SPROOF: A Platform for Issuing and Verifying Documents in a Public Blockchain

Clemens Brunner, Fabian Knirsch and Dominik Engel
Center for Secure Energy Informatics, Salzburg University of Applied Sciences, Puch bei Hallein, Austria

Keywords: Blockchain, Certificate, Privacy-friendly, Digital Document, Pseudonym.

Abstract: Managing educational certificates or records of personal achievements often comes at the cost of handling documents, loss of data or malicious counterfeits. Especially in the case of printed certificates, both the origin and the integrity of certificates are hard to verify. Furthermore, such documents can be lost or destroyed due to unseen circumstances. Reissuing certificates can then be cost intensive, hard or impossible, e.g., if the issuing organization has been closed. While issuing and signing documents digitally solves some of these issues, this still requires centralized trusted infrastructures and still does not allow for easy verification or recovery of lost documents. In this paper, we present SPROOF, a platform for issuing, managing and verifying digital documents in a public blockchain. In the proposed approach, all data needed for verification of documents and issuers is stored decentralized, transparent, and integrity protected. The platform is permissionless and thus no access restrictions apply. Rather, following principles of the Web of Trust, issuers can confirm each other in a decentralized way. Additionally, scalability and privacy issues are taken into consideration.

1 INTRODUCTION

Educational certificates and other records of personal achievements are still most commonly issued as a paper document. These documents are often easy to counterfeit, can be lost and are hard to verify. In order to verify the correctness of such documents for, e.g., a job application, one has to manually contact all issuing institutions for verifying the integrity and validity of the paper document and the printed records. Furthermore, issuing – and reissuing such paper documents in case they get lost – can be a cost and labor intensive process. While documents can be issued and signed digitally, this only solves some of the problems and requires a centralized and trusted infrastructure that has – in the past – already shown to be unreliable in some circumstances (Durumeric et al., 2013). Additionally, traditional digitally signed documents do not allow for easy verification or recovery of lost documents and especially do not support the completeness feature which is introduced below.

In this paper, SPROOF, a platform for storing digital documents in a public blockchain, is presented. The work builds on the initial proposal of such a platform by Brunner (2017). In this paper, first, the architectural building blocks of SPROOF are presented, and second, the detailed protocol that uses a blockchain and a distributed storage is discussed.

As a document, we define a digital file that is granted from an issuer to a receiver, e.g., a diploma granted from a university to a student or records of achievements granted from an company to a customer. Such a document can represent any data that has an issuer and a receiver. The proposed approach uses a blockchain for decentralized, transparent, and integrity protected management of issued documents. The approach is fully permissionless and does not allow single entities to gain control over issued documents or to prevent others to verify documents. Furthermore, validation is easy and can be automatized for a large number of documents from different issuers and for different subjects.

The contribution of this work is twofold: It is shown how documents can be issued, received and verified while being fully decentralized, permissionless and transparent. In addition, the ability to group related documents from the same issuer is outlined. Scalability and privacy issues are taken into consideration.

In order to issue, receive and verify documents, in SPROOF the following roles are defined:

Issuer: The issuer of a document can be a company, an educational institution or basically anyone who

Brunner, C., Knirsch, F. and Engel, D.
SPROOF: A Platform for Issuing and Verifying Documents in a Public Blockchain.
DOI: 10.5220/0007245600150025
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 15-25
ISBN: 978-989-758-359-9

wants to grant a document. The platform itself poses no limitations on issuers and there is no central third party to control issuers.

Receiver: The receiver of a documents can be a student for an educational certificate, an employee or even a company. Similar to issuers, there is no control over receivers.

Verifier: The verifier represents anyone who wants to view and verify the validity of documents. A verifier also wants to authenticate the identity of an issuer or a receiver. Authentication is fully decentralized and follows the principles of the Web of Trust (WoT). This role can be assumed by, e.g., an employer.

These participants interact via a platform for storing and managing digital documents at low cost, with a simple verification feature, and with a reliable storage of data. We define the following desired properties:

Decentralization: The platform is completely decentralized and especially allows the verification of data without a single trusted third party. Furthermore, verification of past documents must be possible even if the issuing institution is not existent anymore.

Permissionless: The platform is permissionless and thus no single entity has control over the participants. Any participant has full access and can add new or has the possibility to revoke own issued documents without being required to register at a third party.

Integrated Issuer Verification: The platform provides built-in mechanisms to verify the identity of issuers. Thus, no additional or centralized channel is needed.

Transparency: The platform is transparent and every participant has read access to validate a given document. Privacy of documents is preserved by not revealing details of the receiver or sensitive content of a digital document, such as the name, without the consent of the receiver.

Completeness: Issuers have the ability to group documents and verifiers can check whether a group of documents is complete or not, i.e., if some document are intentionally hidden by the receiver (e.g., verifying a Bachelor's diploma includes verifying all related courses). This can be enforced by the issuer at the time of granting documents and is explained in detail in Section 4.

The rest of the paper is structured as follows: Section 2 compares SPROOF to state of the art approaches in the field of educational certificate management and with respect to the stated requirements. Section 3 describes the basic building blocks of this work and the proposed protocol. Section 4 then describes the roles and the SPROOF protocol in detail. Section 5 conducts a security analysis of the proposed protocol and Section 6 summarizes this work and gives an outlook to future research.

2 RELATED WORK

In this section, related work in the field of blockchain-based digital document management is presented. Table 1 shows a comparison of such approaches in the field of educational certificates. The related work is evaluated with respect to our initial requirements, which are decentralization, permission management, transparency, support for integrated issuer verification and completeness, as described in the previous section.

The University of Nicosia[1] was the first (2014) to register academic certificates for an online course on the Bitcoin blockchain. A hash of an index document, which contains a list of hashes of all certificates for a specific semester is registered on the blockchain. Their approach is decentralized, permissionless and transparent, but does not allow for integrated issuer verification and for validating the completeness of issued academic certificates.

The MIT Media Lab is working on a project called *Blockcerts*[2]. Their approach is similar to the one implemented by the University of Nicosia, i.e., registering the root hash of a Merkle tree of hashes of documents on a public blockchain. This approach is decentralized, permissionless and transparent. The project is not attempting to map the digital identity to the real identity of an institution and thus does not allow for integrated issuer verification and validation. Additionally, verifying the completeness of issuing documents is not possible.

By Gräther et al. (2018), an approach for a Lifelong Learning Passport (LLP) is presented which is very similar to the approach of *Blockcerts*. Their approach is decentralized, transparent and additionally they support a mechanism for issuer verification. However, they use a hierarchical scheme for issuer accreditation and therefore it is not fully permissionless.

[1] https://digitalcurrency.unic.ac.cy/free-introductory-mooc/self-verifiable-certificates-on-the-bitcoin-blockchain/academic-certificates-on-the-blockchain/ [retrieved: August 16, 2018]

[2] https://www.blockcerts.org/ [retrieved: August 16, 2018]

Table 1: Comparison of related work with respect to decentralization, permission management, transparency, support for integrated issuer verification and completeness.

	Decentralized	Permissionless	Transparent	Integrated Issuer Verification	Completeness
University of Nicosia	✓	✓	✓	✗	✗
Blockcerts	✓	✓	✓	✗	✗
LLP	✓	✗	✓	✓	✗
SPROOF	✓	✓	✓	✓	✓

Verifying the completeness of issuing documents is not possible.

We are not aware of any scheme that meets all of the initial stated requirements and, in particular, resolves the completeness issue in a decentralized, permissionless, and transparent way, which is one of the main contributions of SPROOF.

3 BUILDING BLOCKS

This section introduces the fundamental building blocks for SPROOF. First, the concept of public storage and blockchain is introduced, and the advantages and challenges for using such a technology are briefly discussed. Second, the principles of key management in HD wallets are explained. The latter is crucial for the completeness feature.

3.1 Public Storage

By Nakamoto (2008), *Bitcoin* is proposed as a decentralized, permanent, trustless public ledger. The proposed approach is the first to reliably solve the double spending problem[3] and sets the foundations for the concept of decentralized, permissionless append-only databases, commonly referred to as *blockchain*. In general, a blockchain can be seen as a global state machine where updates are performed by conflicting-free, authenticated transactions. Following the initial approach by Nakamoto (2008), many implementations have been proposed in recent years, also for fields other than financial transactions, see e.g., (Kosba et al., 2016; Christidis and Devetsikiotis, 2016; Knirsch et al., 2018).

For SPROOF we use a public permissionless blockchain, e.g., Bitcoin or Ethereum (Wood, 2017a), in order to create a platform where nobody, not even a selected consortium, has the right to exclude data or participants (Wüst and Gervais, 2017). SPROOF is built on top of a public blockchain and does not intend to develop a new blockchain for this purpose. The blockchain is used by SPROOF in order to have a verifiable global state of ordered pieces of data in a decentralized, transparent manner and without the need of a single trusted platform operator. The use of a blockchain in SPROOF comes with two main issues: scalability and storage costs.

Blockchain implementations often come with limitations on the scalability (Croman et al., 2016), i.e., the number of transactions and the amount of data that can be stored or processed within a certain amount of time. Polkadot (Wood, 2017b; Eyal et al., 2016) proposes a strategy for solving these scalability issues by decoupling the consensus architecture from the state-transition mechanism. This means that all data is accepted to become part of the blockchain, i.e., the data is stored and distributed, but the semantics of that data and thus the actual validity are processed independently and off-chain. For SPROOF we only need the blockchain to register chronologically ordered pieces of data and thus the consensus is built off-chain by processing data with a publicly known rule set separately, the SPROOF protocol.

Storage on a public blockchain is often limited in terms of size (e.g., 80 Bytes of data in Bitcoin) or expensive (Unterweger et al., 2018). To avoid this problem, SPROOF only adds hashes of data to the blockchain within a transaction. The corresponding raw data is then stored in a distributed hash table (DHT). Data stored in such a DHT inherits the immutability and ordering property from the blockchain if a cryptographically secure hash function is used to calculate the hash that is sealed in the blockchain. In order to create a fully decentralized platform, also the DHT needs to be managed in a decentralized way. For example, established DHTs such as IPFS (Benet, 2014) or Swarm[4] can be used.

Blockchains use public-private key cryptography (Diffie and Hellman, 1976) to represent a user and to sign transactions. The public keys can be seen as pseudonyms, because they can be created offline and without the need of an identification process. However, this does not provide full anonymity, since public blockchains are transparent. If an attacker knows that a pseudonym is linked to an identity, the attacker also

[3] The problem that two conflicting transactions spend the same funds twice.

[4] http://swarm-gateways.net/bzz:/theswarm.eth/ [retrieved: August 23, 2018]

has the possibility to see all transactions which have been recorded in the blockchain since the beginning and are linked to that pseudonym (Reid and Harrigan, 2013). A solution to this traceability problem is to generate a new key pair for each transaction, hence to use an address only once. One method to generate keys out of a single seed is explained in the next section.

3.2 Key Management

In most blockchains, users are represented by a unique ID derived from a public-private key pair using the Elliptic Curve Digital Signature Algorithm (ECDSA) (Johnson et al., 2001). In order to solve the traceability problem, a new key pair for each transaction is created. A key derivation function (KDF) is therefore used to derive one or more private keys from a single password[5], master key or a pseudo random number, a so-called *seed* S. In the following, a method to deterministically derive hierarchically structured pseudorandom public-private child keys $(Q_1,d_1),(Q_2,d_2),\dots,(Q_n,d_n)$ out of a single master key pair $(\hat{Q},\hat{d})$, is explained and illustrated in Figure 1. Each child key can be used as a new master key, hence it is possible to build an infinite hierarchical tree. This concept is called a hierarchical deterministic (HD) wallet.

$$\begin{tikzcd}[column sep=small] \text{Seed} \arrow[rr] & & (\hat{Q},\hat{d}) \arrow[dl] \arrow[dr] & \\ & (Q_1,d_1) & \cdots & (Q_n,d_n) \end{tikzcd}$$

Figure 1: Representation of a HD wallet, where child key pairs $(Q_1,d_1),\dots,(Q_n,d_n)$ are derived from a parent key $(\hat{Q},\hat{d})$ and a seed S.

The ECDSA is based on (the assumed hardness of) the elliptic curve discrete logarithm problem (ECDLP), which is denoted as follows: $E(K)$ denotes an elliptic curve over a field K. A generator of the elliptic curve is referred to as $P \in E(K)$ with an order p. These parameters are publicly known. With the private key $d \in K$ it is easy to calculate the public key $Q \in E(K)$, which is a point on the elliptic curve, using the formula $Q = dP$. Recovering d by only using Q and P constitutes breaking one instance of the ECDLP. Although there exists no formal proof, the ECDLP is commonly assumed to be hard to invert if the underlying elliptic curve is properly chosen (Johnson et al., 2001).

The KDF of an HD wallet uses a cryptographically secure hash function $\mathcal{H}(\cdot)$ which maps an index i and a public key $Q \in E(K)$ to an element of K. The index is the number for the child key pairs (Q_i,d_i), which is calculated as follows:

$$d_i = \hat{d} + \mathcal{H}(i,\hat{Q}) \pmod p \tag{1}$$

$$Q_i = d_i P \tag{2}$$

One of the main properties of HD wallets is that each child public key Q_i can be calculated without using (and needing to know) a private key, by $\hat{Q} + \mathcal{H}(i,\hat{Q})P$. This is called *master public key property*.

A known vulnerability of HD wallets, however, is that it is possible to calculate the master private key $\hat{d}$ with the knowledge of the master public key $\hat{Q}$ and an arbitrary child private key d_i, by using the derived formula $\hat{d} = d_i - \mathcal{H}(i,\hat{Q})(\bmod p)$. This vulnerability can be bypassed by allowing so-called hardened child keys, where also the public keys are derived from the master private key, instead of the master public key. Such keys lose the master public key property. Another approach for HD wallets that tolerates key leakage is presented by Gutoski and Stebila (2018).

In SPROOF, HD wallets are used to derive key pairs out of a single seed to generate pseudonyms, which are then used for receiving documents. The use of multiple pseudonyms allows to release only a selected subset of documents to a verifier.

4 SPROOF

In this section, we describe SPROOF, a decentralized, permissionless, integrity-protected and transparent platform for granting, storing and verifying digital documents. There are three basic roles in the upkeep of SPROOF: issuer, receiver and verifier.

For the communication between the users representing these roles, two distinct channels are needed: a public and a private one. The public channel is used for publicly available data that is stored on a blockchain, i.e., the issuing of a document. The private channel is needed to transfer non-publicly available and direct personal or sensitive information required for issuing and verifying documents.

Any information sent over the public channel is denoted as an *event*. Events are the only way to add information to the publicly available data set of SPROOF. Events are signed by the issuer and are sealed and integrity protected with the help of the blockchain and a DHT.

In the following, we first describe the processes to create an issuer, then ways to trust an unknown issuer, the generation of a privacy-friendly representation for receivers and finally, necessary steps to verify a document.

[5]Deriving a key from a password is not recommended (Vasek et al., 2016).

4.1 Issuer

The role of an issuer represents any organization or person who wants to grant documents, e.g., a university. Issuers need to be publicly known, trustworthy and verifiable.

In order to create a new account, an issuer establishes a public-private key pair. The public key is the representation of the issuers public profile P_P in SPROOF and the private key is needed to sign events triggered by the issuer. The key pair itself provides no information about the organization or person behind and is thus pseudonymous. However, issuers need to be identifiable and therefore P_P needs to be linked to the issuers organization. This can be done by adding a new $\mathrm{E}_{\text{Identity Claim}}$ event. This event includes all necessary data to address the issuer, e.g., the name of the company or organization. Since the platform is permissionless and decentralized there are no restrictions for generating such identity claims and there is no single trusted third party to verify the correctness of the provided claims. To increase the trustworthiness of an issuer, additional $\mathrm{E}_{\text{Identity Evidence}}$ events can be provided. These events, also created by the issuer, provide additional evidence by connecting the SPROOF account with already established central trusted platforms, e.g., social media accounts or known public key infrastructures. To link a social media account, the issuer needs to add an $\mathrm{E}_{\text{Identity Evidence}}$ event including a reference to a publicly accessible message, e.g., a post in an online social network, which contains P_P. To link an X.509 certificate, the issuer needs to add an $\mathrm{E}_{\text{Identity Evidence}}$ event including the certificate and a signature over P_P created by the confirmed private key of the X.509 certificate. Note that this process is possible for all types of PKI certificates. This allows to connect several, already established central trusted infrastructures, to P_P. There is no limitation in the number of $\mathrm{E}_{\text{Identity Evidence}}$ events, hence an issuer can add multiple $\mathrm{E}_{\text{Identity Evidence}}$ events to strengthen its P_P.

While the methods to increase trustworthiness of an issuer described above are based on central trusted authorities, this is used as bootstrapping to build a decentralized confirmation network which borrows concepts from the WoT (Caronni, 2000). In a WoT others must be able to confirm the identity of the issuer, by sending a $\mathrm{E}_{\text{Confirm}}$ event. The purpose of a confirmation is that the sender verifies the receivers identity claim. Confirmations are linked to the identity claim that was added last. This is to rule out the possibility of an issuer to maliciously rename itself after collecting some confirmations. Before an issuer confirms another issuer it needs to verify P_P and the provided identity claim.

This can be done based on the identity evidence events or also outside SPROOF, e.g., during a personal meeting. This means that – in return – an issuer may lose its reputation if it confirms a fake issuer. A confirm event contains a boolean value, either a positive or negative trust indicator and arguments to justify the decision. Confirmations can thus be used to create networks of issuers. Given such a network, newly added issuers can quickly gain reputation by a confirmation from a well-known and established issuer. As an example, consider a network of universities. While a newly established university sets up relationships with well-established institutions for research and teaching collaborations it can – in the same way – gain confirmations in SPROOF after a while. Once one or more major institution confirmed the integrity of the new university, this sets up a WoT.

4.2 Receiver

A receiver of a document is, analogously to an issuer, represented with a public key. This public key is used together with the corresponding private key to prove the ownership of a document to a verifier. Reusing the same pseudonym for all documents that a receiver gets would lead to the receiver being only able to share all documents ever received at once to a verifier, which would not be privacy-friendly and also impractical for the receiver. Once a third-party knows the pseudonym, it would be able to view all documents issued in the past and also all future ones. To avoid this traceability problem, fresh pseudonyms can be created for each document exploiting the previously presented properties of HD wallets. Note that only leaves of the pseudonym tree should be used for receiving a document. By doing so, the privacy is preserved by the fact that it is practically impossible to invert a cryptographically secure hash function. Therefore, it is not possible to calculate parent pseudonyms by knowing the corresponding child pseudonyms.

As shown before, the public-private key pairs are deterministically generated out of the random seed S using a HD wallet K_M, as described in Section 3.2. This seed is the main secret and is needed for recovering all derived pseudonyms.

Using K_M, the receiver is able to generate child pseudonyms $P_{I_1}, \ldots, P_{I_n}$. Each of those pseudonyms can be used as a new master key for further sub-pseudonyms for a specific issuer. A pseudonym is shared with an issuer using the private channel. From this pseudonym, the issuer is able to generate further sub-pseudonyms by using the master public key property of HD wallets. Note that this can be done without revealing any information about the corresponding private keys.

The ability to derive sub-keys enables further features, e.g., if an issuer wants to link a document which has dependencies to other already issued ones. This can be the case for a series of educational certificates that build on each other, e.g., required courses for getting a bachelor's degree. Given a parent pseudonym, all descendants are verifiably connected to this parent. If, for instance, pseudonym P_{G_1}, see Figure 2, represents a Bachelor's diploma, all sub-pseudonyms including $P_{D_1}, \dots, P_{D_n}$, which may represent particular courses, are permanently and publicly linked to the Bachelor's diploma. We call this property *forced completeness*, since it can be enforced by the issuer and cannot be hidden. Note that a receiver can still share documents $P_{D_1}, \dots, P_{D_n}$ separately and independently and without revealing the parent pseudonym and thus the corresponding document. If a receiver shares more than one pseudonym that are used for documents issued by the same issuer, which can be avoided by sharing a pseudonym on a higher level of the pseudonym tree, the verifier can conclude that the receiver shows an incomplete information. This is, to the best of our knowledge, a feature that is unique to SPROOF.

The concept of completeness is shown in Figure 2, where the privacy and completeness property are indicated by arrows. Privacy is provided bottom-up, whereas completeness is achieved top-down.

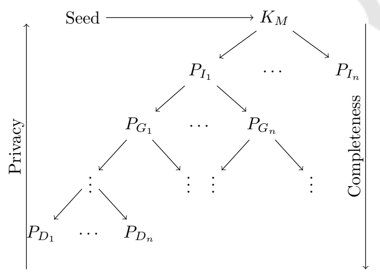

Figure 2: Pseudonym tree with derived keys and documents in the leaves. Completeness is achieved by a unique, easily verifiable path from the top to the bottom and privacy is achieved by the impossibility to retrieve parent keys from a given leaf key.

Note that the pseudonym itself contains no information about the real identity of the receiver, e.g. the name of the person. However, a mean of linking a document to the real identity of the receiver needs to be established. Otherwise, receivers may collaborate and share documents among each other by sharing private keys of their pseudonyms.

To link a document to the real identity of a receiver, the receiver has to create a file including identification data, denoted as ID_R. The data contained in this file must be enough for the issuer – and all possible verifiers – to determine the real identity of the receiver.

One approach to create ID_R is to copy the information printed on an ID-Document, i.e., name, date of birth, etc., or by use of any other trusted third party. Another way is to use a machine readable code of an ID-Document, an image of a passport, or a fingerprint. To protect the privacy of the receiver, only the hash reference of ID_R is added to the document. Given the cryptographic hash reference of some data, it is practically impossible to reconstruct the original data. However, cryptographic hash functions are deterministic, hence an attacker who holds a copy of, knows or guesses the identification data of the receiver is able to calculate the hash value and thus disclose information about the receiver. To avoid this vulnerability, a salt is added to ID_R, to obfuscate the hash reference (Gauravaram, 2012). If we use the same salt value for all hash calculations, the traceability problem arises again because of the resulting hash value being the same for all documents. Therefore, different salt values need to be generated for each document. To reduce the risk of losing the salt values a hierarchical deterministic generation function is used. The seed to generate the master salt S_M is $S||suffix$ where *suffix* is a publicly known fixed phrase, || denotes a concatenation and S is the same seed as used for the generation of K_M. The salt value for a specific child pseudonym in the pseudonym tree is recursively calculated by hashing the parent salt concatenated with the child's index.

Note that the salt value can be seen as a secret needed for disclosing the identification data on a (otherwise publicly visible) document. This value should not be publicly available, since otherwise a verifier or adversary is able to detect all documents by knowing ID_R.

4.3 Document

In this section the processes to create and add a document to SPROOF is described in abstract form and an exemplary list of fields for a document and the necessary steps to grant and revoke documents are presented. Generally, in SPROOF a document is a digital file where the public key of the issuer and the pseudonym of the receiver, including a salted hash of the identification data, is publicly known.

4.3.1 Format

A document consists of an issuer and a receiver, which are always publicly known. The other fields like title, description, evaluation, expiry dates, etc. can either be public or private. For the private fields of the document the issuer just adds a salted hash reference of the private information. Salt values are freshly generated, like that one used for the identification data. The corresponding raw data is then in the hands of the issuer and the receiver only. To allow the verifier to validate the private part of the document, the receiver needs to share the corresponding raw data with the verifier over a private channel. This allows the verifier to recalculate and compare the hash to the one in the document. Also, hybrid methods are possible where, e.g., the title and the description is public, but the grade is hidden in the private part.

4.3.2 Grant Documents

In order to add a document to SPROOF, the receiver has to register at an issuing institution. For this purpose, the receiver chooses a master pseudonym P_I, which has not already been used by another issuer and which represents a new leaf in the pseudonym tree. The receiver then needs to transmit the identification document ID_R, P_I, and the corresponding master salt $SALT_I$ for the pseudonym P_I over a private channel to the issuer. The issuer has to verify if ID_R matches to the real identity of the receiver and check if P_I is not already used as receiver for a document. Once this process is completed, the receiver is registered at an issuer. Furthermore, the receiver permits the issuer, by sharing its pseudonym, to derive new sub-pseudonyms and the corresponding salted hash value of ID_R to grant documents. With this approach the issuer is able to decide the ordering and structuring of its issued documents by deriving new sub-pseudonyms out of the shared master pseudonym. The completeness feature can thus be enforced at the time of granting documents. A verifier can later check the set of all documents granted to a given pseudonym and all derived sub-pseudonyms. The verifier can thus be sure that no document were hidden by the receiver. Documents are granted and stored using the $E_{\text{Grant Document}}$ event. This is illustrated in Figure 3.

4.3.3 Revoke Documents

Documents are not always valid for an unlimited period of time. Sometimes an expiration date is sufficient, e.g., for a first aid course or a driving license. Additionally, an issuer may also decide to revoke a

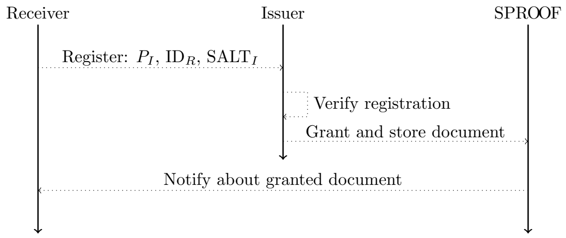

Figure 3: Process for granting a document in SPROOF. The receiver registers at the issuer and passes the pseudonym which should be used for the new document. The issuer verifies the registration and grants a new document.

document, e.g., when it detects plagiarism in a graduation paper or for other reasons. Therefore, an issuer which grants a document has the possibility to revoke it at a later point in time. This is done by adding an $E_{\text{Revoke Document}}$ event, which only the issuing institute is allowed to do. This event includes the hash of the document, and a reason to justify the revocation. The $E_{\text{Revoke Document}}$ event is appended to the public storage and therefore available and transparent to all verifiers.

4.4 Events

Events are the only way to add information to SPROOF and they are sealed in a public blockchain. In this work, only issuers are allowed to add events. The reason is that adding an event requires a transaction on a blockchain. Adding a transaction to a blockchain usually comes at the cost of at least a fraction of cryptographic tokens. We assume that issuers are willing to buy some tokens, but receivers may not. To be considered as valid, each event needs to follow specific rules, as described in Section 4.1. Invalid events are ignored. Note that due to the decoupling of the consensus mechanism of the blockchain from the SPROOF protocol, invalid events may become part of the blockchain data, but are not considered by SPROOF users. Therefore, the publicly available data set of SPROOF is a chronologically ordered list of valid events. A blockchain node only needs to check if the blockchain transaction is valid and does not need to validate if the corresponding data represents a valid SPROOF event. This reduces the costs for a transaction to the blockchain.

Since storage space on a blockchain is often limited and expensive, only the hash reference of data is sealed into a transaction. Adding a new transaction for each event would imply that an issuer, which wants to grant n documents, also needs to add n transactions. This is inefficient and expensive. Therefore, events are combined into a chronologically ordered list and the hash reference of this list of events is then registered into a single transaction, as illustrated in Figure 4. The issuer has to sign this transaction, including the hash reference, and add it to the blockchain as part of a transaction. Once the transaction is included and confirmed, it is traceable and authentic to the issuing institute, integrity-protected and publicly readable.

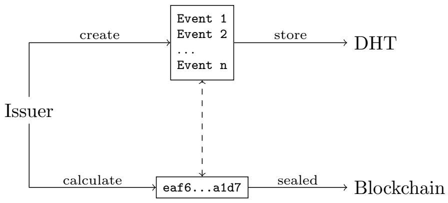

Figure 4: For adding events to SPROOF, one or more events are collected and written to a DHT. The hash reference of that DHT entry is then sealed in the blockchain and thus publicly visible to all participants.

At this point, only the hash reference of events is sealed in the blockchain. The corresponding raw data is stored in a DHT, where the sealed hash value is used to address the raw data. The issuer has to ensure that the raw data is available and complete. A registration of events in the blockchain that does not provide the raw data is transparent visible to all verifiers and would therefore damage the reputation of the issuer. A transaction that is considered for the SPROOF data set is always sent to the publicly known SPROOF blockchain address. Therefore, for validation purposes, a verifier only needs to consider transactions sent to this address.

4.5 Verification

Transactions in SPROOF can be validated by practically anyone in the world. For this purpose, a verifier needs to iterate over all transactions in the blockchain that are sent to the SPROOF address. The verifier then downloads the corresponding raw data from the DHT. After that, the verifier is able to execute each event of the SPROOF protocol and check if it is valid and should be added to a local database. The database represents the precalculated global unique state of SPROOF. Note that this includes also revocation events for documents. Additionally, this database can then be used to view and validate documents and to authenticate issuers and receivers. This client-side validation process can be done programmatically on a trusted computer that is controlled by the verifier. Since hashes can be assumed to be collision free (Damgård, 1988), the data stored in the DHT is immutable. Changing the raw data would results in a different hash reference, not matching the one sealed in the public blockchain.

4.5.1 Receiver

The verifier can validate a receiver by two different approaches. In both approaches, the receiver has to share a pseudonym P_x with the verifier. Using P_x, the verifier is able to find all documents that are granted to P_x or any descendants of P_x. In the first approach the receiver remains anonymous, whereas in the other the receiver has to disclose ID_R. Both approaches are described below.

For the anonymous approach, the verifier has to be convinced by the receiver to be in possession of the private key of a pseudonym P_x. For this purpose, the receiver creates a verification document, which includes the following fields ($Verifier\,Name$, $Blockhash$, P_x) and is signed using the private key that belongs to P_x. This document is shared with the verifier via a private channel. The verifier is now able to check if the provided signature matches to P_x and has to check if the signed *Verifier Name* is correct. The *Blockhash* acts as a decentralized timestamp to detect outdated signatures. This needs to be done in order to reduce the risk of an attacker for reusing the verification document. With this information the verifier can conclude that the receiver knows the private key of P_x, that all documents granted to any derivation path of P_x belong to the receiver and furthermore the verifier knows the minimum age of the signature by comparing the *Blockhash*. Considering that receivers may cooperate and share pseudonyms, it is not always enough to verify a receiver without identification. For the approach where the receiver discloses the used identification data, the receiver shares ID_R, $SALT_x$ and P_x with the verifier. With $SALT_x$ and ID_R the verifier is able to calculate and validate the obfuscated hash values of the identification data for all documents granted to P_x. Finally, the verifier needs to crosscheck ID_R with an official ID-Document to see if it matches to the real identity of the receiver.

4.5.2 Issuer

To decide if an issuer is trustworthy, a verifier can check the publicly available $E_{\text{Identity Evidence}}$ events and decide whether the provided information is sufficient to trust an issuer P_P. Additionally, the linked X.509 certificates can be verified. In case that the $E_{\text{Identity Evidence}}$ events are insufficient, another way is to use the confirmation network to find a path from a known trustworthy party to the issuer.

4.6 Combine Issuer and Receiver

In Section 4.1, the process for issuers to create a public profile is described. This process is not limited to issuers, but also allows receivers to create a public account where the receiver has the possibility to disclose privately received documents and attach them to their public profiles. With the use of HD wallets it is possible to generate, out of a single seed S, multiple hierarchically structured public-private key pairs. The first child can be used as the representation for issuers to grant documents and by receivers to publish documents. The second child can be used as the master key for possible pseudonyms, which is illustrated in Figure 5.

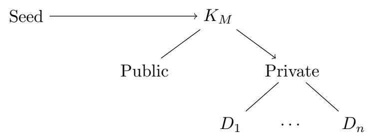

Figure 5: A SPROOF account can be split into a public and a private part. While the public part is used for granting and receiving documents, whereas the private part is used as a master key for new pseudonyms.

To publish a privately received documents to a public profile two signatures are needed. One from the receivers pseudonym and one from the public account. The second signature is implicitly provided by adding the transaction to the blockchain. This is done by triggering an $E_{\text{Link Document}}$ event.

4.7 Summary

In this section, the processes for generating issuers, receivers and processes for granting and revoking documents were presented. The platform is permissionless and therefore provides, for issuers, a decentralized way to add identity claims for bootstrapping accounts. Necessary data to verify the issuer and the document is publicly available without any read restrictions. A privacy-friendly way to generate pseudonyms and link identification data of a receiver to a publicly accessible document has been shown. The generation of pseudonyms enables the platform to fulfill the completeness property. Combining events allows to add data to the blockchain in a scalable way.

5 EVALUATION

In this section the SPROOF protocol is evaluated with respect to maliciously acting issuers, receivers and verifiers. Additionally, general attacks to the SPROOF platform are considered.

A malicious issuer may create a fake profile. Therefore the fake issuer sets up a $E_{\text{Identity Claim}}$ event and adds numerous $E_{\text{Identity Evidence}}$ events to strengthen its fake profile. By consistently creating fake social media accounts, a fake website, etc. this makes it hard to identity a true issuer from a fake one. However the core idea of the WoT is that multiple established and trusted issuers confirm the identity of new issuers. A verifier of a document, attempting to validate the identity of the issuer, can identify such fakes, by starting at one or more known trusted issuers and following the paths to the fake issuer. In case there exist no paths or a majority of negatively rated confirmations only, these are strong indications that the document has been created by a fake issuer. Additionally, a verifier can validate the X.509 certificates linked to the issuer. In case that these X.509 certificates are invalid, linked to non-official websites or not available, this are also strong indicators of a fake issuer. Issuers may revoke documents with a malicious intent and without justification, or publicly release identification data of documents it has previously issued. While this is a general problem, it would only affect the specific documents from this issuer and not the receivers' whole accounts.

A malicious receiver may attempt to collaborate with other receivers to share pseudonyms and thus collecting documents that were issued to another receiver. However, this is prevented by adding identification data to documents, which uniquely identifies a specific receiver, e.g., a fingerprint. Note that this data is not publicly stored in the blockchain, but only the hash reference is linked to a document in order to protect privacy. In case the receiver wants to remain anonymous at the time of verification such an attack is feasible, however it is up to the verifier to allow an anonymous verification at the risk of shared pseudonyms.

A malicious verifier may reuse or publish received documents and the corresponding identification data. In the process of sharing a document to a verifier the *Verifier Name* and the *Blockhash* are contained and signed by the receiver. Reusing a document is practically impossible since this would require to change the *Verifier Name* and the *Blockhash* within the signed data. While publishing received documents cannot be prevented in SPROOF it would only affect the specific documents shared with this malicious verifier.

Malicious attackers may add a huge amount of valid or invalid events at once or seal a hash reference where the raw data stored in the DHT is not available or significantly large. However, adding a transaction to the blockchain is only possible with a signature which is linked to a public key. Adding invalid events or hash references where the raw data is not available will downgrade the reputation of an issuer. A timeout for reading data from the DHT and a limit for the number of events which are allowed to be sealed within one transaction can be used to protect the platform from such attacks.

6 CONCLUSION

In this paper, a platform for managing digital documents has been presented. The paper proposes the architectural building blocks and a protocol for issuing, receiving and verifying digital documents. A blockchain is used to seal hashes of data stored in a Distributed Hash Table and a Hierarchical Deterministic Wallet is employed for key management. This allows for features such as completeness, i.e., the ability to prevent receivers from hiding certain documents. The platform additionally sets up a Web of Trust of issuers and thus provides integrated issuer verification. In summary, the management of documents is fully decentralized, permissionless and transparent. Future work will focus on evaluation of the prototypical implementation and explore the abilities of the proposed scheme for other fields of application. Attribute-based identification for receivers will be integrated by extending the SPROOF protocol.

ACKNOWLEDGMENTS

The overall support of Rainer Böhme from the University of Innsbruck as the supervisor of (Brunner, 2017) and especially the initial idea of using HD wallets to build pseudonym trees to enable the completeness feature is gratefully acknowledged. The authors also like to acknowledge Michael Fröwis and Pascal Schöttle for discussions about this topic. The financial support by the Federal State of Salzburg is gratefully acknowledged. Funding by the Austrian Research Promotion Agency (FFG) under project number 865082 (ProChain) is gratefully acknowledged.

REFERENCES

Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3). Technical report, IPFS.

Brunner, C. (2017). *Eduthereum: A System for Storing Educational Certificates in a Public Blockchain*. Master's thesis, Universität Innsbruck.

Caronni, G. (2000). Walking the Web of Trust. In *9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*, pages 153–158, Gaithersburg, MD, USA. IEEE.

Christidis, K. and Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303.

Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Gün Sirer, E., Song, D., and Wattenhofer, R. (2016). On Scaling Decentralized Blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125, Christ Church, Barbados. Springer.

Damgård, I. B. (1988). Collision Free Hash Functions and Public Key Signature Schemes. *Advances in Cryptology — EUROCRYPT' 87*, pages 203–216.

Diffie, W. and Hellman, M. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

Durumeric, Z., Kasten, J., Bailey, M., and Halderman, J. A. (2013). Analysis of the HTTPS Certificate Ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference (IMC '13)*, pages 291–304, Barcelona, Spain. ACM.

Eyal, I., Gencer, A. E., Sirer, E. G., and van Renesse, R. (2016). Bitcoin-NG: A Scalable Blockchain Protocol. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 45–59, Santa Clara, CA. USENIX Association.

Gauravaram, P. (2012). Security analysis of salt||password hashes. In *International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, pages 25–30, Kuala Lumpur, Malaysia. IEEE.

Gräther, W., Augustin, S., Schütte, J., Kolvenbach, S., Augustin, S., Ruland, R., Augustin, S., and Wendland, F. (2018). Blockchain for Education: Lifelong Learning Passport. In *Proceedings of 1st ERCIM Blockchain Workshop 2018*, Amsterdam. European Society for Socially Embedded Technologies (EUSSET).

Gutoski, G. and Stebila, D. (2015). Hierarchical deterministic Bitcoin wallets that tolerate key leakage. In *19th International Conference on Financial Cryptography and Data Security (FC 2015)*, San Juan, Puerto Rico. Springer.

Johnson, D., Menezes, A., and Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63.

Knirsch, F., Unterweger, A., and Engel, D. (2018). Privacy-preserving Blockchain-based Electric Vehicle Charging with Dynamic Tariff Decisions. *Journal on Computer Science - Research and Development (CSRD)*, 33(1):71–79.

Kosba, A., Miller, A., Shi, E., Wen, Z., and Papamanthou, C. (2016). Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 839–858, San Jose, CA, USA. IEEE.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Technical report.

Reid, F. and Harrigan, M. (2013). An Analysis of Anonymity in the Bitcoin System. In Altshuler, Y., Elovici, Y., Cremers, A. B., Aharony, N., and Pentland, A., editors, *Security and Privacy in Social Networks*, pages 197–223. Springer New York.

Unterweger, A., Knirsch, F., Leixnering, C., and Engel, D. (2018). Lessons Learned from Implementing a Privacy-Preserving Smart Contract in Ethereum. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France. IEEE.

Vasek, M., Bonneau, J., Castellucci, R., Keith, C., and Moore, T. (2016). The Bitcoin Brain Drain: A Short Paper on the Use and Abuse of Bitcoin Brain Wallets. In *20th International Conference on Financial Cryptography and Data Security (FC 2016)*, Christ Church, Barbados. Springer.

Wood, G. (2017a). Ethereum: A Secure Decentralised Generalised Transaction Ledger. Technical report, Ethereum.

Wood, G. (2017b). Polkadot: Vision for a Heterogeneous Multi-Chain Framework. Technical report, Parity.io.

Wüst, K. and Gervais, A. (2017). Do you need a Blockchain. Technical report, International Association for Cryptologic Research.