LEDGER
ledgerjournal.org

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

RESEARCH ARTICLE

Ring Confidential Transactions

Shen Noether,* Adam Mackenzie, the Monero Research Lab†

Abstract. This article introduces a method of hiding transaction amounts in the strongly decentralized anonymous cryptocurrency Monero. Similar to Bitcoin, Monero is a cryptocurrency which is distributed through a proof-of-work "mining" process having no central party or trusted setup. The original Monero protocol was based on CryptoNote, which uses ring signatures and one-time keys to hide the destination and origin of transactions. Recently the technique of using a commitment scheme to hide the amount of a transaction has been discussed and implemented by Bitcoin Core developer Gregory Maxwell. In this article, a new type of ring signature, A Multilayered Linkable Spontaneous Anonymous Group signature is described which allows one to include a Pedersen Commitment in a ring signature. This construction results in a digital currency with hidden amounts, origins and destinations of transactions with reasonable efficiency and verifiable, trustless coin generation. The author would like to note that early drafts of this were publicized in the Monero Community and on the `#bitcoin-wizards` IRC channel. Blockchain hashed drafts are available showing that this work was started in Summer 2015, and completed in early October 2015.[17] An eprint is also available at `http://eprint.iacr.org/2015/1098`.

1. Introduction

Recall that in Bitcoin each transaction is signed by the owner of the coins being sent and these signatures verify that the owner is allowed to send the coins. This is entirely analogous to the signing of a check from your bank. In Bitcoin, value is created through a trustless "mining" process, where users try to solve a mathematical problem, based on a hash function, in order to receive coins. This coin generation phase is trustless in the sense that it requires no central party to distribute value or guarantee its security, and anyone with a computer connected to the internet may participate. In this paper we discuss a protocol allowing for coins to be transmitted anonymously while retaining the trustless coin-generation process.

1.1. Previous Attempts: CryptoNote—CryptoNote and Ring Coin advance the digital signature part of Bitcoin by using "ring signatures" which were originally described by Rivest *et al.* as a "digital signature that specifies a group of possible signers such that the verifier can't tell which member actually produced the signature."[22, 3, 20] The idea, therefore, is to have the origin pubkey of a transaction hidden in a group of pubkeys all of which contain the same amount of coins, so that no one can tell which user actually sent the coins.

The original CryptoNote protocol implements a slight modification of this to prevent double spends.[22] Namely, CryptoNote employs a "traceable ring signature," which is a slight mod-

*S. Noether (shen.noether@gmx.com) is a cryptography researcher at the Monero Research Lab
†1KTexdemPdxSBcG55heUuTjDRYqbC5ZL8H

ification of those described by Fujisaki and Suzuki.[22, 10] This type of ring signature has the benefit of preventing the owner of a coin from signing two different ring signatures with the same pubkey without being noticed on the blockchain. The obvious reason for this is to prevent "double-spending" which, in Bitcoin, refers to spending a coin twice. Ring coin uses a more efficient linkable ring signature which is a very slight modification of the Linkable Spontaneous Anonymous Group signatures described by Liu *et al.*[3, 4, 11]

One possible attack against the original CryptoNote or ring-coin protocol is blockchain analysis based on the amounts sent in a given transaction.[22, 4] For example, if an adversary knows that .9 coins have been sent at a certain time, then they may be able to narrow down the possibilities of the sender by looking for transactions containing .9 coins. This is somewhat negated by the use of the one-time keys used in van Saberhangen's CryptoNote protocol since the sender can include a number of change addresses in a transaction, thus obfuscating the amount which has been sent with a type of "knapsack mixing."[22] However this technique has the downside that it can create a large amount of "dust" transactions on the blockchain, *i.e.* transactions of small amounts that take up proportionately more space than their importance. Additionally, the receiver of the coins may have to "sweep" all this dust when they want to send it, possibly allowing for a smart adversary to keep track of which keys go together in some manner. Furthermore, it is easy to establish an upper and lower bound on the amounts sent.

Another downside to the original CryptoNote set-up is that it requires a given pair of (P,A) of pubkey P and amount A to be used in a ring signature with other pubkeys having the same amount. For less common amounts, this means there may be a smaller number of potential pairs (P',A') available on the blockchain with $A' = A$ to ring signature with. Thus, in the original CryptoNote protocol, the potential anonymity set is perhaps smaller than may be desired. Analysis of the above weaknesses is covered in Noether *et al.*[12]

1.2. CoinJoin and Coin Mixing—One benefit of using the above types of ring signatures over other anonymizing techniques, such as CoinJoin or using coin mixing services, is that they allow for "spontaneous" mixing. With CoinJoin or coin mixers, it is similarly possible to hide the originator of a given transaction, however these techniques may require a delay while participants wait for their coins to be mixed.[13] In addition, although recent schemes such as XIM attempt to mitigate this through probabilistic means, most existing Bitcoin mixers rely on a trusted party.[7] In the case that the trusted party is compromised, the anonymity of the transaction is also compromised.

Some coins such as Dash (originally called Darkcoin),[8] attempt to negate this by using a larger number of trusted mixers (called "masternodes") but this number is still much smaller than the users of the coin. In contrast, with a spontaneous ring signature, transactions can be created by the owner of a given pubkey (this is the spontaneous, or "ad-hoc" property) without relying on any trusted party, and thus providing for safer, faster, and more reliable anonymity.

1.3. Confidential Transactions (Bitcoin)—Bitcoin itself includes, in the Elements sidechain, a method of hiding the amounts in transactions, namely Confidential Transactions itself, as described by Maxwell.[14] This could probably be combined with coin mixing, to provide additional anonymity. However, the downsides to CoinJoin mentioned above would still apply, namely, parties would need to wait and arrange carefully their mixing, rather than being able to disguise the sender of a transaction instantly, as in CryptoNote, by creating an ad-hoc ring signature. Still, this method has the benefit of the trustless coin-generation process of Bitcoin,

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

which is desirable, and the advantage over CryptoNote of hidden amounts.[22]

1.4. Decentralized Anonymous Payment Schemes—Another protocol which achieves anonymous transactions is Zerocash.[5] Zerocash is a highly anonymous database which may be grafted onto another currency to provide anonymous transactions. The Zerocash scheme has the downside that all of the anonymous coins must be pregenerated by a trusted group. If any actor somehow compromises the group and acquires all of the pieces of the master key used in coin generation, then they will be able to generate free coins at will, dropping the value of all coins in existence, and thus ruining the currencies ability to be a store of value. Obviously, it would be risky to graft Zerocash onto a currency which has a lot of infrastructure depending on it, such as Bitcoin itself.

Note that one of the biggest innovations of Bitcoin,[16] was the decentralized distribution model allowing anyone willing to put their computing power to work to participate in the generation of the currency. Some of the benefits of this type of proof-of-work include trustless incentives for securing the network and stronger decentralization (for example, to protect against poison-pill type attacks which may also affect Zerocash.[5] Since Zerocash inherently lacks this mining phase, and must instead be grafted to another currency which has mining, it is unclear whether these benefits are enforceable.

Another downside to Zerocash is the large computational cost it requires to generate transactions. A Zerocash transaction proof takes about 3 minutes to generate, and requires a computer with a significant amount of RAM (although obviously technological advances will reduce this issue).[5] In contrast, using the techniques of this paper, the authors have experimentally generated many transaction proofs per second using only an older laptop with an i3 processor and a small amount of RAM.

Finally, due to the complex mathematics behind Zerocash, it is unclear whether Zerocash allows for anonymous multisignature transactions, in other words, transactions with more than one signer. These type of transactions are useful for purposes such as two-factor authentication, escrow, and joint bank accounts. The methods of this paper allow for anonymous multisignature with no additional modification.

1.5. Ring Confidential Transactions Overview—In this paper, we describe a modification to the Monero protocol, a proof-of-work cryptocurrency extending the original CryptoNote protocol. The modification is based on Confidential Transactions which are used on the Elements side-chain in Bitcoin, except it allows for their use in ring signatures.[14] Therefore, the modification is given the obvious name of Ring Confidential Transactions (Ring CT) for Monero. The authors note that a similar protocol was proposed by Fromknecht a month after the original blockchain-timestamped drafts of this paper were publicized to the Monero and Bitcoin Community.[9, 17]

The Ring CT protocol is a currency which (1) requires no trusted setup, (2) hides the identity of the recipient of a transaction, (3) hides the identity of the originator of a transaction, and (4) hides the amount in a given transaction. The new property (4), which does not exist in CryptoNote,[22] negates most of the downsides to ring signature based currencies noted in Section 1.1.

Properties (1) and (2) are accomplished in the same manner as in CryptoNote.[22] Namely, a strongly-decentralized (*i.e.*, with no privileged party) hash-based "mining" process provides the trustless coin generation, and one-time keys, created as in CryptoNote,[22] hide the identity of the recipient of a transaction.

Property (4) will be the result of then applying Greg Maxwell's "Confidential Transactions"

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

protocol (which consists of signing a Pedersen Commitment to an amount rather than a plaintext amount),[14] however in order to also obtain property (3) while retaining the trustless aspect, some care must be taken. In CryptoNote,[22] hiding the originator of a transaction is accomplished by signing the transaction with a ring signature, rather than an ordinary digital signature. In CryptoNote,[22] each public key in the ring signature must contain the same amount of coin, corresponding to the amount in the output. If the amounts are therefore naively hidden, it is impossible to guarantee that the inputs to the signature have the same amount of coins as the outputs. We solve this by ring-signing on the actual Pedersen Commitments. This ring signature on Pedersen Commitments is discussed in Section 3.2.

Creating a ring signature on the commitments is not enough, since the output commitments could be changed, creating another signature which will also verify, thus allowing the input coins to be spent twice. (We explain this in more detail in Section 4.) To deal with this issue, a generalization of the signatures of Liu *et al.* is created,[11] which allows a user to sign with a vector of keys, rather than a single key. The point of using a key-vector is so that a user can prove they simultaneously have knowledge of both the secret key of the input address and the secret key of the input commitment. (Contrast this with signing two distinct ring signatures: one for the commitment, and one for the input key. This would prove that you know keys for an amount and an address, but not necessarily link the two.) This generalization is called a Multilayered Linkable Spontaneous Anonymous Group signature (MLSAG) and will be discussed in Section 2. The ring signature on the Pedersen Commitments is placed in the bottom row of the MLSAG, and the overall protocol would not be possible without the multilayering.

The final part of the Ring CT protocol deals with proving the output of a given transaction lies within an acceptable range of positive values. As noted by Maxwell,[14] if this "range proof" is not performed, a user might be able to create a transaction outputting a negative value. Since amounts are encoded as elements in a very large finite field, this would result in the creation of free money as the (small) negative value becomes a (large) positive value modulo the field order. These "range-proofs" are described in Section 3.3 and an example construction is given in an appendix.

In summary, this paper takes a number of previously known cryptographic techniques and generalizes and combines them in a novel way to result in a currency with similar anonymity properties to Zerocash,[5] but retaining the trustless coin-generation inherent in Bitcoin.

2. Multilayered Linkable Spontaneous Anonymous Group Signatures

In this section, we define the Multilayered Linkable Spontaneous Anonymous Group signatures (MLSAG) used by the Ring CT protocol. Note that we define these as a general signature, and not necessarily in their use case for Ring Confidential Transactions. An MLSAG is essentially similar to the LSAG's described by Liu *et al.* but rather than having a ring signature on a set of n keys,[11] instead, an MLSAG is a ring signature on a set of n key-vectors.[1] Recall that an LSAG scheme is a triple of algorithms $(GEN, SIGN, VER)$ consisting of a key-generation phase, a signature phase, and a verification phase. The signature phase should produce a single-signer signature on a set of $n \geq 2$ public keys in such a way that it is provably difficult to tell which key belongs to the actual signer, and in such a way that if the signer signs twice, then this will be noticed.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

Definition 1. A **key-vector** is just a collection $\bar{y} = (y_1, ..., y_r)$ of public keys with corresponding private keys $\bar{x} = (x_1, ..., x_r)$.

2.1. LWW Signatures vs. FS Signatures—The ring signatures used in Monero and the original CryptoNote protocol are derived from the traceable ring signatures of Fujisaki and Suzuki.[10] The CryptoNote ring signatures come with a "key-image" which means that a signer can only sign one ring on the block-chain with a given public and private key pair or else their transaction will be marked as invalid.[22] Because of this, one-time keys are used in CryptoNote, which further helps anonymity.

Adam Back noticed that the Linkable Spontaneous Anonymous Group (LSAG) signatures of Liu can be modified to give a more efficient linkable ring signature producing the same effect as the Fujisaki ring signatures.[4, 11, 10] This modification reduces the storage cost on the blockchain essentially in half.

First we recall (almost verbatim) the modification given by Back:[4]

GEN: Let G be the basepoint of cyclic group where the discrete logarithm assumption is assumed to hold (Monero currently uses Ed25519).[6] Find a number of public keys $P_i, i = 0, 1, ..., n$ and a secret index $j \in \{0, 1, ..., n\}$ such that $xG = P_j$ where G is the base-point and x is the signer's spend key. Let $I = xH_p(P_j)$ be the key image corresponding to P_j where H_p is a cryptographically secure hash function returning a point whose logarithm with respect to the base-point G is unknown.

SIGN: Let $\mathfrak{m}$ be a given message to sign (in practice, $\mathfrak{m}$ is a sha512 has of an arbitrary string. Let $\alpha, s_i,\ i \neq j,\ i \in \{1, ..., n\}$ be random values in $\mathbb{Z}_q$ (the Ed25519 base field).

Compute

$$L_j = \alpha G$$

$$R_j = \alpha H_p(P_j)$$

$$c_{j+1} = H_s(\mathfrak{m}, L_j, R_j)$$

where H_s is a cryptographic hash function returning a value in $\mathbb{Z}_q$. Now, working successively in j modulo n, define

$$L_{j+1} = s_{j+1}G + c_{j+1}P_{j+1}$$

$$R_{j+1} = s_{j+1}H_p\left(P_{j+1}\right) + c_{j+1} \cdot I$$

$$c_{j+2} = H_s\left(\mathfrak{m},\ L_{j+1},\ R_{j+1}\right)$$

$$\ldots$$

$$L_{j-1} = s_{j-1}G + c_{j-1}P_{j-1}$$

$$R_{j-1} = s_{j-1}H_p\left(P_{j-1}\right) + c_{j-1} \cdot I$$

$$c_j = H_s\left(\mathfrak{m}, L_{j-1},\ R_{j-1}\right)$$

so that $c_1, ..., c_n$ are defined.

Let $s_j = \alpha - c_j \cdot x_j \ mod\ l$, ($l$ being the Ed25519 curve order) hence $\alpha = s_j + c_j x_j \ mod\ l$ so that

$$L_j = \alpha G = s_j G + c_j x_j G = s_j G + c_j P_j$$

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

$$R_j = \alpha H_p(P_j) = s_j H_p(P_j) + c_j I$$

and

$$c_{j+1} = H_s(\mathfrak{m},\ L_j,\ R_j)$$

and thus, given a single c_i value, the message $\mathfrak{m}$, the P_j values, the key image I, and all the s_j values, then all the other $c_k,\ k \neq i$ can be recovered by an observer. The signature therefore becomes:

$$\sigma = (I, c_1, s_1, ..., s_n)$$

which represents a space savings over CryptoNote,[22] where the ring signature would instead look like:

$$\sigma = (I, c_1, ..., c_n, s_1, ..., s_n)$$

VER: Verification proceeds as follows. An observer computes L_i, R_i, and c_i for all i and checks that $c_{n+1} = c_1$. Then the verifier checks that

$$c_{i+1} = H_s(\mathfrak{m}, L_i, R_i)$$

for all $i \mod n$

LINK: Signatures with duplicate key images I are rejected.

2.1.1 Back signatures vs LWW signatures

The very slight difference between the above signature of Back and that of Liu *et al.* is in the key image.[4, 11] In the scheme described by Liu, I is defined by the equation

$$I = x_j H_p(P_1, P_2, ..., P_n).$$

In other words, that scheme proves that a signer can only sign once with respect to a given set of keys (although they could sign with the same key in many different sets of keys, each having different hashes). In the above scheme of Back,[4] the key image is

$$I = x_j H_p(P_j)$$

which enforces that a signer can only sign with a key once, without their signature being rejected in the LINK phase of the algorithm. This modification, which is similar, but more efficient to the one made by Fujisaki and Suzuki,[10] is more suited for digital currency use, since it prevents double-spending of the value stored in a given key, and also links a signer with their key-image, rather than the group of signers with a key-image. Note that proofs of unforgeability, anonymity, and linkability hold for the above protocol which are only insignificant modifications to the proofs given by Liu.[11] We will give a more general version of these proofs for the MLSAGs.

2.2. MLSAG Signature Scheme—An MLSAG signature scheme is a tuple of algorithms (GEN, $SIGN$, VER, $LINK$):

- GEN: A Probabilistic Polynomial Time (PPT) algorithm which takes as input a security parameter k and outputs a secret key-vector $x = (x_1, ..., x_m)$ and corresponding public key-vector $P = (P_1, ..., P_m)$ with each (x_i, P_i) at security level k.
- SIGN: A PPT algorithm which takes as inputs a security parameter k, a message $\mathfrak{m}$, a secret

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

key-vector x, a secret index π, a set of public key-vectors $P_1,...,xG = P_\pi,...,P_n, n \geq 2$ (this set of public key-vectors is called the key-matrix of the signature), and outputs a signature σ.

- VER: A polynomial time algorithm which takes as inputs a security parameter k, a key matrix $\mathfrak{L}$, a message $\mathfrak{m}$, and a signature σ on $\mathfrak{L}, \mathfrak{m}$, and outputs true or false, depending on whether the signature verifies or not. For completeness, the MLSAG scheme must satisfy VER(SIGN$(\mathfrak{m}, \mathfrak{L}, x), \mathfrak{m}, \mathfrak{L})$=true with overwhelming probability at security level k.
- LINK: A polynomial time algorithm which takes as inputs two signatures σ_1 and σ_2 and outputs a bit 0 if they are linked, and 1 if they are not linked.

We will say that the signature is "valid" in a given setting (*e.g.* a blockchain) if it passes phase VER of the above protocol, and additionaly passes phase LINK of the above protocol as σ_1 with σ_2 being any previous MLSAG signature in the given setting.

2.3. MLSAG Security Model—An MLSAG will satisfy the following three properties of Unforgeability, Linkability, and Signer Ambiguity which are very similar to the definitions given by Liu *et al.*[11]

Definition 2. (Unforgeability) An MLSAG signature scheme is unforgeable if for any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ with signing oracle $\mathcal{SO}$ producing valid signatures, given a list $\mathfrak{L}$ of n public key vectors chosen by $\mathcal{A}$ out of those available in a given setting (such as a blockchain), then $\mathcal{A}$ can only with negligible probability produce a valid signature, when $\mathcal{A}$ does not know any of the corresponding private key vectors.

Definition 3. (Linkability) Let L be the set of all public key-vectors in a given setting (*e.g.* on a given blockchain) at security level k. Let L' be the set of all pairs of key-vectors y, y' consisting of keys in L, and such that y, y' have at least one key in common. Set M to be the set of all messages (in Monero, a message $\mathfrak{m}$ consists of the sha512 hash of an arbitrary string of bytes, so the message space could be considered the set of all byte strings hashable in polynomial time). Define Σ to be the set of all MLSAG signatures $\sigma(x, \mathfrak{L}, \mathfrak{m})$ on L signed with respect to private key-vector x, with key-matrix $\mathfrak{L}$ and on message $\mathfrak{m}$. An MLSAG signature scheme is said to be key-image linkable, if there exists a PPT algorithm $\mathcal{B} : \Sigma \to \{0,1\}$ such that for all secret keys x, x', key-matrices $\mathfrak{L}_1, \mathfrak{L}_2$, and messages $\mathfrak{m}, \mathfrak{m}'$,

$$Pr(\mathcal{B}(\sigma(x, \mathfrak{L}_1, \mathfrak{m}), \sigma'(x', \mathfrak{L}_2, \mathfrak{m}') : (x, x') \in L') = 1)) > 1 - Q(k)$$

$$Pr(\mathcal{B}(\sigma(x, \mathfrak{L}_1, \mathfrak{m}), \sigma'(x', \mathfrak{L}_2, \mathfrak{m}') : (x, x') \notin L') = 0)) > 1 - Q(k)$$

for some polynomial in k.

Definition 4. (Signer Ambiguity) An MLSAG signature scheme is said to be signer-ambiguous if for any PPT algorithm $\mathcal{A}$, taking as inputs a message $\mathfrak{m}$, a set D_t of t private keys, any verifying signature σ signed at a randomly chosen column π on any list of key-vectors $L = (\overline{y}_1, ..., \overline{y}_n)$, then, assuming $\overline{y}_\pi$ does not contain any element with private key in D_t, the probability of $\mathcal{A}$ guessing the secret key is less than $\frac{1}{n-t} + \frac{1}{Q(k)}$, *i.e.*,

$$Pr[\mathcal{A}(\mathfrak{m}, L, D_t, \sigma) \to \pi] < \frac{1}{n-t} + \frac{1}{Q(k)}$$

where Q is some polynomial function taking as input the security parameter k.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

We now describe an instantiation of an MLSAG signature. Proofs that this signature satisfies the above definitions of Unforgeability, Linkability, and Signer Ambiguity are given in Appendix A.

2.4. MLSAG Description—For the Ring CT protocol, which will be described in Section 4, we require a generalization of the Back LSAG signatures described in the previous section which allows for key-vectors (Definition 1) rather than just keys.

Suppose that each potential signer of a key-matrix $\mathfrak{L}$ containing n members has exactly m keys $\mathfrak{L} = \left\{P_i^j\right\}_{j=1,...,m}^{i=1,...,n}$. The intent of the MLSAG ring signature is the following:

- To prove that one of the n signers knows the secret keys to their entire key vector.
- To enforce that if the signer uses any one of their m signing keys in another MLSAG signature, then the two rings are linked, and the second such MLSAG signature (ordered by blockchain height) is discarded.

The reason it is desirable to have a signature with vectors of keys $P = P_1, ..., P_m$ is because, since we are going to replace plaintext amounts with Pedersen Commitments to amounts, we need a signature which shows that a signer is using the correct Pedersen Commitment which has been sent to their address. In other words, one of the public keys in the above signature is going to be a Pedersen Commitment to zero. This will be explained further in Section 3.2.

The algorithm proceeds as follows: Let $\mathfrak{m}$ be a given message. We again work in a cyclic group where the discrete logarithm problem is assumed to hold at security level k, and which has basepoint G. Let π be a secret index corresponding to the signer of the generalized ring. Suppose the signer owns secret / public key pairs (x_π^j, P_π^j) for $j = 1, ..., m$ and let $I_j = x_j H_p\left(P_\pi^j\right)$ be the key image corresponding to P_j, where H_p is a cryptographic hash function returning a point whose logarithm (with respect to G) is unknown. Finally, for $j = 1, ..., m$, $i = 1, ..., \hat{\pi}, ...n$ (where $\hat{\pi}$ means omit the index π) let s_i^j be some random scalars (elements of $\mathbb{Z}_q$). Now, in a manner analogous to Section 2.1, define

$$L_\pi^j = \alpha_\pi^j G$$

$$R_\pi^j = \alpha_\pi^j H_p\left(P_\pi^j\right)$$

for random scalars α_π^j and $j = 1, ..., m$. Now, again analogously to Section 2.1, set:

$$c_{\pi+1} = H_s\left(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m\right).$$

where H_s is a cryptographic hash function returning a scalar, and

$$L_{\pi+1}^j = s_{\pi+1}^j G + c_{\pi+1} P_{\pi+1}^j$$

$$R_{\pi+1}^j = s_{\pi+1}^j H_p\left(P_{\pi+1}^j\right) + c_{\pi+1} I_j$$

and repeat this, incrementing i modulo n until we arrive at

$$L_{\pi-1}^j = s_{i-1}^j G + c_{i-1} P_{i-1}^j$$

$$R_{\pi-1}^j = s_{i-1}^j H_p\left(P_{i-1}^j\right) + c_{i-1} \cdot I_j$$

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

$$c_\pi = H_s\left(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1}\right).$$

Finally, solve for each s^j_π using $\alpha_j = s^j_\pi + c_\pi x_j \mod \ell$. The signature is then given as $\left(I_1, ..., I_m, c_1, s^1_1, ..., s^m_1, s^1_2, ..., s^m_2, ..., s^1_n, ..., s^m_n\right)$, so the complexity is $O(m(n+1))$. Verification proceeds by regenerating all the L^j_i, R^j_i starting from $i = 1$ as in Section 2.1 (which is the special case that $m = 1$) and verifying the hash $c_{n+1} = c_1$. If these are being used in a blockchain setting such as Monero, signatures with key images I_j which have already appeared are then rejected. One can easily show, in a manner similar to Liu *et al.*:[11]

- The probability of a signer generating a valid signature without knowing all "m" private keys belonging to their key vector for index π is negligible.
- The probability of a signer not signing for any key of index π is negligible. (In other words, the key images in the signature necessarily all come from index π.)
- If a signer signs two rings using at least one of the same public keys, then the two rings are linked.

We expand on these points in Appendix A with security proofs that the MLSAG signatures satisfy Unforgeability, Signer Ambiguity, and Linkability.

3. Background on Confidential Transactions

3.1. Confidential Transactions in Bitcoin—Maxwell describes Confidential Transactions which are a way to send Bitcoin transactions with the amounts hidden.[14] The basic idea is to use a Pedersen Commitment and the method is well described in the cited source. In this paper we make a slight modification to the Confidential Transactions machinery in that rather than taking the commitments to sum to zero, we instead sign for the commitment, to prove we know a private key. This is described in more detail in the next section.

3.2. Modification for Ring Signatures—Let G be the basepoint of a given cyclic group which satisfies the discrete logarithm assumption. Let H be a point in $\langle G \rangle$ whose discrete logarithm with respect to G is unknown. (For example, Maxwell takes the cryptographic hash of G and uses this as an x-coordinate for a point on an elliptic curve, from which they recover H.)[14]

Under the discrete logarithm assumption on Ed25519, the probability of an adversary discovering $log_G H$ is negligible. Define $C(a,x) = xG + aH$, the commitment to the value a with mask x. Note that as long as $log_G H$ is unknown, and if $a \neq 0$, then $log_G C(a,x)$ is unknown. On the other hand, if $a = 0$, then $log_G C(a,x) = x$, so it is possible to sign with keypair $(x, C(0,x))$.

Maxwell's scheme includes input commitments $C_{in,i}$, output commitments $C_{out,j}$, corresponding, respectively, to amounts going into and out of a given transaction,[14] and the network verifies that

$$\sum_i C_{in,i} = \sum_j C_{out,j}.$$

However, this does not suffice in Monero: since a given transaction contains multiple possible inputs $P_i, i = 1, ..., n$, only one of which belong to the sender (see Section 4.4 of CryptoNote),[22] then if we are able to check the above equality, it must be possible for the network to see which P_i belongs to the sender of the transaction. This is undesirable, since it removes the anonymity provided by the ring signatures. Thus instead, commitments for the inputs and outputs are created as follows (suppose first that there is only one input)

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

$$C_{in} = x_c G + aH$$

$$C_{out,1} = y_1 G + b_1 H$$

$$C_{out,2} = y_2 G + b_2 H$$

such that $x_c = y_1 + y_2 + z$, $x_c - y_1 - y_2 = z$, y_i are mask values, $z > 0$ and $a = b_1 + b_2$. Here x_c is a special private key the "amount key" known only to the sender, and to the person who sent them their coins, and must be different than their usual private key. In this case,

$$C_{in} - \sum_{i=1}^{2} C_{out,i}$$

$$= x_c G + aH - y_1 G - b_1 H - y_2 G - b_2 H$$

$$= zG.$$

Thus, the above summation becomes a commitment to 0, with private key z, and public key zG, rather than an actual equation summing to zero. Note that z is not computable to the originator of x_c's coins, unless they know both of the y_1, y_2, but even this can be simply mitigated by including an additional change address (the usual case is that the second commitment, with y_2 as mask, is sent to yourself as change).

Since it is undesirable to show which input belongs to the sender, a ring signature consisting of all the commitment equations is created on the ring:

$$\left\{ C_{1,in} - \sum_j C_{j,out}, ..., C_{s,in} - \sum_j C_{j,out}, ..., C_{n,in} - \sum_j C_{j,out} \right\}.$$

This is a ring which can be signed since we know one of the private keys (namely $z + x'$ with z as above and $x'G = P_s$). A verifying signature on this ring proves that the amount into a transaction equals the amount going out of a transaction, however it does not provide linkability or prevent double-spending (since you could change $C_{j,out}$, resulting in a different key-image). Thus, instead of producing a signature only on this ring, the above ring will be used as a row in an MLSAG signature, which also includes the addresses corresponding to the input commitments $C_{i,in}$. Since the addresses correspond to public keys, which cannot be changed, and since they will be in the same column of the MLSAG, the addresses provide linkability. The full protocol is described in Section 4.1.

3.3. Range Proofs—As noted by Maxwell,[14] when replacing plaintext amounts with Pedersen Commitments, one must additionally prove that each output lies in a certain range of positive values. The reason for this, is that since the underlying mathematics for the commitments occurs over a finite field, a small negative amount is in the same equivalence class, modulo the field order, as a very large positive amount. Thus, a user could create a transaction to themselves where the inputs equal the outputs, and the outputs include a small negative (large positive) value, resulting in the creation of free coins. Thus it is necessary to include a "range proof" with each output, proving that each value lies in a given positive range. In this subsection, we recall how this is done by Maxwell.[14] Note that this range proof construction also uses a type of ring-signature,

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

which do not need linkability. These ring signatures are used for a completely different purpose than the MLSAG's (namely the range proof ring signatures are used to prove a single bit is in the set $\{0,1\}$ without giving away which number it actually is) and have no interaction with the MLSAG part of the Ring CT Protocol.

Given that there are 2^{64} atomic units of Monero currency, one could choose $[0,2^{64}]$ to be an acceptable range for the range proof, or alternatively, with the objective of having smaller range proofs, one could use a floating point implementation which would include an exponent and a mantissa in $[0,2^n]$ for some smaller value of n.

The standard method of constructing a range proof, described by Maxwell is to first write the output amount in it's n-ary expansion.[14] For example, in binary, an output amount b would be represented as:

$$b = b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + \cdots + b_r \cdot 2^r$$

with $b_i \in \{0,1\}$. In this case, a range proof is constructed for b by actually proving, for each $i \in \{0,...,r\}$ that $b_i \in \{0,1\}$, thus proving a has an expansion such as the above.

To do this, one writes the commitments, described in the previous section, as $C = aG + bH$ where a is a masking secret key, then one picks $a_0, a_1, ..., a_r$ with sum $\sum_{i=0}^{t} a_i = a$ and proves for each i that

$$C_i = a_i G + b_i 2^i H$$

is a commitment to either 0 or to 2^i, without revealing explicitly which one it is. This is easily accomplished by providing a ring signature on the ring

$$\{a_i G + b_i 2^i H, a_i G + b_i 2^i H - 2^i H\}.$$

If $b_i \notin \{0,1\}$, then neither of the keys in the above ring will be a commitment to zero, and the ring will not be signable. Note that this simple ring signature need not be linkable, and can therefore use the more well-known signature technique of Abe *et al.* rather than Liu *et al.*[2, 11] Finally, the prover provides all C_i and the verifier checks that $\sum_i C_i = C$. Various techniques can then be used to easily aggregate these signatures for each bit in order to achieve additional space savings (see for example Maxwell and Poelstra).[15] One can further implement blockchain pruning, where, after a certain number of blocks or at a checkpoint, these range proofs are not stored by nodes desiring to save additional disk space.

4. Ring CT For Monero Protocol

In this section, we describe the full construction of the Ring CT protocol which combines the MLSAG's of Section 2, the ring commitment scheme of Section 3.2, and the range proofs of Section 3.3.

4.1. Tag-Linkable Ring-CT with Multiple Inputs and One-time Keys—

4.1.1 Transaction Generation:

- Let $\left\{\left(P_\pi^1, C_\pi^1\right), ..., \left(P_\pi^m, C_\pi^m\right)\right\}$ be a collection of addresses / commitments with corresponding secret keys $x_j,\ j = 1,...,m$. Each pair (P_i, C_i), with $i \in \{1,...,m\}$, corresponds to an input public key P_i and commitment C_i in a given group which satisfies the discrete logarithm assumption.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

- Find $q+1$ collections $\left\{\left(P_i^1, C_i^1\right), ..., \left(P_i^m, C_i^m\right)\right\}, i = 1, ..., q+1$ which are not already tag linked in the sense of Fujisaki and Suzuki (page 6).[10] These will serve as additional inputs to the MLSAG to mask the actual input column.
- Decide on a set of output addresses $(Q_i, C_{i,out})$ such that $\sum_{j=1}^{m} C_\pi^j - \sum_i C_{i,out}$ is a commitment to zero.
- Let

$$\mathfrak{R} := \left\{ \left\{ \left(P_1^1, C_1^1\right), ..., \left(P_1^m, C_1^m\right), \left(\sum_{j=1}^{m} C_1^j - \sum_i C_{i,out}\right) \right\}, \right.$$
$$...,$$
$$\left. \left\{ \left(P_{q+1}^1, C_{q+1}^1\right), ..., \left(P_{q+1}^m, C_{q+1}^m\right), \left(\sum_{j=1}^{m} C_{q+1}^j - \sum_i C_{i,out}\right) \right\} \right\}.$$

  be the key matrix which we wish to sign. Note that the last column is a Ring-CT ring in the sense of 3.2.
- Compute the MLSAG signature Σ on $\mathfrak{R}$ with respect to message $\mathfrak{m}$ which is a cryptographically secure hash of the set of $(Q_i, C_{i,out})$
- Compute range proofs, as in Section 3.3, for each $C_{i,out}$.

In this case, by Theorem 6, P_π^j, $j = 1, ..., m$ cannot be the signer of any additional non-linked Ring Signatures in the given superset $\mathcal{P}$ of all such pairs $\mathcal{P} = \{(P, C)\}$ after signing Σ. Furthermore, by the property of Signer-ambiguity, which holds for the MLSAG by Theorem 8, the signer is anonymous to a degree proportional with the number of columns in their chosen key-matrix.

4.1.2 Transaction Verification

Given a Ring-CT transaction generated as above, the transaction can be verified by simply verifying each of the signatures

- Verify the MLSAG part of the transaction with respect to the given message consisting of outputs and commitments $(Q_i, C_{i,out})$ including rejecting the transaction as duplicate if the key-image part of the signature appears as part of a previously received transaction.
- Verify each of the range proofs for the $C_{i,out}$.

By Theorems 6 and 7, the likelihood of a double-spend or fake transaction being verified is negligible with respect to the chosen security parameter.

4.1.3 Storage Cost

Note that the size of the signature Σ on $\mathfrak{R}$ according to definition 4.1 is actually smaller, for $m > 1$, than a current CryptoNote ring signature based transaction which includes multiple inputs.[22] This is because of the size improvements, given by Liu *et al.* to each column.[11] Note also, it is probably not necessary to include the key-image of the commitment entry of the above signature. Further size optimizations are likely possible.

4.2. Conversion from Visible Denominations to Commitments—As Monero currently uses blockchain visible scalars to represent amounts, it is important that there is a way to convert from visible amounts to commitments while preserving anonymity. In fact, this is not difficult. Given a pair (P, a) where P is a public key and a represents an amount, this may be used as the input to a transaction as (P, aH), and it must be checked by the verifier that the input amount a multiplied

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

by the masking point H, indeed gives aH. Thus at the first step, the input amounts will not be hidden, but the outputs of this transaction can be hidden, and all the necessary relations outlined in Section 4 hold. Note that a range proof is not necessary for such an input. The obvious benefit of this method of converting from visible amounts to commitments is that the amount of coins generated by the mining process is trustlessly verifiable. This is an advantage of the Ring CT protocol over payment schemes such as Zerocash which rely on a trusted setup phase.[5]

4.3. Transaction Fees—As Monero is strongly decentralized (*i.e.* proof-of-work) it is necessary to pay miners a transaction fee for each transaction. This helps with the network security to prevent blockchain bloat. These fees must be paid "unmasked" *i.e.* just as bH, rather than $xG+bH$, and for some standardized amount b so that the miner can verify that $b \cdot H = bH$ and thus there is enough money for the transaction fee while still having the equations in terms of H so the necessary relations of Section 4 hold.

4.4. Ring Multisignature—Note that a simple, non-interactive version of a t of m of n ring-multisignature can be done with the MLSAG signatures in a Blockchain setting. (We give an interactive version with better anonymity properties in Noether *et al.*[21]) This allows a group of m participants to create a multisignature such that t out of m must sign for the signature to be accepted as valid, and in addition, it is signer ambiguous which m keys are the participants out of the n keys.

- The n participants in the multisignature create a shared secret x_e and public key P_e and share multisig key-images
$$I_j = x_e H_p(P_e|P_j)$$
with P_j the public key of the participant.
- Any participant of the multisignature selects $n-m$ additional public keys on the blockchain and creates an MLSAG signature with the first row the total n keys, and the second row having each entry the shared key P_e.
- Each signer transmitting part of the multisignature provides the initial $I_j,\ j = 1,...,m$ so that the signature is accepted after t verifying signatures have been transmitted on the blockchain, each corresponding to one of the key images I_j.

An expanded writeup of the ring-multisignature is in the works.

5. Conclusion

The Ring Confidential Transactions protocol provides a strongly decentralized cryptocurrency, having no priviledged party, and with provable security estimates regarding the hiding of amounts, origins and destinations. In addition, coin generation in the Ring Confidential Transactions protocol is trustless and verifiably secure. These properties are a necessity of any cash-like cryptocurrency such as Monero.

Acknowledgement

We would like to thank the Monero team for help and discussion in the creation of this paper and the Monero and Bitcoin Community for support and discussion.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

Conflict of Interest

With respect to disclosure, the first author received several donations totaling between 2 and 3 bitcoins from the Monero community in gratitude for his contribution to this research.

Notes and References

[1] Note that the terminology "group signature" in Liu *et al.*[11] is used in a slightly non-standard way, since it usually refers to a signature scheme which has a trusted manager, while these signatures do not require a trusted manager. Thus Liu et al.[11] have substituted "spontaneous group signature" where ring signature would normally appear in the literature.

[2] Abe, M., Ohkubo, M., Suzuki, K. "1-out-of-n signatures from a variety of keys." *Advances in Cryptology — Asiacrypt 2002* 415-432 (2002).

[3] Back, A. "Bitcoins with homomorphic value (validatable but encrypted)." *Bitcointalk* (2013) (accessed 1 May 2015) `https://bitcointalk.org/index.php?topic=305791.0`.

[4] Back, A. "Ring signature efficiency." *Bitcointalk* (2015) (accessed 1 May 2015) `https://bitcointalk.org/index.php?topic=972541.msg10619684#msg10619684`.

[5] Ben Sasson, E., *et al.* "Zerocash: Decentralized anonymous payments from bitcoin." In IEEE, *2014 IEEE Symposium on Security and Privacy (SP)* 459–474. (2014).

[6] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y. "High-speed high-security signatures." *Journal of Cryptographic Engineering* 2.2 77–89 (2012).

[7] Bissias, G., Ozisik,A. P., Levine, B. N., Liberatore, M. "Sybil-resistant mixing for bitcoin." In ACM, *Proceedings of the 13th Workshop on Privacy in the Electronic Society* 149–158 (2014).

[8] Duffield, E. and Hagan, K. "Darkcoin: Peertopeer cryptocurrency with anonymous blockchain transactions and an improved proof of work system." (2014).

[9] Fromknecht, C. "One-time zero sum ring signature." Github (2016) `https://github.com/cfromknecht/OZcoin/raw/master/whitepaper/zerosum.pdf`

[10] Fujisaki, E., Suzuki, K. "Traceable ring signature." In *Public Key Cryptography–PKC 2007* 181–200. Springer (2007).

[11] Liu, J. K., Wei, V. K., Wong, D. S. "Linkable spontaneous anonymous group signature for ad hoc groups." In *Information Security and Privacy* 325–335. Springer (2004).

[12] Mackenzie, A., Noether, S., M. C. Team. "Improving obfuscation in the cryptonote protocol." (2015) `https://lab.getmonero.org/pubs/MRL-0004.pdf`

[13] Maxwell, G. "Coinjoin: Bitcoin privacy for the real world, august 2013." *Bitcointalk* (2013) (accessed 1 July 2015) `https://bitcointalk.org/index.php?topic=279249.0`

[14] Maxwell, G., "Confidential Transactions." (2015) (accessed 1 June 2105) `https://people.xiph.org/~greg/confidential_values.txt`

[15] Maxwell, G., Poelstra, A. "Borromean ring signatures." (accessed 12 December 2016) `https://github.com/Blockstream/borromean_paper/raw/master/borromean_draft_0.01_34241bb.pdf`

[16] Nakamoto, S. "Bitcoin: A peer-to-peer electronic cash system." (2008).

[17] Noether, S. "Mininero." (2015) `https://github.com/ShenNoether/MiniNero`.

[18] Noether, S. "Ringct demo in c++." (2016) `https://github.com/ShenNoether/RingCT`

[19] Noether, S. "Ringct demo in python." (2016) `https://github.com/ShenNoether/RingCT-Python`

[20] Rivest, R. L., Shamir, A., Tauman, Y. "How to leak a secret." In *Advances in Cryptology — ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001* 552–565. Springer (2001).

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

[21] Noether, S., Mackenzie, A., M. C. Team. "Ring multisignature." (April 2016) https://shnoe.wordpress.com/2016/03/22/ring-multisignature/

[22] van Saberhagen, N. "Cryptonote v 2. 0." (2013) https://cryptonote.org/whitepaper.pdf

Appendix A: Security Proofs

1.1. MLSAG Unforgeability—This follows similarly to Theorem 1 of Liu *et al.*[11] We work in a cyclic group with basepoint G and satisfying the discrete logarithm assumption at security level k. Let H_s and H_p random oracles returning a scalar and group element respectively, and $\mathcal{SO}$ be a signing oracle which returns valid MLSAG signatures. Assume there is a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ with the ability to forge an MLSAG (*i.e.*, create a verifying MLSAG without knowing the signing key-vector) from a list of key vectors $\mathcal{L}$ with non-negligible probability

$$Pr\left(\mathcal{A}\left(\mathcal{L}\right) \rightarrow \left(m,\sigma\right) : VER\left(\mathcal{L},m,\sigma\right) = True\right) > \frac{1}{Q_1\left(k\right)}$$

where Q_1 is a polynomial inputting a security parameter k and where (m,σ) is not one of the signatures returned by $\mathcal{SO}$. Assume that $\mathcal{A}$ makes no more than $q_H + nq_S$ (with n the number of keys in $\mathcal{L}$) queries to the signing oracles H_s, H_p and $\mathcal{SO}$ respectively. The oracles H_s and H_p are assumed independent and random and are consistent given duplicate queries. The signing oracle $\mathcal{SO}$ is also allowed to query H_s and H_p. Given $\mathcal{A}$, we will show it is possible to create a PPT adversary $\mathcal{M}$ which uses $\mathcal{A}$ to find the discrete logarithm of one of the keys in $\mathcal{L}$.

If $\mathcal{L}$ is a set of key vectors $\{\overline{y_1},...,\overline{y_n}\}$ each of size r, (*i.e.* $\overline{y_i} = \left(y_1^i,...,y_r^i\right)$ with $y_1,...,y_r$ public keys) then a forged signature

$$\sigma = \left(c_1, s_1, ..., s_n, y_0\right)$$

produced by $\mathcal{A}$ must satisfy

$$c_{i+1} = H_s\left(\mathfrak{m}, L_i^1, R_i^1, ..., L_i^m, R_i^m\right)$$

where the i are taken mod n, and the L_i^j, R_i^j are defined as in Section 2.4. The new adversary $\mathcal{M}$ may call $\mathcal{A}$ to forge signatures a polynomial number of times and will record each Turing script $\mathcal{T}$ whether or not the forgery is successful.

Lemma 5. *Lemma 1 (from Liu et al.[11]) Let $\mathcal{M}$ invoke $\mathcal{A}$ to obtain a transcript $\mathcal{T}$. If $\mathcal{T}$ is successful, then $\mathcal{M}$ rewinds $\mathcal{T}$ to a header $\mathfrak{H}$ and re-simulates $\mathcal{A}$ to obtain transcript $\mathcal{T}'$. If $Pr\left(\mathcal{T}\ succeeds\right) = \varepsilon$, then $Pr\left(\mathcal{T}'\ succeeds\right) = \varepsilon$.*

Theorem 6. *Using the random oracle model, the probability that an adversary $\mathcal{A}$ forges a verifying MLSAG signature is negligible under the discrete logarithm assumption.*

Proof. We follow the notation introduced above. Similarly to Theorem 1 of Liu *et al.*,[11] since the probability of guessing the output of a random oracle is negligible, therefore, for each successful forgery $\mathcal{A}$ completes with transcript $\mathcal{T}$, there are $m_{\mathcal{T}}$ queries to H_s matching the n queries used to verify the signature. Thus let $X_{i_1},...,X_{i_m}$ denote these queries used in verification for the i^{th} such forgery and let π be the index corresponding to the last such verification query for a given forgery

$$X_{i_m} = H_s\left(m, L_{\pi-1}^1, R_{\pi-1}^1, ..., L_{\pi-1}^{m_{\mathcal{T}}}, R_{\pi-1}^{m_{\mathcal{T}}}\right).$$

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

(Intuitively, π corresponds to what would be the secret index of the forged signature, since it corresponds to the last call to the random oracle for the given signature).

An attempted forgery σ produced by $\mathcal{A}$ is an (ℓ,π)-forgery if $i_1 = \ell$ and π is as above (so this forgery corresponds to queries ℓ through $\ell+\pi$). By assumption, there exists a pair (ℓ,π) such that the probability that the corresponding transcript $\mathcal{T}$ gives a successful forgery, $\varepsilon_{\ell,\pi}(\mathcal{T})$, satisfies

$$\varepsilon_{\ell,\pi} \geq \frac{1}{m_{\mathcal{T}}(q_H + m_{\mathcal{T}} q_S)} \cdot \frac{1}{Q_1(k)} \geq \frac{1}{n(q_H + n q_S)} \cdot \frac{1}{Q_1(k)}.$$

Now, rewinding $\mathcal{T}$ to just before the ℓ^{th} query, and again attempting a forgery on the same set of keys, (and letting H_s compute new coin flips for all of it's succeeding queries) then by Lemma 5, it follows that the probability that $\mathcal{T}'$ is also a successful forgery satisfies

$$\varepsilon_{\ell,\pi}(\mathcal{T}') \geq \frac{1}{n(q_H + n q_S)} \cdot \frac{1}{Q_1(k)}.$$

Therefore, the probability that both $\mathcal{T}$ and $\mathcal{T}'$ correspond to verifying forgeries σ and σ' is non-negligible:

$$\varepsilon_{l,\pi}(\mathcal{T} \text{ and } \mathcal{T}') \geq \left(\varepsilon_{l,\pi}(\mathcal{T})\right)^2.$$

As new coin-flips have been computed for the random oracle outputs of H_s, it follows that with overwhelming probability, that for each j we have $s_\pi^j \neq s_\pi'^j$ and $c_\pi \neq c'_\pi$. Thus we can solve for any private key of index π:

$$x_\pi^j = \frac{s_\pi'^j - s_\pi^j}{c_\pi - c'_\pi} \; mod \; q$$

which contradicts the discrete logarithm assumption.

1.2. MLSAG Linkability—

Theorem 7. *(Key-Image Linkability) The probability that a PPT adversary $\mathcal{A}$ can create two verifying (and unlinked in the given setting) signatures σ, σ' signed with respect to key vectors $\overline{y}$ and $\overline{y}'$ respectively such that there exists a public key y in both $\overline{y}$ and $\overline{y}'$ is negligible.*

Proof. Suppose to the contrary that $\mathcal{A}$ has created two verifying signatures σ and σ' both signed with respect to key vectors $\overline{y}$ and $\overline{y}'$ respectively such that there exists a public key y in both $\overline{y}$ and $\overline{y}'$. Let y appear as element j of $\overline{y}$, and as element j' element of $\overline{y}'$. By Theorem 6, it holds with overwhelming probability that there exists indices π and π' for the public keys in σ and σ' respectively such that

$$L_\pi^j = s_\pi^j G + c_\pi y_\pi^j$$
$$R_\pi^j = s_\pi^j H_p\left(y_\pi^j\right) + c_\pi I_j$$

and

$$L_{\pi'}^{j'} = s_{\pi'}^{j'} G + c_{\pi'} y_{\pi'}^{j'}$$
$$R_{\pi'}^{j'} = s_{\pi'}^{j'} H_p\left(y_{\pi'}^{j'}\right) + c_{\pi'} I_{j'}$$

with

$$log_G L_\pi^j = log_{H_p\left(y_\pi^j\right)} R_\pi^j$$

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

and

$$log_G L_{\pi'}^{j'} = log_{H_p\left(y_{\pi'}^{j'}\right)} R_{\pi'}^{j'}$$

Letting x denote the private key of y, $y = xG$, then after solving the above for I_j and $I_{j'}$ it follows that $I_j = xH_p\left(y_\pi^j\right) = xH_p(y)$ and similarly $I_{j'} = xH_p(y)$. Thus the two signatures include $I_j = I_{j'}$, and therefore, since duplicate key images are rejected, one of them must not verify.

1.3. MLSAG Anonymity—To prove the anonymity of the above protocol in the random oracle model, let H_s, H_p be random oracles modeling discrete hash functions returning, as in the previous proofs, a scalar and group element. Let $\mathcal{A}$ be an adversary against anonymity. I construct an adversary $\mathcal{M}$ against the Decisional Diffie Helman (DDH) assumption as follows. The DDH assumption says that given a tuple $(G, aG, bG, \gamma G)$, the probability of determining whether $\gamma G = abG$ is negligible.

Theorem 8. *Ring CT protocol is signer-ambiguous under the Decisional Diffie-Helman assumption.*

Proof. (Similar proof to Theorem 2 of Liu *et al.*)[11] Assume that the Decisional Diffie-Helman problem is hard in the cyclic group generated by G and suppose there exists a PPT adversary $\mathcal{A}$ against signer ambiguity. Thus given a list $\mathcal{L}$ of n public key-vectors of length m, a set of t private keys $\mathcal{D}_t = \{x_1, ..., x_t\}$, a valid signature σ on $\mathcal{L}$ signed by a user with respect to a key-vector $\bar{y}$ such that the corresponding private key-vector $\bar{x} = (x_1^\pi, ..., x_m^\pi)$ satisfies $x_j^\pi \notin \mathcal{D}_t$, then $\mathcal{A}$ can decide π with probability

$$Pr(\mathcal{A} \to \pi) > \frac{1}{n-t} + \frac{1}{Q(k)}$$

for some polynomial $Q(k)$. We construct a PPT adversary $\mathcal{M}$ which takes as inputs a tuple (G, aG, bG, c_iG) where $i \in \{0,1\}$ is randomly chosen (and not a priori known to $\mathcal{M}$), $c_1 = ab$, and c_0 is a random scalar, and outputs i with probability

$$Pr(\mathcal{M}(G, aG, bG, c_iG) \to i) \geq \frac{1}{2} + \frac{1}{Q_2(k)}$$

for some polynomial $Q_2(k)$.

Consider an algorithm SIMNIZKP (similar to the one defined by Fujisaki and Suzuki[10]) which takes as input scalars a, c , a private key vector $\bar{x}$, a set of public key-vectors $\bar{y}_i, i = 1, ..., m$, an index π, and a message $\mathfrak{m}$ and acts on these as follows:

1. Generate random scalars $s_1, ..., s_m$ and, a random scalar $c_\pi \leftarrow H_s$.
2. For j indexing $\bar{x}$, set

$$L_\pi^1 = aG$$

$$R_\pi^1 = cG$$

and for all other j

$$L_\pi^j = s_\pi^j G + c_\pi y_\pi^j$$

$$R_\pi^j = s_\pi^j H_p\left(y_\pi^j\right) + c_\pi x^j H_p\left(y_\pi^j\right)$$

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34

3. Compute a random output from the random oracle

$$c_{\pi+1} \leftarrow H_s\left(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m\right).$$

4. For each i, working mod m, compute

$$L_i^j = s_i^j G + c_i y_\pi^j$$

$$R_i^j = s_i^j H_p\left(y_i^j\right) + c_i x^j H_p\left(y_\pi^j\right)$$

$$c_{i+1} \leftarrow H_s\left(\mathfrak{m}, L_i^1, R_i^1, ..., L_i^m + R_i^m\right).$$

and note that at the last step when $i = \pi - 1$, then c_{i+1} is already determined, to maintain consistency with the random oracle output.

Note that regardless of whether $\overline{x}$ is the actual private key corresponding to $\overline{y}$, due to the fact that consistency is maintained by the random oracles in subsequent calls, the above signature verifies. If $\overline{x}$ is actually the private key-vector of $\overline{y}$, then there is no difference between SIMNIZKP and an actual signature.

Finally, given a tuple (G, aG, bG, c_iG) where a, b are randomly selected scalars, with $c_1 = ab$, c_0 a random element, $i \in \{0,1\}$, $\mathcal{M}$ takes the following steps to solve the Decisional Diffie Helman Problem with non-negligible probability. $\mathcal{M}$ grabs a random $\gamma \leftarrow H_s$ from the random oracle and takes a private / public key-vector pair $(\overline{x}, \overline{y})$, and then computes s such that $a = s + \gamma x$. Now $\mathcal{M}$ performs SIMNIZKP with arbitrarily selected key-vectors $\{\overline{y_i}\}_{i=1,...,n}$ such that $\overline{y} = \overline{y}_\pi$, $a \to a$, $c_i \to c$ some message $\mathfrak{m}$, and $\overline{x} \to \overline{x}$.

If it is the case that $i = 1$, then $c = ab$, then

$$log_G aG = log_{bG} cG = a$$

and due to the fact that $\mathcal{A}$ is assumed to be able to find π with non-negligible probability, then there is a non-negligible probability over $\frac{1}{2}$ that $\mathcal{A}$ returns 1 (upon which $\mathcal{M}$ returns 1). If $i = 0$, then $\mathcal{A}$ returns 1 only with probability $\frac{1}{2}$, and so for some non-negligible probability over $\frac{1}{2}$, $\mathcal{M}$ returns the same value as $\mathcal{A}$, and thus solves the Decisional Diffie-Helman problem for randomly chosen scalars with non-negligible probability over $\frac{1}{2}$, which is a contradiction.

Appendix B: Ring CT Demo Code

In our repositories we have created a simple demonstration in C++, and Python respectively of the Ring Confidential Transactions protocol outlined in Section 4.1.[18, 19]

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.34