RESEARCH BULLETIN MRL-9999

Threshold Ring Confidential Transactions

August 31, 2017
Brandon Goodell

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

Abstract

This research bulletin extends [4] 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 [3].

1 Introduction and Background

Multi-factor authentication plays a critical role in modern-day security. A user in control over several devices may wish to distribute the right to fashion a signature across all of those devices, so that a malicious user who has gained control over any one device requires other devices to complete a signature. Moreover, 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. By generalizing ring signatures to allow for multi-factor authentication, these protocols may employ new multi-factor authentication features.

If the number of users cooperating in the construction of a signature is not secret, multi-signature schemes can be constructed from any signature scheme (ring signature or otherwise) by simply requiring each participating user to present a separate signature; some implementations combine these together using boolean AND circuits as in the Borromean ring set-up described in []. We call these schemes *transparent* because it is clear to an adversary how many keys are used in a signature. On the other hand, 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 whether more than one key is used in the construction of a signature. We call these multi-signature schemes *opaque*. We say that a multi-signature scheme is t-of-n or t-of-n *threshold* if a set of n keys are specified and any t of them may be used to fashion a signature.

This leads us to the construction of opaque t-of-n threshold signatures, where a single joint public key and signatures from that key are computed by at least t members of a group of n without revealing that any collaboration took place. An opaque t-of-n threshold ring signature is merely a usual ring signature fashioned collaboratively and secretly by a subset of t members from some group of n members.

Note that a 1-of-n threshold signature scheme is equivalent to 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. Moreover, for multi-factor authentication purposes, $(n-1)$-of-n and n-of-n schemes are of particular interest. Due to this, we consider only t-of-n threshold ring signatures with $n-1 \leq t \leq n$, with $t > 1$, and with $n > 1$.

1.1 Our Contribution

We describe an implementation of t-of-n LSAG signatures first described by previous MRL contributors Shen Noether in [4] and implemented for use in Monero by contributor `Luigi`.

1.2 Notation and Prerequisites

We denote the message space $\mathcal{M} = \{0,1\}^*$. We let $\mathbb{G}$ be an additive group with prime order q an an arbitrary generator G. We denote $H_{\mathcal{X},\mathcal{Y}}(-)$ as a cryptographic hash function from the space $\mathcal{X}$ to $\mathcal{Y}$. For example $H_{\mathbb{G},\mathbb{Z}_q}(-)$ is a cryptographic point-to-scalar function. We assume that any two hash functions $H_{\mathcal{X},\mathcal{Y}}(-)$ and $H_{\mathcal{X}',\mathcal{Y}'}(-)$ are statistically independent.

2 MLSAG Ring Signatures

We briefly describe LSAG ring signatures in the sense of [3] and their MLSAG variant used in Monero. Let Q denote the set of ring members (public keys) $Q = \{Y_1, \dots Y_L\}$. Let i' be a distinguished index in this set corresponding to the signing key $Y_{i'}$. Let $y_{i'} \in \mathbb{Z}_q$ such that $Y_{i'} = y_{i'}G$. Let m be a message. A signer computes a key image I based on the private key $y_{i'}$.

For each i, the signer computes an elliptic curve point from the i^{th} ring member public key as $H_i := H_{\mathbb{G},\mathbb{G}}(Y_i)$. The signer selects a random secret $u \in_R \mathbb{Z}_q$, computes an initial temporary pair of points $\underline{k}_{i'} = (uG, uH_{i'})$ and computes an initial commitment $c_{i'+1} := H_{\mathcal{M},\mathbb{G}}(m, \underline{k}_{i'})$. The signer sequentially proceeds through indices $i = i'+1, i'+2, \dots, n, 1, 2, \dots, i'-1$ in the following way. The signer selects at random secret $s_i \in_R \mathbb{Z}_q$, computes the next temporary pair of points $\underline{k}_i = (s_iG + c_iY_i, s_iH_i + c_iI)$, and computes the next commitment $c_{i+1} := H_{\mathcal{M},\mathbb{G}}(m, \underline{k}_i)$. 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 $s_{i'} := u - c_{i'}y_{i'}$ and published the signature $(c_1, s_1, \dots, s_n, I)$.

A signature $(c_1^*, s_1^*, \dots, s_n^*, I^*)$ 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 $\underline{y} = (y_1, \dots, y_w)$, the j^{th} component of this key vector is used to generate a temporary secret as above described $\underline{k}_{i,j}$, so the i^{th} temporary secret is the list $\underline{k}_i = (\underline{k}_{i,1}, \underline{k}_{i,2}, \dots, \underline{k}_{i,w})$ yielding the associated commitment

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

2.1 Extending to Threshold Signatures

Generally, we wish to allow a group of users with the public keys $X = \{X_1, X_2, \dots, X_n\}$ 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 Y. In the n-of-n case, implementation works in the following manner. Given message m, a group of friends with the public keys $X = \{X_1, X_2, \dots, X_n\}$ wish to collaboratively construct a signature using all their private keys without revealing any collaboration took place. They first must compute a shared public key $Y = \sum_{j=1}^{n} X_j$ and publish it. They select their ring to be a set of public keys $Q = \{Y_1, Y_2, \dots, Y_L\}$ such that $Y = Y_{i'}$ for some secret index i'. Each user can compute $H_i = H_{\mathbb{G},\mathbb{G}}(Y_i)$. For each $1 \leq j \leq n$, the j^{th} signer (who owns the secret key x_j) computes a partial key image $I_j = x_j H_{i'}$, picks a random initial temporary secret u_j, and shares the triple of points $I_j, u_j G, u_j H_{i'}$ with the other signers.

Now any of these users may compute the key image $J = \sum_j I_j$, the random points $u_{i'}G = \sum_j u_j G$ and $u_{i'}H_{i'} = \sum_j u_j H_{i'}$. Using these, any of these users may compute the first in the sequence of commitments $c_{i'+1} := H_{\mathcal{M},\mathbb{G}}(m, u_{i'}G, u_{i'}H_{i'})$. The group decides upon random values $s_{i'+1}, \dots, s_L, s_1, \dots, s_{i'-1}$ and computes each $c_{i+1} := H_{\mathcal{M},\mathbb{G}}(m, s_i G + c_i Y_i, s_i H_i + c_i I)$ for $i = i'+1, \dots, i'-1$. All threshold members then use $c_{i'}$ to compute their $s_{i',j} = u_j - c_{i'} x_j$. Each member may reveal their value of $s_{i',j}$ without fear and the threshold members may then compute the value $s_{i'} = \sum_j s_{i',j}$ and publish the signature $(c_1, s_1, \dots, s_L, J)$. 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 $\{X_1, X_2, \dots, X_n\}$ form a coalition. Each pair of users has a shared secret scalar $z_{i,j} = H_{\mathbb{G},\mathbb{Z}_q}(x_i X_j)$ 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 $Y := \sum_{1 \leq i \leq n} \sum_{i < j \leq n} Z_{i,j}$.

3 Security Models

In this section we present a few security definitions we use later on. Our definition of *unforgeability* begins with the definition from [3], which is stronger than notions of unforgeability put forth previously (c.f. [7]), but which was expanded further in [2].

Definition 3.1 [Ring Signature Existential Unforgeability against Adaptive Chosen Message Attack] Let $Q = \{Y_1, \dots, Y_L\}$ be a ring of public keys. Let $\mathcal{SO}(-)$ be a ring signing oracle that accepts as input any non-empty subset Q' and message m' and produces a ring signature σ' that always passes verification. Let $\mathcal{A}$ be any PPT adversary with oracle access to $\mathcal{SO}$. A ring signature scheme is *existentially unforgeable with respect to adaptive chosen message attacks* if, for any message m and subset Q^* not in the transcript of queries from $\mathcal{A}$ to $\mathcal{SO}$, the output $(Q^*, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{SO}}(Q)$ passes verification with only negligible probability.

This definition, however, only applies to ring signatures and ignores the possibility that the adversary has corrupted some public keys. We first expand the notion to take into account the idea that the adversary may corrupt public keys by considering the adaptive chosen message attack with insider corruption first described in [2].Expanding that definition to threshold signatures requires a modification to allow for the adversary to control a sub-threshold number of private keys.

We need to ensure unforgeability even if the adversary has corrupted some, but not a threshold number, of public keys used in the computation of shared keys in the ring. We do this by checking that each threshold/shared public key in the ring has had the signing oracle queried on fewer of its parent keys than its threshold. Moreover, if any of the parent keys of any of the shared keys in the ring are themselves shared keys, the problem repeats: for each public key contributing to some shared public key contributing to some shared public key contributing to ... some shared public key in the ring, we must check that the signing oracle has been queried with fewer of the parent keys than the associated threshold for each of them. Turtles all the way down.

THIS IS A NOVEL DEFINITION HOLY WOW! THIS TOOK ME TOTALLY BY SURPRISE! NONE OF THE PROOFS I THOUGHT WOULD WORK WILL WORK! WOW! Learn something new every day.

Definition 3.2 [Threshold Ring Signature Existential Unforgeability against Insider Adaptive Chosen Message Attack] Let $Q = \{B_1, \ldots, B_L\}$ be a set of public keys. Let $\mathcal{SO}(-)$ be a ring signing oracle that accepts as input any non-empty subset Q' and message m' and produces a ring signature σ' that always passes verification. We say a threshold ring signature scheme is *existentially unforgeable with respect to insider adaptive chosen message attacks* if for any message m, for any subset $Q^* \subseteq Q$ such that:

(a) Q^* is not in the transcript of queries from $\mathcal{A}$ to $\mathcal{SO}$,

(b) if B is any t_B-of-n_B threshold public key that contributes to the computation of any shared public key in Q^*, and if $B' = \{B'_1, \ldots, B'_{n_B}\}$ is the set of public keys used to compute B, then at most $t-1$ distinct elements of B' appear in the transcript of $\mathcal{A}$ to $\mathcal{SO}$..

the output $(Q^*, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{SO}}(Q)$ passes verification with only negligible probability.

Our definition of *signer ambiguity* also comes from [3], informally stated this way: given any ring signature σ on message m with ring of public keys $Q = \{X_1, \ldots, X_n\}$, an adversary can determine public key $X_{i'}$ corresponding to the secret key $x_{i'}$ used in the construction of σ with only negligible probability.

This definition of signer ambiguity may be strengthened, as in [1] or [7], by requiring that adversaries gain at most a negligible advantage even when some or all of the private keys of the ring are revealed. For usage in Monero (and other CryptoNote protocols), this stronger definition of signer ambiguity is impossible to satisfy due to our construction of key images. 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), but that is beyond the scope of this document.

4 Proposed Implementation

We provide an implementation of the above scheme in the spirit of the original CryptoNote methodology, wherein user secret keys and public keys are both ordered pairs of keys, i.e. $sk = (sk_1, sk_2)$ and $pk = (pk_1, pk_2)$. We expand upon the MLSAG generalization to key vectors in Section **??**. Following terminology from [8], we refer to (sk_1, pk_1) as the *view keypair* and (sk_2, pk_2) and the *spend keypair*.

The keys $(sk, pk) \leftarrow$ `GenUserKey` are chosen by selecting $a, b \leftarrow \mathbb{Z}_q$, computing $A := aG$, $B := bG$, and setting $(sk, pk) := ((a, b), (A, B))$. `ThrRingSign`, given a message M, a set of public keys $\{(A_1^*, B_1^*), \ldots, (A_n^*, B_n^*)\}$, a threshold $t \in \{n-1, n\}$ with $t \geq 2$ and $n \geq 2$, and a secret index $1 \leq i' \leq L$, generates a t-of-n signature as follows.

(1) A random secret view key $a^* \leftarrow \mathbb{Z}_q$ is chosen and distributed to the threshold members. The public view key is $A^* = a^*G$.

(2) (All threshold signatories) If $t = n$, the public spend key B^* is computed as $B^* := \sum_{i=1}^{n} B_i^*$. If $t = n - 1$, each pair of users computes the shared secrets $z_{i,j} = H_{\mathbb{G},\mathbb{Z}_q}(b_i^* B_j^*)$ and publishes the public values $Z_{i,j} = z_{i,j}G$. The public spend key is $B^* := \sum_{i=1}^{n-1} \sum_{j=i}^{n} Z_{i,j}$.

(3) (All threshold signatories) The signatories decide upon a set of public keys to include as false signatories in their ring, they decide upon a secret index i' at which they store their shared public key inside the ring, and for $i \neq i'$ they decide on a random scalar $s_i \leftarrow \mathbb{Z}_q$. Denote the set of public keys $Q = \{(A_1, B_1), \ldots, (A_L, B_L)\}$ where $(A_{i'}, B_{i'}) = (A^*, B^*)$.

(4) (All threshold signatories) The j^{th} signatory selects a random $u_j \leftarrow \mathbb{Z}_q$, computes $H_i := H_{\mathbb{G},\mathbb{G}}(B_i)$ for each index $1 \leq i \leq L$, and computes the points $u_j G$ and $u_j H_{i'}$. The j^{th} signatory sends the points $u_j G, u_j H_{i'}$ to the other threshold members with authentication.

(5) (All threshold signatories) If $t = n$, the j^{th} signatory computes the partial key image $I_j := b_j^* H_{\mathbb{G},\mathbb{G}}(B^*)$ and sends I_j to the other threshold members with authentication. If $t = n - 1$, the j^{th} signatory computes the partial key image $I_j := \sum_{i=1}^{n} z_{i,j} H_{\mathbb{G},\mathbb{G}}(B^*)$ and sends I_j to the other threshold members with authentication.

(6) (Some signer) Some threshold member computes $I = \sum_j I_j$, the random point $u_{i'}G = \sum_j u_j G$, and the random point $u_{i'}H_{i'} = \sum_j u_j H_{i'}$. That member then
   (i) computes $c_{i'+1} = H_{\mathcal{M},\mathcal{G}}(m, u_{i'}G, u_{i'}H_{i'})$ and
   (ii) computes $c_{i+1} = H_{\mathcal{M},\mathcal{G}}(m, s_i G + c_i B_i, s_i H_i + c_i I)$ for $i \neq i'$.

(7) (Some signer) The threshold member from the previous step sends $c_{i'}$ to all other signers with authentication.

(8) (All threshold signatories) If $t = n$, each signatory computes their personal $s_{i',j} := u_j - c_{i'} b_j^*$ and the group computes (with authentication) their sum $\sum_j s_{i',j}$. If $t = n-1$, each signatory computes $s_{i',j} = u_j - c_{i'} \sum_{i=1}^{L} z_{i,j}$ and the group computes (with authentication) the sum.

(9) Any signatory may now publish $(c_1, s_1, \ldots, s_N, I)$.

The resulting signature takes the same form as LSAG signatures as in [3]. 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 4.1 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.

5 Security Proofs

If we assume all threshold members are honest, the scheme reduces to the usual LSAG signature as in [3]. However, this is an undesirable assumption for applications in cryptocurrency. Definition 3.1 is inadequate in the threshold setting because, if some B_i is a t_i-of-n_i shared public key in the ring, the adversary may query the oracle $\mathcal{SO}$ to sign messages on behalf of some of the n_i members sharing the public key B_i without violating the conditions of Definition 3.1. Do to this, we consider Definition 3.2 to describe adaptive chosen message attacks where insiders.

Since we may regard any $(n-1)$-of-n instantiation of the above scheme as an n^*-of-n^* instantiation, it is sufficient to prove that any n-of-n instantiation is secure. The 1-of-1 instantiation is merely the LSAG signature from [3]; in this setting, Definitions 3.1 and 3.2 coincide, so we only must concern ourselves with n-of-n instantiations with $n \geq 2$.

The strength of the security proof from [3] rests on novel rewind-on-success simulations. Rewind simulations were first presented as the forking lemma in [6] and the heavy row lemma in [5]; rewind-on-success simulations are first presented in [3]. With a master PPT $\mathcal{M}$ invoking a PPT adversary $\mathcal{A}$ to obtain a transcript $\mathcal{T}$ in an attack game on some scheme Π may, the rewind-on-success simulation will, upon finding a success in $\mathcal{T}$, rewind $\mathcal{T}$ to some point, header h and "begin again" to seek an additional success. Resimulating $\mathcal{A}$ with new random data, $\mathcal{M}$ generates a new transcript $\mathcal{T}^*$ where $\mathcal{T}$ and $\mathcal{T}^*$ are identical up to (and including) header h. It is established in [3, Lem E.1] that the probabilities of success of $\mathcal{T}$ and $\mathcal{T}^*$ are identical (although this says nothing of their independence). Thus an attacker who can find one success with non-negligible probability can find any finite number they desire using rewind-on-success with non-negligible probability.

Thus, if the adversary can successfully compute one forged signature on a message with non-negligible probability, say $\sigma = (c_1, s_1, \ldots, s_N, J)$, then that adversary can rewind and compute a second forged signature with the same key image but different random values s^*_i, say $\sigma^* = (c^*_1, s^*_1, \ldots, s^*_N, J)$, also with non-negligible probability. In checking that these forgeries satisfy verification, the adversary must compute the commitments in the LSAG signature, and hence must query some hash function H at least once per commitment. So the adversary must make as many queries to H as there are ring members, L, each of the form $H(J, m, sG+cB, sH(B)+cJ)$, where

B is a public key in the ring and J is the key image associated to the signature. In computing the first forgery, the adversary must compute some first commitment $H(m, J, uG, vG') = H(m, J, sG + cB, sH(B) + cJ)$, where u, v, and the base point G' are each unknown before rewinding. After rewinding and computing a second forgery, the adversary has the system of equations

$$\begin{aligned} uG &= sG + cB \\ uG &= s^*G + c^*B \\ vG' &= sH(B) + cJ \\ vG' &= s^*H(B) + c^*J \end{aligned}$$

The adversary can then compute the secret key $b = \frac{s-s^*}{c^*-c}$, solving the discrete log problem $B = bG$.

This proof extends directly to MLSAG signatures; this was claimed in [4] but the proof therein contained a mistake, which we correct here:

Shen's security theorem and corrected proof

Theorem 5.1 *

6 Elaboration

The signatories must make several joint decisions in the process described under Section 4. We are vague in their description because the implementation of these steps can be done in many ways.

For example, in Step 3, the signatories decide upon a ring and a secret index to store their public keys. This may be done by merely having some member do it randomly (introducing a sort of Byzantine General problem), or using some deterministic (but seemingly random) method chosen ahead of time in meatspace based on the input message. The secret index should appear to be uniformly random, regardless of method employed. Note that one of the signatories can publicly communicate a sort of encryption of the secret index without harm by merely publishing the ring of public keys Q; other signatories know their own public keys, so they can inspect the ring and determine the secret index without any further information from the first signatory and without observers being able to discern which index is the threshold key.

Also in Step 3, the signatories decide on the values of s_i randomly. One method is for each signatory to compute their own $s_{i,j}$ and computing the shared sums $s_i = \sum_j s_{i,j}$. This requires a lot of interaction; another method is to simply let one threshold member do it and communicate the values to the group with authentication. Note that the values s_i are eventually made public in the signature and there is no harm in sharing these values without encryption. Similarly, in step 8, we compute the sum $\sum_j s_{i',j}$; since each user keeps their u_j secret, they can reveal their $u_j - c_{i'} b_j^*$ or $u_j - c_{i'} \sum_{i=1}^{L} z_{i,j}$ without risking their private keys, so there is no harm in sharing these values without encryption.

To communicate a message with authentication, we use the HMAC scheme: for the j^{th} threshold member to communicate a message M to the i^{th} threshold member,

a shared secret $t_{i,j}$ is computed and $(M, \texttt{HMAC}(t_{i,j}, M))$ is sent. For a CCA-secure encrypt-and-authenticate scheme [?], two users generate two shared secrets $t_{i,j}$, $t^*_{i,j}$. The sending user computes the ciphertext $C = \texttt{Enc}(t_{i,j}, M)$ and the authentication codes $\tau = \texttt{HMAC}(t^*_{i,j}, C)$ and sends (C, τ) to the receiver. If a receiver sees some (C, τ), they can check if τ is a valid HMAC on C for any of their shared secrets. If so, they can decrypt C with the other shared secret.

EVERYTHING BELOW THIS LINE IS NOT WRITTEN BY ME. It is reference copy-pasta from another document as I write the above.

6.1 Security Models

group-changing attack: can attacker change ring without changing sig? Multiple-Known-Signature Existential Forgery [On the security models of threshold ring signature schemes. Liu and Wong]

Unforgeability: LWW LSAGs definition of unforgeability.

A t-of-n threshold signature σ on a message M with set of public signing keys P^* will not pass `Verify` except with negligible probability unless σ is a non-$\perp$ output of `RingSign`. Against fixed-ring attacks: same ring, different message. against chosen subring attacks: any subring, different message. against adversarially chosen pubkeys: honest users tricked into participating in signatures with adversarially chosen ring members. against insider corruption: like chosen subring + adversary can corrupt or subpoena users into giving up their secret keys.

Anonymity/linkable signer ambiguity: Basic, wrt Adversarially chosen keys, Attribution attacks and full key exposure. [Ring Signatures: Stronger Definitions, and Constructions without Random Oracles.]

From LWW: By signer anonymity, we require that given a signature with respect to a group of keys, it is infeasible to identify which private key is used to generate the signature. In this paper, we focus on providing near maximum privacy in such a way that the chance of guessing correctly which key of a set of n keys generated by G with security parameter k is used to generate a given signature is negligibly greater than $1/n$ provided that the signing key is chosen at random and the adversary only knows the public keys (but not the private keys). If t private keys of the set are also known by the adversary, where $t < n$, but the signing private key is not one of them, then the success probability of the

adversary should still be negligibly greater than$1/(n-t)$.

The notion of signer anonymity above is different from those of

31, 1

which require that revealing private keys does not reduce the level of signer anonymity. However, we specify the weaker form for supporting the property of culpability. On the other hand, the original notion of [31, 1] should be used if exculpability is required.

Traceability

Linkability:

6.2 Implementation

6.3 Security Proofs

7 Further Analysis

Efficiency and comparisons

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. Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. *Advances in Cryptology, Asiacrypt 2002*, pages 639–645, 2002.
2. 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.
3. 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.
4. Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.
5. Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In *Annual International Cryptology Conference*, pages 354–369. Springer, 1998.
6. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Eurocrypt*, volume 96, pages 387–398. Springer, 1996.
7. Ronald Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *Advances in Cryptology, ASIACRYPT 2001*, pages 552–565, 2001.
8. Nicolas van Saberhagen. Cryptonote v 2. 0, 2013.