RING CONFIDENTIAL TRANSACTIONS

SHEN NOETHER MRL LABS

Abstract. This article introduces a method of hiding transaction amounts in the strongly decentralized anonymous cryptocurrency Monero. Similar to Bitcoin, Monero is cryptocurrency which is distributed through a proof of work "mining" process. 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 Multi-layed Linkable Spontaneous ad-hoc group signature is described which allows for hidden amounts, origins and destinations of transactions with reasonable efficiency. The previous draft of this article has been publicized in the Monero Community for a couple of months now and the new content here are some slightly updated proofs- see [Snoe] for earlier blockchain timestamped drafts. A longer expository version will follow.

1. Introduction

1.1. **Spontaneous (Ad Hoc) Ring Signatures in CryptoCurrencies.** 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 CryptoNote [CN] and Ring Coin [B2] the idea to use "Ring Signatures" originally described in [RST] as a "digital signature that specifies a group of possible signers such that the verifier can't tell which member actually produced the signature." The idea therefore is to have the origin pubkey of a transaction hidden in a group of pubkeys all which contain the same amount of coins, so that no one can tell which user actually sent the coins. The CryptoNote protocol as described in [CN] implements a slight modification of this to prevent double spends. Namely in [CN] a slight modification of the Traceable Ring signatures described in [FS] are employed. The benefit of using these "traceable" ring signatures, is that they do not allow a the owner of a coin to sign 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 a coin which has been spent twice. Ring coin [B2, B] uses a more efficient linkable ring signature which is a slight modification of the Linkable Spontaneous Anonymous Group signatures described in [LWW].

The main benefit of these type of ring signatures over, say coinjoin[GMc], is that it allows for "Spontaneous" mixing. In coinjoin, it is similarly possible to hide the originator of a given transaction, however these lack the ability to be sent at will, and in practice have to be sent to a centralized CoinJoin server where transactions are combined by a trusted party, so if that trusted party is compromised, then the anonymity of the transaction is comprimised. Some coins such as Dash asdf cite, attempt to negate this by making a larger number of trusted mixers (in this case 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 aka ad-hoc property) without relying on any trusted server, and thus providing for safer anonymity.

One possible attack against the cryptonote or ring-coin protocol [CN, B2] is blockchain analysis based on the amounts sent in a given transaction. 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 [CN] since the sender can include a number of change addresses in a transaction, thus obfuscating the amount which has been sent. 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.

Another downside to the CryptoNote set-up is that it requires a given pair of (P, A) of Pubkey and amount 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. This can have impacts to linkability property if lots of ring signatures are created using the same keys.

1.2. **Ring CT for Monero.** To negate both of the downsides to ring signature based cryptocurrency described in the last two paragraphs of the previous section, in this paper, I describe a modification to the

Monero protocol, a proof-of-work cryptocurrency based on CryptoNote which allows the amounts sent in a transaction to be hidden. This modification is based on the Confidential Transactions [GM] which are used on the lightning side-chain in bitcoin, except it allows for their use in ring signatures. Therefore, the modification is given the obvious name of Ring CT for Monero.

In order to preserve the property that coins cannot be double spent, a generalization of the LSAG's is described, a Multilayered Linkable Spontaneous Anonymous Group Signature (MLSAG) which allows for combining Confidential Transactions with a ring signature in a way such that multiple inputs and outputs are possible, anonymity is preserved, and double-spending is prevented.

1.3. Strongly Decentralized Anonymous Payment Schemes.

The Ring CT protocol allows hidden amounts, origins, and destinations for transactions which is somewhat similar to Zerocash [Z]. One possible differentiator is the use of proof of work for coin generation is possible with Ring CT as opposed to in ZeroCash, it seems all coins must be pregenerated by a trusted group.

Note that one of the biggest innovations in Bitcoin [SN], 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.

One final obvious benefit of the proof-of-work coin generation is it makes Ring CT immune to a powerful actor somehow aquiring all the pieces of the master key used in coin generation. Since there is an obvious large incentive (the ability to generate free money and unmask all senders of transactions) to aquire all pieces of the trusted generation key, this is fairly important. Especially since if a powerful actor had access to the generation key of a DAP scheme, then it seems it would negate the anonymity and privacy purposes of the currency.

1.4. **Acknowledgements.** I would like to thank Monero team for lots of help and discussion in the creation of this paper and the Monero and Bitcoin Community for support and discussion.

2. Multilayered Linkable Spontaneous Ad-Hoc Group Signatures

2.1. Informal def.

In this section, I define the Multilayered Linkable Spontaneous ad-hoc group signatures (MLSAG) used by the the Ring CT protocol. Note that I define these as a general signature, and not necessarily in their use case for ring CT. Note that a MLSAG is essentially similar to the LSAG's described in [LWW], but rather than having a ring signature on a set of n keys, instead, an MLSAG is a ring signature on a set of n key-vectors.

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

2.2. **LWW signatures vs FS signatures.** The ring signatures used in Monero and the CryptoNote protocol are derived from the traceable ring signatures of [FS]. The [CN] ring signatures come with a "key-image" which means that a signer can only sign one ring on the blockchain with a given public and private key pair. Because of this, one-time keys are used in CryptoNote, which further helps anonymity.

The [CN] ring signatures (asdf basic description)

In [B], Adam Back noticed that the Linkable Spontaneous Ad Hoc Group signatures of [LWW] can be modified to give a more efficient linkable ring signature producing the same effect as the [FS] ring signatures. This modification reduces the storage cost on the blockchain essentially in half.

First I recall the modification given in [B]:

Keygen: Find a number of public keys $P_i, i = 0, 1, ..., n$ and a secret index j such that $xG = P_j$ where G is the ed25519 basepoint and x is the signers spend key. Let $I = xH(P_j)$ where H is a hash function returning a point (in practice $toPoint(Keccak(P_k))$). Let m be a given message.

SIGN: Let $\alpha, s_i,\ i \neq j,\ i \in \{1, ..., n\}$ be random values in $\mathbb{Z}_q$ (the ed25519 base field).

Compute

$$L_j = \alpha G$$

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

$$c_{j+1} = h(m, L_j, R_j)$$

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

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

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

$$c_{j+2} = h(m,\ L_{j+1},\ R_{j+1})$$

$$\cdots$$

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

$$R_{j-1} = s_{j-1}H\left(P_{j-1}\right) + c_{j-1} \cdot I$$
$$c_j = h\left(m, L_{j-1},\ R_{j-1}\right)$$

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

Let $s_j = \alpha - c_j \cdot x \ mod \ l$, ($l$ being the ed25519 curve order) hence $\alpha = s_j + c_j x \ mod \ l$ so that

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

$$R_j = \alpha H\left(P_j\right) = s_j H\left(P_j\right) + c_j I$$

and

$$c_{j+1} = h\left(m,\ L_j,\ R_j\right)$$

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

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

which represents a space savings over [CN, 4.4].

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

$$c_{i+1} = h\left(m, L_i, R_i\right)$$

for all i.

LINK: Signatures with duplicate key images I are rejected.

Note that essentially proofs of unforgeability, anonymity, and linkability hold for the above protocol which are essentially similar to the proofs given in [LWW]. (I will give a more generalized version of these proofs for the MLSAG's).

2.3. **MLSAG Description.** For use with the Ring CT protocol which will be described in section 4, I require a generalization of the Back LSAG [B] modification which allows for key-vectors (definition 1) rather than just keys. This signature is described as follows

Now suppose that each signer of a (generalized) ring containing n members has exactly m keys $\left\{P_i^j\right\}_{j=1,...,m}^{i=1,...,n}$. The intent of the MLSAG ring signature is the following:

- Exactly one of the n signers has given a signature on all m of their keys.
- If the signer uses any one of their m keys in another MLSAG signature, then the two rings are linked.

The algorithm proceeds as follows: Let $\mathfrak{m}$ be a given message. Let π be a secret index corresponding to the signer of the generalized ring. For $j = 1, ..., m$, let $I_j = x_j H(P_\pi^j)$, and 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. Now, in an analogous manner to section the standard LSAG signature, define

$$L_\pi^j = \alpha_j G$$
$$R_\pi^j = \alpha_j H\left(P_\pi^j\right)$$

for a random scalar α and $j = 1, ..., m$. Now, again analogously to section **??**, set:

$$c_{\pi+1} = H\left(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m + R_\pi^m\right).$$
$$L_{\pi+1}^j = s_{\pi+1}^j G + c_{\pi+1} P_{\pi+1}^j$$
$$R_{\pi+1}^j = s_{\pi+1}^j H\left(P_{\pi+1}^j\right) + c_{\pi+1} I_j$$

and repeat this incrementing i modulo n until we arrive at

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

$$R_{\pi-1}^j = s_{i-1}^j H\left(P_{i-1}^j\right) + c_{i-1} \cdot I_j$$
$$c_\pi = H\left(\mathfrak{m}, L_{\pi-1}^1, R_{\pi-1}^1, ..., L_{\pi-1}^m + R_{\pi-1}^m\right).$$

Finally, solve for each s_π^j using $\alpha_j = s_\pi^j + c_\pi x_j \ mod \ q$. Now the signature is given as $(I_1, ..., I_m, c_1, s_1^1, ..., s_1^m, s_2^1, ..., s_2^m, ..., s_n^1, ..., s_n^m)$, so the complexity is $O(m(n+1))$. Now verification proceeds by regenerating all the L_i^j, R_i^j starting from $i = 1$ as in section 2.2(where $m = 1$) 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 rejected). One can easily show, in a manner similar to [LWW]:

- The probability of a signer generating a valid signature without knowing all "m" private keys 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.

I expand on these points below.

2.4. **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 in [LWW].

Definition 2. (Unforgeability) An MLSAG signature scheme is unforgeable if for any PPT algorithm $\mathcal{A}$ with signing oracle $\mathcal{SO}$ producing valid signatures, then for a list of n public key vectors chosen by $\mathcal{A}$, then $\mathcal{A}$ can only with negligible probability produce a valid signature when $\mathcal{A}$ does not know one of the corresponding private key vectors.

Remark 3. In the following definition, note that I include rejecting duplicate key images as part of the verification criteria for the MLSAG, which is slightly different than described in [LWW].

Definition 4. (Linkability) Let L be the set of all public keys in a given . An MLSAG signature scheme on L is key-image linked if the probability of a PPT adversary $\mathcal{A}$ creating two signatures σ, σ' signed with respect to key-vectors $\overline{y}$ and $\overline{y}'$ each containing the same public key $y_i = y_i'$ in L and each verifying, is negligible.

Definition 5. (Signer Ambiguity) An MLSAG signature scheme is said to be signer ambiguous if given any verifying signature σ on key-vectors $(\overline{y}_1, ..., \overline{y}_n)$ and any set of t private keys, none of the same index nor of the secret index, then the probability of guessing the secret key is less than $\frac{1}{n-t} + \frac{1}{Q(k)}$

2.5. **MLSAG Unforgeability.** This follows similarly to [LWW, Theorem 1]. Let H_1 and H_2 random oracles, 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 from a list of key vectors L with non-negligible probability

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

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_1, H_2 and $\mathcal{SO}$ respectively. The oracles H_1 and H_2 are assumed independent and random and are consistent given duplicate queries. The signing oracle $\mathcal{SO}$ is also allowed to query H_1 and H_2. Given $\mathcal{A}$, I 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 L is a set of key vectors $\{\overline{y_1}, ..., \overline{y_n}\}$ each of size r, (i.e. $\overline{y_i} = (y_1^i, ..., y_r^i)$ with $y_1, ..., y_r$ public keys) then a forged signature

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

produced by $\mathcal{A}$ must satisfy

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

where the i are taken mod n, and the L_i^j, R_i^j are defined as in section 2.3 The new adversary$\mathcal{M}$ may call $\mathcal{A}$ to forge signatures a polynomial number of times and will record each turing transcript $\mathcal{T}$ whether or not the forgery is succesful.

Lemma 6. .[LWW, Lemma 1] *Let $\mathcal{M}$ invoke $\mathcal{A}$ to obtain a transcript $\mathcal{T}$. If $\mathcal{T}$ is succesful, then $\mathcal{M}$ rewinds $\mathcal{T}$ to a header H and resimulates $\mathcal{A}$ to obtain transcript $\mathcal{T}'$. If $Pr\left(\mathcal{T}\text{ succeeds}\right) = \epsilon$, then $Pr\left(\mathcal{T}'\text{ succeeds}\right) = \epsilon$.*

Proof. Follows easily from the cited Theorem. □

Theorem 7. *The probability of an adversary $\mathcal{A}$ forges a verifying MLSAG signature is negligible under the discrete logarithm assumption.*

Proof. I follow the notation introduced above. Similarly to [LWW, Theorem 1], since the probability of guessing the output of a random oracle is negligible, therefore, for each succesful forgery $\mathcal{A}$ completes with transcript $\mathcal{T}$, there are $m_{\mathcal{T}}$ queries to H_1 matching the n queries used to verify the signature. Thus let 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 verifcation query for a given forgery

$$X_{i_m} = H_1\left(m, L_{\pi-1}^1, R_{\pi-1}^1, ..., L_{\pi-1}^{m_{\mathcal{T}}}, R_{\pi-1}^{m_{\mathcal{T}}}\right).$$

(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 succesful forgery $\epsilon_{\ell,\pi}(\mathcal{T})$ satisfies

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

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_1 compute new coin flips for all of it's suceeding queries) then by Lemma 6, it follows that the probability that $\mathcal{T}'$is also a succesful forgery satisfies

$$\epsilon_{\ell,\pi}\left(\mathcal{T}'\right) \geq \frac{1}{n\left(q_H + nq_S\right)} \cdot \frac{1}{Q_1\left(k\right)}.$$

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

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

As new coin-flips have been computed for the random oracle outputs of H_1, it follows that with overwhelming probability there is j such that $s_\pi^j \neq s_\pi'^j$ and $c_\pi \neq c_{\pi+1}$. Thus we can solve for the 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. □

2.6. **MLSAG Linkability.** In order to show that MLSAG's are linkability according to definition 4

Theorem 8. *(Key-Image Linkability) The probability that a PPT adversary $\mathcal{A}$ can create two verifying 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, 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 7, it holds with overwhelming probability that there exists indices π and π' for the public keys in σ and σ' respectively such that

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

and

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

with

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

and

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

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

2.7. **MLSAG Anonymity.** To prove the anonymity of the above protocol in the random oracle model, let H_1, H_2 be random oracles modeling discrete hash functions. Let $\mathcal{A}$ be an adversary against anonymity. I construct an adversary $\mathcal{M}$ against Decisional Diffie Helman (DDH) assumption assumption as follows. Recall that a DDH triple is a tuple of group elements (A, B, C, D) such that $log_A C = log_B D$ the DDH asumption says that given a tuple $(G, aG, bG, \gamma G)$, the probability of determining whether $\gamma G = abG$ is negligible.

Theorem 9. *Ring CT protocol is signer-ambiguous under the Decisional Diffie-Helman assumption.*

Proof. (Similar proof to [LWW, Theorem 2]) Suppose there exists a PPT adversary $\mathcal{A}$ against signer ambiguity. Thus given a list 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 L signed by a user with respect to a key-vector $\overline{y}$ such that the corresponding private key-vector $\overline{x} = (x_1^\pi, ..., x_m^\pi)$ satisfies $x_j^\pi \notin \mathcal{D}_t$, then $\mathcal{A}$ can decide π with probability

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

for some polynomial $Q\left(k\right)$. I construct a PPT adversary $\mathcal{M}$ which takes as inputs a triple $\left(aG, bG, c_i G\right)$, with $c_1 = ab$, c_0 a random element, $i \in \{0, 1\}$ random, and gives as outputs 1 if $c = ab$ and 0 otherwise and such that

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

for some polynomial $Q_2\left(k\right)$.

Consider an algorithm SIMNIZKP (similar to [FS]) which takes as input scalars a, c , a private key vector $\overline{x}$, a set of public key-vectors $\overline{y}_i, i = 1, ..., m$, 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 $c_\pi \leftarrow H$.
2. For j indexing $\overline{x}$, set

$$L_\pi^1 = aG$$
$$R_\pi^1 = cG$$

and for all other j

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

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

3. Compute a random output from the random oracle

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

4. For each i, working mod m, compute

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

$$R_i^j = s_i^j H\left(y_i^j\right) + c_i x^j H\left(y_i^j\right)$$

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

and just 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 triple (aG, bG, c_iG) where a, b are randomly selected scalars, with $c_1 = ab$, c_0 a random element, $i \in \{0, 1\}$ to solve the Decisional Diffie Helman Problem with non-negligible probability, $\mathcal{M}$ grabs a random $\gamma \leftarrow H$ 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 contradicts the DDH assumption. □

3. Background on Confidential Transactions

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

3.2. **Modification for ring signatures Intuitive Explanation.** Let G be the ed25519 basepoint. Let[1]

$$H = toPoint\left(cn\_fast\_hash\left(123456 \cdot G\right)\right)$$

Note on the choice of scalar 123456. In the curve group of ed25519, not every cn_fast_hash is itself a point in the group of the basepoint G. The scalar 123456 is chosen so that the hash is a point in the group of the basepoint (i.e. $H = \psi G$ for some unknown ψ), so that all the usual elliptic curve math holds. Under the discrete logarithm assumption on ed25519, the probability of finding an x such that $xG = H$ is negligible.

Remark 10. (Choice of H) Only 1 out of 8 hashes lead to points in the group of the ed25519 basepoint used by Monero. The author tried $H\left(12G\right), H\left(123G\right), H\left(1234G\right), ...$ and finally arrived at $H\left(123456G\right)$ in the group of the basepoint. Note that this is contrary to secp.. where every hash results in a point, so in Bitcoin $H = toPoint\left(G\right)$ asdf check if $toPoint\left(G\right)$ is in the basepoint group, and double check all these

Define $C\left(a, x\right) = xG + aH$, 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 $log_G C\left(a, x\right)$ is unknown. On the other hand, if $a = 0$, then $log_G C\left(a, x\right) = x$, so it is possible to sign with sk-pk keypair $\left(x, C\left(0, x\right)\right)$.

In [GM], there are input commitments, output commitments, and the network checks that

$$\sum Inputs = \sum Outputs.$$

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 [CN, 4.4]), 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,

[1] $H = MiniNero.getHForCT()$ in terms of the code at [Snoe]

commitments for the inputs and outputs are created as follows (suppose first that there is only one input)

$$\begin{aligned} C_{in} &= x_c G + aH \\ C_{out-1} &= y_1 G + b_1 H \\ C_{out-2} &= y_2 G + b_2 H \end{aligned}$$

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,

$$\begin{gathered} C_{in} - \sum_{i=1}^{2} C_{out-i} \\ = x_c G + aH - y_1 G - b_1 H - y_2 G - b_2 H \\ = zG. \end{gathered}$$

Thus, the above summation becomes a commitment to 0, with $sk = z$, and $pk = zG$, 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 then they are receiving the coins, and presumably remember which pubkey they sent them to originally, and so there is no additional unmasking.

Since it is undesirable to show which input belongs to the sender, a ring signature consisting of all the input commitments $C_i, i = 1, ..., s, ..., n$ (where s is the secret index of the commitment of the sender), adding the corresponding pubkey (so commitments and pubkeys are paired (C_i, P_i) only being allowed to be spent together) and subtracting $\sum C_{out}$ is created:

$$\left\{ P_1 + C_{1,in} - \sum_j C_{j,out}, ..., P_s + C_{s,in} - \sum_j C_{j,out}, ..., P_n + C_{n,in} - \sum_j C_{j,out} \right\}.$$

This is a ring signature which can be signed since we know one of the private keys (namely $z + x'$ with z as above and $x'G = P_s$). In fact, since we know, for each i both the private key for P_i and the private key for $P_i + C_{i,in} - \sum_j C_{j,out}$ we can perform a signature as in section 2.3.

As noted in [GM], it is important to prove that the output amounts [2] $b_1, ...b_n$ all lie in a range of positive values, e.g. $(0, 2^{16})$. This can be accomplished essentially the same way as in [GM]:

[2]since input commitments could potentially be just inherited from the previous transaction, it suffices to consider the output amounts

- Prove first $C^{(j)}_{out-i} \in \{0, 2^j\}$ for all $j \in \{0, 1, ..., 16\}$. This is done as in [GM]: for example, $C^0_{out-i} = y^0_i G + b^0_i H$ where $b^0_i \in \{0, 1\}$. Let
$$C'^0_{out-i} = C^0_{out-i} - H = y_i G + b^0_i H - H$$
so that if $b^0_i = 0$, then $C'^0_{out} = y_i G$ and if $b^0_i = 1$, then $C^0_{out} = y_i G$, and in either case, the ring signature on$\{C^0_{out}, C'^0_{out}\}$ can be signed for.
  - Note that $\sum_j y^j_i = y_i$
- By carefully choosing the blinding values for each j, ensure that
$$\sum_{j=1}^{16} C^{(j)}_{out-i} = C_{out-i}.$$
- By homomorphicity of the commitments, $b_i = \sum_j \delta_{ji} 2^j$, where δ_{ji} is the j^{th} digit in the binary expansion of b_i.

Thus in total, by the above, the sum of inputs into a transaction equals the outputs, yet the specific input (and it's index!) is hidden. In addition, the outputs are positive values.

4. Ring CT For Monero Protocol

4.1. **Protocol Description.**

Definition 11. (Tag-Linkable Ring-CT with Multiple Inputs and One-time Keys)

- Let $\{(P^1_\pi, C^1_\pi), ..., (P^m_\pi, C^m_\pi)\}$ be a collection of addresses / commitments with corresponding secret keys $x_j,\ j = 1, ..., m$.
- Find $q + 1$ collections $\{(P^1_i, C^1_i), ..., (P^m_i, C^m_i)\}, i = 1, ..., q + 1$ which are not already tag linked in the sense of [FS, page 6].
- Decide on a set of output addresses $(Q_i, C_{i,out})$ such that $\sum_{j=1}^m C^j_\pi - \sum_i C_{i,out}$ is a commitment to zero.
- Let
$$\mathfrak{R} := \left\{ \left\{ \left(P^1_1, C^1_1\right), ..., \left(P^m_1, C^m_1\right), \left(\sum_j P^j_1 + \sum_{j=1}^m C^j_1 - \sum_i C_{i,out}\right) \right\}, \right.$$
$$...,$$
$$\left. \left\{ \left(P^1_{q+1}, C^1_{q+1}\right), ..., \left(P^m_{q+1}, C^m_{q+1}\right), \left(\sum_j P^j_{q+1} + \sum_{j=1}^m C^j_{q+1} - \sum_i C_{i,out}\right) \right\} \right\}.$$
be the generalized ring which we wish to sign. Note that the last column is a Ring-CT ring in the sense of section 4.
- Compute the MLSAG signature Σ on $\mathfrak{R}$.

In this case, $P_\pi^j, j = 1, ..., m$ 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.

Remark 12. Space complexity of the above protocol. Note that the size of the signature Σ on $\mathfrak{R}$ according to definition 11 is actually smaller, for $m > 1$ than a current CryptoNote [CN] ring signature based transaction which includes multiple inputs. This is because of the size improvements, given by [LWW], to each column. 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.

References

[B] Back, Adam. Bitcointalk post on modification to LWW signatures. https://bitcointalk.org/index.php?topic=972541.msg10619684#msg10619684

[B2] Back, Adam. Bitcointalk post on “Ring Coin.” https://bitcointalk.org/index.php?topic=305791.0

[CN] van Saberhagen, Nicolas. "Cryptonote v 2. 0." HYPERLINK “https://cryptonote. org/whitepaper. pdf” https://cryptonote. org/whitepaper. pdf (2013).

[FS] Fujisaki, Eiichiro, and Koutarou Suzuki. "Traceable ring signature." Public Key Cryptography–PKC 2007. Springer Berlin Heidelberg, 2007. 181-200.

[GMc] Maxwell, Greg. "CoinJoin: Bitcoin privacy for the real world, August 2013." Bitcoin Forum. URL: https://bitcointalk. org/index. php.

[GM] Maxwell, Gregory. “Confidential Transactions.” `https://people.xiph.org/~greg/confidential_values.txt`

[LWW] Liu, Joseph K., Victor K. Wei, and Duncan S. Wong. "Linkable spontaneous anonymous group signature for ad hoc groups." Information Security and Privacy. Springer Berlin Heidelberg, 2004.

[RST] Rivest, Ronald L., Adi Shamir, and Yael Tauman. "How to leak a secret." Advances in Cryptology—ASIACRYPT 2001. Springer Berlin Heidelberg, 2001. 552-565.

[Snoe] Noether, Shen. MiniNero, (2015), GitHub repository, `https://github.com/ShenNoether/MiniNero`

[SN] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Consulted 1.2012 (2008): 28.

[Z] Ben Sasson, Eli, et al. "Zerocash: Decentralized anonymous payments from Bitcoin." Security and Privacy (SP), 2014 IEEE Symposium on. IEEE, 2014.