Fully Anonymous Transferable Ecash

Hitesh Tewari and Arthur Hughes

School of Computer Science and Statistics
Trinity College Dublin
Ireland
{hitesh.tewari,arthur.hughes}@scss.tcd.ie
http://www.scss.tcd.ie

Abstract. Numerous electronic cash schemes have been proposed over the years ranging from Ecash, Mondex to Millicent. However none of these schemes have been adopted by the financial institutions as an alternative to traditional paper based currency. The Ecash scheme was the closest to a system that mimicked fiat currency with the property that it provided anonymity for users when buying coins from the Bank and spending them at a merchant premises (from the Bank's perspective). However Ecash lacked one crucial element present in current fiat based systems i.e., the ability to continuously spend or transfer coins within the system. In this paper we propose an extension to the Ecash scheme which allows for the anonymous transfer of coins between users without the involvement of a trusted third party. We make use of a powerful technique which allows for distributed decision making within the network - namely the Bitcoin blockchain protocol. Combined with the proof-of-work technique and the classical discrete logarithm problem we are able to continuously reuse coins within our system, and also prevent double-spending of coins without revealing the identities of the users.

Keywords: Anonymous Ecash, Transferable Coins, Delegated Signatures, Blockchain, Proof-of-Work

1 Introduction

In today's increasingly digital world we are surrounded by electronic payment systems in everyday life. People regularly use debit and credit cards to make payments at point-of-sale (PoS) terminals, and on the Internet with e-commerce retailers. More and more people are also starting to make use of electronic funds transfer (EFT) mechanisms, as Banks try to reduce their costs, and eliminate the use of cheques. Near field communications (NFC) systems combined with mobile technology promises to bring more convenient and quicker ways for users to pay and conduct peer-to-peer (P2P) funds transfer. However, for all the technological advances over the past three decades *cash is still king*. According to a recent report by the San Francisco Federal Reserve Bank two-thirds of all transactions below $10, and nearly half of all transactions below $25 are in cash [1].

There are a myriad of reasons for this - including the ease of use of cash transactions, lower transaction fees for merchants, lack of access to credit for

individuals etc. However, one of the main reasons still for the prevalence of cash in everyday life is the fact that cash transactions are *anonymous*. In the majority of cases when a person spends cash with a merchant at a PoS, there is no requirement to share any personal information with them. Also, the merchants and Banks cannot identify the payee through the use of the serial number on the Bank notes even if they collude. On the other hand, with the proliferation of payment cards, consumers are willingly handing over huge amounts of personal details to retailers, who in turn sell this information onto marketing agencies. The downside of the *cash only* approach is that it is not possible to pay for goods and services at e-commerce retailers - a segment of the market that is growing rapidly and popular with young technology savvy consumers.

It is therefore quite obvious that what is required is an electronic version of cash that is *fully anonymous*. Just such a scheme (Ecash) was proposed by David Chaum [2] more than thirty years ago. However it failed to take off as a real alternative to physical cash. The first possible reason for its failure was that Ecash lacked an important property that most users associate with fiat currency i.e. *transferability of coins*. Within the Ecash scheme it was possible to anonymously withdraw coins from a Bank, and at a later date spend them at a merchant premises. In order for the coins to be accepted by the merchant they had to be deposited with the Bank for verification in the Ecash scheme. However, once deposited the merchant was not able to reuse the coins i.e. transfer them to another user as change or reuse them to buy more goods from a wholesaler.

Over the years, a number of other electronic cash schemes have been proposed to address the transferability issue [3]. These schemes usually embed an obfuscated version of the identity of the current owner into the coin history. As long as all the users abide by the rules of the system their identities remain anonymous. However if a user tries to double spend a coin then their identity (and possibly the identity of other users) will be revealed to the Bank [4]. We believe that the inability to *prevent double-spending* of coins in these schemes is the second possible reason which excluded them from being widely adopted for financial transactions.

A recent paper on anonymous transferable electronic cash makes use of a construct called *malleable signatures* [5], where given a signature, one can derive another signature for an allowed transformation - thereby delegating the signing authority to a third party. This scheme allows for anonymous transfer of e-cash without the involvement of any trusted third parties. However this scheme only proposes a double-spending detection protocol and does not prevent the double-spending of coins.

In this paper, we introduce a *fully anonymous* and *transferable* electronic cash scheme. Our proposed system uses the *blind signature protocol* [6] to anonymize the identities of all users in the system, and allows for anonymous transfer of coins. In addition, we make use of the Bitcoin inspired *blockchain* [7] and Hashcash inspired *proof-of-work* [8] algorithms to collectively verify the authenticity of transactions, and prevent double-spending in the network. Our contribution to the state-of-the-art is a novel delegated signature scheme in conjunction with

the discrete logarithm problem (DLP) [9] and distributed verification, to allow for the secure anonymous transfer coins from one user to another without the need to contact a trusted third party (TTP). In the rest of this paper, we provide an overview of some of the technologies that we use as building blocks for our protocol, followed by detailed description of our system, and a brief security analysis.

2 Related Work

In the following sections we provide the reader with an overview of some of the existing techniques that we are going to use within our proposed system, such as the blind signature protocol, and proof-of-work concepts. We assume that the reader is familiar with the discrete logarithm problem. A detailed description of discrete logs can be found in [10].

2.1 Blind Signature Protocol

The first electronic cash protocol which was truly anonymous was proposed by David Chaum [6][11]. The scheme is essentially an on-line software solution whereby a buyer can spend Ecash coins with any merchant participating in the system. By examining the coins (alone) neither the issuer or the merchant are able to determine the identity of the customer. The protocol is designed such that the issuer is not able to detect the serial numbers of coins that it issued to users of the system (at the time of issue), even if it colludes with other participants in the system.

The scheme uses the *blind signature protocol* which allows a user (Alice) to mint a coin and hide the serial number of the coin using a blinding factor. Alice forwards the unsigned blinded coin to the Ecash Bank. As long as the coin satisfies certain criteria, the Bank signs the coin with its private coin signing key, without knowledge of the serial number. This feature allows for truly *anonymous cash*. On receiving the signed coin back from the Bank, Alice removes the blinding factor, and uses the coin to pay for goods at a merchant (Bob) participating in the system. The blinding factor is a random number used to obfuscate the serial number of the electronic coin from the Bank. On receipt of the coin, Bob immediately forwards it to the Ecash Bank for verification. The Bank maintains an *ever-growing* database of the serial numbers of all coins that have been spent in the system and is thus able to detect double-spending.

Mathematically the blind signature scheme is comprised of the following:

- A set $\mathcal{M}$ which is the encrypted and non-encrypted message data and two functions $PK, SK : \mathcal{M} \to \mathcal{M}$ which are an asymmetric encryption/decryption pair:
  - $SK(PK(m)) = m$ and $PK(SK(m)) = m$ for all m in $\mathcal{M}$;
  - $PK(m)$ is relatively easy to compute for all m in $\mathcal{M}$;
  - PK's effect on an m in $\mathcal{M}$ is extremely difficult to undo (invert) without knowing SK, i.e. if we are given an e in $\mathcal{M}$ which is the image of some m in $\mathcal{M}$ under PK ($e = PK(m)$) and where this m is unknown then it is extremely difficult to compute m without knowing SK.
- There is a binary 'product' $\cdot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ on $\mathcal{M}$ the set of encrypted and decrypted message data which forms a group $\langle \mathcal{M}, \cdot \rangle$ with unit 1 (identity element) over $\mathcal{M}$.
- The decryption function $SK : \mathcal{M} \to \mathcal{M}$ distributes over the binary product $\cdot$ on $\mathcal{M}$ i.e. $SK : \langle \mathcal{M}, \cdot \rangle \to \langle \mathcal{M}, \cdot \rangle$ is a group homomorphism.

Let m be the coin's serial number, (PK, SK) be the Bank's asymmetric encryption/decryption key pair, r is a random element chosen from the group $\mathcal{M}$. The sender encrypts r using the Bank's public key forming the blinding factor $(PK(r))$, and computes the product of the serial number with this blinding factor to form the blinded coin serial number:

$$m' = m \cdot PK(r)$$

The Bank in turn signs the blinded serial number with its private key:

$$s' = SK(m')$$

Returns the coin to the user who removes blinding factor:

$$s = s' \cdot r^{-1}$$

The user now has a coin signed with the Bank's private key:

$$\begin{aligned} s &= SK(m') \cdot r^{-1} \\ &= SK(m \cdot PK(r)) \cdot r^{-1} \\ &= (SK(m) \cdot SK(PK(r))) \cdot r^{-1} \\ &= (SK(m) \cdot r) \cdot r^{-1} \\ &= SK(m) \cdot (r \cdot r^{-1}) \\ &= SK(m) \cdot 1 \\ &= SK(m) \end{aligned}$$

2.2 Bitcoin and Blockchains

Bitcoin is a *decentralized*, *pseudo-anonymous* electronic cash scheme [7]. The Bitcoin protocol is decentralized in the sense that the participants collectively verify all of the transactions in the network. The security of Bitcoin is based

around the assumptions that a majority of the nodes in the network are honest, and that the proof-of-work (PoW) algorithm employed will deter any sybil attacks [12], as the amount of computational resources required will be greater than 50% of the network resources.

However, it is important to note that all Bitcoin transactions are stored in a public ledger known as the *blockchain*, which can be parsed by others in an attempt to ascertain the identity of participants in the system. The blockchain is a timestamped public ledger of all transactions that have ever been conducted on the Bitcoin network. The first block in the chain is known as the *genesis block*, followed by blocks that have been created by *miners*.

Miners in the Bitcoin network are nodes that keep a complete and up to date version of the blockchain, and compete to try to be the first to add the next valid block into the blockchain, so that they can earn some bitcoins and or transaction fees. Valid transactions are irreversibly locked into the blockchain using the *proof-of-work* algorithm by the miners who work for a reward for solving the next PoW problem (see section 2.3 below).

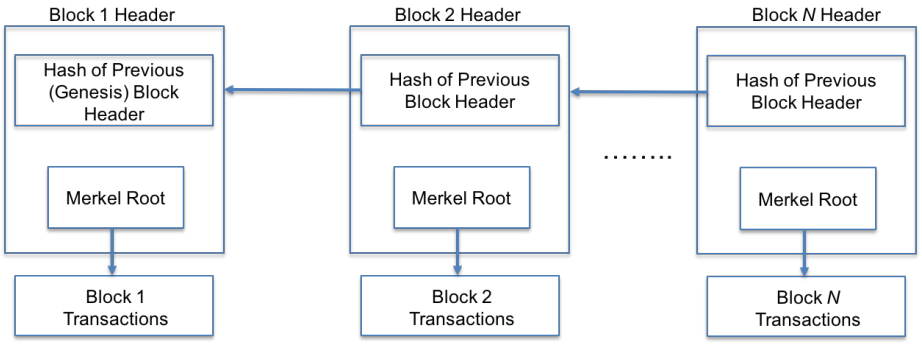

Fig. 1. Blockchain Structure

Each new block contains one or more new transactions that have been received by the miner within a specified time interval (e.g. every ten minutes). These are repeatedly hashed in pairs to form a *Merkel tree* [13]. The root of the Merkel tree along with the hash of the previous block is stored in the block header thereby chaining all the blocks together (see figure 1). This ensures that a transaction cannot be modified without modifying the block that records it and all following blocks. This property of the blockchain makes double spending of bitcoins very difficult.

2.3 Proof-of-Work

The problem with the above *collective verification* approach is that Alice can still cheat Bob by introducing a large number of nodes into the network that are

controlled by her [12]. These nodes could then all reply back to Bob saying that the coin given to him by Alice was legitimate and trick Bob into accepting the transaction. To prevent this from happening the Bitcoin protocol employs the Hashcash PoW algorithm [8].

The Hashcash system was proposed by Adam Black in 1997 as a means to prevent spammers from sending large amounts of unsolicited emails to users on the Internet. A cryptographic hash function (e.g. SHA-256) takes an arbitrary length input and produces a fixed length output. Hash functions are designed to be *collision resistant* i.e., it is computationally hard to find two inputs x and y which are different and which produce the same output $H(x) = H(y)$. For SHA-256 this requires on average 2^{128} or 4×10^{38} attempts to find a collision, which is a near impossible task given current computational capabilities [14].

Hashcash simplifies this requirement considerably by only looking for *partial collisions*. A k-bit partial collision is where only the first k most significant bits match. In practice the hash output is preceded by k zero bits. For example, say one is required to find a partial collision for a string s. Let $s =$ "hello, world!" such that the hash output begins with four zeros ('0000') [15]. We vary the length of the string s by concatenating an integer value x as a string to the end called a *nonce* and incrementing it each time until we find the desired result. Concatenating the nonce $x = 0$ does not produce a match:

$$H(hello, world!0) => 1312af178c253f84028d480 \ ...$$

We keep on incrementing the value of x until we reach the number $x = 4250$ which gives us the desired result, with four zeros at the start of the hash output:

$$H(hello, world!4250) => 0000c3af42fc31103f1f \ ...$$

This task can be made more or less difficult by varying the number of zeros required to obtain the partial collision. As per the above example, a relatively simple proof-of-work problem that requires four zeroes at the start of the hash output is quick to solve. A more difficult proof-of-work problem might require a longer run e.g., ten consecutive zeros which would take a longer time to solve.

A miner (Trudy) in the Bitcoin network is someone that wants to earn some bitcoins and or transaction fees. Trudy verifies each transaction that is broadcast on the network, within a particular time period, by consulting her version of the blockchain looking for previous incompatible transactions, and then adds them into a block of pending transactions that have not yet been endorsed by the network peers. In order to convince her peers in the network that these pending transactions are valid, she attempts to solve the PoW problem, which requires a substantial amount of computational effort on her part. Since all previous blocks are chained together, this ensures that untrustworthy peers have to work harder than honest peers if they want to modify previous blocks, and include them in the blockchain.

The Bitcoin PoW problem requires the hash of a blocks header to be lower than or equal to a number known as the *target*. The target is a 256-bit number that all Bitcoin clients share. The SHA-256 hash of a transaction block's header

must be lower than or equal to the current target for the block to be accepted by the network. The lower the target value the more difficult it is to generate a block. Trudy broadcasts the results of her computation (i.e. her version of the transaction block and her calculated nonce) to the network. The other network participants can easily verify it and include the block into the blockchain. The system is designed such that any *forks* in the chain can automatically be resolved, and that all network participants eventually agree on a single version of the blockchain.

3 System Overview

Our payment protocol extends the Ecash system by allowing coins to be transferred from one use to another without revealing the identity of any of the participants in the system by examining the coin alone. We combine the blind signature protocol, blockchain and proof-of-work techniques, with the classical discrete logarithm problem to allow for the *anonymous transfer* of coins from one user to another in the system.

By anonymous transfer of coins we mean that the Bank cannot link the serial number of coins with the identities of the users that bought or transacted with the coins. Once a coin is added to the global blockchain the serial number of the coin can be seen by anyone who examines the blockchain - however there is no information within the coin parameters to identify the current owner of the coin. Similarly when the coin is transferred between users in the system the two parties involved in the transaction will share some information between them (such as a shipping or email address) - however no personal identifiers are added into the coin's transaction history that may reveal the identity of the two participants when it is included into the blockchain. As long as the two parties involved in the transaction keep their transaction details secret, the Bank or any other entity is not be able to identify them.

A coin in our system consist of a 4-tuple such that:

$$\text{Coin} = \langle SN, val, H(m_1), SK_{val}(m_1) \mid\mid ... \mid\mid SK_{n-1}(m_n) \rangle$$

where SN is the serial number of the coin, val is the value of the coin, $H(m_1)$ is the hash of the first transaction (m_1), and $(SK_{val}(m_1) \mid\mid ... \mid\mid SK_{n-1}(m_n))$ is a number of signed transaction entries concatenated together, each of which represents the transfer of a coin from one participant to another in the system. The first transaction is always signed by the Bank which mints coins in the system. Each subsequent coin transaction entry reveals a secret quantity associated with the previous transaction (proving ownership of the coin to the intended recipient), and also ties in a new coin secret and public coin transfer key of the new owner. The first three components of the coin's tuple $(SN, val, H(m_1))$ are used to uniquely index the coin within the global blockchain.

3.1 Coin Transaction Entry (m_i)

A coin transaction entry (m_i) is composed of the following parameters:

$$m_i = SN \mid\mid DLP_i \mid\mid x_{i-1} \mid\mid PK_i \mid\mid H(m_{i-1})$$

- A serial number (SN) for the coin which remains constant.
- A discrete logarithm problem $\alpha_i^{x_i} \equiv \beta_i (mod\ p_i)$ which we denote ($DLP_i = (p_i, \alpha_i, \beta_i)$) where:
  - A large prime p_i which allows the formation of the multiplicative group $\mathbb{Z}_{p_i}^*$.
  - The large prime p_i must be chosen in such a way that the multiplicative group $\mathbb{Z}_{p_i}^*$ contains a large subgroup G of prime cardinality (as the cardinality of $\mathbb{Z}_{p_i}^*$ is $p_i - 1$ which is not prime). Doing this prevents Pholig-Hellman attack on the DLP [16] [17].
  - Elements α_i and x_i from the subgroup G with prime cardinality where α_i is chosen to be a primitive element of the subgroup (which will exist because G has been chosen to have prime cardinality).
  - The x_i is the secret quantity which must only be revealed when transferring the coin to another user.
- The x_{i-1} is the secret associated with the previous DLP.
- The public key component (PK_i) of a RSA *coin transfer* key pair (PK_i, SK_i).
  - This is an ephemeral key pair which is generated by the current coin owner on a per transaction basis. This allows the owner of a coin to create a *delegated signature* when transferring the coin without the need to contact a central authority.
  - The public key (PK_i) is embedded into the coin transaction entry m_i and is locked into the transaction by the blind signing of the previous coin's owner or the Bank when the coin is minted. During subsequent coin transfers each owner creates their own delegated signature on the transaction parameters.
- The hash of the previous transaction data $H(m_{i-1})$.

A user forms the coin transaction data (m_i) by concatenation of the serial number (SN), the public discrete logarithm parameters (p_i, α_i, β_i) which we collectively refer to as DLP_i in our protocol, their public ephemeral coin transfer key (PK_i) for this transaction, plus x_{i-1} (the previous secret) and $H(m_{i-1})$ (the hash of the previous transaction data).

$$m_i = SN \mid\mid DLP_i \mid\mid x_{i-1} \mid\mid PK_i \mid\mid H(m_{i-1})$$

In general, each coin consists of a number of signed transactions ($SK_{i-1}(m_i)$), where m_n is the last transaction entry associated with the current owner of the coin (see figure 2). The value x_{i-1} is the DLP secret of the previous coin transaction (m_{i-1}) and is only revealed when proving ownership of the coin. In the case of m_1 the value of x_0 is null as this is the first transaction in the list. In addition, we chain all of the coin transactions together by including the hash of the previous transaction $H(m_{i-1})$ in the next transaction (m_i). Again for transaction m_1 the value of $H(m_0)$ is null as it is the first transaction in the list.

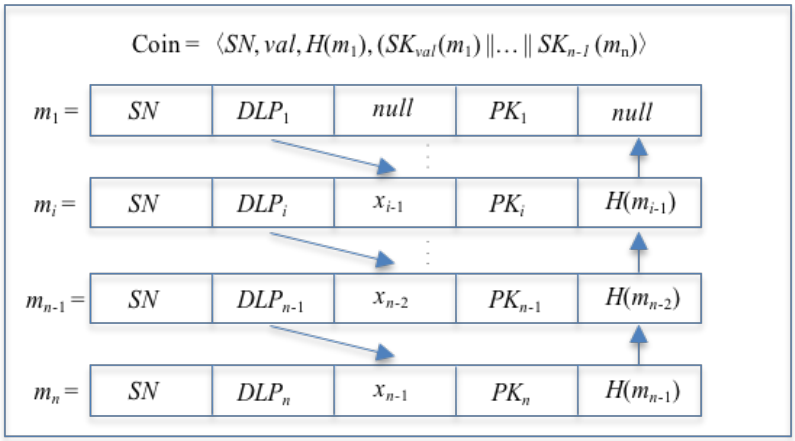

Fig. 2. Structure of a Coin

3.2 Minting a Coin

Figure 3 shows the steps of how Alice gets the Bank to blindly sign the first coin transaction (m_1) with its private signature key for a particular coin value (SK_{val}).

Firstly, Alice constructs the coin's first transaction data m_1 and then blinds this value forming m'_1 and sends this value to the Bank along with her account number and the value of the coin she requires.

Secondly, the Bank has a number of signature keys for different denominational values ($1, $5, $10, ...), and uses the private signature key (SK_{val}) corresponding to the val parameter supplied by Alice during their first message exchange. The Bank deducts the corresponding amount of fiat money from Alice's account, checks that the blinded message m'_1 adheres to certain parameters (such as the message length), and signs the coin's blinded first transaction m'_1 with its private signature key corresponding to val (SK_{val}). The Bank then sends the blindly signed coin's first transaction data (s') in the second message exchange to Alice.

Thirdly, Alice removes her blinding factor r from s' to reveal the coin's first transaction data signed with the Bank's private signature key (SK_{val}) for value (val). Alice now has the coin's first transaction data signed by the Bank without having revealed the contents of the transaction data to the Bank. Only she knows the DLP secret (x_1) which must be revealed by her in order to prove ownership of the coin. In addition, she alone knows the private ephemeral coin transfer key (SK_1) which is used to transfer ownership of the coin to another user or recipient in the network.

Finally, she broadcasts the newly minted coin on the P2P network to be included in the next transaction block of the global blockchain. Note, that there

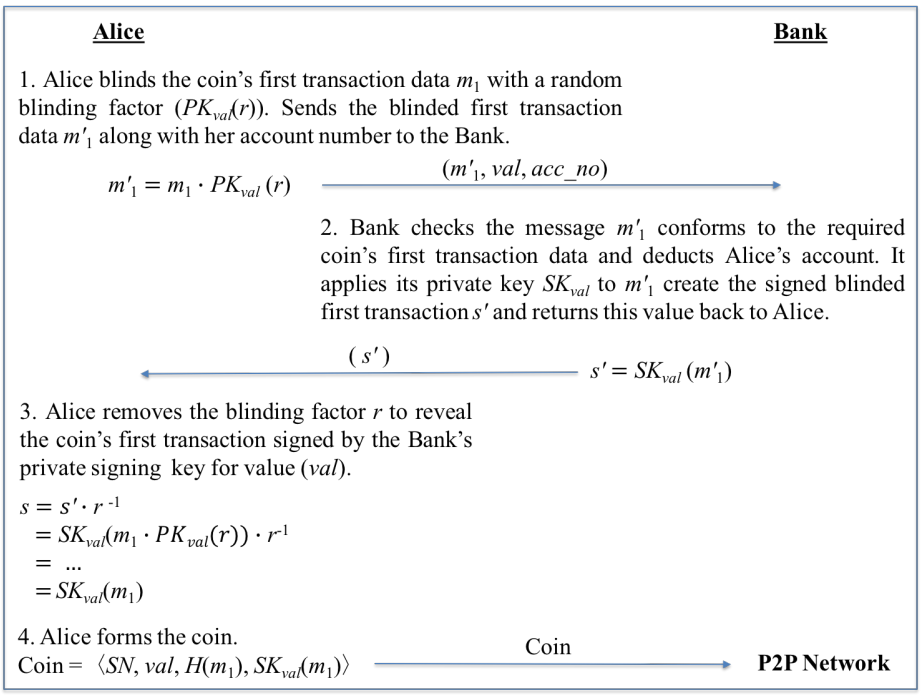

Fig. 3. Anonymous Minting of Coins by the Bank

are no identifying parameters in the coin transaction history that can link the coin to Alice's identity.

Note that all communications between the Bank and Alice are conducted over a secure channel (e.g. SSL), and the Bank has no record of the coin transactions that it just signed for Alice. Also, note that it is Alice's responsibility to ensure that she forms the correct coin transactions as badly formed coins will not be accepted by the other nodes in the system and will not get included into the blockchain.

3.3 Anonymous Coin Transfer

Coins in this scheme are self-contained and consist of a list of signed transactions $(SK_{val}(m_1) \mid\mid \ldots \mid\mid SK_{n-1}(m_n))$, one for each time the coin is transferred from one user to another. A recipient is able to verify each transaction in the list in turn without having to contact a central authority or TTP. This is achieved by the recipient by first downloading the coin from the global blockchain using the coin's public parameters $(SN, val, H(m_1))$ as an index into the blockchain. The recipient then traverses the coin transaction list staring at $SK_{val}(m_1)$ and ending at $SK_{n-1}(m_n)$, ensuring that each transaction is valid.

The first transaction (m_1) is always signed by the Bank whose public key (PK_{val}) for value (val) is known to all participants. A user that asked the Bank to mint the coin embedded their public coin transfer key (PK_1) into m_1. During subsequent coin transfers the public key (PK_{i+1}) of the recipient of the coin is locked into the next transaction (m_{i+1}) by the owner when they create a blind signature using their coin transfer private key (SK_i). We then make use of a global ledger and the PoW algorithm to lock in valid coins into the blockchain and to prevent the double-spending.

In order for Alice to transfer a coin to another user or spend coins at a merchant premises she needs to fulfill two requirements. The first is that she must be able to prove ownership of the coin that she trying to spend by revealing the DLP secret (x_n) for the last transaction (m_n) for the coin. Once the recipient (Bob) has verified the DLP $\alpha_n^{x_n} \equiv \beta_n \ (mod\ p_n)$, he generates a new transaction (m_{n+1}) to be signed with Alice's private coin transfer key (SK_n). He does this by applying Alice's public coin transfer key (PK_n) to a random number (r) to form a blinding factor ($PK_n(r)$). Note Alice's public coin transfer key (PK_n) can be found in the last transaction m_n in the coin history. Bob blinds his new coin transaction m_{n+1} with the blinding factor to produce m'_{n+1} and sends this to Alice to sign. Alice checks that m'_{n+1} has the correct structure and applies her private coin transfer key (SK_n), and returns the blindly signed coin transaction. Bob removes r to reveal the new coin transaction signed with the private coin transfer key of Alice ($SK_n(m_{n+1})$). Finally, Bob updates the coin history and broadcasts the new version of the coin to the P2P network to be included in the next transaction block of the global blockchain. Only when Bob sees the new version of the coin appearing in the global blockchain plus an additional six further transaction blocks does he complete the transaction. This process typically takes about one hour to complete.

Figure 4 shows in detail the coin transfer protocol between Alice and Bob. In the first message exchange Alice sends the coin's serial number (SN), its value (val), the hash of the first transaction $H(m_1)$, and the DLP secret (x_n) for the last transaction (m_n). Bob downloads the coin from the global blockchain using the SN, val and $H(m_1)$ parameters associated with the coin as an index into the blockchain. He verifies that x_n is the solution to the discrete logarithm problem (DLP_n) of the last transaction (m_n). As Alice did previously, Bob generates a new coin transaction which consists of a new set of DLP parameters $(\alpha_{n+1}, \beta_{n+1}, p_{n+1})$ and secret x_{n+1} such that $\alpha_{n+1}^{x_{n+1}} \equiv \beta_{n+1} \ (mod\ p_{n+1})$, and a new ephemeral coin transfer key pair (PK_{n+1}, SK_{n+1}) which are only know to him such that the next transaction (m_{n+1}) is a follows:

$$m_{n+1} = SN \ || \ DLP_{n+1} \ || \ x_n \ || \ PK_{n+1} \ || \ H(m_n)$$

The new transaction (m_{n+1}) consists of a concatenation of the serial number, Bob's DLP public parameters (DLP_{n+1}), his public coin transfer key (PK_{n+1}), the DLP secret (x_n) for the *previous* transaction (m_n), and a hash of the *previous* transaction $H(m_n)$, thereby linking the two transactions together. He then blinds the transaction (m_{n+1}) by multiplying a blinding factor $PK_n(r)$ where r is a

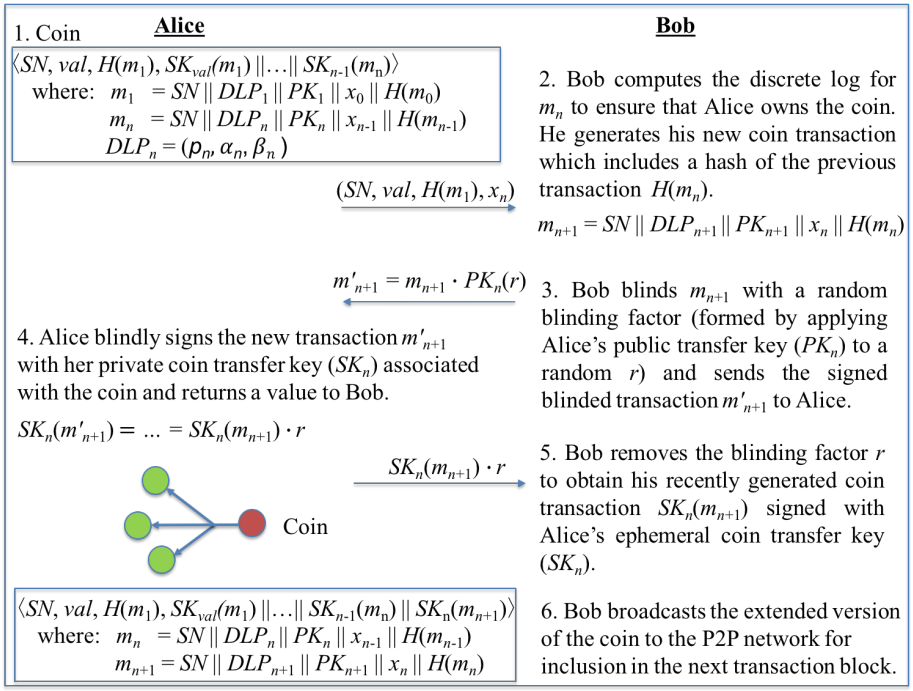

Fig. 4. Coin Transfer Protocol

random number PK_n is Alice's public coin transfer key which can be obtained from the previous transaction (m_n) in the coin's transaction history. Alice blindly signs the new transaction (m'_{n+1}) with her private coin transfer key (SK_n) and returns the blinded new transaction to Bob. Bob removes the blinding factor to reveal the new signed coin transaction $(SK_n(m_{n+1}))$ and updates the coin history with this new signed transaction, and broadcasts the updated coin to the P2P network.

Once the coin is locked into the global blockchain Bob knows that the transaction on the coin has been accepted by the network as being valid and he can now spend the coin, as only he knows the two secret quantities (x_{n+1}, SK_{n+1}) that allow him to prove ownership of the coin, and to transfer the coin legitimately to another user. As long as Alice and Bob do not reveal the transaction details to any third parties their identities will remain anonymous within the system.

3.4 Transaction Verification and Locking

In our system whenever a merchant or user is presented with the serial number of a coin or coins, they first consult the global blockchain and download a copy of the coin(s). We make use of the Bitcoin blockchain protocol and Hashcash proof-

of-work algorithm to collectively verify a coin's authenticity and prevent double-spending of coins within the network. Each node in the system independently verifies the authenticity of the coin's transaction history before forwarding it to its peers. Coins that do not conform to the correct structure are silently dropped from the network and do not get included into the blockchain. As in the Bitcoin protocol we have *miners* in the payment network that periodically solve the PoW problem for a given transaction block which consist of all transactions that have been verified and broadcast to the network in the last x minutes. The miners are rewarded by the Bank for solving the PoW and locking transactions into the global blockchain.

3.5 Size of Coins

We recall from section 3, that a coin in our system has the structure whereby the first three parameters of the coin remain constant, while the transaction list grows with each coin transfer:

$$\langle SN, val, H(m_1), SK_{val}(m_1) \mid\mid ... \mid\mid SK_{n-1}(m_n)\rangle$$

$$\text{where: } m_n = (SN \mid\mid DLP_n \mid\mid x_{n-1} \mid\mid PK_n \mid\mid H(m_{n-1}))$$

- 512 bit serial number (SN) - 64 bytes
  - Size of val is negligible
- Hash of the first transaction $H(m_1)$ - 32 bytes
- DLP public params (p_n, α_n, β_n) of 1024 bits each - 384 bytes
  - x_{n-1} is 32 bytes
- 3072 bit secret coin transfer key (SK_n) - 384 bytes
  - Public coin transfer key (PK_n) is negligible in length
- $H(m_{n-1})$ - 32 bytes

If we make use of the above parameter lengths then each coin transaction (m_n) requires ∼512 bytes. Using RSA encryption with optimal asymmetric encryption padding (OAEP) [18] gives us a length of ∼768 bytes ($SK_{n-1}(m_n)$ - for the two ciphertext blocks that are generated). Adding in the fixed part of the coin parameters (96 bytes) gives us a final coin length of ∼865 bytes for a coin with a single transaction. After 1000 transactions the size of the coin would still be less than 1MB. If a coin's storage requirements exceed certain limits then it can be easily removed from the system by the Bank and a new coin can be issued in its place.

In practice, the client wallet only needs to securely store the following 4-tuple value $\langle SN, val, H(m_1), x_n, SK_n\rangle$ for the last transaction (m_n) for each coin that the user owns. Therefore in order to store the coin secret parameters the client wallet requires ∼ 512 bytes for each coin.

3.6 Removing Coins from Circulation

As indicated in the previous section it may be necessary to remove coins from the payment network if their size exceeds a pre-defined system limit or if Alice decided to *cash-in* her coin. The size of a coin will grow with each transaction, and in the case of frequently transacted coins the system defined size limit may be exceeded. In such circumstances, a special procedure can be initiated whereby the client wallet would automatically transfer the coin to the Bank which would in turn credit the user for the value of the coin. The Bank would then broadcast the coin with the *last* coin transaction where the DLP parameters and the coin transfer key values would be null. This final coin transaction will be an indication to all users that the coin has now been taken out of circulation from the payment network. This approach implies that the user that cashes-in a coin reveals their updated account balance to their Bank.

4 Security Considerations

In the following sections we discuss some of the security issues associated with our scheme:

4.1 Transferability

Only the owner of the coin knows the DLP secret (x_n) for the last coin transaction (m_n) which is required to be exposed to the intended recipient in order for the coin to accepted by that user. We make use of the owner's coin transfer key to blindly sign the new coin transaction (which is generated by the intended recipient of a coin). This allows for the new transaction (m_{n+1}) which consists of the previous DLP secret (x_n), the new DLP_{n+1} parameters, and the new coin transfer key (PK_{n+1}) to be embedded into the coin history. We also lock in the hash of the previous transaction $H(m_n)$ into the new transaction (m_{n+1}) thereby creating a chain of transactions. These measures together prevent the transaction list to be modified without knowledge of the correct signature key, and there is no requirement to contact the Bank or any TTP during the transfer of coins in the system. This satisfies our transferability requirement without any central authority involvement.

4.2 Anonymity

There are no identifying parameters within a coin's transaction history. An owner of a coin who wishes to transfer it to someone else must reveal the DLP secret for the last transaction in the coin transaction history to the intended recipient, and must blindly sign the new coin parameters with their secret coin transfer key. When an owner of the coin signs the new transaction in order to transfer the coin to a recipient they make use of a per-transaction coin transfer ephemeral key. This ephemeral key should never be used again for another transaction by the user. This prevents anyone from analyzing the blockchain and linking the transactions to a particular user, which therefore guarantees anonymity.

4.3 Double-Spending

The blockchain is a timestamped public ledger of all valid coins that have ever been transacted on the payment network. Coins containing non-valid transactions are automatically dropped by fully functioning and trusted nodes in the system, and not allowed to propagate further through the network. The first block in the chain is known as the genesis block, followed by blocks that have been generated by miners. Each new block contains one or more new coins that have been received by a miner on the P2P network in the last m minutes. These are repeatedly hashed in pairs to form a Merkel tree. The root of the Merkel tree along with the hash of the previous block is stored in the block header thereby chaining all the blocks together. Each of the miners then tries to solve a hard problem called the proof-of-work, and whoever solves it first is rewarded by the Bank for their efforts. This ensures that a transaction cannot be modified without modifying the block that records it and all following blocks. This property of the blockchain makes double-spending of coins very difficult.

4.4 Security Questions

In the section below we pose a number of security questions and try to provide a reasonable defence for each scenario:

Can the Bank trick Alice into revealing coin serial numbers?

– A scenario can occur where the Bank can act as a malicious entity and only mint one coin at a time, and wait until the coin has been locked into the global blockchain in order to match user identities to serial numbers of coins. We can prevent this by designing the client software to wait for a least X new blocks to be added to the blockchain, and a minimum of Y valid transactions to have been broadcast on the P2P network before the client broadcasts its newly minted coin into the network. We believe that this measure introduces enough randomness into the system to prevent the Bank from carrying out such an attack.

Can Alice fraudulently procure coins from the Bank?

– In our system, we have multi-denominational coins ($1, $5, $10, ...). The Bank has a separate signature key (SK_{val}) for each denomination. The Bank will only sign a blinded coin with the corresponding signature key if Alice has a sufficient amount of fiat currency in her account which the Bank can successfully deduct. It is Alices responsibility to ensure that she has formed the correct coin parameters as the coin will not be accepted by other users in the system otherwise.

Can Alice spend a coin multiple times?

– We make use of the network participants to verify a new transaction on a coin when it is being transferred from one user to another, and use the

PoW algorithm to lock in the new coin parameters into a global ledger. The recipient of a coin first consults the blockchain and downloads the last transacted copy of the coin. This prevents someone like Alice from spending a coin multiple times.

Can Bob insert a malformed transaction into the coin history?

– Bob can send a malformed transaction to be signed by Alice. However, for the coin to be included in the blockchain all other participants will want to verify the chain of transactions $(m_1, ..., m_n)$. Failure to do so will result in the coin being rejected and silently being dropped from the P2P network. The serial number (SN) in each transaction should be same as the one initially signed by the Bank, and each subsequent transaction (m_{i+1}) must be signed with the private coin transfer key corresponding to the public key (PK_i) locked into the previous transaction (m_i). Also the DLP secret (x_i) for the previous transaction has to be revealed in the new transaction.

Can Trudy try to spend a coin that she does not own?

– In order to spend a coin a user must be able to reveal the DLP secret, and also sign the new transaction with the private key corresponding to the public key locked into the last transaction (m_{i-1}) on the coin. Although, all the coin transactions are stored in a global publicly accessible blockchain, each user must securely store some secret parameters, namely the DLP secret and the secret coin transfer key associated with coins that they currently own. If these parameters were to be revealed accidentally or stolen maliciously, then those coins could be anonymously spent by others in the system. It is imperative that each user undertakes measures to securely store the secret parameters on a secure device with additional file or disk encryption to prevent unauthorized access.

5 Conclusion

In this paper, we have presented a fully anonymous version of the Ecash scheme with unlimited transferability of coins. We are able to achieve this by embedding a *coin transfer public key* into each transaction entry for the coin such that the recipient can use the owner's coin transfer public key to request the owner to *blindly sign* the new coin transaction parameters. In effect, we have been able to create a *delegated signature* scheme to allow for the transfer of coins within the system without the need to contact a trusted third party.

In addition, unlike other schemes which allow for *post-fact detection* of the double-spending of coins and reveal the identity of participants in the system, our scheme *prevents* double-spending of coins within the system. We do this by making use of the distributed decision making techniques used in the Bitcoin system. Specifically we make use of the Bitcoin blockchain and Hashcash proof-of-work algorithms to lock-in all valid transactions into a global ledger.

Finally, we never reveal the identity of users in the system. We use the classic hard discrete logarithm problem to prove ownership of the coin during a payment transaction or coin transfer. In addition we make use of a novel per-transaction ephemeral coin transfer key which allows us to anonymously transfer coins in the network.

References

1. B. Bennett, D. Conover, S. O'Brien, and R. Advincula, "Cash Continues to Play a Key Role in Consumer Spending: Evidence from the Diary of Consumer Payment Choice", *San Francisco Federal Reserve Bank*, `http://www.frbsf.org/cash/publications/fed-notes/2014/april/cash-consumer-spending-payment-diary`, April 2014
2. D. Chaum, "Security Without Identification: Transaction Systems to Make Big Brother Obsolete", *Communications of the ACM*, vol 28, no 10, pages 1030-1044, October 1985
3. F. Baldimtsi, M. Chase, G. Fuchsbaur and M. Kholweiss, "Anonymous Transferable E-Cash", *Proceedings of Public-Key Cryptography - PKC 2015*, 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Springer-Verlag, pages 101-124, 2015.
4. D. Chaum, A. Fiat and M. Naor, "Untraceable Electronic Cash", *Proceedings of Crypto 88*, Advances in Cryptology, Springer-Verlag, vol 403, pages 319-327, 1988
5. M. Chase, M. Kholweiss, A. Lysyanskaya and S. Meiklejohn, "Malleable Signatures: New Definitions and Delegateable Anonymous Credentials", *Proceedings of IEEE Computer Security Foundations Symposium*, pages 199-213, 2014
6. D. Chaum, "Blind Signatures for Untraceable Payments", *Proceedings of Crypto 82*, Advances in Cryptology, Springer-Verlag, pages 199-203, 1982
7. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", `http://www.bitcoin.org`, 2008
8. A. Black, "Hashcash Proof-of-Work System", `http://www.hashcash.org/`, May 1997
9. W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions Information Theory*, vol 22, no 6, pages 472-492, 1976
10. C. Paar and J. Pelzl, "Understanding Cryptography", Springer-Verlag, pages 392, October 2011
11. D. Chaum, "Blind Signatures System", *Proceedings of Crypto 83*, Advances in Cryptology, Springer-Verlag, 1983
12. J. Douceur, "The Sybil Attack", *Proceedings of First InternationalWorkshop, IPTPS 2002*, Peer-to-Peer Systems, Springer-Verlag, pages 251-260, 2002
13. R.C. Merkel,"Protocols for Public Key Cryptosystems", *Proceedings of the Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980
14. Q. Dang, "Recommendation for Applications Using Approved Hash Algorithms", NIST Special Publication 800-107, Revision 1, `http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf`, August 2012
15. M. Nielsen, "How the Bitcoin Protocol Actually Works", `http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/`, December 2013

16. S. Pohlig and M. Hellman, “An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance”, *IEEE Transactions on Information Theory*, vol 24, pages 106110, January 1978
17. J. Hoffstein, J. Pipher and J. H. Silverman, “An Introduction to Mathematical Cryptography”, Sprinfer-Verlag, pages 533, November 2009
18. M. Bellare andP. Rogaway, “Optimal Asymmetric Encryption – How to encrypt with RSA”, *Proceedings of Eurocrypt ’94* , Advances in Cryptology, Springer-Verlag, vol 950