Ring Signatures

Implementation of Linkable Spontaneous Anonymous Group Signatures

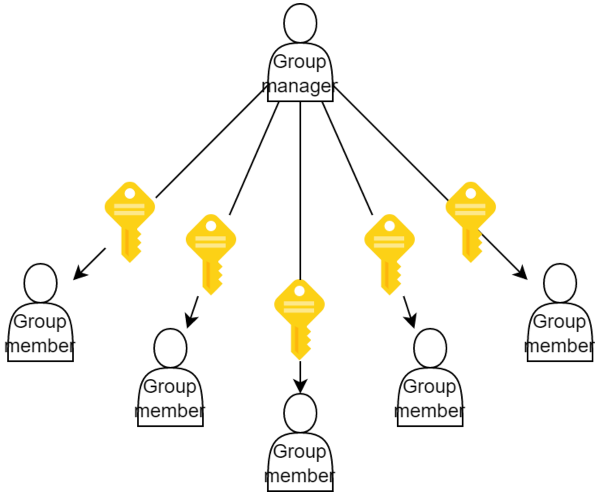
Markus Eggimann

Content

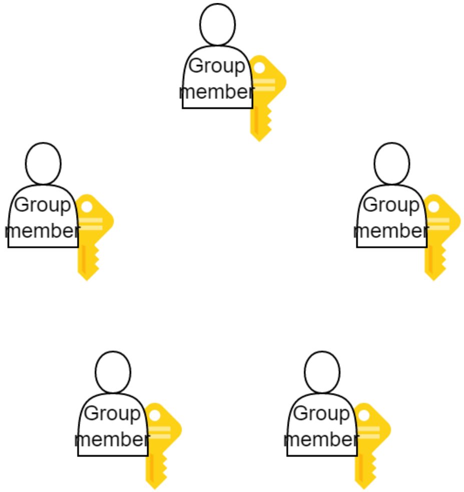
Definitions I – Signature Schemes

Single Signer Scheme

Group Signature Scheme



Ring Signature Scheme



Definitions II – Properties

- **Anonymity**
  - Signer remains anonymous
- **Spontaneity**
  - No group setup or coordination necessary
- **Linkability**
  - Two signatures of the same signer can be linked

Definitions III – Interface

- **Ring Signature Scheme Interface**

```
public interface IRingSigner
{

    Signature Sign(
        byte[] message,
        BigInteger[] publicKeys,
        BigInteger signerPrivateKey,
        int signerPublicKeyIndex);

    bool Verify(
        byte[] message,
        Signature signature,
        BigInteger[] publicKeys);

}
```

Selected Ring Signature Schemes

Spontaneous anonymous group signature scheme (SAG)

- Rivest, Shamir, Tauman (2001)
- Satisfies Anonymity, Spontaneity
- Based on public/private key pairs
- Motivation: safe whistleblowing

Linkable spontaneous group signature scheme (LSAG)

- Liu, Wei, Wong (2004)
- Satisfies Anonymity, Spontaineity, Linkability

```
bool SignedBySameSigner(
    Signature signature1,
    Signature signature2);
```

- Based on public/private key pairs
- Motivation: E-Voting

LSAG Algorithm I – Overview

Signing

Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute
$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute
$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$
4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

Verification & Linking

A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z_i' = g^{s_i} y_i^{c_i}$, $z_i'' = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

For a fixed list of public keys L, given two signatures associating with L, namely $\sigma_L'(m') = (c_1', s_1', \cdots, s_n', \tilde{y}')$ and $\sigma_L''(m'') = (c_1'', s_1'', \cdots, s_n'', \tilde{y}'')$, where m' and m'' are some messages, a public verifier after verifying the signatures to be valid, checks if $\tilde{y}' = \tilde{y}''$. If the congruence holds, the verifier concludes that the signatures are created by the same signer. Otherwise, the verifier concludes that the signatures are generated by two different signers.

LSAG Algorithm I – Overview

Signer private key

Public keys

Hash functions

"Nonces"

"Challenges"

Signing

Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

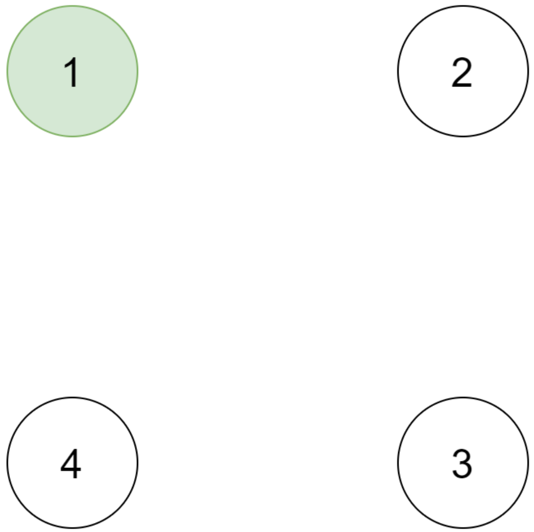
Verification

A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z'_i = g^{s_i} y_i^{c_i}$, $z''_i = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z'_i, z''_i)$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z'_n, z''_n)$. If yes, accept. Otherwise, reject.

For a fixed list of public keys L, given two signatures associating with L, namely … $(c'_1, s'_1, \cdots, s'_n, \tilde{y}')$ and $\sigma''_L(m'') = (c''_1, s''_1, \cdots, s''_n, \tilde{y}'')$, where m' and …ome messages, a public verifier after verifying the signatures to be …cks if $\tilde{y}' = \tilde{y}''$. If the congruence holds, the verifier concludes that the signatures are created by the same signer. Otherwise, the verifier concludes that the signatures are generated by two different signers.

LSAG Algorithm II – Signing



Given message $m \in \{0,1\}^*$, list of public key $\boxed{L = \{y_1, \cdots, y_n\}}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

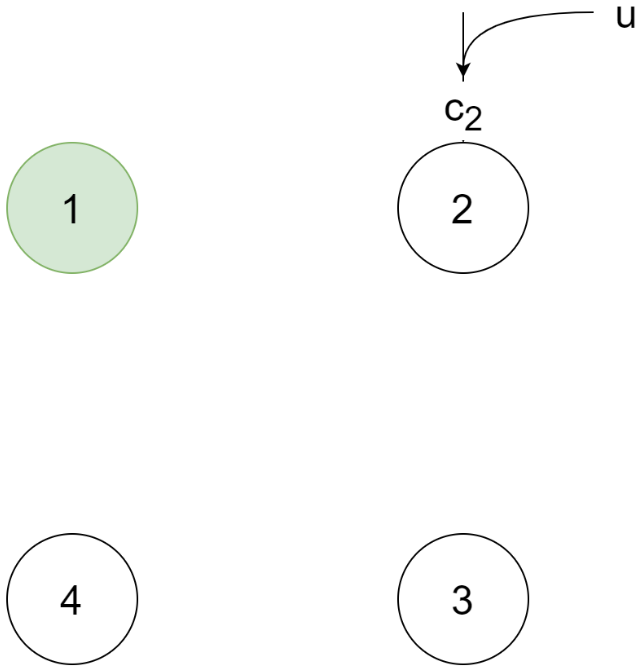
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \mod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing

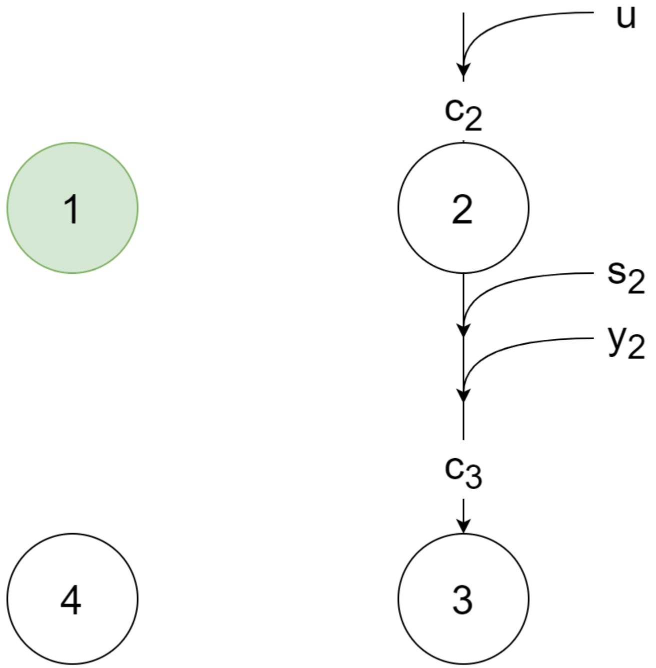


Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \le \pi \le n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute
$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute
$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$
4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing



Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

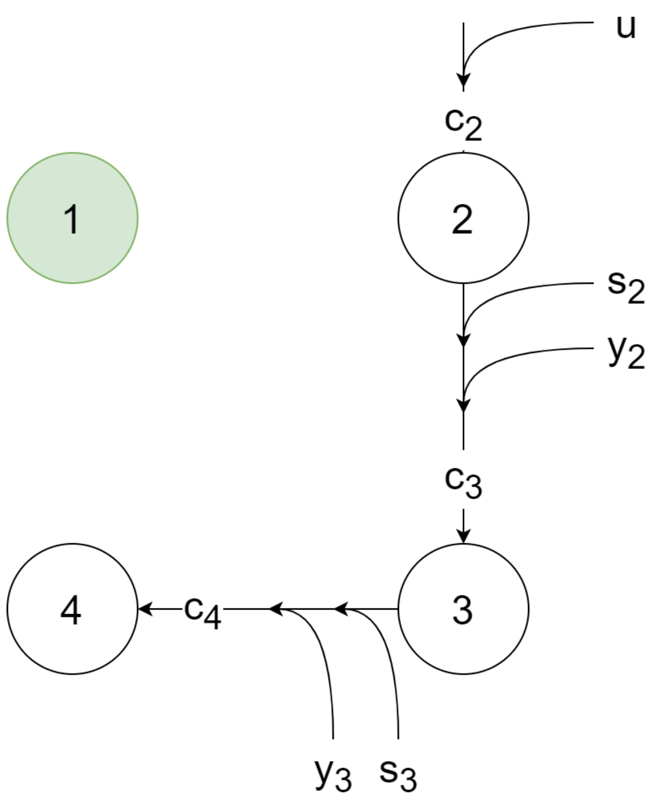
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing



Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \le \pi \le n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

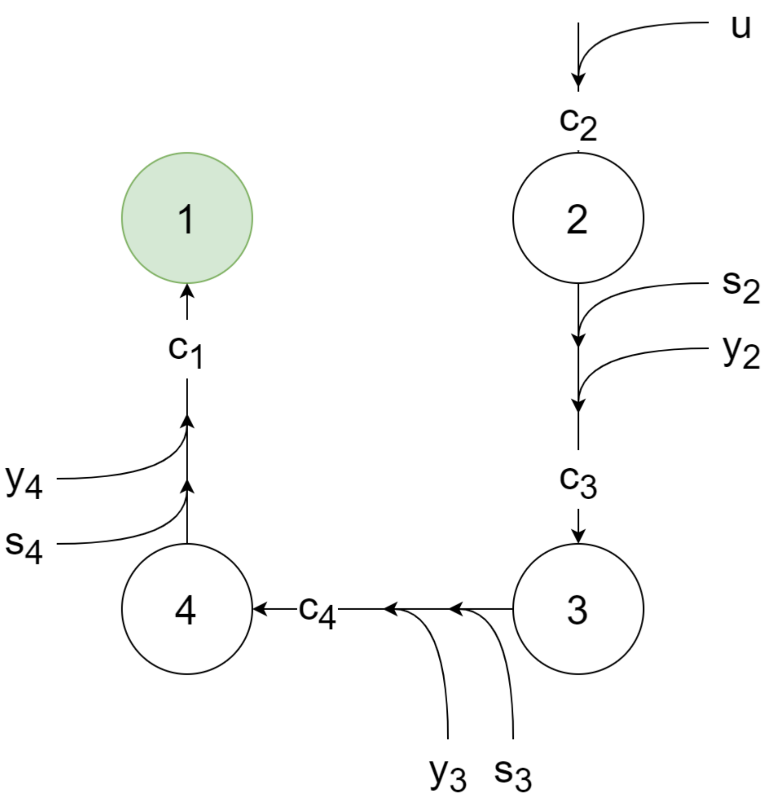
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing



Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \le \pi \le n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

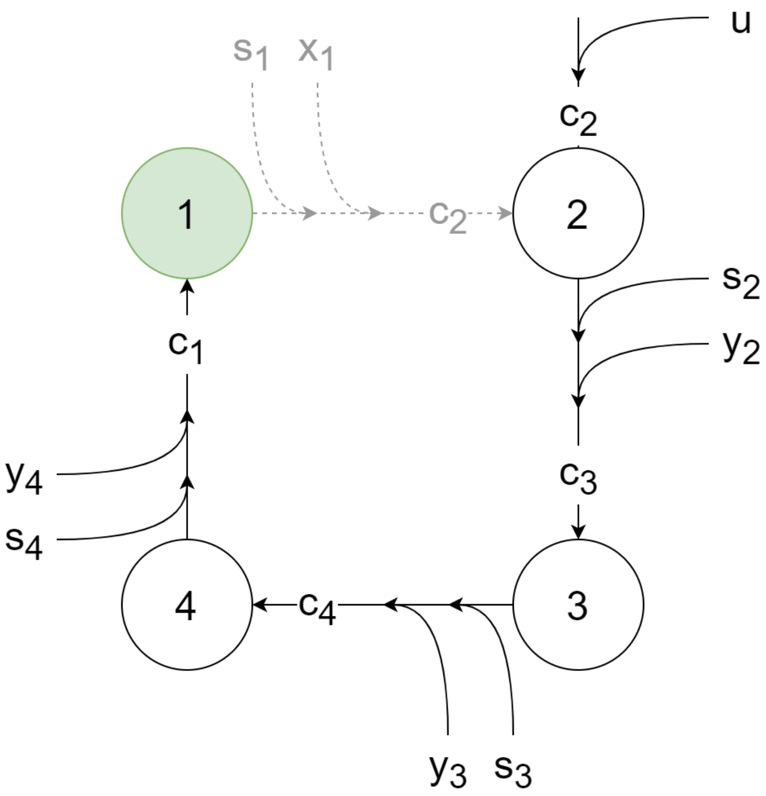
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing



Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

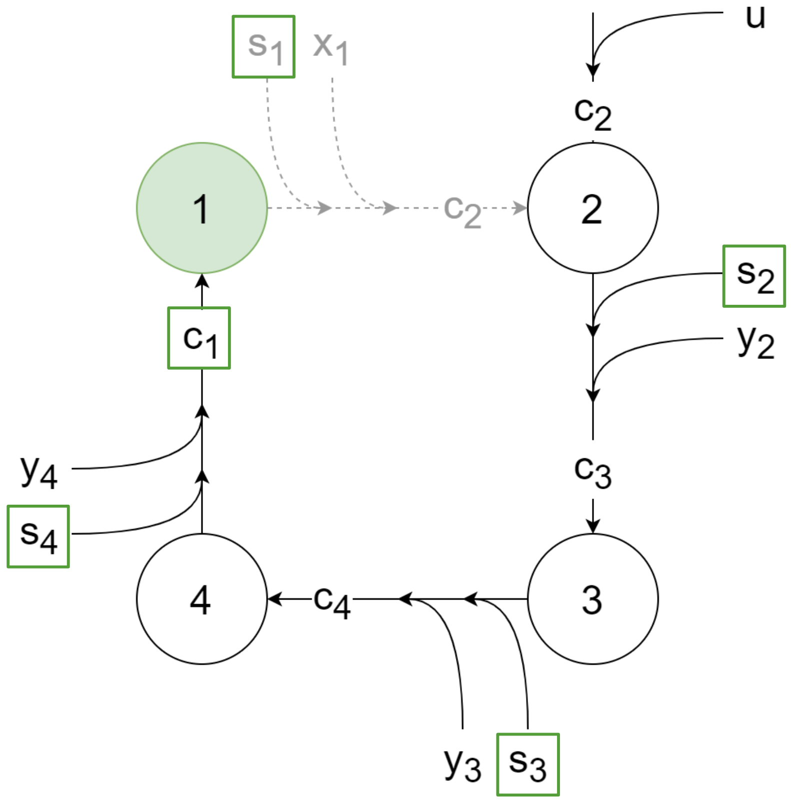
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing



Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\tilde{y} = h^{x_\pi}$.
2. Pick $u \in_R \mathbb{Z}_q$, and compute

$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$

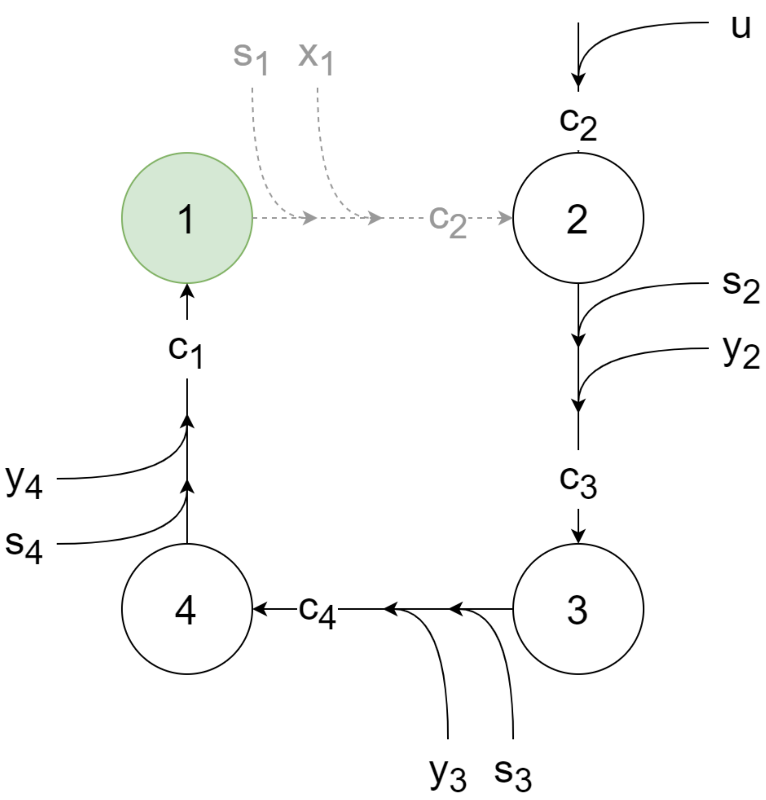
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute

$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$

4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$.

LSAG Algorithm II – Signing

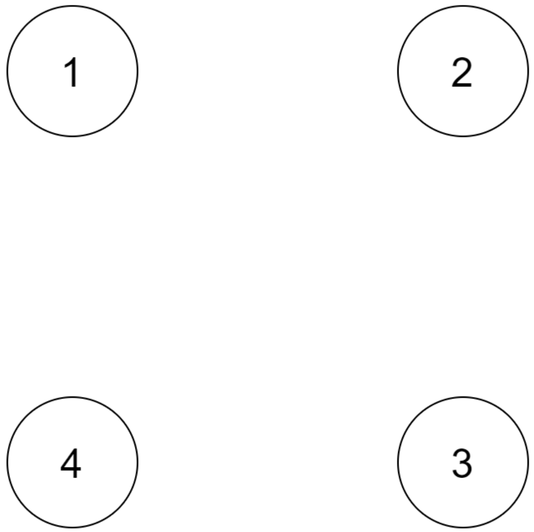


Given message $m \in \{0,1\}^*$, list of public key $L = \{y_1, \cdots, y_n\}$, private key x_π corresponding to y_π $1 \leq \pi \leq n$, the following algorithm generates a LSAG signature.

1. Compute $h = H_2(L)$ and $\boxed{\tilde{y} = h^{x_\pi}}$
2. Pick $u \in_R \mathbb{Z}_q$, and compute
$$c_{\pi+1} = H_1(L,\ \tilde{y},\ m,\ g^u,\ h^u).$$
3. For $i = \pi+1, \cdots, n, 1, \cdots, \pi-1$, pick $s_i \in_R \mathbb{Z}_q$ and compute
$$c_{i+1} = H_1(L,\ \tilde{y},\ m,\ g^{s_i} y_i^{c_i},\ h^{s_i} \tilde{y}^{c_i}).$$
4. Compute $s_\pi = u - x_\pi c_\pi \bmod q$.

The signature is $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \boxed{\tilde{y}})$.

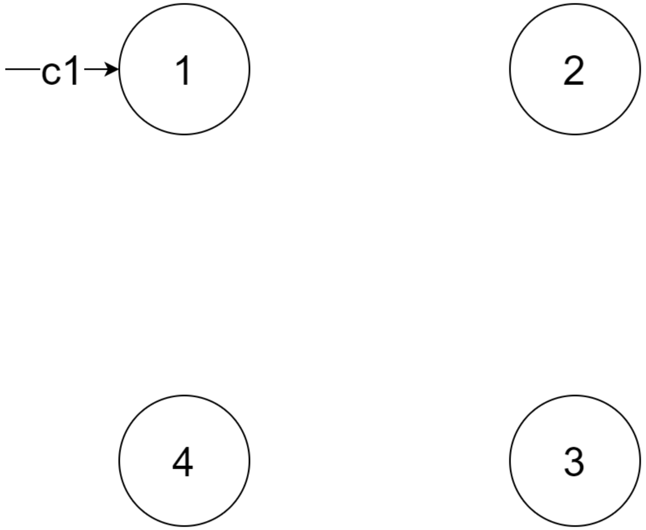
LSAG Algorithm III – Verification



A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z_i' = g^{s_i} y_i^{c_i}$, $z_i'' = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

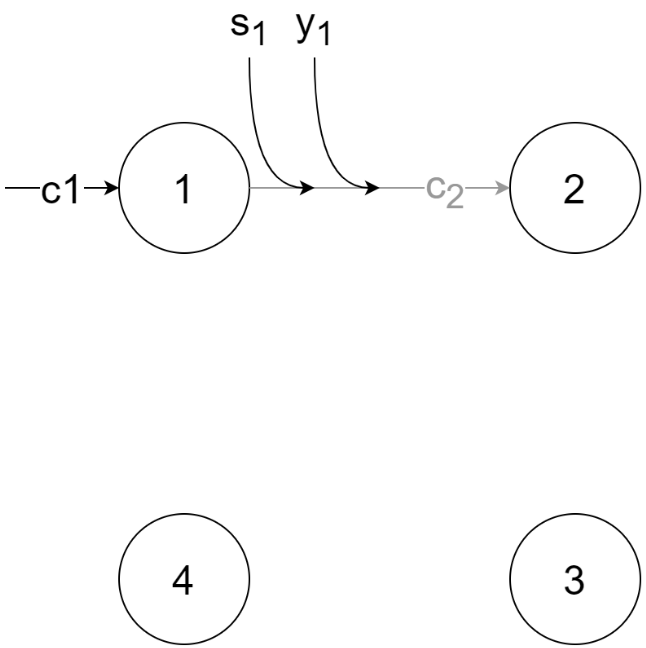
LSAG Algorithm III – Verification



A public verifier checks a signature $\boxed{\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})}$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z_i' = g^{s_i} y_i^{c_i}$, $z_i'' = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

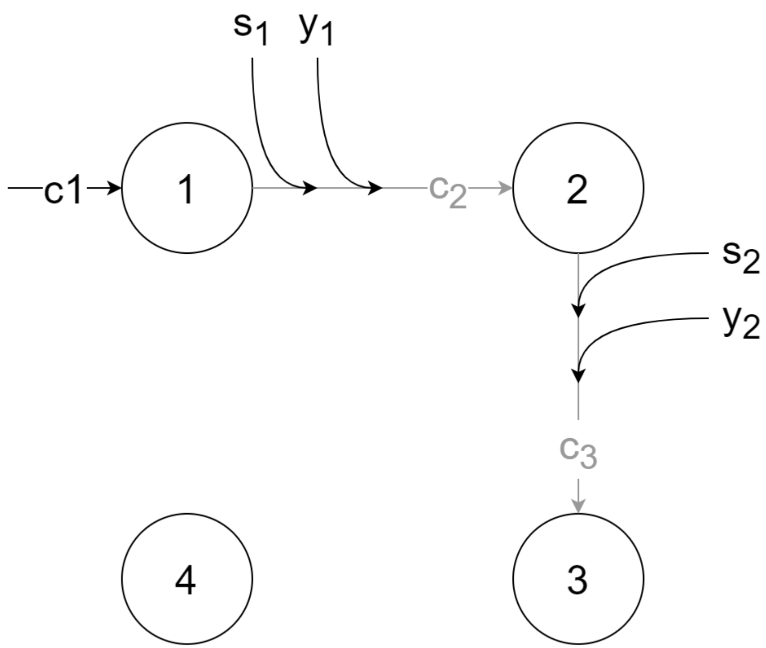
LSAG Algorithm III – Verification



A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z'_i = g^{s_i} y_i^{c_i}$, $z''_i = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z'_i, z''_i)$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z'_n, z''_n)$. If yes, accept. Otherwise, reject.

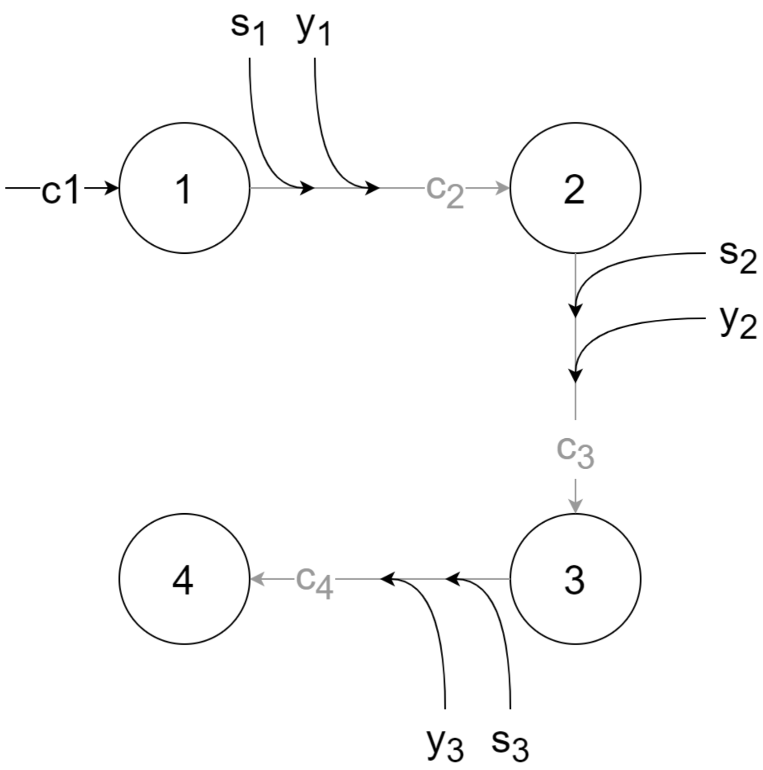
LSAG Algorithm III – Verification



A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z'_i = g^{s_i} y_i^{c_i}$, $z''_i = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z'_i, z''_i)$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z'_n, z''_n)$. If yes, accept. Otherwise, reject.

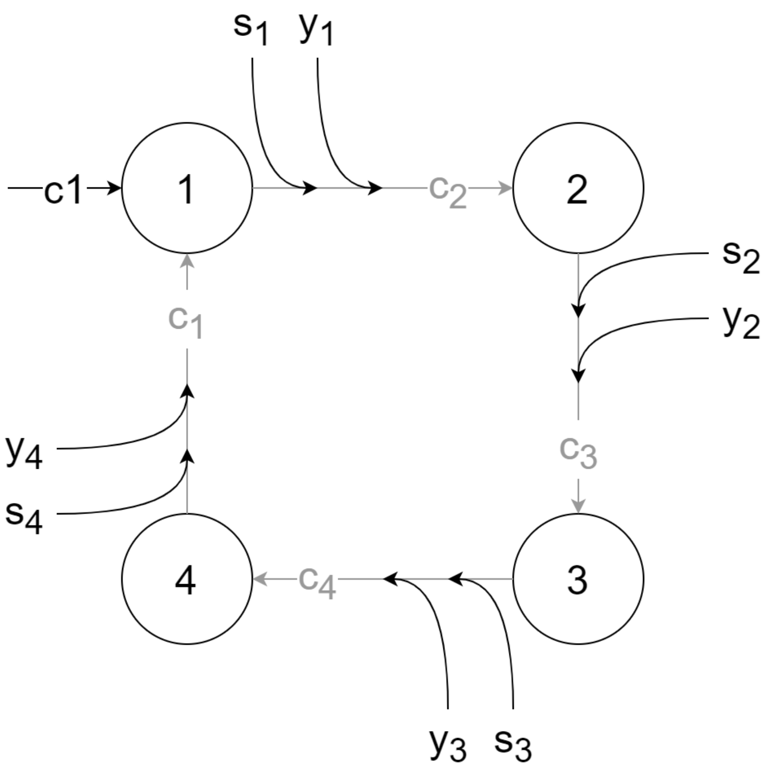
LSAG Algorithm III – Verification



A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z'_i = g^{s_i} y_i^{c_i}$, $z''_i = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z'_i, z''_i)$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z'_n, z''_n)$. If yes, accept. Otherwise, reject.

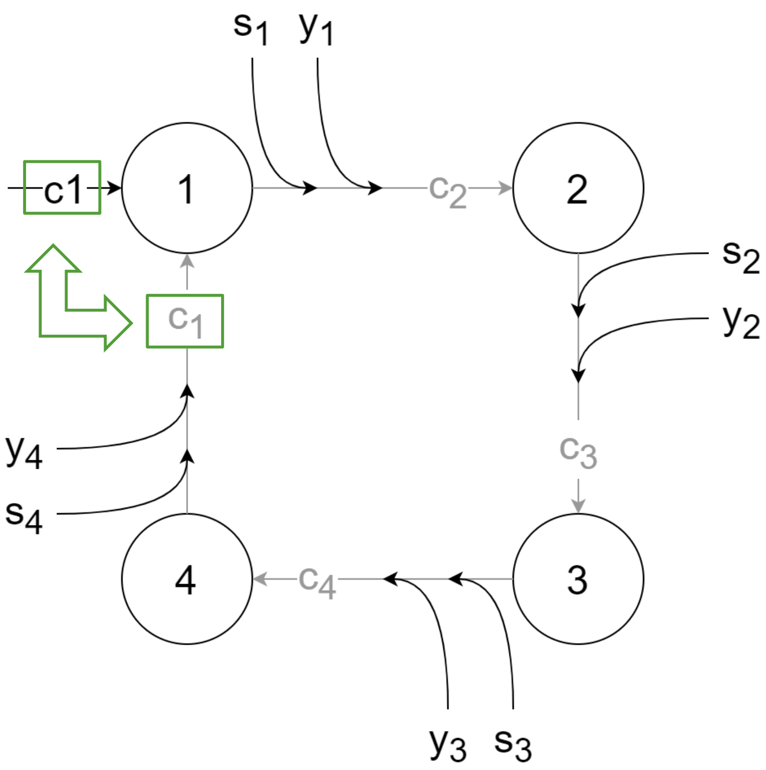
LSAG Algorithm III – Verification



A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z_i' = g^{s_i} y_i^{c_i}$, $z_i'' = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

LSAG Algorithm III – Verification



A public verifier checks a signature $\sigma_L(m) = (c_1, s_1, \cdots, s_n, \tilde{y})$ on a message m and a list of public keys L as follows.

1. Compute $h = H_2(L)$ and for $i = 1, \cdots, n$, compute $z_i' = g^{s_i} y_i^{c_i}$, $z_i'' = h^{s_i} \tilde{y}^{c_i}$ and then $c_{i+1} = H_1(L, \tilde{y}, m, z_i', z_i'')$ if $i \neq n$.
2. Check whether $c_1 \stackrel{?}{=} H_1(L, \tilde{y}, m, z_n', z_n'')$. If yes, accept. Otherwise, reject.

LSAG Algorithm IV – Linking

- Two signatures σ_1, σ_2 are signed by same signer when $\tilde{y}_1 = \tilde{y}_2$

For a fixed list of public keys L, given two signatures associating with L, namely $\sigma'_L(m') = (c'_1, s'_1, \cdots, s'_n, \tilde{y}')$ and $\sigma''_L(m'') = (c''_1, s''_1, \cdots, s''_n, \tilde{y}'')$, where m' and m'' are some messages, a public verifier after verifying the signatures to be valid, checks if $\boxed{\tilde{y}' = \tilde{y}''}$. If the congruence holds, the verifier concludes that the signatures are created by the same signer. Otherwise, the verifier concludes that the signatures are generated by two different signers.

Demonstration

- Implemented using C# and .NET Framework 6
- Functional tests with Xunit Framework
- Benchmarks using Benchmark.Net Framework

Demonstration

Performance I

Method	Ring Size [n]	Signature Size* [KB]	Mean [ms]	Error [ms]	StdDev [ms]	Allocated [KB]
SignMessage	10	44	107.9	1.95	1.63	246
VerifySignature	10	44	108.8	2.15	2.01	252
SignMessage	100	404	1,093.0	18.60	17.40	4,723
VerifySignature	100	404	1,119.7	4.76	4.22	4,723
SignMessage	1000	4,004	11,477.0	220.37	235.79	272,279
VerifySignature	1000	4,004	11,896.2	234.29	304.64	272,188

BenchmarkDotNet=v0.13.1, OS=Windows 10.0.22000
11th Gen Intel Core i7-11800H 2.30GHz, 1 CPU, 16 logical and 8 physical cores
* Theoretical number, actual size smaller as only necessary bytes are stored

Performance II

- Signature length: $|c_1| + |s_1| + \ldots + |s_n| + |y'| + |y_1| + \ldots + |y_n|$
- Signature length linear in ring size
- Computational complexity linear in ring size
- BUT: Implementation not optimized for performance

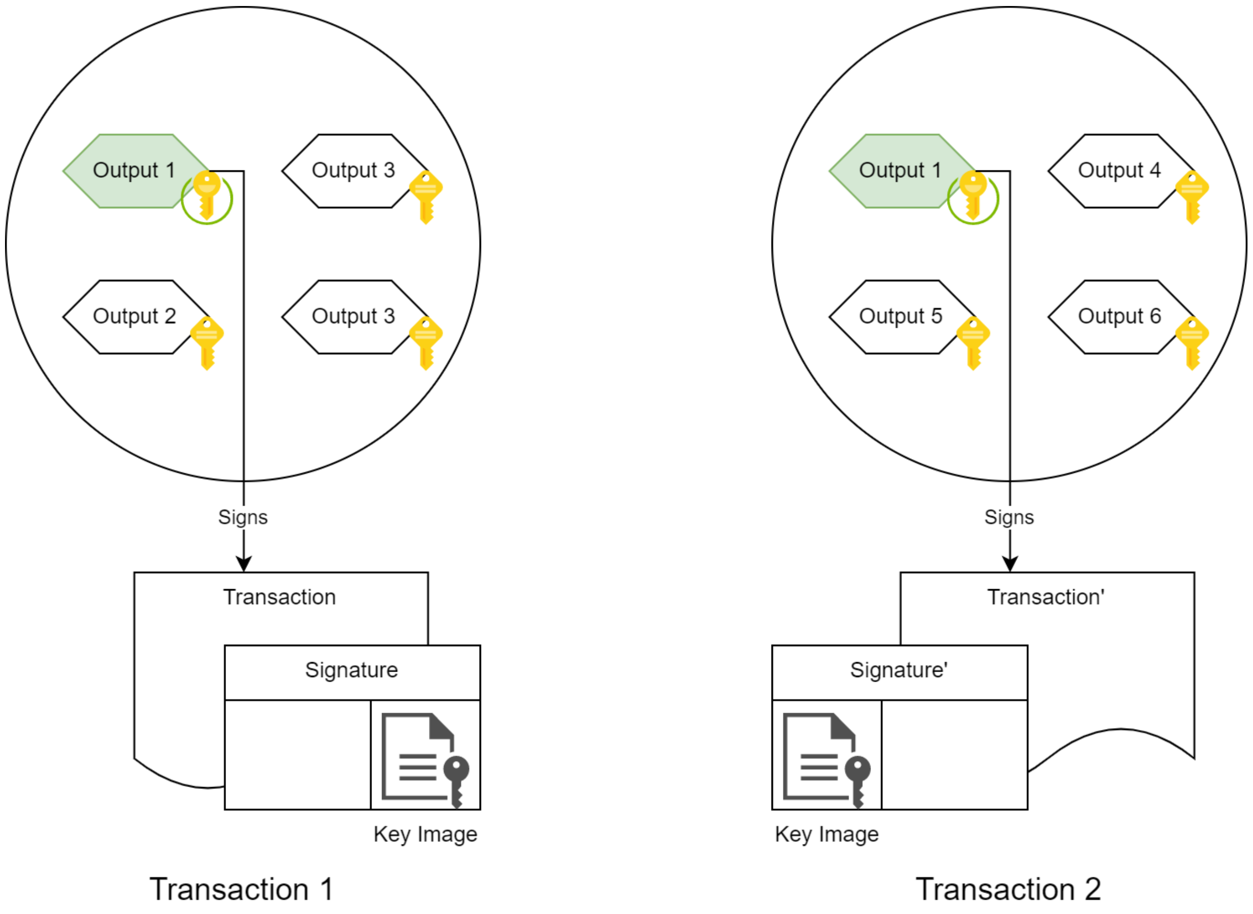
Applications I – Monero

- Generalization of LSAG: MLSAG
- Shen Noether, Monero Research Labs (2015)
- Works with multiple key *vectors* instead of multiple keys
- Prevents double spending

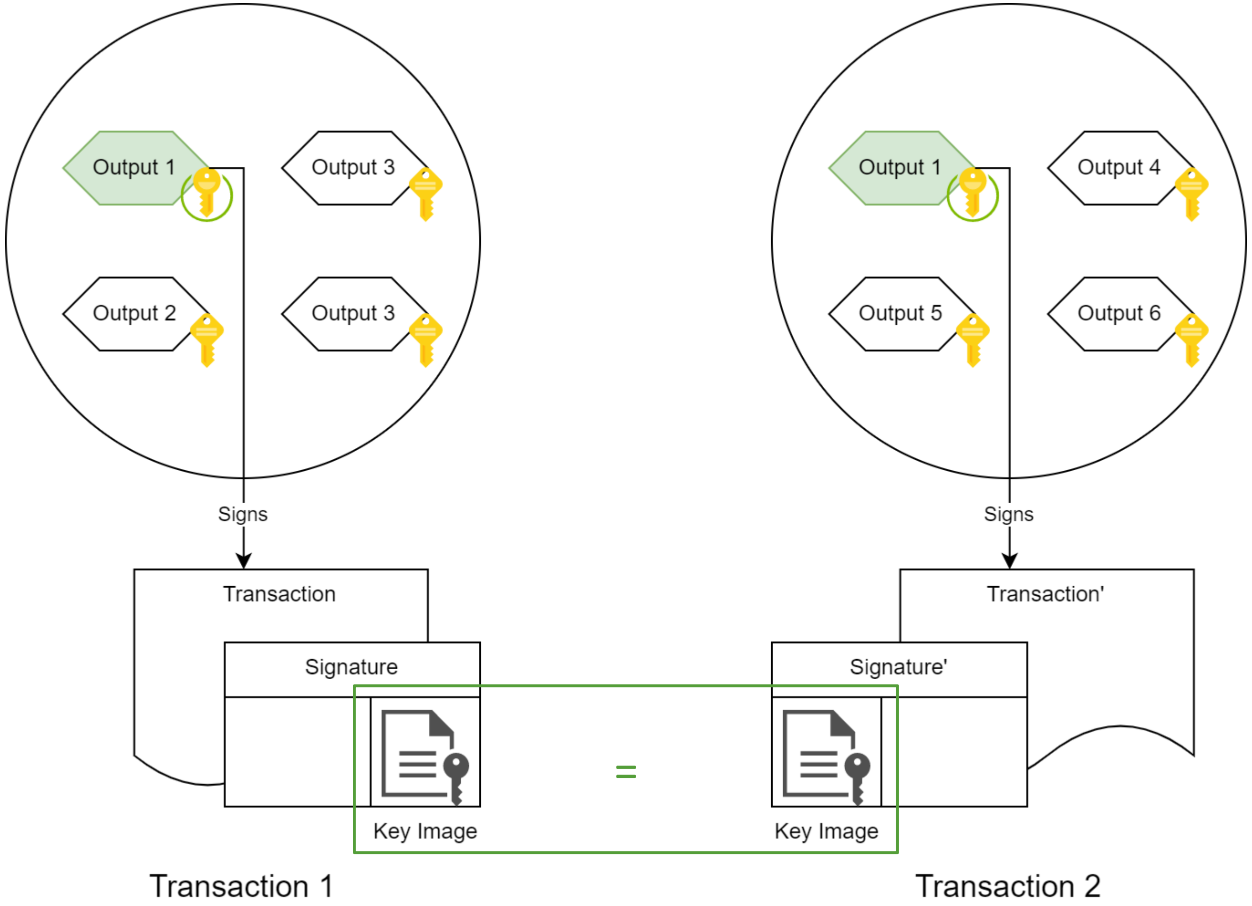
Applications I – Monero

1. Sender derives one-time receiver address
2. Sender sends money to that address
3. Receiver observes blockchain for transactions
4. If transaction is for him, uses private view key to unlock private spend key
5. Receiver uses spend key if he wants to send the money further

Applications I – Monero

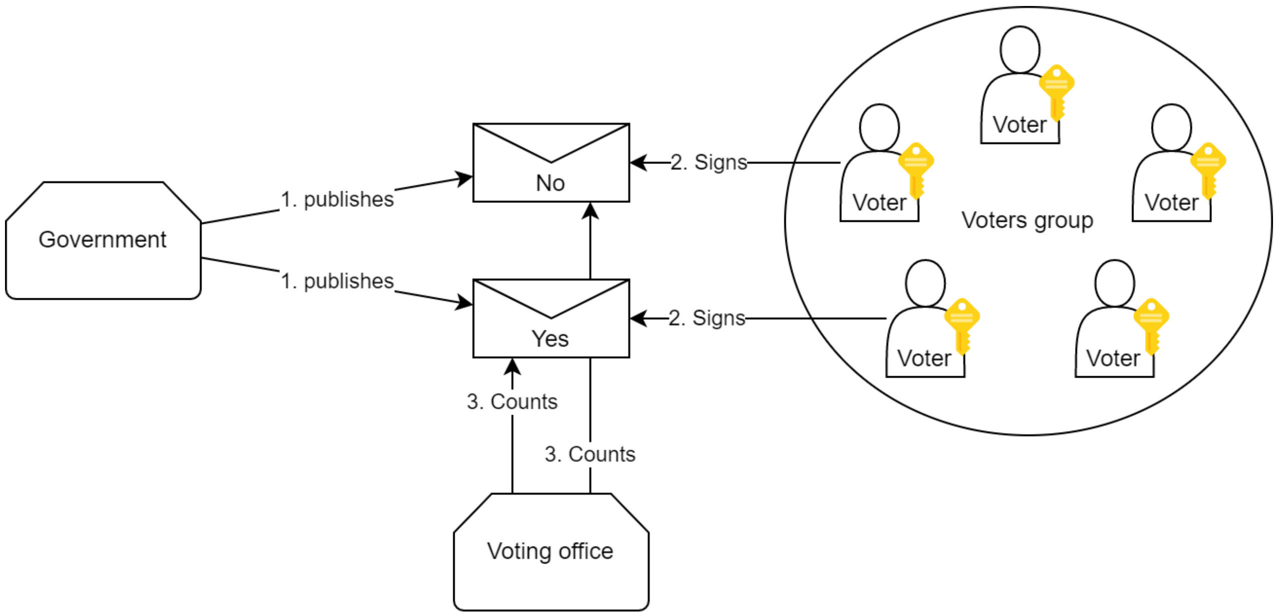


Applications I – Monero

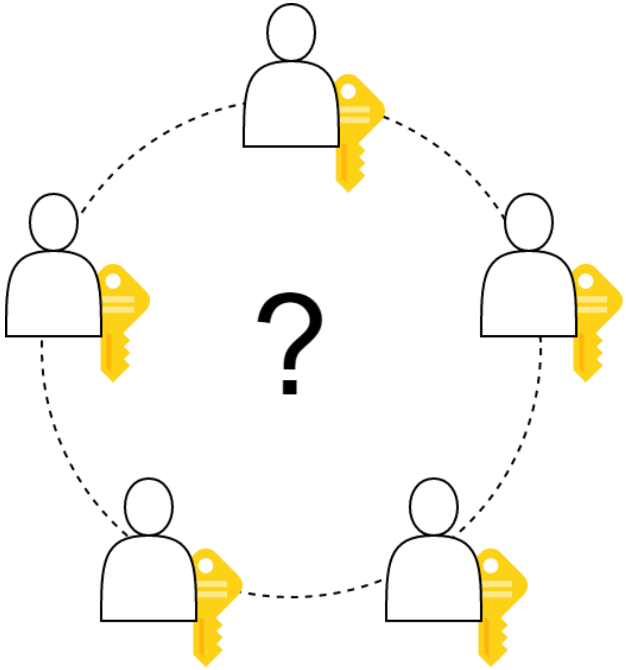


Applications II – E-Voting

- No Registration Phase
- Government issues m_{yes} and m_{no}
- Voter signs one message
- Multiple votes not possible



Discussion



References I

Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups
Joseph K. Liu, Victor K. Wei, and Duncan S. Wong
2004

How to Leak a Secret
Ronald L. Rivest, Adi Shamir, and Yael Tauman
in Advances in Cryptology – ASIACRYPT 2001 Proceedings
2001

CryptoNote WhitePaper v2.0
https://web.archive.org/web/20201028121818/https://cryptonote.org/whitepaper.pdf
Nicolas van Saberhagen
2013

Ring Confidential Transactions
https://eprint.iacr.org/2015/1098.pdf
Shen Noether
2015

References II

LSAG implementation
https://github.com/meggima/seminar-crypto-2022
Markus Eggimann
2022

LSAG implementation
https://github.com/sorrge/LSAG
sorrge
2013