Practical solutions to save bitcoins applied to an identity system proposal

Daniel Augot[1,2,3], Hervé Chabanne[4,5] and William George
[1]*Inria, France*
[2]*Laboratoire LIX, École Polytechnique & CNRS UMR 7161, France*
[3]*Université Paris-Saclay, France*
[4]*Idemia, France*
[5]*Télécom ParisTech, France*

Keywords: Bitcoin, Atomic swaps, Off-chain payment channels, Identity system

Abstract: In a recent work by Augot et al. (2017), a scheme is proposed to build an identity system on top of the Bitcoin network. However, this proposal incurs very high costs since Bitcoin transactions require heavy fees. The current work introduces modifications to their scheme to make it more cost efficient while preserving its potential. Namely, we build on features of Bitcoin's scripting language, which allows swapping coins between two compatible blockchains, and also on off-chain transactions.

1 INTRODUCTION

In recent years, there has been much discussion of the possibility of using blockchains for identity management (Yang et al., 2016). Generally, proposals in this sense attempt to take advantage of the decentralized space offered by blockchains to provide a less top-down model of identity management, empowering users to take a greater control over their identity in the spirit of Person Identity Management Systems (PIMS) (Abiteboul et al., 2015).

There have been many, varied proposals for how blockchains can be used in identity management. Among projects that are currently targeting enterprise clients, uPort (Lundkvist et al., 2017) has offered a solution based on Ethereum, while Sovrin (Tobin et al., 2016) has created a solution based on a permissionned blockchain. Proposals for blockchain based identity systems in government include that of the e-Estonia program (Prisco, 2015), `https://e-estonia.com/solutions/security-and-safety/ksi-blockchain/`, `https://e-estonia.com/wp-content/uploads/faq-a4-v02-blockchain.pdf`. See (Jacobovitz, 2016; Dunphy and Petitcolas, 2018) for overviews of projects in this space.

Particularly relevant to our work is a certificate issuing scheme built onto the Bitcoin blockchain proposed by MIT MediaLabs (Nazaré et al., 2016), where a certificate is associated to a Bitcoin transaction. This system has the interesting property that revocation of a certificate is done by emitting a Bitcoin transaction whose input consumes the output of the certificate; as such, protection against attacks using revoked certificates is inherited from the immutability and double spend protections of Bitcoin's blockchain. These ideas are then further developed in (Augot et al., 2017a) and (Augot et al., 2017b). Both of these works integrate the use of Brands selective disclosure credentials (Brands, 2000), first used in the context of blockchains in (Garman et al., 2016), for user privacy. These user privacy ideas are then integrated with the transactional structure of Bitcoin in order to effectively manage revocation of identity documents, developing the ideas of (Nazaré et al., 2016), however (Augot et al., 2017a) and (Augot et al., 2017b) achieve this in somewhat different ways.

Principal advantages (Augot et al., 2017a) include:

- a flexible user experience, particularly allowing users to coordinate micro-identities from different issuers;
- security from Bitcoin's blockchain for
  - enabling monitoring (such as by imposing a limit of uses of the identity), or building reputation, based on the usage history of an identity record
  - checking that a record has not been revoked that can be performed by a simplified payment verification client (SPV), rather than requiring a full node;
- the Identity Provider (see below), $\mathcal{IP}$, must be on-

line only to issue and revoke identities, and does not need to continuously manage revocation status such as, e.g. in the Online Certificate Status Protocol. Essentially, $\mathcal{IP}$ has outsourced its role of maintaining a live database of its certificates (revoked or nor) to the miners and the underlying of P2P network of Bitcoin.

The downside of the system of (Augot et al., 2017a) is that the number of required Bitcoin transactions results in currently non-viable fees. See Sec. 3 for further details.

The proposal of (Augot et al., 2017b) is more financially viable. In this system, each Bitcoin transaction in the protocol corresponds to the root of an entire Merkle tree of user identity documents that are being issued. This proposal, however, lacks certain good properties of the system of (Augot et al., 2017a), as it requires verifiers to be a full node for checking the revocation status, instead of being a light, SPV client, and that users must place some amount of trust in an intermediary.

In the present work, we propose a number of modifications to the architecture of (Augot et al., 2017a) that are designed to make this system more affordable, while preserving as many of its positive features as possible. These modifications are based on tools such as atomic swaps and off-chain protocols, in some cases inspired by proposals to make the Bitcoin network itself more cost efficient.

We begin in Sec. 2 by introducing some notions and notations related to Bitcoin. In Sec. 3, we recall the system of (Augot et al., 2017a). In Sec. 4, we present the first adaption of (Augot et al., 2017a) which allows moving the scheme from the Bitcoin blockchain to Litecoin where lower fees makes the scheme more economical. In Sec. 5, we show how identity services can be delivered off-chain. Sec. 6 concludes.

2 BITCOIN AND SOME NOTATIONS

Types of transaction outputs. Bitcoin employs a limited scripting language whose expressive capacities to determine how an output can be spent we make use of in this work. Bitcoin transaction outputs typically fall into one of the following types (see (Antonopoulos, 2015)):

- P2PKH: these are the most commonly occurring Bitcoin transactions. The hash of a public key is specified, then to spend the output, a transaction must be signed with the corresponding private key.
- P2SH: these transactions outputs include the hash of a script. To spend this output, the script must be presented, and whatever conditions in that script must be satisfied. P2SH outputs give Bitcoin a certain flexibility in their scripting language. Notable uses include
  - multisig outputs where the script specifies how many and among which public keys must be used to sign the spending transaction
  - bounties, where the spender must present some fixed secret value R
- `OP_RETURN`: these outputs have an amount associated to them of 0 BTC, hence they are provably non-spendable. They allow for one to place up to 80 bytes of arbitrary data in the transaction, which is then recorded in the blockchain.

Fees. The difference between the value of inputs to a Bitcoin transaction and its outputs is paid in fees to the miner who finds the block in which this transaction is included. As each miner chooses which transactions he wants to include in a block and as the total size of blocks is limited, a market has developed for what fees need to be paid for a transaction to be included in a block in a timely manner `https://bitcoinfees.21.co/`.

In our work we present several models of transactions and we indicate the fees paid by $\mathrm{F}_{\texttt{NAME-OF-TRANSACTION}}$.

Dust. Non-`OP_RETURN` transaction outputs must have positive amounts assigned to them. Moreover, in order for a transaction to be considered "standard" by network participants using the Bitcoin Core software (Antonopoulos, 2015), each non-`OP_RETURN` output must have an amount that is at least .000005 bitcoin `https://github.com/bitcoin/bitcoin/blob/0.14/src/primitives/transaction.h`. In this work, we will generally denote by $\mathcal{D}$ the smallest amount of bitcoin that can be attached to a transaction output so that the transaction is considered standard. Non-standard transactions will not be circulated by network participants, although blocks that contain non-standard yet valid transactions are still accepted.

Full nodes and obtaining network information. A full node stores all of the information in every Bitcoin block. Thus, a full node can verify that the root of the Merkle tree of transactions in a block corresponds to what is published in the block header. The downside to operating a full node is that, as the block chain grows in size, this requires substantially bandwidth and storage.

In contrast, the Simplified Payment Verification protocol (SPV), which was already proposed by

Satoshi (Nakamoto, 2008), allows for one to verify that a given transaction has been included in the blockchain without having to operate a full node. This is particularly important for merchants who want to see that a given transaction paying them for a service has processed without having to download the entire history of the network. An SPV client downloads the block headers for all blocks, then when they want to verify a given transaction they communicate with full nodes and obtain all of hashes along the Merkle branch of that transaction. As such, they can verify that the transaction correctly hashes with the hashes along that branch to the root in the block header.

Timelocks. The most basic type of timelock in Bitcoin, which uses the field nLockTime and has been present since the original creation of the network, specifies a time/block before which a transaction cannot be added to the blockchain `https://en:bitcoin:it/wiki/Timelock`.

There are other, newer, more exotic types of timelocks, such as those that control when a single transaction output can be spent `https://en:bitcoin:it/wiki/Timelock`, but we will only require nLockTime in our work.

Notations. For the Bitcoin transaction described in this paper, we use the following notations: "$\mathtt{a}$ for an amount: x" means that x coins are sent to address $\mathtt{a}$, "$\mathtt{MSIG}i\_j(\mathtt{a}^{(1)},\ldots,\mathtt{a}^{(j)})$" means that i valid signatures are required from the addresses $\mathtt{a}^{(1)}\ldots,\mathtt{a}^{(j)}$

For more complex output scripts encoded with a P2SH output, we use simple Boolean logic with "and" and "or", and occasionally, the notation "needs(R)" means that, for the cryptographic hash function H, $H(R)$ for a random R has been included in the output script, and that R must be provided in the redeeming transaction.

Also, given a transaction $\mathtt{TX}_{\mathtt{NAME}}$, we denote by $\mathtt{F}_{\mathtt{NAME}}$ the associated fees received by the miners

3 IDENTITY MANAGEMENT SCHEME OF AUGOT ET AL.

We are building our work upon the model of (Augot et al., 2017a). In particular, we imagine similar use cases and have mostly the same actors that participate in our system.

Actors. The actors in the work of (Augot et al., 2017a) are:

- Users – $\mathcal{USR}$ – are people that obtain and use identity documents through this system.
- Identity providers – $\mathcal{IP}$ – are entities, such as banks and utility companies, or governments, which play some role in issuing documents for use in identity. Typically, $\mathcal{IP}$ will physically verify the documents of a user (passport, etc) and issue a record on the blockchain.
- Service providers – $\mathcal{SP}$ – are entities to whom a user needs to prove facts about her identity, on the basis of a document issued by $\mathcal{IP}$. An example of a service provider is a library that requires proof of address through a utility bill.

These participants will each have a Bitcoin address: $\mathtt{a}_{\mathcal{USR}}$, $\mathtt{a}_{\mathcal{IP}}$, $\mathtt{a}_{\mathcal{SP}}$. The addresses of $\mathcal{IP}$, $\mathcal{SP}$, should be well-known, while $\mathcal{USR}$ may wish to have several addresses $\mathtt{a}^{(i)}_{\mathcal{USR}}$ which should not be linked to addresses the user has in Bitcoin for other purposes, for instance moving bitcoins around, in order to avoid de-anonymization of these addresses. The user's addresses may be used with different identity providers.

Transaction structure. We now recall the scheme of (Augot et al., 2017a). First, $\mathcal{USR}$ presents real-life proofs of her identity to $\mathcal{IP}$, and a Bitcoin address $\mathtt{a}^{(i)}_{\mathcal{USR}}$ that she controls. $\mathcal{IP}$ then constructs a Brands commitment $h_{\mathtt{a}^{(i)}_{\mathcal{USR}}}$ that blindly encodes the user's identity information (see (Augot et al., 2017a) for details). Then, $\mathcal{IP}$ publishes an identity record containing this information via a Bitcoin transaction $\mathtt{TX}_{\mathtt{PUBLISH}}$ of the following form:

> $\underline{\mathtt{TX}_{\mathtt{PUBLISH}}}$ (Fees: $\mathtt{F}_{\mathtt{PUBLISH}}$)
> Input Address:
> $\quad\mathtt{a}_{\mathcal{IP}}$ for an Amount: V+$\mathcal{D}$+$\mathtt{F}_{\mathtt{PUBLISH}}$
> Output Addresses:
> $\quad\mathtt{a}^{(i)}_{\mathcal{USR}}$ for an Amount: $\mathcal{D}$
> $\quad\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}},\mathtt{a}_{\mathcal{IP}})$ for an Amount: V
> $\quad\mathtt{OP\_RETURN}\left(h_{\mathtt{a}^{(i)}_{\mathcal{USR}}}\right)$

This transaction uses an input that belongs to the address $\mathtt{a}_{\mathcal{IP}}$. Hence, it is signed by $\mathcal{IP}$'s private key, and as $\mathtt{a}_{\mathcal{IP}}$ is well-known, anyone in the system can verify that this transaction is from $\mathcal{IP}$. The transaction output $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}},\mathtt{a}_{\mathcal{IP}})$ serves doubly as an *authentication token* and a *revocation token*. As this output is a 1 of 2 multisignature, either $\mathcal{USR}$ or $\mathcal{IP}$ can spend it. $\mathcal{USR}$ will spend this output to use her identity (see below), whereas $\mathcal{IP}$ will spend it to revoke $\mathcal{USR}$'s identity. There is also an output to the address of $\mathtt{a}^{(i)}_{\mathcal{USR}}$, with only a symbolic "dust" amount $\mathcal{D}$ (See Sec. 2), to link this record to $\mathtt{a}^{(i)}_{\mathcal{USR}}$. Finally

the $\texttt{OP\_RETURN}(h_{\mathtt{a}^{(i)}_{\mathcal{USR}}})$ contains the Brands commitment.

Then when $\mathcal{USR}$ wants to use her identity record to convince an $\mathcal{SP}$ of an aspect of her identity (for example to prove that she is over 18 to a bar, or to prove that she has a valid driver's license to a traffic policeman, without revealing here name), she creates a Brands proof of the statement $\mathcal{SP}$ requires, and communicates it to $\mathcal{SP}$. Then she issues a Bitcoin transaction of the form $\texttt{TX}_{\texttt{REQUEST}}$ below. $\mathcal{USR}$ may also wish to have the proof archived. To archive in Bitcoin's blockchain, the $\texttt{OP\_RETURN}$ field is generally not large enough to contain the proof, so if there is a need to attest to it on-chain, $\mathcal{USR}$ can provide a link to an external storage service (cloud or P2P network) where the proof is stored and a hash of the proof. We refer to such a reference as "proof-ref" in the following $\texttt{TX}_{\texttt{REQUEST}}$.

> $\underline{\texttt{TX}_{\texttt{REQUEST}}}$ (Fees: $\texttt{F}_{\texttt{REQUEST}}$)
> Input Address:
> $\texttt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ for an Amount: V
> Output Addresses:
> $\mathtt{a}_{\mathcal{SP}}$ for an Amount: $\texttt{F}_{\texttt{ACCEPT}} + \mathcal{D}$
> $\texttt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ for an Amount: $V - (\texttt{F}_{\texttt{REQUEST}} + \texttt{F}_{\texttt{ACCEPT}} + \mathcal{D})$
> $\texttt{OP\_RETURN}$("proof-ref") OR no $\texttt{OP\_RETURN}$

Finally, when convinced by the proof, $\mathcal{SP}$ grants access to service, and in order to enable a user to build a reputation, $\mathcal{SP}$ sends a transaction of the form $\texttt{TX}_{\texttt{ACCEPT}}$ to $\mathcal{IP}$ acknowledging that the user's identity from $\mathcal{IP}$ has been used.

> $\underline{\texttt{TX}_{\texttt{ACCEPT}}}$ (Fees: $\texttt{F}_{\texttt{ACCEPT}}$)
> Input Address: $\mathtt{a}_{\mathcal{SP}}$ for an Amount: $\texttt{F}_{\texttt{ACCEPT}} + \mathcal{D}$
> Output Address: $\mathtt{a}_{\mathcal{IP}}$ for an Amount: $\mathcal{D}$

Note that in $\texttt{TX}_{\texttt{REQUEST}}$, the input is the authentication token, and one of the outputs is again of the same form $\texttt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$. Thus, this output can be used as the input for a subsequent $\texttt{TX}_{\texttt{REQUEST}}$. The trace of this sequence of linked outputs of the same form ties all of the user's authentication request together, and $\mathcal{SP}$ can trace back a received $\texttt{TX}_{\texttt{REQUEST}}$ all the way to the $\texttt{TX}_{\texttt{PUBLISH}}$ to find $h_{\mathtt{a}^{(i)}_{\mathcal{USR}}}$.

Whenever $\mathcal{IP}$ wants to revoke the user's identity, he can trace forward the linked list of spent outputs to the most recent $\texttt{TX}_{\texttt{REQUEST}}$ and spend the output $\texttt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ to himself. An advantage of using the same output to authenticate and to revoke (in contrast to (Nazaré et al., 2016) where revocation is controlled by its own output) is that seeing that $\texttt{TX}_{\texttt{REQUEST}}$ has been included by miners in a recent block can be sufficient to convince $\mathcal{SP}$ that that output has not been previously spent and hence that a document has not been revoked (up to that block).

The amounts attached to the various outputs in these transactions are calibrated so that sufficient fees are paid and outputs all surpass the dust limit. Otherwise, the rest of the value is passed on with the authentication token, and the value is calibrated for a number of uses of the token. Alas, (Augot et al., 2017a) estimates that the cost of establishing an identity for N uses in this manner to be approximately $(5.2N + 2.6)$ USD, based on Bitcoin fees and prices as of September 2017. This high cost is essentially due to the high cost of Bitcoin transaction fees and has in fact gotten worse with the rising price of Bitcoin. At peak (December 2017) estimates of cost for this system are near $(38.73N + 19.71)$ USD.

4 CHAINED IDENTITIES ACROSS ATOMIC SWAPS

A first idea to modify the scheme of (Augot et al., 2017a) to reduce costs is to take advantage of lower transaction costs on the blockchain of some other cryptocurrency, and to "migrate" the authentication token from Bitcoin to the blockchain of the other currency. There are several other cryptocurrencies whose transactional structures have the necessary elements and support scripts similar to Bitcoin (recalled in Sec. 2). For example, this is the case of Litecoin https://www.coindesk.com/information/comparing-litecoin-bitcoin/, https://www.cryptopia.co.nz/Forum/Thread/981 and Bitcoin Cash https://www.bitcoin.com/info/differences-between-bitcoin-cash-bcc-and-bitcoin-btc. These currencies have substantially lower fees https://bitinfocharts.com/; at the time of writing, the maximum average fees of Bitcoin (resp. Litecoin, resp. Bitcoin Cash) was reached in December 2017 at $55.16 USD (resp. $1.505, $.904). However, the total hash power of these blockchains also varies greatly: Bitcoin has a hash rate of 13.4 EH/s, Litecoin 90 TH/s, and Bitcoin Cash 950 PH/s. Thus, in light of the dependence of the security of our identity scheme on the resistance of the blockchain to 51% attacks, applications with higher security requirements may want to remain on Bitcoin despite the high fees, while lower security requirements can use the token migrated to Litecoin.

The identity authentication token can be passed from one blockchain to another, using an *atomic swap* with a *chain switcher service* $\mathcal{CS}$. For simplicity, we exemplify the concept with Bitcoin (BTC) and Litecoin (LTC). In the following, the identity provider $\mathcal{IP}$

has a well-known address $\mathtt{a}_{I\mathcal{P}}$ on the BTC blockchain and $\mathtt{a}'_{I\mathcal{P}}$ on the LTC blockchain (the same holds for $\mathcal{CS}$, and for some $\mathcal{USR}$'s addresses $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}^{'(j)}_{\mathcal{USR}}$). In particular, this gives $I\mathcal{P}$ the power to limit on which chains issued identities can migrate, since the protocol needs $I\mathcal{P}$ to also have an address on the second chain.

In the following, we let R be a random value and $h = H(R)$ for a given hash function $H(.)$.

> $\mathtt{TX}_{\mathtt{BTC\text{-}SWAP}}$ (Fees: $\mathtt{F}_{\mathtt{BTC\text{-}SWAP}}$ in BTC)
> Input Address: $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{I\mathcal{P}})$ for an Amount: V
> Output Addresses:
> $\mathtt{a}_{\mathcal{CS}}$ for an Amount: $\mathcal{D}$
> $P2SH$(needs(R) and signed by $\mathtt{a}_{\mathcal{CS}}$, OR signed by $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{CS}}$, OR signed by $\mathtt{a}_{I\mathcal{P}}$) for an Amount: $\mathrm{V}-\mathcal{D}-\mathtt{F}_{\mathtt{BTC\text{-}SWAP}}$ (in BTC)
> $\mathtt{OP\_RETURN}(h$, txid of $\mathtt{TX}_{\mathtt{LTC\text{-}SWAP}})$

> $\mathtt{TX}_{\mathtt{LTC\text{-}SWAP}}$ (Fees: $\mathtt{F}_{\mathtt{LTC\text{-}SWAP}}$ in LTC)
> Input Address: $\mathtt{a}'_{\mathcal{CS}}$ for an Amount: W
> Output Addresses:
> $P2SH$(needs(R) and signed by $\mathtt{a}^{'(j)}_{\mathcal{USR}}$, OR signed by $\mathtt{a}^{'(j)}_{\mathcal{USR}}$ and $\mathtt{a}'_{\mathcal{CS}}$) for an Amount: $\mathrm{W}-\mathtt{F}_{\mathtt{LTC\text{-}SWAP}}$ (in LTC)
> $\mathtt{OP\_RETURN}$(txid of last $\mathtt{TX}_{\mathtt{REQUEST}}$ issued on BTC)

Before signing their respective swap transactions, $\mathcal{USR}$ and $\mathcal{CS}$ have each other signed backout transactions of the form:

> $\mathtt{TX}_{\mathtt{BTC\text{-}BACKOUT}}$ (Timelock: t, Fees: $\mathtt{F}_{\mathtt{BTC\text{-}BACKOUT}}$ in BTC)
> Input Address:
> $P2SH$(needs(R) and signed by $\mathtt{a}_{\mathcal{CS}}$, OR signed by $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{CS}}$ OR signed by $\mathtt{a}_{I\mathcal{P}}$) for an Amount: $\mathrm{V}-\mathcal{D}-\mathtt{F}_{\mathtt{BTC\text{-}SWAP}}$
> Output Address: $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{I\mathcal{P}})$ for an Amount: $\mathrm{V}-\mathcal{D}-\mathtt{F}_{\mathtt{BTC\text{-}SWAP}}-\mathtt{F}_{\mathtt{BTC\text{-}BACKOUT}}$ (in BTC)

> $\mathtt{TX}_{\mathtt{LTC\text{-}BACKOUT}}$ (Timelock: t', Fees: $\mathtt{F}_{\mathtt{LTC\text{-}BACKOUT}}$ in LTC)
> Input Address:
> $P2SH$(needs(R) and signed by $\mathtt{a}^{'(j)}_{\mathcal{USR}}$, OR signed by $\mathtt{a}^{'(j)}_{\mathcal{USR}}$ and $\mathtt{a}'_{\mathcal{CS}}$) for an Amount: $\mathrm{W}-\mathtt{F}_{\mathtt{LTC\text{-}SWAP}}$
> Output Address: $\mathtt{a}'_{\mathcal{CS}}$ for an Amount: $W-\mathtt{F}_{\mathtt{LTC\text{-}SWAP}}-\mathtt{F}_{\mathtt{LTC\text{-}BACKOUT}}$ (in LTC)

Here, as the LTC swap transaction must be created first, $\mathcal{CS}$ should choose R, and communicate $h = H(R)$ to $\mathcal{USR}$. In particular, the timelock t' should be longer than the timelock t. Then, once R has been made public and the two swap transactions are accepted to their respective blockchains, to establish a usable authentication token as in Section 3, $\mathcal{USR}$ issues herself a transaction of the form:

> $\mathtt{TX}_{\mathtt{CREATE\text{-}LTC\text{-}TOKEN}}$ (Fees: $\mathtt{F}_{\mathtt{CREAT\text{-}\ LTC\text{-}TOKEN}}$)
> Input Address:
> $P2SH$(needs(R) and signed by $\mathtt{a}'_{\mathcal{CS}}$, OR signed by $\mathtt{a}^{'(j)}_{\mathcal{USR}}$ and $\mathtt{a}'_{\mathcal{CS}}$) for an Amount: $\mathrm{W}-\mathtt{F}_{\mathtt{LTC\text{-}SWAP}}$
> Output Address: $\mathtt{MSIG1\_2}(\mathtt{a}^{'(j)}_{\mathcal{USR}}, \mathtt{a}'_{I\mathcal{P}})$ for an Amount: $\mathrm{W}-\mathtt{F}_{\mathtt{LTC\text{-}SWAP}}-\mathtt{F}_{\mathtt{CREAT\text{-}\ LTC\text{-}TOKEN}}$

$\mathcal{USR}$ can now continue to use her identity on the LTC blockchain exactly as in (Augot et al., 2017a) to authenticate to service providers. It is essential that the path of the authentication token from one blockchain to the other is not broken, to trace forward (for revocation) and backward (for usage) the authentication token. To preserve the link, the txid of the $\mathtt{TX}_{\mathtt{LTC\text{-}SWAP}}$ transaction is included in the $\mathtt{OP\_RETURN}$ of the $\mathtt{TX}_{\mathtt{BTC\text{-}SWAP}}$ transaction. As such, a verifier following $\mathcal{USR}$'s transaction history can trace the token in either direction through the swap. A service provider who receives a $\mathtt{TX}_{\mathtt{REQUEST}}$ on chain LTC can find the last transaction of this identity on chain BTC, and the corresponding $\mathtt{TX}_{\mathtt{PUBLISH}}$ on BTC. Similarly, $I\mathcal{P}$ can revoke the identity, by tracing the authentication token to the new chain and spend the $\mathtt{MSIG1\_2}(\mathtt{a}^{'(j)}_{\mathcal{USR}}, \mathtt{a}'_{I\mathcal{P}})$. Again, as in (Augot et al., 2017a), albeit on a different blockchain, an $\mathcal{SP}$ that wants to verify that an identity has not been revoked again only needs to run an SPV client on the LTC chain, see Section 3.

Recall that, in a P2SH output, only the hash is placed in the output and the script is not revealed until the output is spent. Thus, in order for $I\mathcal{P}$ to be able to revoke $\mathcal{USR}$'s identity after $\mathtt{TX}_{\mathtt{BTC\text{-}SWAP}}$ and before the $\mathtt{TX}_{\mathtt{CREATE\text{-}LTC\text{-}TOKEN}}$ transaction is issued, $I\mathcal{P}$ needs to be able to present the script for the P2SH output for $\mathtt{TX}_{\mathtt{BTC\text{-}SWAP}}$. Ideally, it would be enough to just include the script in the $\mathtt{OP\_RETURN}$ of this transaction so that it would be available to $I\mathcal{P}$. However, this script must include: a) $h = H(R)$, b) the three 32-byte public keys of $\mathcal{USR}$, $I\mathcal{P}$, and $\mathcal{CS}$, c) the txid of the transaction on the other chain. This is too large for the 80 bytes limit of the $\mathtt{OP\_RETURN}$. So, as a workaround, as $I\mathcal{P}$ already knows $\mathtt{a}_{I\mathcal{P}}$ and $\mathtt{a}^{(i)}_{\mathcal{USR}}$, we propose to include only the part of the script that $I\mathcal{P}$ does not know, namely the necessary txid and h should be placed in the $\mathtt{OP\_RETURN}$. Meanwhile, an output with a minimal amount of $\mathcal{D}$ is paid to $\mathtt{a}_{\mathcal{CS}}$ so that $I\mathcal{P}$ has this address

as well.

We have proposed that $\mathcal{CS}$ serves the role of providing coins on the new chain in exchange for the old. However, a user does not need to depend on any specific $\mathcal{CS}$ for this role. Indeed, $\mathcal{IP}$ could also perform this role, or in a more decentralized and trustless way, the user herself can play this role, assuming she already possesses some coins on the new chain in an address $\mathtt{a}'^{(j)}_{\mathcal{USR}}$, that she is willing to publicly tie to her identity address $\mathtt{a}^{(i)}_{\mathcal{USR}}$. Due to the symmetry of this protocol, the user can go back to the chain on which her identity was originally issued at some later time via the same transfer mechanism.

We leverage on the security of (Augot et al., 2017a) for this proposal. Namely, the system depends on properties of an ideal blockchain: consistency of the chain of transactions inputs and outputs, and the global availability of the data, as BTC and LTC in our example, and immutability of the associated ledgers.

5 AUTHENTICATION TOKENS USED IN OFF-CHAIN CHANNELS

As we saw in Section 3, the central problem in the protocol of (Augot et al., 2017a) is the bandwidth it required in the Bitcoin network and the associated transaction costs. So we are inspired by the Lightning Network (Joseph Poon and Thaddeus Dryja, 2016) and its proposals to allow secure, off-chain transactions in the Bitcoin network.

A new actor is introduced, the *service enabler*, $\mathcal{SE}$, whose role is to play as an intermediary between $\mathcal{USR}$ and $\mathcal{SP}$. $\mathcal{USR}$ will only need to open a channel with $\mathcal{SE}$, without needing to open many channels with different $\mathcal{SP}$. In our mind, these service enablers, $\mathcal{SE}$, are seen as well-established, trusted companies that have a reputation to preserve and might periodically be subject to audits. Our proposal relies on their integrity to resist certain attacks that involve collusion with $\mathcal{USR}$ (see below). Nonetheless, $\mathcal{USR}$ is not required to place an absolute trust in $\mathcal{SE}$ as we envision several competing service enablers, and $\mathcal{USR}$ will be able to shift an identity document from one to another.

We will see in this section that using such off-chain transactions in an identity management scheme is possible, preserving some (but not all) of the advantages and protections provided by the Bitcoin network in (Augot et al., 2017a). For instance, as $\mathcal{USR}$ can use different $\mathcal{SE}$ while keeping her identity, some kind of decentralization is preserved.

$\mathcal{USR}$ will dedicate a "part" of her identity to off-chain authentication by opening a payment channel with $\mathcal{SE}$. Note that as this channel will serve for authentication tokens and not real life payments, all "payments" will be unidirectional, and there will never be a need for $\mathcal{USR}$ to receive payments back through this channel.

When the channel is opened, the authentication token (which again is also used for revocation by $\mathcal{IP}$) normally of the form $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ becomes

$P2SH$(signed by 2 of 2 of $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{SE}}$ OR by $\mathtt{a}_{\mathcal{IP}}$).

As such, this output can be spent by both $\mathcal{USR}$ and $\mathcal{SE}$ together, but it can also be spent at any time by $\mathcal{IP}$ revoking the identity and rendering what has passed in the channel moot.

We now detail the sequence of transactions to open, use, and close such channel while preserving identity information. We begin with the last $\mathtt{TX}_{\mathtt{REQUEST}}$ that the user issued ((Augot et al., 2017a), see Section 3).

> $\underline{\mathtt{TX}_{\mathtt{REQUEST}}}$ (Fees: $\mathtt{F}_{\mathtt{REQUEST}}$)
> Input Address: $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ for an Amount: V
> Output Addresses:
> $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}$
> $\mathtt{OP\_RETURN}$("proof-ref")

The user takes the authentication token output from this address and uses it as the input to following transaction $\mathtt{TX}_{\mathtt{OPEN}}$, which $\mathcal{USR}$ does not yet publish.

> $\underline{\mathtt{TX}_{\mathtt{OPEN}}}$ (Fees: $\mathtt{F}_{\mathtt{OPEN}}$)
> Input Address: $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}$
> Output Addresses:
> $P2SH$(signed by 2 of 2 of $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{SE}}$ OR by $\mathtt{a}_{\mathcal{IP}}$) for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}$
> $\mathtt{OP\_RETURN}$($\mathtt{a}_{\mathcal{SE}}$ formatting, other info)

Before publishing $\mathtt{TX}_{\mathtt{OPEN}}$, $\mathcal{USR}$ creates a backout transaction that spends the multisig output of $\mathtt{TX}_{\mathtt{OPEN}}$ back to herself (with a timelock), and has $\mathcal{SE}$ sign this transaction. This way, if $\mathcal{SE}$ does not close the channel, $\mathcal{USR}$ can close it and recover her input, by signing and sending $\mathtt{TX}_{\mathtt{BACKOUT}}$ to the blockchain.

> $\underline{\mathtt{TX}_{\mathtt{BACKOUT}}}$ (Timelock, Fees: $\mathtt{F}_{\mathtt{BACKOUT}}$)
> Input Address: $P2SH$(signed by 2 of 2 of $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{SE}}$ OR by $\mathtt{a}_{\mathcal{IP}}$) for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}$
> Output Address: $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}}, \mathtt{a}_{\mathcal{IP}})$ for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}-\mathtt{F}_{\mathtt{BACKOUT}}$

Thus, a payment channel from $\mathcal{USR}$ to $\mathcal{SE}$ has been opened. When using her identity to obtain a service from $\mathcal{SP}$:

1. $\mathcal{USR}$ communicates to $\mathcal{SE}$ the identity of $\mathcal{SP}$
2. $\mathcal{SE}$ asks $\mathcal{SP}$ what $\mathcal{USR}$ needs to prove about her identity to use the service
3. $\mathcal{USR}$ makes a Brands proof interacting with $\mathcal{SE}$ against the $h_{\mathtt{a}^{(i)}_{\mathcal{USR}}}$ provided in $\mathtt{TX}_{\mathtt{PUBLISH}}$ (see Section 3) to prove the required property
4. $\mathcal{USR}$ issues to $\mathcal{SE}$ a transaction of the following form:

> $\underline{\mathtt{TX}_{\mathtt{USE\text{-}CHANNEL}}}$ (Fees: $\mathtt{F}_{\mathtt{FIRST\text{-}USE}}$)
> Input Address: *P2SH*(signed by 2 of 2 of $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{SE}}$ OR by $\mathtt{a}_{\mathcal{IP}}$) for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}$
> Output Addresses:
> $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}},\mathtt{a}_{\mathcal{IP}})$ for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}-\mathtt{F}_{\mathtt{FIRST\text{-}USE}}$
> $\mathtt{OP\_RETURN}(k)$ with $k=1$

5. $\mathcal{SE}$ confirms to $\mathcal{SP}$ that $\mathcal{USR}$'s identity has the required property
6. Each time $\mathcal{USR}$ uses this channel to authenticate to maybe another $\mathcal{SP}$, a counter k is incremented. At the k-th use of the channel, a transaction is issued of the following form using the same UTXO from the above $\mathtt{TX}_{\mathtt{OPEN}}$:

> $\underline{\mathtt{TX}_{\mathtt{USE\text{-}CHANNEL}}}$ (Fees: $\mathtt{F}_{\mathtt{kth\text{-}USE}}$)
> Input Address:
> *P2SH*(signed by 2 of 2 of $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{SE}}$ OR by $\mathtt{a}_{\mathcal{IP}}$) for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}$
> Output Address:
> $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}},\mathtt{a}_{\mathcal{IP}})$
> for an Amount: $\mathrm{V}-\mathtt{F}_{\mathtt{REQUEST}}-\mathtt{F}_{\mathtt{OPEN}}-\mathtt{F}_{\mathtt{kth\text{-}USE}}$
> $\mathtt{OP\_RETURN}(k)$

Whenever $\mathcal{SE}$ wants to close the channel, he signs and publishes $\mathtt{TX}_{\mathtt{OPEN}}$ and $\mathtt{TX}_{\mathtt{USE\text{-}CHANNEL}}$, with the highest index k. $\mathcal{SE}$ can then spend the last P2SH output. $\mathcal{USR}$ is then left with an authentication token (marked in red in the above diagrams) that she can use for future authentications which will be chained to her usage history. In addition, if $\mathcal{SE}$ becomes hostile, and does not close the channel, $\mathcal{USR}$ can recover her funds using the $\mathtt{TX}_{\mathtt{BACKOUT}}$ transaction when the timelock is reached.

Any observer, such as a service provider, can check the highest index k when the channel is closed, and learn how many times $\mathcal{USR}$ used her identity during the off-chain portion of the history. Furthermore, privacy is gained, since these observers cannot see on-chain to which service providers $\mathcal{USR}$ authenticated. This proposal thus requires a heavy dependence on $\mathcal{SE}$ to confirm to $\mathcal{SP}$ that the protocol has been correctly executed, and to properly close the channel with the highest k. This places this protocol in the style of (Augot et al., 2017b), where a service enabler plays a crucial role. But here the dependency is lighter, since $\mathcal{SE}$ plays its role only in this off-chain mode: $\mathcal{USR}$ also has the possibility to close the channel and use her identity purely on-chain, without depending anymore on $\mathcal{SE}$.

We have similar issues as in Section 4 concerning the requirement that $\mathcal{IP}$ must have the script in the P2SH output in order to be able to revoke an identity after $\mathtt{TX}_{\mathtt{OPEN}}$ and before the channel is closed. Again, with three 32-byte public keys the script is too big for the 80 bytes limit of an $\mathtt{OP\_RETURN}$. Thus, we again include in the $\mathtt{OP\_RETURN}$ $\mathtt{a}_{\mathcal{SE}}$ and the format of script with the other public keys removed as $\mathcal{IP}$ already knows $\mathtt{a}_{\mathcal{IP}}$ and $\mathtt{a}^{(i)}_{\mathcal{USR}}$.

While the channel is open, it is the responsibility of $\mathcal{SE}$ to check that a user's identity has not been revoked. In order to do this, $\mathcal{SE}$ must be capable of determining that the P2SH transaction output in $\mathtt{TX}_{\mathtt{OPEN}}$ has not yet been spent. Thus, $\mathcal{SE}$ must be a full node. This is consistent with the fact that, in the Lightning Network in general the intermediaries need to be full nodes (Gulbrandsen, 2016) to check for broadcasting of obsolete transactions. Moreover, after the channel is closed, either by $\mathcal{SE}$ using with $\mathtt{TX}_{\mathtt{OPEN}}$ and $\mathtt{TX}_{\mathtt{USE\text{-}CHANNEL}}$, or $\mathcal{USR}$ using $\mathtt{TX}_{\mathtt{BACKOUT}}$, the authentication token, which was on the channel

P2SH(signed by 2 of 2 of $\mathtt{a}^{(i)}_{\mathcal{USR}}$ and $\mathtt{a}_{\mathcal{SE}}$ OR by $\mathtt{a}_{\mathcal{IP}}$)

becomes again $\mathtt{MSIG1\_2}(\mathtt{a}^{(i)}_{\mathcal{USR}},\mathtt{a}_{\mathcal{IP}})$ and, back on the blockchain, can be used either by $\mathcal{USR}$ who can resume using her identity, or $\mathcal{IP}$ who can revoke it.

Trust is put into $\mathcal{SE}$, in particular to properly perform the verification of whether an identity is revoked or not, and properly close channel with the highest index k. (Note $\mathcal{USR}$ will refuse to sign a $\mathtt{TX}_{\mathtt{USE\text{-}CHANNEL}}$ with a higher index that what is justified, so $\mathcal{SE}$ cannot cheat $\mathcal{USR}$ out of uses.) Furthermore, a hostile $\mathcal{SE}$ may collude with $\mathcal{USR}$, and not publish the $\mathtt{TX}_{\mathtt{OPEN}}$ and $\mathtt{TX}_{\mathtt{kth\text{-}USE}}$ transactions $\mathtt{TX}_{\mathtt{CLOSE}}$: $\mathcal{USR}$ can then claim the reimbursement transaction from the $\mathtt{TX}_{\mathtt{BACKOUT}}$ after the timelock. This would allow a user to continue using her identity as if the transactions that took place off-chain had never happened. In the Lightning Network, this would cost $\mathcal{SE}$ the money that had been transacted on the channel; here, as the amounts are mostly symbolic, this attack has a low

cost. This is similar to (Augot et al., 2017a) where exists the risk that $\mathcal{SP}$ might accept a user identity without demanding a $\texttt{TX}_{\texttt{REQUEST}}$. Thus, the trust model service providers have towards a group of $\mathcal{SE}$'s is similar to how web browsers trust certificate authorities in PKIs; the system is only as secure (against $\mathcal{SE}$ - $\mathcal{USR}$ collusion) as the least trustworthy $\mathcal{SE}$.

6 CONCLUSION

Taking back existing mechanisms to our advantage – atomic swaps and off-chain payments – we have presented two methods to reduce the costs of the proposal of (Augot et al., 2017a) to the point of rendering it financially viable while nonetheless preserving its advantages in terms of flexible user experience and identity provider controls, particularly for revocation, which can only be violated by an attacker capable of committing Bitcoin double spending attacks. We believe that our ideas can be reused in other contexts.

REFERENCES

Abiteboul, S., André, B., and Kaplan, D. (2015). Managing your digital life. *Commun. ACM*, 58(5):32–35.

Antonopoulos, A. M. (2015). *Mastering Bitcoin*. O'Reilly Media, Sebastopol, California.

Augot, D., Chabanne, H., Chenevier, T., George, W., and Lambert, L. (2017a). A user-centric system for verified identities on the bitcoin blockchain. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshop, CBT 2017*, pages 390–407.

Augot, D., Chabanne, H., Clémot, O., and George, W. (2017b). Transforming face-to-face identity proofing into anonymous digital identity using the Bitcoin blockchain. In *PST2017 - International Conference on Privacy, Security and Trust*. See also arxiv.org/abs/1710.02951.

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

Dunphy, P. and Petitcolas, F. A. P. (2018). A first look at identity management schemes on the blockchain. *CoRR*, abs/1801.03294.

Garman, C., Green, M., and Miers, I. (2016). Accountable privacy for decentralized anonymous payments. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016*, pages 81–98.

Gulbrandsen, A. (2016). Bitcoin Lightning Network FAQ. Online, https://medium.com/@AudunGulbrands1/lightning-faq-67bd2b957d70.

Jacobovitz, O. (2016). Blockchain for identity management. Technical Report 16-02, Lynne and William Frankel Center for Computer Science (Ben Gurion University). https://www.cs.bgu.ac.il/%7Efrankel/TechnicalReports/2016/16-02.pdf.

Joseph Poon and Thaddeus Dryja (2016). The Bitcoin Lightning Network: Scalable off-chain instant payments. Online, https://lightning.network/lightning-network-paper.pdf.

Lundkvist, C., Heck, R., Torstensson, J., Mitton, Z., and Sena, M. (2017). uPort: A platform for self-sovereign identity. https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Online, http://bitcoin.org/bitcoin.pdf.

Nazaré, J., Hamilton, K., and Schmidt, P. (2016). Digital certificates project. online, source code available at https://github.com/digital-certificates. http://certificates.media.mit.edu.

Prisco, G. (2015). Estonian government partnerts with bitnation to offer blockchain notarization services to e-residents. https://bitcoinmagazine.com/articles/estonian-government-partners-with-bitnation-to-offer-blockchain-notarization-services-to-e-residents-1448915243/.

Tobin, A., Reed, D., and Windley, P. J. (2016). The inevitable rise of self-sovereign identity. Online, https://sovrin.org/wp-content/uploads/2017/07/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf.

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