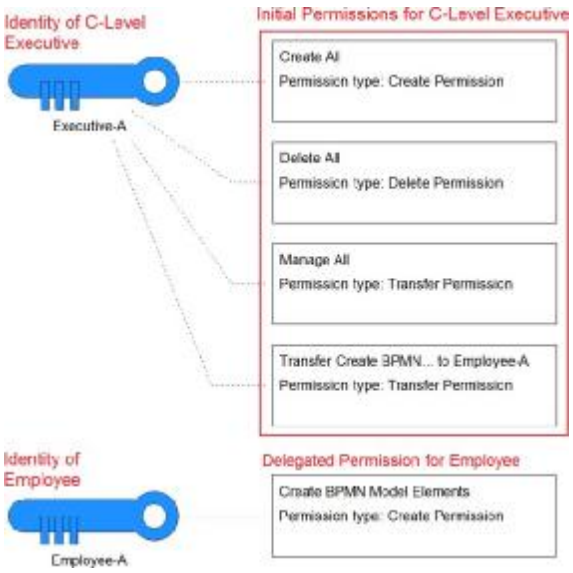

Figure 8: Sample instance of a permission model type

The identity of this Employee A in the form of the corresponding public key together with a Create-Permission is thus added to a copy of the permission model, which is obtained from the previous block. This is shown in the lower part of Figure 8. The updated permission model is then submitted to the miner as a new transaction. For this purpose, the CEO has to sign the proposed block with the private key of her digital signature. The miner checks the signature and whether this identity is allowed to conduct the changes based on the last permission model. As this is the case for the CEO identity, the block proposal is accepted and the new permission model becomes part of the blockchain. Equipped with the new rights, Employee A can now create BPMN model elements and submit them to the blockchain. Upon the receipt of the block proposal, the miner verifies the identity and the permissions and adds the information to the Knowledge Blockchain. This results in further blocks in the Knowledge Blockchain that represent the content of the business process model as shown in Figure 9.

The following tasks can now be accomplished based on the information stored in the blockchain. First, it can be transparently monitored by all parties with access to the Knowledge Blockchain, how the represented knowledge evolves, how new entities are added, and who is responsible for adding them. Besides this identification of the provenance of knowledge, also the delegation scheme behind the Knowledge Blockchain becomes visible. It can exactly be tracked, when the CEO delegated the mentioned Create-Permission and to whom. If necessary, process model patterns can be specified and tested for their containment in the Knowledge Blockchain without revealing the content of the BPMN model (zero-knowledge proof). By reverting to the example of the account opening process this could be the task ”Ask customer for ID”, which is complimentary for any bank to check the identity of a future customer. In the current version of the Knowledge Blockchain, according attribute hash values could be calculated for this pattern and compared with the Merkle tree stored in the most recent block to proof its existence.

6. Related Work

Concerning existing approaches which make similar propositions with regard to the organization and evolution of knowledge as discussed for Knowledge Blockchains, we identified the following areas: approaches that allow to track changes in enterprise models with their provenances and allow to restrict and delegate access to models and collaborative modeling approaches and mechanisms in modeling tools for multi-user features.

Document management and versioning control systems (VCS) allow storing any number of electronic documents, like source code files or models, in a well-defined state as a version together with their author. Well-known VCS include systems using a centralized repository, such as CVS [7] and SVN [8]. Distributed Version Control Systems (DVCS) such as Git additionally permit users to create and synchronize distributed repositories and have grown in popularity [9]. While VCS offer version control at the level of individual files, they can also be used to collaboratively access models. Altmanninger et al. [2] evaluate various VCS for this purpose. They show that such systems usually allow comparisons of models and the detection of syntactic conflicts. For the purposes of knowledge evolution, such systems can be leveraged to track model changes together with their respective authors. The comparison can take model elements into account, while a commit or fetch of an artifact is conducted on a per-model-basis. Access to a repository can be restricted, however, it is not possible to restrict the modeling of individual models, model elements or to delegate such permissions.

Collaborative modeling approaches such as COMA [27], DREAM [4], and CoMoMod by Dollmann et al. [10] are concerned with the collaboration during the creation of models to build models with multiple participants. Such approaches provide means to contribute to a model. However, knowledge evolution in the form of tracking model changes and access restrictions are not covered by these approaches.

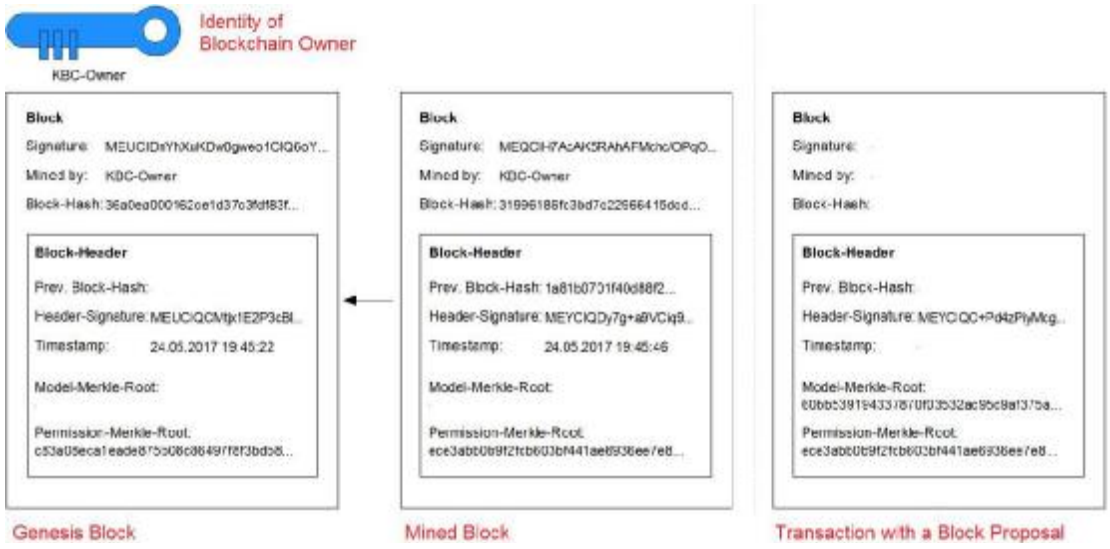

Figure 9: Detailed view of the sample instance of the Knowledge Blockchain model type in ADOxx

On a tool level, professional modeling tools such as ADOxx [12] or MetaEdit+ [19] provide multi-user features. Models can be organized in repositories and versioned. Much like in VCS, versions and authors of individual model versions may be tracked, however, there is no enforcement of individual access restrictions at the platform level. This means that there is no system-enforced process by which new models or model elements are proposed, checked against permissions and possibly added to a repository.

With regard to existing approaches, version control systems are typically used to provide knowledge evolution features through extensions and implementations in modeling tools. The model creation process is covered by collaborative modeling approaches that focus on the creation process. A blockchain-based method facilitates collaborative aspects by allowing identities to make signed changes to models, which are incorporated only if platform-enforced access rules allow it. Through the generation of irrevocable blocks with the changed models, versioning is implicit. In contrast to existing approaches, cryptography enforces access control and allows changes on a per model- and model-element-basis that can be traced back to their provenance.

7. Conclusion and Outlook

In this paper, we have presented the concept of Knowledge Blockchains for storing the knowledge expressed in enterprise models in an immutable and tamper-resistant way. It has been illustrated which benefits can be gained from such an approach in the context of knowledge management. The approach in its current stage contains several limitations that are to be tackled in future versions. For example, it is currently not foreseen to uniquely identify also the elements of the modeling language using UUIDs and to ensure the correct instantiation of models based on the modeling language. This is currently assumed to be implicitly handled by the used modeling platform. In addition, also the current implementation on the ADOxx platform has several shortcomings. First and foremost, the implementation so far only contains a subset of the required rules for checking the conformance of blocks during mining.

Regarding future work, extensions of the UUID scheme using URI will be evaluated. In addition, the approach will be described in a more formal way, e.g. using a formalism such as FDMM [13], to ensure a common understanding of all technical details.