RESEARCH BULLETIN MRL-9999

Threshold Ring Confidential Transactions

September 29, 2017

Brandon Goodell* and Sarang Noether

*Correspondence:
bggoode@g.clemson.edu
Monero Research Lab

Abstract

This research bulletin extends [3] by constructing a t-of-n threshold multi-layered linkable spontaneous anonymous group signature scheme (t-of-n MLSAG) in the same style as the LSAG schemes put forth by [2].

1 Introduction and Background

Ring signatures can play a critical role in establishing user anonymity (or at least user ambiguity) during message authentication. Due to this utility, ring signatures enjoy application in many cryptocurrency protocols. Multisignatures play a critical role in off-chain transactions for cryptocurrencies (e.g. the Bitcoin Lightning Network) and for message authentication in general (e.g. multi-factor authentication). It is natural to extend the notion of ring signatures to ring multisignatures for implementation in cryptocurrencies to enjoy signer-ambiguous multisignatures. We investigate t-of-N threshold multisignatures. A multisignature scheme is a t-of-N *ring threshold multisignature* (RTM) scheme if any set of N keys may be specified as a coalition of signers and assigned a shared public key such that any t coalition members may collaborate to fashion a signature on behalf of the shared public key. A usual digital signature scheme is a 1-of-1 multisignature scheme, so we can regard all keys as shared public keys (just perhaps with a coalition of only one member).

If the number of users cooperating in the construction of a signature is not secret, naive multisignature schemes can be constructed from any signature scheme (ring signature or otherwise) by simply requiring each participating user to present a separate signature. More sophisticated implementations combine these together using boolean AND circuits as in the Borromean ring set-up described in [] for efficiency reasons.

If a user does not desire to reveal to an adversary how many devices were used for some multi-factor authentication, it should be difficult for an adversary to determine the size of a coalition behind some shared public key. This property should be satisfied even if the adversary can persuade the party (or parties) controlling the shared public key to sign arbitrary messages chosen by the adversary. We introduce the security definition of *coalition-indistinguishable* multisignature schemes against adaptive chosen message attacks: given a shared t-of-N public key X, an adversary prompts the coalition in control of X to produce signatures on a set of messages and should be unable to guess any information about t or N.

Note that this property cannot be satisfied if the adversary has corrupted any public key in the coalition. If some coalition member for X, say X', is a shared t'-of-N' public key, this means that if the adversary has corrupted t' or more members of the coalition for X', then coalition-indistinguishability is impossible. Thus we investigate the notion of *subthreshold corruption oracle access*.

Note that a 1-of-N threshold signature scheme may be accomplished by simply handing out an identical set of keys to N individuals: whoever decides to use them first will be able to fashion a signature. We consider this a degenerate case. On the other hand, a 1-of-1 signature scheme is a usual signature scheme, as we mentioned.

1.1 Our Contribution

1.2 Notation and Prerequisites

2 MLSAG Ring Signatures

We briefly describe LSAG ring signatures in the sense of [2] and their MLSAG variant used in Monero. Let Q denote the set of ring members (public keys) $Q = \{Y_1, \dots Y_L\}$. Let k be a distinguished index in this set corresponding to the signing key Y_k. Let $y_k \in \mathbb{Z}_q$ such that $Y_k = y_k G$. Let m be a message. A signer computes a key image J based on the private key y_k.

For each i, the signer computes an elliptic curve point from the i^{th} ring member public key as $H_i := H_p(Y_i)$. The signer selects a random secret $u \in_R \mathbb{Z}_q$, computes an initial temporary pair of points uG and uH_k, and computes an initial commitment $c_{k+1} := H_p(m, uG, uH_k)$. The signer sequentially proceeds through indices $i = k+1, k+2, \dots, n, 1, 2, \dots, k-1$ in the following way. The signer selects at random secret $s_i \in_R \mathbb{Z}_q$, computes the next temporary pair of points $s_i G + c_i Y_i$ and $s_i H_i + c_i J$, and computes the next commitment $c_{i+1} := H_p(m, s_i G + c_i Y_i, s_i H_i + c_i J)$. The signer continues proceeding through the set of keys until commitments c_i have been computed for each $i = 1, ...L$. The signer then computes $s_k := u - c_k y_k$ and publishes the signature $(c_1, s_1, \dots, s_n, J)$.

A signature $(c_1^*, s_1^*, \dots, s_n^*, J^*)$ on m can be verified to have been generated by at least one member of the ring's private key in the following way: for each $i = 1, 2, \dots, L$, the verifier computes $z_i = s_i^* G + c_i^* Y_i$ and $z_i' = s_i^* H_i + c_i^* I^*$ and uses these to compute the $(i+1)^{th}$ commitment $c_{i+1} = H_{\mathcal{M},\mathcal{G}}(m, z_i, z_i')$. After computing $c_2, c_3, \dots, c_L, c_1$, the verifier approves of the signature if and only if $c_1 = c_1^*$.

The MLSAG generalization, where user keys are represented by vectors, is straightforward. For a user with secret key $\underline{y} = (y_1, \dots, y_w)$, each component of this key vector is used to generate a temporary pair of points (like uG, uH_k in the first step of constructing a ring signature, or $s_i G + c_i Y_i$, $s_i H_i + c_i J$). This provides the associated commitment

$$c_{i+1} := H_{\mathcal{M},\mathcal{G}}\left(m, \{(s_{i,j} G + c_i Y_{i,j}, s_{i,j} h_{i,j} + c_i I)\}_{j=1}^{w}\right).$$

2.1 Extending to Threshold Signatures

Generally, we wish to allow a coalition of users with the public keys $X = \{X_1, X_2, \dots, X_N\}$ (where each $X_i = x_i G$) to collaboratively fashion a shared public key Y (which we call a t-of-N *shared public key*) such that any subset of at least t

of the users in X can collaborate to fashion a signature on a message with corresponding public key Y. In the N-of-N case, one such implementation works in the following manner.

Given message m, a coalition of friends with the public keys $X = \{X_1, X_2, \ldots, X_N\}$ compute a shared public key $Y_{\texttt{shared}} := \sum_{j=1}^{n} X_j$, which is published. They select their ring to be a set of public keys $Q = \{Y_1, Y_2, \ldots, Y_L\}$ such that $Y = Y_k$ for some secret index s. Each user can compute $H_i = H_p(Y_i)$ for each $Y_i \in Q$. For each $1 \leq j \leq N$, the j^{th} signer (who owns the secret key x_j) computes a partial key image $I_j = x_j H_k$, picks a random initial temporary secret u_j, and shares the triple of points $I_j, u_j G, u_j H_k$ with the other signers. Now any of these users may compute the key image $J = \sum_j I_j$, the random points $u_k G = \sum_j u_j G$ and $u_k H_k = \sum_j u_j H_k$.

Using these, any of these users may compute the first in the sequence of commitments $c_{k+1} := H_p(m, u_k G, u_k H_k)$. The group decides upon random values $s_{k+1}, \ldots, s_L, s_1, \ldots, s_{k-1}$ and computes each $c_{i+1} := H_p(m, s_i G + c_i Y_i, s_i H_i + c_i J)$ for $i = k+1, \ldots, k-1$. All threshold members then use c_k to compute their $s_{k,j} = u_j - c_k x_j$. The signers share their $s_{k,j}$ with the other signers. Any threshold member may then compute the value $s_k = \sum_j s_{k,j}$ and publish the signature $(c_1, s_1, \ldots, s_L, J)$. Any user may verify this signature corresponds to the N-of-N shared public key Y.

The above set-up extends naturally to an $(N-1)$-of-N set-up. As before, a set of N public keys $\{X_1, X_2, \ldots, X_N\}$ form a coalition. Each pair of users has a shared secret scalar $z_{i,j} = H_s(x_i X_j)$ with associated public point $Z_{ij} = z_{i,j} G$. There are $\frac{N(N-1)}{2}$ such pairs; if any $N-1$ members get together, all of the associated shared secrets are known. Hence, we may simply instantiate the $(N-1)$-of-N threshold as an N^*-of-N^* set-up with $N^* = \frac{N(N-1)}{2}$, wherein all values $z_{i,j}$ are necessary to fashion a signature with the public key $Y_{\texttt{shared}} := \sum_{1 \leq i \leq n} \sum_{i < j \leq n} Z_{i,j}$.

This implementation may not satisfy very strong security definitions. For example, in the above description, an adversary with knowledge of the underlying public keys X_i can trivially test whether a certain public key Y in some ring Q is a multisignature key by simply computing $\sum_i X_i$. We discuss this and other properties in greater detail.

3 Security Models

In this section we present definitions we use later on. We immediately make use of the following observation: in a threshold multisignature scheme, coalitions of users collaborate to generate new threshold keys from sets of old keys and to generate signatures on messages using those threshold keys. This collaboration uses secret information each coalition member wishes to keep private from the rest of the coalition (and certainly any adversary outside of the coalition). In order to make rigorous any notion of security in an environment where threshold keys can be used to create threshold keys, we introduce the idea of the *depth* of a threshold public key.

We say a usual public key Y (as in a usual ring signature scheme) has *depth* $\texttt{depth}(Y) := 0$. For a threshold public key Y, we denote the set of public keys used to compute threshold ring signatures on behalf of Y as $\texttt{pub}(Y)$. We say that a t-of-N threshold public key Y has $\texttt{depth}(Y) := 1$ if each of the N elements in $\texttt{pub}(Y)$

have depth 0. For any set of public keys S, we define $\texttt{pub}\,(S) = \cup_{Y\in S}\texttt{pub}(Y)$. In the case that some public key in $\texttt{pub}(Y)$ is, itself, a threshold public key, we abuse operator notation to denote $\texttt{pub}\circ\texttt{pub}(Y) = \texttt{pub}^2(Y)$, and so on. For consistency, we define $\texttt{pub}^0(Y) = \{Y\}$. Hence, we iteratively define depth in the following way. We say that a threshold public key Y has $\texttt{depth}(Y) := \max_{Y'\in\texttt{pub}(Y)}\{\texttt{depth}(Y')+1\}$. Define the set $\tilde{Y} := \cup_{i=0}^{d_Y}\texttt{pub}^i(Y)$. Note that each threshold public key $Y' \in \texttt{pub}(Y)$ has depth at most $d_Y - 1$ and if Y is t_Y-of-N_Y, then $|\texttt{pub}(Y)| = N_Y$.

In our threshold signature scheme, the private keys for any given threshold public key are never revealed. The only private keys known by any user are the private keys associated with their usual public key. We denote $\texttt{priv}(\tilde{Y})$ as the set of all private keys used to compute any threshold ring signature on behalf of Y.

Definition 3.1 [Ring Threshold Multisignature Scheme] A ring threshold multisignature (RTM) scheme is a quadruple of PPT algorithms, $(\texttt{GEN}, \texttt{MERGE}, \texttt{SIG}, \texttt{VER})$ that, respectively, generate private-public keypairs for users, merge sets of public keys and threshold numbers into new public keys, fashion signatures on messages given a ring of public keys, and verifies signatures. We later restrict Definition 3.1 so as to only generate N-of-N and $(N-1)$-of-N signatures. Formally:

(i) $\texttt{GEN}(-)$ takes as input a security parameter 1^λ and outputs a 1-of-1 threshold keypair (x, X) where x is a private key with associated public key X.

(ii) $\texttt{MERGE}(-,-)$ takes as input keypairs $S = \{(x_1, X_1), (x_2, X_2), \ldots, (x_n, X_N)\}$ and a positive integer (threshold) t and outputs a t-of-N public key X for fashioning t-of-N threshold multisignatures.

(iii) $\texttt{SIG}(-,-,-,-)$ takes as input message M, ring of public keys $R = \{X_1, \ldots, X_L\}$, secret index s, and set $\texttt{priv}(\tilde{X}_s)$ and outputs signature σ.

(iv) $\texttt{VER}(-,-,-)$ takes as input a message M, a ring of public keys R, and a signature σ, and outputs a bit $b \in \{0, 1\}$.

We require the scheme to be *complete* in the sense that, for any message M, any ring of public keys $R = \{X_i\}_{i=1}^L$, and for any index $1 \le s \le L$, $\texttt{VER}(M, R, \texttt{SIG}(M, R, s, \texttt{priv}(\tilde{X}_s))) = 1$.

Definition 3.2 formalizes the idea that an adversary should not be able to link a public key with the threshold size, t, the coalition size, N, or the depth of the coalition except with negligible probability.

Definition 3.2 [Coalition Indistinguishable Keys] Let $\mathcal{A}$ be a PPT adversary and let $D(-)$, $M(-)$ be positive polynomials.

(i) $\mathcal{A}$ selects a random triple of integers (t_0, N_0, d_0) such that $2 \le t_0 \le N_0 \le M(\lambda)$ and $0 \le d_0 \le D(\lambda)$.

(ii) A random triple of integers (t_1, N_1, d_1) are selected such that $2 \le t_1 \le N_1 \le M(\lambda)$ and $0 \le d_1 \le D(\lambda)$. A random bit b is selected. A coalition S is generated such that $0 \le \texttt{depth}(S) \le d_b$ and $|S| = N_b$.

(iii) The key $X_b \leftarrow \texttt{MERGE}(t_b, S)$ is sent to $\mathcal{A}$.

(iv) $\mathcal{A}$ outputs a bit b'. This counts as a success if $b = b'$.

We say an RTM scheme has Coalition Indistinguishable Keys (CIK) if the adversary can succeed with probability only negligibly more than $1/2$.

Definition 3.2 does not take into account the fact that the adversary may learn the public keys of the coalition S generated in step (ii) above. Due to this, the naive implementation in Section 2.1 does not satisfy coalition indistinguishability for keys: an adversary who learns the public keys of a coalition can certainly check whether a certain elliptic curve point is their sum! We may also modify that implementation so that $Y_{\texttt{shared}} = \sum_{i=1}^{N} \sum_{j=1}^{N} H_s(Z_{i,j}, \mu_{i,j})G$, which clearly re-uses keys; this presents no algebraic danger, and allows for recursive implementation.

The implementation in Section 2.1 may be tweaked to satisfy CIK by having coalition members compute the shared key as $Y_{\texttt{shared}} := \sum_j H_s(X_j)G$ (in the N-of-N case) or as $Y_{\texttt{shared}} := \sum_{1 \leq i < j \leq N} H_s(Z_{i,j}, \mu_{i,j})G$ (in the $(N-1)$-of-N case). Still, each $H_s(X_j)G$ or $H_s(Z_{i,j})G$ must be communicated to the coalition, and if the adversary discovers these scalars, the adversary may simply check whether a given public key Y is the sum. Hence, these points must be communicated with encryption. If we allow users to compute their shared key as $Y_{\texttt{shared}} = \sum_j H_s(X_j, \mu_j)G$ for some constants μ_j, then we may also reduce the advantage enjoyed by an adversary in the game of Definition 3.2, even if that adversary is granted (limited) corruption oracle access.

We may strengthen Definition 3.2 to allow $\mathcal{A}$ to obtain signatures on arbitrary messages:

Definition 3.3 [Coalition Indistinguishable Keys and Signatures] Let $\mathcal{A}$ be a PPT adversary and let $D(-)$, $M(-)$ be a non-negative polynomial. Let $\mathcal{SO}(-,-)$ be a signing oracle.

(i) $\mathcal{A}$ selects a random triple of integers (t_0, N_0, d_0) such that $2 \leq t_0 \leq N_0 \leq M(\lambda)$ and $0 \leq d_0 \leq D(\lambda)$.

(ii) A random triple of integers (t_1, N_1, d_1) are selected such that $2 \leq t_1 \leq N_1 \leq M(\lambda)$ and $0 \leq d_1 \leq D(\lambda)$. A random bit b is selected. A coalition S is generated such that $0 \leq \texttt{depth}(S) \leq d_b$ and $|S| = N_b$.

(iii) The key $X_b \leftarrow \texttt{MERGE}(t_b, S)$ is sent to $\mathcal{A}$. $\mathcal{A}$ is granted oracle access to $\mathcal{SO}$.

(iv) $\mathcal{A}$ outputs a bit b'. This counts as a success if $b = b'$.

We say an RTM scheme has Coalition Indistinguishable Keys and Signatures (CIKS) if the adversary can succeed with probability only negligibly more than $1/2$.

Note that by giving $\mathcal{A}$ the public key X_b and signing oracle access, if $\mathcal{A}$ can discern non-negligible information about threshold t or coalition size N or depth d by inspecting a signature σ, then $\mathcal{A}$ succees non-negligibly at the game in Definition 3.3. Hence, a CIKS RTM scheme is an RTM scheme through which attackers are unable to link threshold size, coalition size, or depth to any given public key or signature.

In addition to coalition indistinguishability, we wish our signature scheme to have the usual property of digital signatures of *existential unforgeability* in the face of adaptive chosen message attacks, as well as the ring signature property of *signer ambiguity*. Variations of these security models describing insider corruption and adversarially generated keys appear in [1]. One may consider further expanding these definitions of existential unforgeability and signer ambiguity to take into account subthreshold corruption.

Definition 3.4 [Subthreshold Oracle Access] Given any set of public keys $Q = \{Y_1, \dots, Y_L\}$, we say that any PPT adversary $\mathcal{A}$ with access to an oracle $\mathcal{O}(-)$ has had *subthreshold oracle access* to Q if, for any $Y \in Q$ and $Y' \in \tilde{Y}$, at most $t_{Y'} - 1$ members of $\texttt{pub}(Y')$ appear in the transcript between $\mathcal{A}$ and $\mathcal{O}(-)$.

However, allowing for subthreshold corruption oracle access immediately degrades the definitions of signer amgibuity and CIK(S). Consider, for example, the definition of signer ambiguity against adversarially generated keys from [1].

Definition 3.5 [Signer Ambiguity v. Adversarially Generated Keys] Let $L(-)$ be a positive polynomial. Let $\mathcal{A}$ be a PPT adversary. Let $\mathcal{A}$ have access to a signing oracle $\mathcal{SO}(-,-,-)$ that outputs the valid signature $\texttt{SIG}(-,-,-,-)$. Consider the following game:

(i) A set of keypairs is generated $\{(x_i, X_i)\}_{i=1}^{L(\lambda)} \leftarrow \texttt{GEN}(1^\lambda)$ and the set of public keys $S = \{X_i\}_{i=1}^{L(\lambda)}$ is sent to $\mathcal{A}$.

(ii) $\mathcal{A}$ outputs a message M, a ring of public keys $R = \{Y_1, \dots, Y_{L^*}\}$, and two distinct indices $i_0 \neq i_1$ such that $Y_{i_0}, Y_{i_1} \in S$.

(iii) A random bit b is chosen and $\sigma \leftarrow \texttt{SIG}(M, R, i_b, y_{i_b})$ is sent to $\mathcal{A}$.

(iv) $\mathcal{A}$ outputs a bit b'. The game counts as a success if $b = b'$ and neither (i_0, M, R) nor (i_1, M, R) appear in the transcript between $\mathcal{A}$ and $\mathcal{SO}$.

We say the RTM scheme is *signer ambiguous with respect to adversarially generated keys* if the probability that $\mathcal{A}$ succeeds is negligibly close to $1/2$ (with respect to λ).

We may be tempted to expand this definition to take into account an adversary who has subthreshold corruption oracle access to $\tilde{Y}_{i_0}$ and $\tilde{Y}_{i_1}$. However, this definition is (in some sense), as good as we can get for signer ambiguity in a RTM scheme made linkable by key images as described in Section 2.1. Indeed, every coalition member computes the key image based solely on their private keys and can therefore check if a signature has been computed on behalf of the shared public key of the coalition. Hence, if an adversary is granted corruption oracle access (or even subthreshold corruption oracle access) to a signing coalition, signer ambiguity is violated. Similarly, an adversary with subthreshold corruption oracle access also violates CIK(S), because the adversary can gain non-trivial amounts of information about both threshold and coalition size during the corruption process.

We do not consider these to be serious violations of the security definitions, because the adversary is applying their corruption oracle access to information used to compute the associated signing public key, partially corrupting the signers. We speculate that modifications to our construction of key images may allow for stronger notions of signer ambiguity in the future without sacrificing robustness against double-spend attacks (e.g. by taking key images as homomorphic commitments, two commitments may be linked if their difference is a commitment to zero without revealing their masks), but that is beyond the scope of this document.

On the other hand, unforgeability of a RTM scheme must take into account subthreshold corruption oracle access. Multisignatures must not be forgeable by a subthreshold collection of malicious coalition members, otherwise they have no utility as signatures.

Definition 3.6 [Existential Unforgeability v. Adaptive Chosen Message and Subthreshold Insider Corruption] Let $\mathcal{A}$ be a PPT adversary and $L(-)$ and $D(-)$ be non-negative polynomials. $\mathcal{A}$ is given access to a signing oracle $\mathcal{SO}$ and a corruption oracle $\mathcal{CO}$. Consider the following game:

(i) Key pairs $\{(x_i, X_i)\}_{i=1}^{L(\lambda)}$ are generated using $\texttt{GEN}(1^\lambda)$ and $\texttt{MERGE}$ such that the set $S = \{X_i\}_{i=1}^{L(\lambda)}$ satisfies $\texttt{depth}(S) \leq D(\lambda)$. The set S is given to $\mathcal{A}$.

(ii) $\mathcal{A}$ outputs a ring R, a message M, and a signature σ. The game counts as a success if $R \subseteq S$, no public key $Y \in R$ appears in the transcript between $\mathcal{A}$ and $\mathcal{CO}$, $\mathcal{A}$ has had subthreshold corruption oracle access to R, $\texttt{VER}(M, R, \sigma) = 1$, and for each index k, (k, M, R) does not appear in the queries between $\mathcal{A}$ and $\mathcal{SO}$.

A scheme in which an adversary is only negligibly likely to succeed is said to be *existentially unforgeable with respect to adaptive chosen message attacks and subthreshold insider corruption* or merely *st-EUF* for subthreshold existentially unforgeable.

4 Proposed Implementation

We provide an implementation of the above scheme in the spirit of the original CryptoNote methodology, restricted to N-of-N and $(N-1)$-of-N coalitions of `depth` zero keys. For reference CryptoNote addresses, user secret keys and public keys are both ordered pairs of keys, i.e. private key (a, b) and public key (A, B) (where, for some group generator G in a group $\mathbb{G}$ where DDH holds, $A = aG$ and $B = bG$). Following terminology from [5], we refer to (a, A) as the *view keypair* and (b, B) and the *spend keypair*. We let $\Pi = (\texttt{GEN}^*, \texttt{ENC}^*, \texttt{AUTH}^*, \texttt{VER}^*, \texttt{DEC}^*)$ be a secure encrypt-then-authenticate scheme (where $\Pi^* = (\texttt{GEN}^*, \texttt{ENC}^*, \texttt{DEC}^*)$ is a secure encryption sub-scheme and $\Pi' = (\texttt{GEN}^*, \texttt{AUTH}^*, \texttt{VER}^*)$ is a secure message authentication sub-scheme). We use both of these schemes for computing sums, so the encryption and decryption algorithms may be taken as an homomorphic encryption scheme.

`GEN` generates the secret key $z = (a, b)$ by selecting a, b from an i.i.d. uniform distribution on $\mathbb{Z}_q$, and computing $Z = (A, B)$ with $A := aG$ and $B := bG$. `GEN` then outputs (z, Z).

`MERGE` takes as input a threshold t and a set of key pairs $S = \{(z_1, Z_1), \ldots, (z_n, Z_N)\}$ (where each $Z_i = (A_i, B_i)$) such that $N - 1 \leq t \leq N$ and $2 \leq t$.

(1) Each member of the coalition selects constants μ_i, γ_i for the multisig address[1].

(2) Each member derives a partial secret keypair (a_i^*, b_i^*) where $a_i^* = H_s(a_i, \mu_i)$ and $b_i^* = H_s(b_i, \gamma_i)$.

(3) Each member uses Π' to send (A_i^*, B_i^*) to the coalition.

(4) If $t = N$, then the coalition uses Π to collaboratively compute the shared secret view key $a^* = \sum_i a_i^*$[2], uses Π' to collaboratively compute the shared public spend key $B^* = \sum_{i=1}^N B_i^*$, and then uses Π to collaboratively compute the key image $J = \sum_{i=1}^N b_i^* H_p(B^*)$.

---

[1] One way to do this is for a user to attach an plaintext label to a multisig address like L = "Escrow wallet for Joe's Coffee" and to compute constants with hash functions and salts, $H_s(L, v)$

[2] Note that although secret information is about a_i not being directly shared with the coalition, the result of the computation is, in fact, a secret key, a^*.

(5) If $t = N - 1$, then:
   (a) For each i, j, a partial shared secret view key $\alpha_{i,j} := H_s(a_i^* A_i^*)$ and a partial shared secret spend key $\beta_{i,j} := H_s(b_i^* B_i^*)$ is computed by either participant i or j.
   (b) Set $N^* := \frac{N(N+1)}{2}$, $S^* := \{((\alpha_{i,j}, \beta_{i,j}), (\alpha_{i,j}G, \beta_{i,j}G))\}_{1 \leq i < j \leq N}$, and run $\texttt{MERGE}(N^*, S^*)$.
   (c) Any coalition member may compute the key image J and publish the CryptoNote public key (A^*, B^*).

$\texttt{SIG}$ takes as input a message M, a set of public keys $\{(A_1, B_1), \ldots, (A_L, B_L)\}$, a secret index $1 \leq k \leq L$ such that (A_k, B_k) is a t-of-N threshold public key pair with $2 \leq N - 1 \leq t \leq N$, and a set of t private keys $\{(a_i^*, b_i^*)\}_{i=1}^t = \texttt{priv}\left(\widetilde{(A_k, B_k)}\right)$.

(1) A set $\{s_{k+1}, s_{k+2}, \ldots, s_{k-1}\}$ of i.i.d. observations of uniform random variables are generated and shared among the coalition using Π'.[3]
(2) The j^{th} signatory selects a random $u_j \leftarrow \mathbb{Z}_q$, computes $H_i := H_p(B_i)$ for each index $1 \leq i \leq L$, and computes the points $u_j G$ and $u_j H_k$. The coalition uses Π' to collaboratively compute $u_k G := \sum_j u_j G$ and $u_k H_k := \sum_j u_j H_k$.
(3) Some threshold member computes $c_{k+1} = H_p(m, u_k G, u_k H_k)$ and iteratively computes $c_{i+1} = H_p(m, s_i G + c_i B_i, s_i H_i + c_i J)$ for $i = k+1, k+2, \ldots, k-1$.
(4) The threshold member from the previous step uses Π' to send c_k to all other signers with authentication.
(5) If $t = N$, each signatory computes their personal $s_{k,j} := u_j - c_k b_j^*$. If $t = N-1$, each signatory computes $s_{k,j} = u_j - c_k \sum_{i=1}^L z_{i,j}$. The coalition uses Π' to collaboratively compute $s_k = \sum_j s_{k,j}$
(6) Any signatory may now publish $(c_1, s_1, \ldots, s_N, J)$.

Remark 4.1 The resulting signature takes the same form as LSAG signatures as in [2]. Modifying the above to appropriately to take into account key vectors provides the generalization to MLSAG signatures. Thus the verification algorithm for these signatures is identical to the verification algorithm for usual MLSAG signatures and we omit its description.

Remark 4.2 Each u_j is kept secret from the other users and is generated randomly when the signature process begins. Certainly if u_j is revealed to another signatory, since the values of s_j and c_i are communicated in with authentication but not encryption, revealing the value $u_j - c_{i'} x_j$ can lead an observer to deduce x_j. Encryption does not solve the problem if threshold members are untrustworthy.

Similarly, if some value of u_j is re-used twice with the same private key, an observer can deduce the private key. Indeed, assuming we are using a hash function resistant to second pre-image attacks, the commitments from two signature processes $c_{i'}, c_{i'}^*$ are unequal except with negligible probability even if the other threshold members are colluding. Hence since $s_{i',j} = u_j - c_{i'} x_j$ and $s_{i',j}^* = u_j - c_{i'}^* x_j$, an observer may solve for the private key x_j. Don't re-use values of u_j, keep them secret, generate them randomly.

[3] We recommend that a coordinating user randomly selects these using a cryptographic random number generator; only the user coordinating the signature needs these values.

Remark 4.3 Note that users in $(N-1)$-of-N processes are prompted to select constants μ, γ multiple times for multiple sets of keys. If our hash function $H_s(-)$ is suitably secure, the lazy user can re-use the same constants μ and γ without concern; nevertheless, it is recommended that users do not re-use constants in `MERGE`.

Remark 4.4 The above description exploits the notion that an $(N-1)$-of-N signature is an N^*-of-N^* signature for $N^* = \frac{N(N+1)}{2}$, allowing to appear in the defining sum $Y_{\texttt{shared}} = \sum_{i,j} H_s(Z_{i,j}, \mu_{i,j})$ multiple times. This allows `MERGE` to be called at one level of recursion, and ensured consistency in step (5) above.

Uhm. I got nervous about what happens if $z_{i,j}$ repeat and the uniqueness of $s_{k,j}$... I think the algebra works out...

5 Security Proofs

Note that for our restricted RTM described in Section 4 only accepts thresholds t such that $2 \leq N-1 \leq t \leq N$ and only merges keys of depth zero.

To deal with our security proofs, we use the fact proven in [4] that the sum of a uniform random variable with any indepenent random variable in $\mathbb{Z}/m\mathbb{Z}$ results in a uniform random variable (and conversely when m is prime). Hence, no PPT algorithm will be able to distinguish between a uniform random variable U and a sum of uniform random variables, $\sum_i U_i$.

Assume H_s, H_p in the RTM implementation from Section 4 are cryptographic hash functions under the random oracle model whose outputs are statistically indistinguishable from a uniform distribution except with non-negligible probability, and whose outputs are independent of one another. Assume the discrete log hardness assumption holds.

Theorem 5.1 The RTM implementation from Section 4 is CIKS.

Proof Either the key pair $X_b = (A, B)$ received by $\mathcal{A}$ in step (iii) of Definition 3.3 is N-of-N (and thus has depth 1 and threshold $t = N$), $(N-1)$-of-N (and thus has depth 1 and threshold $t = N-1$), or a usual 1-of-1 key pair (and thus has depth 0). Since $N-1 \leq t \leq N$, non-negligible information about t is equivalent to non-negligible information about N. Moreover, $(N-1)$-of-N key pairs are N^*-of-N^* key pairs. Thus, we really only need to deal with two cases: an N-of-N key pair with $N > 1$ (where depth is necessarily $d = 1$) or a 1-of-1 key pair (necessarily with depth $d = 0$).

If (A, B) is an N-of-N key pair, then $A^* = \sum_i H_s(a_i, \mu_i)G$ and $B^* = \sum_i H_s(b_i, \gamma_i)G$. Since H_s is uniformly distributed in its output space with independent outputs, the sum of any number of digests is also uniformly distributed in output space [4]. On the other hand, if (A, B) is a 1-of-1 key pair, then A and B are each independent uniform random variables, so no PPT algorithm can determine whether A or B is a sum or not. This establishes that the scheme is CIK.

Granting signing oracle access does nothing for $\mathcal{A}$. Signatures on a message m with ring Q take the form $(c_1, s_1, \ldots, s_L, J)$, where c_1 and each s_i are observations from independent uniform random variables; these coordinates contain no information about the signing coalition of users. The key image is $J = \sum_i b_i^* H_p(B^*)$; write this $J = b^* H_p(B^*)$. The adversary may compute $H_p(B_i)$ for each public spend key B_i in Q, but without the ability to compute the discrete log of J with respect to

these points, $\mathcal{A}$ can gain only negligible information about b^*. Moreover, even if $\mathcal{A}$ gains non-negligible information about b^*, since each b_i^* was fairly generated using $\texttt{GEN}(1^\lambda)$, the adversary still will not be able to determine whether b^* is a sum or not, and if so, will not be able to determine how many contributing elements are in the sum. This establishes that the scheme is CIKS.

□

Theorem 5.2 The RTM implementation from Section 4 is st-EUF.

Proof □

6 Further Analysis

6.1 Efficiency and comparisons

6.2 Elaborations

On the other hand, we may be tempted to strengthen Definition 3.2 to take into account corruption oracle access on the part of the adversary. Unfortunately this leads to certain problems with the security definition. However, we can tweak the implementation in Section 2.1 so that coalition members select constants μ_j (or select random shared secrets $\mu_{i,j}$) and compute the shared key as $Y_{\texttt{shared}} := \sum_j H_s(X_j, \mu_j)G$ in the N-of-N case (or as $Y_{\texttt{shared}} := \sum_{1\leq i<j\leq N} H_s(Z_{i,j}, \mu_{i,j})G$ in the $(N-1)$-of-N case, respectively). Without knowledge of the values of μ_j, even if the adversary corrupts all the public keys in S, then $\mathcal{A}$ cannot successfully run `MERGE` for each value $1 \leq t \leq |S|$ to check the results by hand in comparison against the key X_b from step (iii). Moreover, each μ_j (or $\mu_{i,j}$) is a secret scalar. These are never made public, but also the associated public points $\mu_j G$ and $\mu_{i,j}G$ are never directly published. Thus, if the participating coalition members keep each μ_* secret, then even a very powerful adversary with oracle access for computing discrete logs will still be unable to discern whether some Y is a coalition key.

Special Thanks: We would like to issue a special thanks to the members of the Monero community who used the GetMonero.org Forum Funding System to support the Monero Research Lab. Readers may also regard this as a statement of conflict of interest, since our funding is denominated in Monero and provided directly by members of the Monero community by the Forum Funding System.

References

1. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC*, volume 6, pages 60–79. Springer, 2006.
2. Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *ACISP*, volume 4, pages 325–335. Springer, 2004.
3. Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.
4. Paola Scozzafava. Uniform distribution and sum modulo m of independent random variables. *Statistics & probability letters*, 18(4):313–314, 1993.
5. Nicolas van Saberhagen. Cryptonote v 2. 0, 2013.