Transforming face-to-face identity proofing into anonymous digital identity using the Bitcoin blockchain

Daniel Augot, Hervé Chabanne, Olivier Clémot, William George

Transforming face-to-face identity proofing into anonymous digital identity using the Bitcoin blockchain

Daniel Augot * † ‡, Hervé Chabanne § ¶, Olivier Clémot §, William George † * ‡
*INRIA, Palaiseau, France
†Laboratoire LIX, École Polytechnique & CNRS UMR 7161, Palaiseau, France
‡ Université Paris-Saclay, Palaiseau, France
§OT-Morpho, Issy-les-Moulineaux, France
¶Télécom ParisTech, Paris, France

***Abstract*—The most fundamental purpose of blockchain technology is to enable persistent, consistent, distributed storage of information. Increasingly common are authentication systems that leverage this property to allow users to carry their personal data on a device while a hash of this data is signed by a trusted authority and then put on a blockchain to be compared against. For instance, in 2015, MIT introduced a schema for the publication of their academic certificates based on this principle. In this work, we propose a way for users to obtain assured identities based on face-to-face proofing that can then be validated against a record on a blockchain. Moreover, in order to provide anonymity, instead of storing a hash, we make use of a scheme of Brands to store a commitment against which one can perform zero-knowledge proofs of identity. We also enforce the confidentiality of the underlying data by letting users control a secret of their own. We show how our schema can be implemented on Bitcoin's blockchain and how to save bandwidth by grouping commitments using Merkle trees to minimize the number of Bitcoin transactions that need to be sent. Finally, we describe a system in which users can gain access to services thanks to the identity records of our proposal.**

***Index Terms*—Bitcoin blockchain, Identity proofs, Discrete Logarithm REPresentation (DLREP)**

I. Introduction

It is a common occurrence in modern life to authenticate part of one's identity by appealing to an existing relationship with a third-party service. Users have primary forms of identification such as passports and drivers' licenses issued by governments, but they may also need to use secondary forms of identification from trusted sources whose business model nevertheless is not that of an identity provider. For example, in order to establish one's address before buying a cellular plan or obtaining a library card, one can provide a utility bill. One might want to replace this system by one in which a digital record is issued to users that can serve the role of the paper utility bill. A simple way of doing this is for the identity issuer to sign a digital identity document that the user can then store on her device. However, in such a system, it can be difficult for the issuer to update or revoke the document if there are changes to the user's identity. Issuers can create revocation lists, such as the ones used in public key infrastructures, but using these lists can pose subtle challenges [1]. We will propose a system for this setting that uses properties of blockchains to provide for streamlined, integrated revocation and updating of identities.

II. Related work

There are several proposals to use blockchains to store identity information. For a survey of such proposals see [2] and [3]. In particular, MIT Media Labs [4] proposes a system in which academic certificates can be verified against records stored in the Bitcoin blockchain and which are considered revoked if the issuer spends a Bitcoin transaction output. The Blockstack project [5] has implemented decentralized versions of PKI and DNS using the Bitcoin blockchain. The uPort [6] project, which has been implemented in the Ethereum [7] blockchain, includes a smart contract that allows a user who has lost her device to regain access to an identity after being authenticated by several pre-designated contacts. Other proposals involve new blockchains that are designed for their specific applications. Namecoin [8] and Certcoin [9] have implemented DNS and PKI respectively on the Namecoin blockchain. In [10], a system is proposed to use zero-knowledge proofs and either Namecoin or Bitcoin to issue identity credentials in settings such as anonymous peer-to-peer networks, where one does not have trusted credential issuers. While using an application designed blockchain can provide greater flexibility, they have less mining power than Bitcoin, which can have security implications for schemes built on top of them, see the security analysis of the Namecoin blockchain in [5]. Similarly, IDCoins [11] uses a custom blockchain whose proof of work is related to the generation of GPG/PGP keys. These keys are then used to create a web of trust. In [12] a system is proposed to store user personal information such as the GPS data from their phone in a distributed hash table; then this hash table is coupled with a blockchain that stores pointers to the data and permissions on how it may be used. The Estonian government has built an electronic records system based on the Guardtime KSI blockchain [13], [14], which is a permissioned, namely only authorized parties can publish transactions to this blockchain. The proposition of ChainAnchor [15] attempts to create a semi-permissioned

structure on existing blockchains, potentially including the Bitcoin blockchain, by changing the incentive structure of miners to promote transactions that have passed a layer of authentication.

III. Our contribution

We present a identity management system based on the Bitcoin blockchain that allows for a very flexible user experience, providing identity documents that can be used in a wide variety of use cases. To achieve this flexibility, it is necessary to provide protections for user privacy, particularly considering the public nature of information on blockchains. To do this, we incorporate into our protocol a zero-knowledge, selective disclosure identity scheme due to Brands [16]. This scheme corresponds particularly well to our proposal as its security is based on cryptographic primitives that are already used in Bitcoin. Moreover, by combining the potential to only reveal the information about one's identity required for a given authentication with identity records stored on the global, widely accessible Bitcoin network, we empower users to take greater control over their identities.

Compared to most previous work that uses blockchains to offer comparably general identity documents [13], [11], we consider it an advantage that we work within the structure of the existing Bitcoin blockchain, the most established and secure blockchain and the blockchain that is supported by the greatest amount of mining power, in a way that is consistent with Bitcoins' existing incentive structure. By doing so, our system has the same security as Bitcoin itself while minimizing the additional infrastructure required. Compared to [10], which explores the possibility of using either Bitcoin or Namecoin, we further develop ways in which one can embed aspects of identity management into the transactional structure of Bitcoin.

Specifically, we build on the ideas of using this transactional structure to encode revocation of certificates in [4] and updating of logs in [17]. These methods allow us to provide for an integrated, streamlined mechanism for revocation and updating of the identity documents issued by our system. In doing so, we resolve several technical challenges related to the interplay between revocation, the structure of Bitcoin, and Brands' anonymous credentials. Particularly, we provide a means for the user to have a secret required for authenticating against an identity record that continues to perform this role even as the user's identity record is updated.

IV. Background

Our system will involve commitments to users' identities being written into the Bitcoin blockchain. The commitment scheme that we make use of, due to Brands [16], will allow users to disclose selective elements of their identity against this commitment in keeping with the principle that only necessary information about a user should be circulated and stored. Also, we will notice that Brands selective disclosure uses similar cryptographic primitives to Bitcoin, discrete logarithms and hash functions, so the security of our two building blocks are related.

A. Brands selective disclosure scheme

We follow [16]. Suppose we want to make selective disclosures involving an identity with n fields, $(X_1, \ldots, X_n)$. (For example, X_1 may represent a user's name, X_2 her nationality, etc). Let q be a prime number and G a group of order q, in which the discrete logarithm is hard. For our purposes, we will take G to be the Koblitz elliptic curve secp256k1, (we use multiplicative notation for compatibility with [16]). Note that by using the same group that is used in the Bitcoin signature protocol, we reduce the number of different cryptographic primitives on which our system depends. Let $g_0, g_1, \ldots, g_n \in G$.

Furthermore, Brands notes that there is the need for an auxiliary random X_0. This will prevent an attacker who knows some of the X_j fields for a user from performing a dictionary attack in which she guesses values for the other X_j.

Definition 1: The tuple $(X_0, X_1, \ldots, X_n) \in \mathbb{Z}_q^{n+1}$ is called a Discrete Logarithm REPresentation (DLREP) of $h = \prod_{j=0}^{n} g_j^{X_j} \in G$ with respect to $(g_0, g_1, \ldots, g_n)$.

In order for a prover $\mathcal{P}$ to establish knowledge of a DLREP of h to a verifier $\mathcal{V}$, the following procedure is performed [16, §2.4.3]

1) $\mathcal{P}$ generates $n+1$ random, secret numbers $a_0, a_1, \ldots, a_n$. Let $A = \prod_{j=0}^{n} g_j^{a_j}$. $\mathcal{P}$ sends A to $\mathcal{V}$.
2) $\mathcal{V}$ provides a challenge number c.
3) $\mathcal{P}$ computes $b_j = a_j + cX_j$, $j = 0, 1, \ldots, n$ and sends them to $\mathcal{V}$.
4) The verifier $\mathcal{V}$ checks that $\prod_{j=0}^{n} g_j^{b_j} h^{-c} = A$ holds.

We denote such an interactive proof π. Note that it is essential that $\mathcal{P}$ knows all of the X_j in order to be able to perform step 3. Particularly, X_0 which is secret and generated randomly acts as a sort of key to be able to perform these proofs.

More generally, in [16, Chapter 3], it is shown how to prove arbitrary satisfiable Boolean statements about the X_j's using such a Discrete Logarithm REPresentation without revealing other information. Thus, there is a great deal of flexibility in what a user can prove about her identity. She can, for example, provide a proof that she is a French citizen AND that she lives at a certain address, or that she is under 18 OR over 65. In this way, the user can prove (true) statements about her identity that contain an arbitrary number of ANDs, ORs, and NOTs in such a way that a verifier learns only the content of the statement. Specifically, Brands shows:

Proposition 1: [16, Proposition 3.6.1] There is a constructive protocol for demonstrating arbitrary satisfiable Boolean formulas in which X_0 does not appear such that:

1) The protocol is complete and sound
2) The protocol is a proof of knowledge of a DL-representation h with respect to the tuple $(g_0, \ldots, g_n)$
3) For any distribution of $(X_1, \ldots, X_n)$, whatever information a verifier in an adaptively chosen formula attack can compute about $(X_1, \ldots, X_n)$ can also be computed using merely its a priori information and the status of the formulas requested.

Moreover, Brands [16] shows that if the discrete logarithm problem is difficult, DLREP is one-way and collision-intractable, preventing forgery attacks.

We make two remarks on subtleties involved in the use of this protocol:

Remark 1: Note that $\mathcal{P}$ should not re-use the same A with different challenges c, see Section 5.4 of [16] on limited use identities; however, one may prove multiple different formulas about the same DLREP h if the A is changed for each challenge.

Remark 2: One should be careful to not publish two commitments using the same X_0. Namely, one should not publish $h = \prod_{j=0}^{n} g_j^{X_j}$ and $h' = \prod_{j=0}^{n} g_j^{X'_j}$, with $X_0 = X'_0$, as this would allow potential attackers who can guess plausible values for (some of) the X_j and X'_j to calculate $h/h' = \prod_{j=1}^{n} g_j^{X_j - X'_j}$, which no longer contains a blinding factor and might permit a dictionary attack.

We will see in Section VII that the issue discussed in Remark 2 presents difficulties in issuing an identity which can be updated by an identity issuer even as the user maintains X_0 secret. We will overcome this problem by splitting X_0 into two pieces of information, both of which are necessary to perform the selective disclosure proofs: X_{00}, which is only known by the user, and X_{01}, which can be updated by the issuer, see Sections VII-A, VII-B.

B. Bitcoin relevant notions

Bitcoin transactions are made up of inputs and outputs; in fact, all bitcoins exist in the form of Unspent Transaction Outputs (UTXOs) [18], [19, Chapter 5]. A given transaction can have several inputs, each of which was an output of some previous Bitcoin transaction. Each transaction also has one or more outputs each of which has an amount of bitcoin attached to it. For each input, a script must be provided establishing the right to spend the input. (A script which details what must be established is included in the transaction that issued this input as an UTXO.) Most transaction outputs correspond to a Bitcoin "address," which is generally the hash of the public key that can spend that output. Then to spend that output one must merely provide this public key and sign the transaction with the private key. A special output type that is relevant to our work is that of `OP_RETURN` outputs; each such output contains up to 80 bytes of space in which the sender of a transaction can store arbitrary information. A requirement of `OP_RETURN` outputs is to have zero bitcoins associated to them; as such, they are provably not spendable, avoiding the necessity for miners to store them as UTXOs. Non-`OP_RETURN` outputs must have positive bitcoin amounts assigned to them, and in fact there is a minimum amount per output in order for a transaction to be considered "standard" and be included in the blocks of miners who use the Bitcoin Core software [19]. This amount varies slightly based on the output, but as of version 0.14 (March 2017) of Bitcoin Core [20], it is on the order of .000005 bitcoin, which is around .01 USD (1 BTC=2280 USD, May 2017).

When creating a Bitcoin transaction, a user broadcasts a "raw transaction" to the nodes containing: the amount of bitcoin to associate to each output, the script setting up the requirements to spend each output, and the scripts for each input that satisfy the requirements established when the corresponding output was created. These requirements typically include signing the transaction with a private key corresponding to the previous output. Each transaction then has a transaction identifier (txid) which is the hash of this raw transaction. A Merkle tree is formed from these transaction identifiers the root hash of which is included in the block header. Hence, the raw transaction, including which inputs and outputs were involved in a given transaction, is committed to in the blockchain in an immutable way.

V. Actors and protocol structure

In our system, a user ($\mathcal{USR}$) wishes to authenticate her identity to a service provider ($\mathcal{SP}$). We will have the following additional actors:

- Identity Verifiers ($\mathcal{IV}$) - organization such as a bank or utility company that has an existing relationship with $\mathcal{USR}$ and can provide justification for aspects of her identity.
- Service Enablers ($\mathcal{SE}$) - capable of verifying the records that various $\mathcal{IV}$ have created for $\mathcal{USR}$ in the blockchain and conveying information about these records to $\mathcal{SP}$. The public key $pk_{\mathcal{SE}}$ of $\mathcal{SE}$ should be well-known.

Just like a bank or utility company in a traditional setting, we assume that $\mathcal{IV}$ has a complete knowledge of $\mathcal{USR}$'s identity record. Thus $\mathcal{USR}$ must confer to $\mathcal{IV}$ a great deal of trust to not misuse this personal information. $\mathcal{IV}$ publishes commitments to identity documents into the Bitcoin blockchain on the basis of which $\mathcal{USR}$ can justify her identity, and $\mathcal{IV}$ publishes updates to these records if necessary.

When required to demonstrate that her identity satisfies some requirements determined by $\mathcal{SP}$ in order to obtain a service, $\mathcal{USR}$ will prove this information to $\mathcal{SE}$. Thus, $\mathcal{SP}$ must trust $\mathcal{SE}$ to accurately relay whether the user meets the requirements and not to perform fraudulent authentications. Moreover, $\mathcal{SE}$ will also learn personal information about the user, namely whatever $\mathcal{USR}$ had to demonstrate to $\mathcal{SP}$ and the fact that $\mathcal{USR}$ obtained a service from $\mathcal{SP}$. (In contrast, $\mathcal{SP}$ will not necessarily learn anything about $\mathcal{USR}$ beyond the fact that a client of $\mathcal{IV}$ that satisfies its requirements obtained a service through the intermediary of $\mathcal{SE}$.) $\mathcal{USR}$ needs to trust $\mathcal{SE}$ to not misuse this personal information; however, we will see that $\mathcal{USR}$ only needs to reveal partial information about her identity to $\mathcal{SE}$. Also, we imagine that there are several companies willing to fill the role of the service enabler, so if $\mathcal{USR}$ is worried that $\mathcal{SE}$ is developing too complete a profile on her, she can perform future authentications through some other service enabler.

For a further discussion of the assumptions we make on $\mathcal{IV}$ and $\mathcal{SE}$, see the discussion on our security model in Section X.

Remark 3: The service enabler has infrastructure allowing it to manage the authentication operations that may be required

of a user by a service provider. Some sophisticated service providers may prefer to perform these operations themselves rather than make use of a third party. Similarly, identity verifiers can also perform the role of the service enabler themselves while still retaining the some of the advantages of using a blockchain, particularly in terms of ease of revocation and updates. However, by outsourcing the service enabler role to a specialized entity, $\mathcal{IV}$ can reduce infrastructure costs. Also, $\mathcal{IV}$ does not need to be as "lively" as $\mathcal{SE}$. Namely, whereas $\mathcal{SE}$ must be continuously online to enable authentications, $\mathcal{IV}$ only needs to come online periodically to publish or update identities. Also note that if one combines the $\mathcal{SE}$ role with either the $\mathcal{IV}$ or the $\mathcal{SP}$ role, there is a pooling of information from the different roles that the user may not be comfortable with, and the users ability to change $\mathcal{SE}$ periodically for greater anonymity as discussed above would be limited. Thus, users may prefer the model with a distinct $\mathcal{SE}$.

Remark 4: Note that a given user may have accounts with several different $\mathcal{IV}$ (her bank, her utility company, her university, etc) and want to authenticate herself to several different $\mathcal{SP}$. However, she can authenticate through the same $\mathcal{SE}$ for each of these accounts; thus $\mathcal{SE}$ can serve as a single service sign-on.

In practice, we imagine $\mathcal{USR}$ interacting with $\mathcal{SE}$ via a website or a mobile phone app. The user's identity information will be entered into her computer or phone, but not disclosed to $\mathcal{SE}$. Note that the user's device must be able to open a secure communications channel with $\mathcal{SE}$ and be able to perform the cryptographic operations of Section IV-A. The communication can be established in a standard way with TLS. Denote $pk_{\mathcal{USR}}$, $pk_{\mathcal{SE}}$, $sk_{\mathcal{USR}}$, $sk_{\mathcal{SE}}$, the public and private (secret) keys of $\mathcal{USR}$ and $\mathcal{SE}$ respectively. The TLS channel corresponding to these keys is denoted $TLS_{pk_{\mathcal{USR}},pk_{\mathcal{SE}}}$. Establishing these keys can be performed when $\mathcal{USR}$ sets up an account with $\mathcal{SE}$, independently of the enrollment steps of Section VII. These steps should be anonymous for the user; namely it is not required that $\mathcal{USR}$ gives her real identity to $\mathcal{SE}$ during this process, while $\mathcal{SE}$ should have a well-known and trusted public key.

VI. Set-up phase

In order for service enablers to verify the authenticity of records issued by identity verifiers, we must begin with a set-up phase in which identity verifiers communicate to service enablers the Bitcoin address $a_{\mathcal{IV}}$ that they will use to issue new identity records in the enrollment phase, Section VII. $\mathcal{IV}$ should also communicate to $\mathcal{SE}$ the points $g_0 \dots g_n$ in secp256k1 that it will use for its selective disclosure protocols as in IV-A.

VII. Enrollment phase

During the enrollment phase, $\mathcal{IV}$ will publish the root of a Merkle tree that commits to the (up-to-date) identities of all of its users $\mathcal{USR}_1, \dots, \mathcal{USR}_N$. $\mathcal{IV}$ will update this information (at most) once per Bitcoin block, namely about once every 10 minutes, via a Bitcoin transaction that we will denote $\mathrm{TX}_{\mathrm{ENROLL}}$.

A. Calculation of DLREP

We detail the steps required for $\mathcal{IV}$ to calculate a DLREP encoding $\mathcal{USR}_i$'s identity.

- Each new user $\mathcal{USR}_i$ establishes her identity to $\mathcal{IV}$ by showing primary identity documents such as a passport of a driver's license. This is typical of the process of opening a bank account, for example. Denote the relevant user identity fields by $X_1, \dots, X_n$.
- $\mathcal{USR}_i$ chooses a secret, random X_{00}, and communicates $h_{00} = g_0^{X_{00}}$ to $\mathcal{IV}$ along with a Brands proof that she knows an X_{00} such that h_{00} has the appropriate form (preventing an abusive user from submitting an h_{00} of an another form such as ag_0^b for some chosen a and b).
- $\mathcal{IV}$ chooses a random X_{01}, which is securely communicated to $\mathcal{USR}_i$. $\mathcal{USR}_i$ sets $X_0 = X_{00} + X_{01}$. (We will see in Section VII-B that this manner of collectively choosing X_0 between $\mathcal{USR}_i$ and $\mathcal{IV}$ will be useful when performing updates to issued identities.)
- $\mathcal{IV}$ computes $h_{\mathcal{USR}_i} = h_{00} \cdot g_0^{X_{01}} \cdot \prod_{j=1}^{n} g_j^{X_j} = \prod_{j=0}^{n} g_j^{X_j}$.

B. Updating a DLREP and X_0

Sometimes a user's identity fields will change. She may change nationalities or she may turn 18 and no longer be marked as a minor. In this case, $\mathcal{IV}$ should have a convenient mechanism for updating the user's identity record which requires computing an updated $h'_{\mathcal{USR}_i}$. This presents the problem that, as we saw in Remark 2, $\mathcal{IV}$ should not publish two Brands commitments $h_{\mathcal{USR}_i}$ and $h'_{\mathcal{USR}_i}$ that share the same X_0. However, it may not always be practical for $\mathcal{IV}$ and $\mathcal{USR}_i$ to manually regenerate a new secret X_0 as in Section VII-A; there may be situations where $\mathcal{IV}$ is notified by governmental agencies such as tax or immigration services of a change in a user's identity, or a user may notify $\mathcal{IV}$ of a change in her identity in writing and may not be online to re-perform the exchange of X_0. Hence, $\mathcal{IV}$ should be able to unilaterally make changes to $\mathcal{USR}_i$'s records.

We propose that for each update, the X_0 should be updated as well. Suppose a given $\mathcal{USR}_i$'s identity is being updated for the k-th time to commit to an identity of $X_1^{(k)}, \dots, X_n^{(k)}$. Then, X_0 will be replaced by $X_0^{(k)} = X_{00} + H^k(X_{01})$, where H^k is SHA-256 applied k times. Even though $\mathcal{IV}$ does not know X_{00}, it can compute $g_0^{X_0^{(k)}} = h_{00} \cdot g_0^{H^k(X_{01})}$. Then, $\mathcal{IV}$ can compute the updated Brands commitment:

$$h^{(k)}_{\mathcal{USR}_i} = h_{\mathcal{USR}_i} \cdot g_0^{X_0^{(k)}} \cdot \left(g_0^{X_0}\right)^{-1} \cdot \prod_{j=1}^{n} g_j^{X_j^{(k)} - X_j}.$$

This schema has the result that

- If a field of $\mathcal{USR}_i$'s identity has changed, she can compute offline the new $X_0^{(k)}$, without having to re-perform an online communication with $\mathcal{IV}$ to transmit g^{X_0}, as she knows both X_{00} and X_{01}.
- Only $\mathcal{USR}_i$ knows X_{00}, thus only $\mathcal{USR}_i$ has the information necessary to construct X_0 and perform Brands proofs based on the commitment $h_{\mathcal{USR}_i}$.

Furthermore, we want this process to preserve the same privacy guarantees as (static) Brands proofs. We argue that, in the random oracle model, for an attacker who does not possess X_{01} or any of the $H^k(X_{01})$, the $X_{00} + H^k(X_{01})$ are randomly distributed (modulo the order of the elliptic curve). Hence, from the perspective of such an attacker, this process is equivalent to $\mathcal{USR}_i$ and $\mathcal{IV}$ re-performing the enrollment phase of Section VII to re-issue the identity with a new manually chosen, random X_0. See the appendix for more detail.

Remark 5: We recommend that at any given time, $\mathcal{IV}$ only store the $H^k(X_{01})$ necessary to perform the next update. As a result, even if $H^k(X_{01})$ is stolen from $\mathcal{IV}$'s servers, as long as this theft is detected before the user's identity is next updated, a thief will not have the means to remove the blinding factor and perform a dictionary attack against any of the previously published versions of the user's identity. Recall that these protections are based on the arguments of the appendix, which are themselves based on the work of Brands [16].

Remark 6: Note that, in the computation of $h^{(k)}_{\mathcal{USR}_i}$, $\mathcal{IV}$ does not need to know the values of the X_j that do not change, $X_j^{(k)} = X_j$. This can potentially allow identity verifiers to delete user information that they never expect to have to update later, even as the user can continue to their identity record to prove this aspect of their identity. As laws regulating the storage of personal data become more stringent, such as under the soon to come into effect General data protection regulation of the European Union [21], $\mathcal{IV}$ storing less customer data can mean less investment required in compliance departments and lower financial liabilities for data breaches.

C. Publication of DLREPs

$\mathcal{IV}$ will now post to the blockchain the root $r_{\mathcal{IV},t}$ of a Merkle tree that commits to the state at time t of the identities for each of its users. (Note that these are not the same Merkle trees that exist natively in Bitcoin, see IV-B, even though the same data structure is used.) To compute $r_{\mathcal{IV},t}$, $\mathcal{IV}$ must both add new users to the tree and edit the information for existing users if their DLREP has been updated as above. We describe how to proceed. Standard trees are dynamic structures, where leaves corresponding to an entry may move in the tree as its data evolves and grows in size. In our case, we want to keep fixed the path to a user's entry, so she can not authenticate using a previous, out-of-date, authentication path in the tree.

As a consequence, we use a Merkle tree that is very large relative to the number of users to allow for future growth. Specifically, based on a simplified version of the data structure proposed in [22], we use a virtual tree with 2^{256} leaves the paths to which are given using each bit of a SHA-256 hash to indicate which binary branch to take at each node. We denote $\mathcal{USR}$'s position in the tree by $\iota_{\mathcal{USR}_i}$, or the index of $\mathcal{USR}_i$ (see Figure 2 for an example). In [22] this path is given by the hash of verifiable random function of a username; here, $\mathcal{IV}$ may attribute to each user an account number and then take the hash of this value, $\iota_{\mathcal{USR}_i} = H(\text{account number}_{\mathcal{USR}_i})$. $\mathcal{IV}$ will publish in an `OP_RETURN` the root of this tree (at time t) $r_{\mathcal{IV},t}$ and also a commitment that tracks all changes made to the tree $cm_{\mathcal{IV},t}$ as below.

- For each user $\mathcal{USR}_i$ of $\mathcal{IV}$, an index $\iota_{\mathcal{USR}}$ is calculated as above. $\mathcal{IV}$ creates a 256 layer deep Merkle tree in which the leaf at each $\iota_{\mathcal{USR}_i}$ is $H(h_{\mathcal{USR}_i}, \text{metadata}_{\mathcal{USR}_i})$, where H is SHA-256. The metadata can include information such as an expiry data of the issued identity or other limitations on its use. All leaves that do not correspond to an $\iota_{\mathcal{USR}_i}$ contain an empty leaf. This tree has root $r_{\mathcal{IV},t}$.
- If any updates have been made to pre-existing identities, then the leaf consisting of the user's (now updated) information should remain in the same location of the tree $\iota_{\mathcal{USR}_i}$. In particular, if the metadata has changed for an identity, such as the extension of its expiration date, etc, this can be appended to or replace the previous metadata.
- For the first commitment $\mathcal{IV}$ makes, it takes $cm_{\mathcal{IV},1} = r_{\mathcal{IV},1}$. Later, if $\mathcal{IV}$ has already computed $cm_{\mathcal{IV},t-1}$, it computes $cm_{\mathcal{IV},t} = H(cm_{\mathcal{IV},t-1}, r_{\mathcal{IV},t})$.
- $\mathcal{IV}$ publishes to the Bitcoin blockchain a transaction, $\text{TX}_{\text{ENROLL}}$, of the following form:

Input Addresses	Output Addresses
$\text{a}_{\mathcal{IV}}$	$\text{a}_{\mathcal{IV}}$ $\texttt{OP\_RETURN}(r_{\mathcal{IV},t}, cm_{\mathcal{IV},t})$

The input from the address $a_{\mathcal{IV}}$ should be the output of the previous $\text{TX}_{\text{ENROLL}}$. One should calibrate to have one new transaction of this form in each block, namely approximately once every ten minutes.

As this transaction represents encoded semantic meaning in our system rather than financial data, the amounts of the inputs and outputs are secondary. Indeed, in the manner of [23], [24], [4], the amounts should be at or near the minimum output amounts to be considered a standard transaction ([19], [20]).

Remark 7: Note that the mechanism by which we create a Merkle tree of the $H(h_{\mathcal{USR}_i}, \text{metadata}_{\mathcal{USR}_i})$ is similar to the structure of how [22] creates a public key registry, as implemented into a blockchain in [17]. In [22], $\iota_{\mathcal{USR}_i}$ is calculated with a keyed, verifiable function of a username in a way that is designed to prevent third parties from determining the position in the tree corresponding to a given username. This is done to prevent such third parties from tracking whether the information corresponding to that username has changed by tracking changes in the intermediate hashes along the branch to its location, which we see below must be circulated to other users that have indices near $\iota_{\mathcal{USR}_i}$. As $\iota_{\mathcal{USR}_i}$ in our case is not computed from public information, these precautions are less relevant but are nonetheless compatible with our system at the expense of additional overhead in the case that the policies of $\mathcal{IV}$ require them.

Remark 8: Note that none of the history of the identity records emitted by $\mathcal{IV}$ can be changed without changing $cm_{\mathcal{IV},t}$. As such, one could eventually migrate this system to another blockchain by simply transferring the last value $cm_{\mathcal{IV},t}$, which ensures the integrity of all of the preceding information.

Remark 9: Rather than forming a hash tree of the identity records of $\mathcal{IV}$'s users, it would be more straightforward for each user to have their $h_{\mathcal{USR}_i}$ included in an `OP_RETURN` in a separate Bitcoin transaction. This is the model for [4], where each certificate is issued via its own transaction. Unfortunately, this presents scaling problems. First, as the number of transaction which $\mathcal{IV}$ has to issue increases, this increases the costs paid in transaction fees (compare to our estimates of these costs in Section XI). Second, Bitcoin blocks are currently limited to 1MB in total size and calibrated so that there is one block per 10 minutes [19]. As a result, there is a limit on the total number of Bitcoin transactions that can issued of about 7 transactions per second [5]. While there are efforts to increase this limit [25], our system if widely adopted risks having a number of users that exceeds what could be supported by the network if each must have her own issuing transaction.

D. Information to be distributed by $\mathcal{IV}$

In order for a user $\mathcal{USR}_i$ to be able to authenticate against her identity $h_{\mathcal{USR}_i}$, she will need to be able to demonstrate that $h_{\mathcal{USR}_i}$ is present in the last published Merkle tree. For this, the user will need the txid of one of $\mathcal{IV}$'s issuing transaction which can then be traced to the most up-to-date information published by $\mathcal{IV}$, $\text{txid}_{\mathcal{IV}}$, the location of her entry in the Merkle tree, $\iota_{\mathcal{USR}_i}$, and the intermediate hashes of the tree along the branch where this entry is located. If $\mathcal{USR}_i$'s identity is updated, in the most recent $r_{\mathcal{IV},t}$ this branch will lead to the updated commitment $cm_{\mathcal{IV},t}$, so $\mathcal{USR}_i$ will no longer be able to authenticate her old identity against it.

$\mathcal{USR}_i$ will need to know if her identity is updated. We imagine that if some field of $\mathcal{USR}_i$'s identity has been invalidated by $\mathcal{IV}$, that $\mathcal{IV}$ must notify her. Thus, $\mathcal{USR}_i$ should have up-to-date values of her $X_1 \dots X_n$. She can track the number of notifications of updates k that she receives in order to be able to calculate the current value of $X_0^{(k)} = X_{00} + H^k(X_{01})$.

We also envisage $\mathcal{IV}$ distributing non-sensitive information to the various service enablers that is nevertheless important to the functioning of the system. It makes sense for this information to be stored by $\mathcal{SE}$ whose primary role, in contrast to $\mathcal{IV}$, is in the facilitation of the identity system. Thus, we can avoid $\mathcal{IV}$ having to be "lively" and ready to distribute information at any given time to various users. As these $\mathcal{SE}$ *are* required by their business model to be lively so as to be able to enable authentications, this additional role does not substantially burden them.

Specifically, if any user's entry in the Merkle tree is updated, that will affect the intermediate hashes that then need to be made available to the other users a service which can be provided by $\mathcal{SE}$. Hence, we imagine $\mathcal{SE}$ storing the tree information $(H(h_{\mathcal{USR}_i}, \text{metadata}_{\mathcal{USR}_i}), \iota_{\mathcal{USR}_i})$ so that these intermediate hashes can be recalculated as necessary. See Section XII for estimation on how much data must be stored. On the other hand, if $\mathcal{IV}$ distributes all this information to all of the $\mathcal{SE}$ in the market, this results in a substantial redundancy of a significant amount of data. Hence, the best course of action may be to have $\mathcal{IV}$ distribute the entire tree to a small number of service enablers, then when a user wants to use her identity, she can download the required information from one of these service enablers whether that is the $\mathcal{SE}$ that she will later correspond with in the proof of identity phase (see Section VIII) or not.

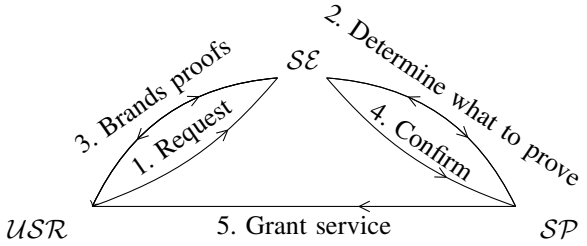

Figure 1. Schema of interactions between $\mathcal{USR}$, $\mathcal{SE}$, and $\mathcal{SP}$ during the proof of identity phase.

E. Comments on updating and revocation

As a special case of updating, $\mathcal{IV}$ can revoke an identity by replacing $h_{\mathcal{USR}_i}$ with an empty string.

$\mathcal{SE}$ can be assured of having the most recent information published by $\mathcal{IV}$, because if $\mathcal{SE}$ checks an out-of-date entry, it will see that the output to $\mathtt{a}_{IV}$ has been spent. $\mathcal{SE}$ can then follow a chain of transactions to the last published Merkle root.

Our system takes advantage of the fact that, in Bitcoin, a decentralized, Byzantine fault resistant, highly available system of nodes is tracking whether updates are made to their list of UTXOs. As any update made to our system implies an update tracked by these nodes, this provides an effective mechanism to track updates to our system. See [4] and [17] for architectures that employ a similar idea in different use cases.

VIII. Proof of identity phase

In this section we show how $\mathcal{USR}$ can use an identity that she has been issued by $\mathcal{IV}$ via the process of Section VII to authenticate herself and gain access to a service provided by $\mathcal{SP}$. We suppose that $\mathcal{USR}$ has downloaded the intermediate hashes on the path to her entry of the Merkle tree as in Section VII-D from some service enabler (that is not necessary the same as $\mathcal{SE}$ below). As we discussed in Section V, $\mathcal{USR}$ and $\mathcal{SE}$ should have previously properly set-up the necessary keys to open the secure communications channel $TLS_{pk_{\mathcal{USR}}, pk_{\mathcal{SE}}}$. (This might occur upon $\mathcal{USR}$ downloading $\mathcal{SE}$'s mobile app and opening an account). Then, the following steps are performed, where communication between $\mathcal{USR}$ and $\mathcal{SE}$ is done through this channel:

In more detail, over the TLS channel between $\mathcal{USR}$ and $\mathcal{SE}$, $\mathcal{USR}$ communicates to $\mathcal{SE}$ what service she is requesting from which service provider. Then, $\mathcal{SE}$ creates a session id and then communicates with $\mathcal{SP}$ to establish what information about $\mathcal{USR}$'s identity needs to be established and provides a list $\{\mathcal{IV}_1, \dots, \mathcal{IV}_m\}$ of identity verfiers from which the service provider accepts documents. Again over the TLS channel between $\mathcal{USR}$ and $\mathcal{SE}$, $\mathcal{USR}$ communicates the txid, $h_{\mathcal{USR}}$, and

User data: $pk_{\mathcal{USR}}$, $sk_{\mathcal{USR}}$, $pk_{\mathcal{SE}}$, $\text{txid}_{\mathcal{IV}}$, $\iota_{\mathcal{USR}}$, branch of Merkle tree, $X_0, \ldots X_n$.

1: Request $\mathcal{USR} \rightarrow \mathcal{SE}$: (Name of service, $\mathcal{SP}$)
2: Determine what to prove: $\mathcal{SE} \leftrightarrow \mathcal{SP}$:
- $\mathcal{SE} \rightarrow \mathcal{SP}$: (Name of service, Session id)
- $\mathcal{SP} \rightarrow \mathcal{SE}$: (Info to prove, $\{\mathcal{IV}_1, \ldots, \mathcal{IV}_m\}$, Session id)

3: Prove: $\mathcal{USR} \leftrightarrow \mathcal{SE}$:
- $\mathcal{SE} \rightarrow \mathcal{USR}$: (Info to prove, $\{\mathcal{IV}_1, \ldots, \mathcal{IV}_m\}$, Session id)
- $\mathcal{USR} \rightarrow \mathcal{SE}$: ($\text{txid}_{\mathcal{IV}_j}$, $h_{\mathcal{USR}}$, branch of Merkle tree, Session id)
- $\mathcal{SE}$ uses the provided branch and the publicly available, most recent version of $r_{\mathcal{IV},t}$ to check that $h_{\mathcal{USR}}$ is present in the tree and up-to-date
- $\mathcal{USR} \leftrightarrow \mathcal{SE}$: (interactive proof π, Session id)
- $\mathcal{SE}$ checks the proof

4: Confirm: $\mathcal{SE} \rightarrow \mathcal{SP}$: (Session id)
5: Grant of service: $\mathcal{SP} \rightarrow \mathcal{USR}$ (if this service is digital, it might pass through the established connections between $\mathcal{SP}$ and $\mathcal{SE}$ en route to $\mathcal{USR}$)

the necessary branch of the Merkle tree corresponding to her relevant identity as established in Section VII. Using the txid, $\mathcal{SE}$ can follow the chain of updates, as in Section VII-B, to obtain the most up-to-date version of $r_{\mathcal{IV},t}$. Then, $\mathcal{SE}$ can use $h_{\mathcal{USR}}$ and the intermediate hashes on the branch to compute a value that can be compared to $r_{\mathcal{IV},t}$ to confirm the presence of $h_{\mathcal{USR}}$ in the most up-to-date version of the tree, see Figure 2. $\mathcal{SE}$ can also verify that this identity was issued by an $\mathcal{IV}$ on the list provided by $\mathcal{SP}$. $\mathcal{SE}$ makes interactive requests of knowledge which $\mathcal{USR}$ establishes via the interactive proof π. Note that the ability of $\mathcal{USR}$ to provide these proofs depends on her knowledge of $X_0, \ldots X_n$. Particularly, X_0, which only $\mathcal{USR}$ should know as she randomly generates the secret X_{00} component, acts as a sort of key to the use of this identity record.

Once $\mathcal{SE}$ is satisfied with the proofs provided by $\mathcal{USR}$, $\mathcal{SE}$ sends a confirmation to $\mathcal{SP}$, and $\mathcal{SP}$ grants the service to $\mathcal{USR}$.

Remark 10: While the Brands proofs protect the user from having to reveal unnecessary identity information to $\mathcal{SP}$, $\mathcal{SE}$ nevertheless observes details about $\mathcal{USR}$'s transactions, specifically with which service providers $\mathcal{USR}$ is interacting. If $\mathcal{SE}$ notices suspicious behavior on the part of the user, $\mathcal{SE}$ might refuse to process a transaction or contact $\mathcal{IV}$ with a warning in a manner akin to how fraudulent credit card activity is detected.

Remark 11: Note that, if $\mathcal{SP}$ is not willing to completely trust any single service enabler, a user might be required to perform this process with several different $\mathcal{SE}$. A smart phone app with which the user has already exchanged keys to establish TLS connections with each $\mathcal{SE}$ could effectively coordinate this process.

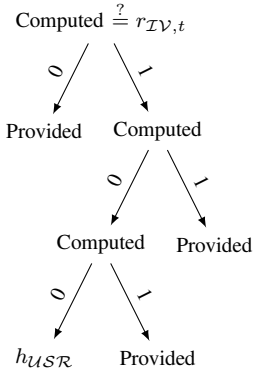

Figure 2. The verifications performed by $\mathcal{SE}$ to prove that $h_{\mathcal{USR}}$ is a branch in the tree using the intermediate hashes of the branch provided by $\mathcal{USR}$. Namely, at each step moving up from the bottom, $\mathcal{SE}$ computes the hash of what has already been computed with the next provided intermediate hash and compares the ultimate result with $r_{\mathcal{IV},t}$. Note that $\iota_{\mathcal{USR}}$ in this example, which gives the position of $h_{\mathcal{USR}}$ in the tree, is 100.

IX. Key loss

We have seen that users must store the following information: $pk_{\mathcal{USR}}$, $sk_{\mathcal{USR}}$, $pk_{\mathcal{SE}}$, $\text{txid}_{\mathcal{IV}}$, and $X_0, \ldots X_n$ in order to use to perform authentications (and she must also have the intermediate hashes on the branch of the Merkle tree on which their identity record is located, possibly re-downloading them from a service enabler). We consider the contingencies if a user loses access to some or all of this information due, for example, to a lost phone.

Note that $sk_{\mathcal{USR}}$ serves only to establish a secure communications with $\mathcal{SE}$, allowing them to exchange (potentially) sensitive information. Particularly, as this key corresponds to an account that can be obtained anonymously, $sk_{\mathcal{USR}}$ is not the basis for proving user identity, and serves mainly to protect against man-in-the-middle attacks, where the attacker intercepts and replays messages from $\mathcal{USR}$ in order to be granted access to $\mathcal{SP}$. This has the consequence that if $sk_{\mathcal{USR}}$ is lost, $\mathcal{USR}$'s identity is not compromised, nor is her ability to perform Brands proofs. Hence, in this case, $\mathcal{USR}$ should simply generate a new pair $sk'_{\mathcal{USR}}$, $pk'_{\mathcal{USR}}$, and communicate $pk'_{\mathcal{USR}}$ to $\mathcal{SE}$.

On the other hand, if X_0 is lost, and there is not a backup available to the user, she should contact $\mathcal{IV}$, again with her physical identity documents to justify her identity. Then $\mathcal{IV}$ can issue an update to an existing identity record to establish a new X'_0 as in Section VII-B.

X. Security model

As we discussed in Section V, $\mathcal{USR}$ must trust both $\mathcal{IV}$ and $\mathcal{SE}$ to handle her personal information appropriately; however, the information that $\mathcal{USR}$ must convey to $\mathcal{SE}$ is generally much more limited than that conveyed to $\mathcal{IV}$. Technically, $\mathcal{USR}$ and $\mathcal{SE}$ should have the capacity to perform interactive Brands proofs. As we saw in Proposition 1, the exchange of Brands proofs reveals nothing about an identity except what statements are being proved. Nonetheless, these statements may be themselves sensitive. By encrypting them, our system

protects this information in the same matter as a tradition online authentication system even as the information that the user must provide to $\mathcal{SE}$ is minimized.

We assume that $\mathcal{IV}$ properly issues updates to an identity $h_{\mathcal{USR}}$ by overwriting the previous version of that identity at index $\iota_{\mathcal{USR}}$. If $\mathcal{IV}$ fails to do this $\mathcal{USR}$ may have the opportunity to continue to authenticate herself against an identity that is out-of-date but appears valid to $\mathcal{SE}$. Note that if $\mathcal{IV}$ realizes that this type of error has been made (either due to coding error, infiltration by hackers, etc), then it can correct these errors in the next update, revoking redundant entries.

Abstractly, we make use of the *public ledger* functionality of Bitcoin, namely that it acts as a "bulletin board" on which anyone can post messages and read the messages that have previously been posted. Specifically, we require that Bitcoin have the properties of *liveness*, i.e. every honest participant will have its posted messages seen by every honest participant after some delay, and *persistence*, namely that every posted message will indefinitely be seen at the same position by all participants, see [26] and [27]. We also depend on the integrity of the Bitcoin transaction verification procedure to check the validity of transactions which includes checking that each non-generation transaction has inputs corresponding to previous transaction outputs, etc. The Bitcoin core protocol has been proven to have these properties when quantitative bounds are assumed on the relative power of the adversary to the honest players, in terms of computing power in [26], or in terms of computing power and influence over the peer-to-peer network in [27]. However, these results are theoretical; in addition to the risk that a hostile adversary may become powerful enough to invalidate the quantitative bounds, there is also the possibility that coding bugs or other accidental situations may cause problems such as small forks, peer-to-peer failures, etc [28].

More concretely, the Bitcoin aspect of our protocol serves to allow $\mathcal{IV}$ to update an identity in an unequivocable manner, and provides neutral infrastructure that is computationally protected against unauthorized modifications. The consequence for our system of a fork would be to allow a user to continue to authenticate against a non-updated copy of an identity that should have been updated. This can be particularly consequential if $\mathcal{IV}$ attempts to revoke a user identity. As a result, service providers and service enablers might consider suspending particularly sensitive authentications if $\mathcal{SE}$ detects indications of a fork.

Note that the service enabler is the only actor in our schema that needs to be capable of verifying the status of Bitcoin transactions. In order to obtain the most up-to-date information on the network available, $\mathcal{SE}$ should operate a full node. (However, a user who wants to emulate the checks performed by $\mathcal{SE}$ to see whether her identity is still valid could obtain information about the Bitcoin network from running a Simplified Payment Verification client [19] or even one or more block explorers, accepting the risk that attacks against these methods might hide from her an update to her identity that would be seen by a full node.)

It is common practice for merchants accepting bitcoin to wait several confirmations (several mined blocks) before accepting a transaction as valid. A common rule-of-thumb is to wait six confirmations (approximately one hour) after a transaction to be confident that this transaction cannot be reversed [19, Chapter 2]. In our scheme, the only party who can double spend is $\mathcal{IV}$, who we do not generally expect to issue malicious, contradictory updates. If $\mathcal{IV}$ issues an update transaction that is included in an orphaned block, but ultimately does not make it into the main chain, $\mathcal{IV}$ can merely include those user updates in a reissued transaction later.

$\mathcal{SE}$ and $\mathcal{SP}$ may still want to wait for a transaction to be included in a few blocks so as to avoid authenticating users against ephemeral records for auditing reasons. However, since only $\mathcal{IV}$ can perform double spending in our scheme, our system does not require the same level of caution against double spending attacks that is employed for substantial financial transactions in Bitcoin.

Finally, as Bitcoin is integrated into our structure, a network attack that makes it impossible for $\mathcal{IV}$ to issue new transactions or for $\mathcal{SE}$ to obtain information on the latest transactions would function as a denial of service attack on our system. This reflects the importance of using a well-established blockchain such as Bitcoin with a large network rather than a smaller, newer blockchain to be as robust as possible against such attacks.

XI. Bitcoin transactions costs

We estimate the cost in Bitcoin transaction fees for an $\mathcal{IV}$ to use this system. As the $\texttt{TX}_{\texttt{ENROLL}}$ transaction has one `OP_RETURN` output containing two SHA-256 hashes, P2PKH output to $a_{\mathcal{IV}}$, —and one input (corresponding to a P2PKH output of a previous transaction), the total size of its raw transaction is approximately 265 bytes [19]. The amount of bitcoin that must be paid in fees for a transaction of a given sizes fluctuates based on market forces as miners must choose which transactions to include in the block they are mining with limited space; current (May 2017) estimates [29] suggest that a fee of .0000036 bitcoin per byte is sufficient to have a high likelihood that a transaction will be included in the next block. This results in a fee for $\texttt{TX}_{\texttt{ENROLL}}$ of approximately .000954 bitcoin or 2.17 USD (1 BTC=2280 USD, May 2017). If $\mathcal{IV}$ has enough clients to justify emitting a $\texttt{TX}_{\texttt{ENROLL}}$ every 10 minutes, this would require transaction fees of approximately 312 USD per day.

XII. Storage and bandwidth requirements

Consider that $\mathcal{IV}$ manages $N = 2^{26} \approx 67$ million users, the size of a large bank. The dynamics of updates are low since user identity is rather static, so for example, we consider that a fraction $f_{\text{daily}} = 1\%$ of users have their identity fields changing daily. This means a fraction $f = f_{\text{daily}}/(6 \cdot 24)$ updates every Bitcoin block.

$\mathcal{IV}$ will issue updates which will be reflected in the root hash of the Merkle tree created in the enrollment phase. However, as discussed in Section VII-D, during the interaction between $\mathcal{SE}$ and $\mathcal{USR}$ during the proof of identity phase, these parties need to have access to the intermediate hashes of this Merkle tree so that they can verify that $h_{\mathcal{USR}}$ is present in the most recent version of the tree.

A. $\mathcal{SE}$ stores information on behalf of $\mathcal{IV}$

In order to be able to calculate the appropriate intermediate hashes for any potential user of $\mathcal{IV}$'s system, $\mathcal{SE}$ needs to store all of the (occupied) leaves of the tree. This means storing, for each leaf, a Brand commitment and the index of this leaf in the tree $(h_{\mathcal{USR}}, \iota_{\mathcal{USR}})$, which consists of a total of 64 bytes per leaf. Hence, $\mathcal{SE}$ must store $N \cdot 64$ bytes, or under our assumptions $2^{26} \cdot 64 = 4.3$ gigabytes for each $\mathcal{IV}$ for which this service is provided. In an average block then $\mathcal{SE}$ will need to update $f \cdot N \cdot 64$ bytes, approximately 298 kilobytes each 10 minutes for each $\mathcal{IV}$.

Additionally, in order to maintain a full node, $\mathcal{SE}$ must download all new blocks, namely approximately 1 MB every 10 minutes, independent of the number of identity verifiers.

B. $\mathcal{USR}$ stores information necessary for her own authentications

For $\mathcal{USR}$ to store the intermediate hashes with which she can verify her own identity against the root hash of the tree, she does not need to store the entire tree. Instead, she can merely store the single path down the tree against which her identity will be hashed, to be presented to $\mathcal{SE}$ during the proof of identity phase. A priori, this would require storing 256 hashes each of 32 bytes, each of which will have to be regularly updated as the other users update their information. However, as most of the leaves of the tree are empty, in fact, $\mathcal{USR}$ will on have average lower storage and bandwidth requirements.

Based on the same argument as in [22, Section 5.3], an average user will only have $\log_2(N)$ many non-empty intermediate hashes on her authentication path. If for each intermediate hash, $\mathcal{USR}$ also stores its location in the tree, this corresponds to $64\log_2(N)$ bytes, or under our assumptions to storage of 1664 bytes. Similarly, as there are $f \cdot N$ modification made to the tree per block, on average only $\log_2(f \cdot N)$ elements of $\mathcal{USR}$'s authentication path in the tree will change. For each change, $\mathcal{USR}$ requires the new value of this intermediate hash and its position in the tree. Namely, $\mathcal{USR}$ must download $64\log_2(f \cdot N)$ bytes of information for a total of approximately 780 bytes each block under our assumptions.

Thus, when a user wants to use her identity, she can download the required information from one of these $\mathcal{SE}$ whether that is the service enabler that she will correspond with in the proof of identity phase or not. Depending on how much time has passed since the user was last online, she may need to download between 780 and 1664 bytes as discussed in Section XII-A.

XIII. Comparison of revocation mechanism to revocation in PKIs

We briefly consider the advantages and disadvantages of the update and revocation mechanism presented in Section VII to the standard revocation systems that are used in public key infrastructures: certificate revocation lists (CRLs) and the Online Certificate Status Protocol (OCSP).

Even though use of CRLs is limited to revoking certificates, rather than allowing for more complex updates, their size, which is linear in the number of revocations, can become quite large. The median certificate has a revocation list on the order of 50 KB, and CRLs of several megabytes are not uncommon [1]. We saw in Section XII that the amount of information required to verify the status of a user's identity against an up-to-date $r_{\mathcal{IV},t}$ is logarithmic in the number of $\mathcal{IV}'s$ users, on the order of 1 KB. On the other hand, as also discussed in Section XII, the bandwidth required to run a Bitcoin full node to be able to confirm that $r_{\mathcal{IV},t}$ is, in fact, up-to-date is roughly 1MB every ten minutes, independent of the number of identity verifiers. There are nonetheless situations where $\mathcal{SE}$ is interacting with many identity verifiers, particularly when $\mathcal{SE}$ is performing the role of a single sign on service, where our update model would be advantageous from a bandwidth perspective. Moreover, a user that runs a Simplified Payment Verification client to validate that her identity is up-to-date, need only download 80 byte block headers every ten minutes and approximately 1 KB of information for each verification of $r_{\mathcal{IV},t}$ [19].

OCSP was developed to alleviate the bandwidth concerns of CRLs at the expense of requiring Certificate Authorities to be "lively" in responding to requests on revocation status in real time. As discussed in Section V, we seek to avoid requiring such liveliness from identity verifiers.

Moreover, as shown in [1], a number of widely used, modern browsers accept revoked certificates in certain circumstances. In contrast, note that our update mechanism is integrated into the issuing system so that checking that a record is up-to-date is done in the same process as checking that that record exists at all. Hence, we minimize the possibility for authentication to be separated from appropriate revocation.

Finally, denial of service attacks can be performed on public key infrastructures by preventing verifiers from downloading CRLs or communicating with a OCSP server. In our system, as we saw in Section X, such attacks are defended by the infrastructure of the Bitcoin network. In exchange for the protections of this network, potentially non-negligible Bitcoin fees must be paid, as seen in Section XI.

As such, while we are not proposing to completely replace PKI models of revocation with blockchain based models (indeed, the TLS connections we propose using during the proof of identity phase make use of traditional public key infrastructures), we argue that the revocation and update mechanisms we have presented are a useful tool that will be beneficial in certain use cases.

XIV. Conclusion

We have presented an architecture in which users can authenticate themselves against records established by "identity verifiers," such as a bank or a utility company, even as these verifiers do not need to store the personal data of the user directly, following the spirit of [21]. As the business model of the verifers does not necessarily focus on identification, we outsource much of the operation of this system to a third party "service enabler" who nonetheless learns no more than what is necessary about the identity of the user. By making use of the Bitcoin blockchain, we allow for a verifier to update or revoke a user's identity in a streamlined fashion.

REFERENCES

[1] Y. Liu, W. Tome, L. Zhang, D. R. Choffnes, D. Levin, B. M. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's PKI," in *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, 2015, pp. 183–196. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815685

[2] D. Yang, J. Gavigan, and Z. Wilcox-O'Hearn, "Survey of confidentiality and privacy preserving technologies for blockchains," Nov. 2016, https://z.cash/static/R3_Confidentiality_and_Privacy_Report.pdf.

[3] O. Jacobovitz, "Blockchain for identity management," Dec. 2016, https://www.cs.bgu.ac.il/%7Efrankel/TechnicalReports/2016/16-02.pdf.

[4] J. Nazaré, K. Hamilton, and P. Schmidt, "Digital certificates project," online, source code available at https://github.com/digital-certificates, Consulted 2016 December, http://certificates.media.mit.edu.

[5] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC '16), Denver, CO, USA, June 22-24, 2016. Proceedings*, 2016, pp. 181–194.

[6] C. Lundkvist, R. H. ans Joel Torstensson, Z. Mitton, and M. Sena, "uPort: A platform for self-sovereign identity," 2017.

[7] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Online: http://gavwood.com/paper.pdf, EIP-150 REVISION, Consulted September 2017.

[8] "Namecoin," Consulted 2017 July, https://namecoin.org/.

[9] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A decentralized public key infrastructure with identity retention," *ArXiv e-prints*, 2014.

[10] C. Garman, M. Green, and I. Miers, "Accountable privacy for decentralized anonymous payments," in *Financial Cryptography and Data Security*, 2016.

[11] "IDCoins," Consulted 2017 April, https://github.com/IDCoin/IDCoin.

[12] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Symposium on Security and Privacy Workshops, SPW 2015, San Jose, CA, USA, May 21-22, 2015*. Los Alamitos, CA, USA: IEEE Computer Society, 2015, pp. 180–184.

[13] "Estonian e-residency," Consulted 2017 March, https://e-estonia.com/e-residents/about/.

[14] G. Prisco, "Estonian government partnerts with Bitnation to offer blockchain notarization services to e-residents," 2015 November, https://bitcoinmagazine.com/articles/estonian-government-partners-with-bitnation-to-offer-blockchain-notarization-services-to-e-residents-1448915243/.

[15] T. Hardjono, N. Smith, and A. S. Pentland, "Anonymous identities for permissioned blockchains," Jan. 2016, http://www.the-blockchain.com/docs/MIT-ChainAnchor-DRAFT.pdf.

[16] S. Brands, *Rethinking Public Key Infrastructures and Digital Certificates (Building in Privacy)*. Cambridge, MA, USA: MIT Press, 2000.

[17] A. Tomescu and S. Devadas, "Catena: Efficient non-equivocation via Bitcoin," Online, https://people.csail.mit.edu/devadas/pubs/catena.pdf, 2016.

[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Online, http://bitcoin.org/bitcoin.pdf, 2008.

[19] A. M. Antonopoulos, *Mastering Bitcoin*. Sebastopol, California: O'Reilly Media, 2010.

[20] T. B. C. developers, "Bitcoin transactions primitives code," Consulted 2017 March, https://github.com/bitcoin/bitcoin/blob/0.14/src/primitives/transaction.h.

[21] European Parliament and European Council, "Regulation (EU) 2016/679 of the European Parliament and of the Council," 2016, http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1495547057161&uri=CELEX:32016R0679.

[22] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "Bringing deployable key transparency to end users," in *24th USENIX Security Symposium (USENIX Security 2015) Berkeley, CA, USA, August 2015*, 2015, pp. 383–398.

[23] A. Shomer, "The Colored Coins protocol," Consulted 2017 March, https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification/wiki.

[24] F. Charlon, "Open assets protocol (oap/1.0)," Online, https://github.com/OpenAssets/open-assets-protocol/blob/master/specification.mediawikii, 2011.

[25] O. Beigel, "Segwit vs. Bitcoin Unlimited and Bitcoin's fork explained simply," 2017 April, https://99bitcoins.com/bitcoin-fork-segwit-vs-bitcoin-unlimited-explained-simply/.

[26] J. A. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, ser. Lecture Notes in Computer Science, E. Oswald and M. Fischlin, Eds., vol. 9057. Springer, 2015, pp. 281–310.

[27] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, ser. Lecture Notes in Computer Science, J. Coron and J. B. Nielsen, Eds., vol. 10211. Springer, 2017, pp. 643–673.

[28] V. Buterin, "Bitcoin network shaken by blockchain fork," 2013 March, https://bitcoinmagazine.com/articles/bitcoin-network-shaken-by-blockchain-fork-1363144448/.

[29] "Predicting Bitcoin fees for transactions," Consulted 2017 May, https://bitcoinfees.21.co/.

APPENDIX

We use the framework of Brands' thesis [16] to discuss and analyze the security of our update method in VII-B. In particular, [16, Definition 2.3.1] introduces the notion of *instance generators* for one-way and collision intractable functions. In the DLREP case, an instance generator is a way to generate the group and the g_i's, and also to generate $X_0, \ldots, X_n$. In Section IV-A, we have essentially recalled Brands standard DLREP instance generator, and as soon as X_0 is random, the DLREP function is one-way and collision intractable [16, Proposition 2.3.3]. As noted by Brands [16, page 61], if an instance generator is indistinguishable from the basic DLREP instance generator, the security theorems of his thesis still hold.

We prove here that, under the random oracle model (i.e. the hash function H is a random function), the instance generator provided by the method in VII-B is indistinguishable from the basic method in IV-A. In IV-A, X_0 is generated at random, while in VII-B, $X_0^{(k)} = X_{00} + H^k(X_{01})$, where X_{00} and X_{01} are random. Since H is modeled as a random function, and since the addition is done in a cyclic group, $X_0^{(k)}$ is as random as X_0 in the basic method, thus, the instance generators are statistically indistinguishable. This proves that each updated $h^{(k)}_{\mathcal{USR}_i}$ is a one-way and collision intractable function of X_{00}, X_{01}, and $X_1^{(k)}, \ldots, X_n^{(k)}$.

Now we discuss the security with respect to a malicious $\mathcal{IV}$. [16, Proposition 2.4.8] and the following discussion also shows that the basic protocol for proving knowledge of a DLREP in IV-A is complete and sound for the basic instance generator. Being indistinguishable, the same holds true for our instance generator in VII-B. Thus, if a dishonest $\mathcal{IV}$, knowing X_{01}, wants to use $h^{(k)}_{\mathcal{USR}_i}$ to authenticate as the user, it has to know $X_0^{(k)} = X_{00} + H^k(X_{01})$ (completeness), which is equivalent to $\mathcal{IV}$ knowing X_{00} (subtraction). Thus, it is not feasible for $\mathcal{IV}$ to authenticate without knowing the secret X_{00}, which is protected by the hardness of the discrete logarithm problem. This continues to hold for each successive updated $h^{(k)}_{\mathcal{USR}_i}$.