Monero Transactions

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

September 23, 2019

Transactions in Monero

- Suppose Alice wants to spend coins from an address P she owns
- Alice assembles a list $\{P_0, P_1, \ldots, P_{n-1}\}$ where $P_j = P$ for exactly one j
- Alice knows x_j such that $P_j = x_j G$
- Key image of P_j is $I = x_j H_p(P_j)$ where H_p is a point-valued hash function
    - Distinct public keys will have distinct key images
- A linkable ring signature over $\{P_0, P_1, \ldots, P_{n-1}\}$ will have the key image I of P_j
    - Signature proves Alice one of the private keys
    - Double spending is detected via duplicate key images
- One cannot say if a Monero address belongs to the UTXO set or not

Linkable Spontaneous Anonymous Group Signatures

- Consider an elliptic curve group E with cardinality L and base point G
- Let $x_i \in \mathbb{Z}_L^*, i = 0, 1, \ldots, n-1$ be private keys with public keys $P_i = x_i G$
- Suppose a signer knows only x_j and not any of x_i for $i \neq j$
- The **key image** corresponding to P_j is $I = x_j H_p(P_j)$
- For a given message m, the signer generates the LSAG signature as follows:
  1. Picks $\alpha, s_i, i \neq j$ randomly from $\mathbb{Z}_L$
  2. Computes $L_j = \alpha G$, $R_j = \alpha H_p(P_j)$, and $c_{j+1} = H_s(m, L_j, R_j)$
  3. Increasing j modulo n, computes

$$\begin{aligned}
L_{j+1} &= s_{j+1} G + c_{j+1} P_{j+1} \\
R_{j+1} &= s_{j+1} H_p(P_{j+1}) + c_{j+1} I \\
c_{j+2} &= H_s(m, L_{j+1}, R_{j+1}) \\
&\vdots \\
L_{j-1} &= s_{j-1} G + c_{j-1} P_{j-1} \\
R_{j-1} &= s_{j-1} H_p(P_{j-1}) + c_{j-1} I \\
c_j &= H_s(m, L_{j-1}, R_{j-1})
\end{aligned}$$

  4. Computes $s_j = \alpha - c_j x_j \implies L_j = s_j G + c_j P_j, R_j = s_j H_p(P_j) + c_j I$
  5. The ring signature is $\sigma = (I, c_0, s_0, s_1, \ldots, s_{n-1})$
- Verifier computes L_j, R_j, remaining c_j's, and checks that $H_s(m, L_{n-1}, R_{n-1}) = c_0$
- Signatures with duplicate key images I will be rejected

LSAG Structure

- Rationale for choice of key image $I = x_j H_p(P_j)$
  - By collision resistance of H_p, I is unique for a given P_j
  - I does not reveal P_j as x_j is unknown to observers
  - Discrete log of $H_p(P_j)$ is unknown
- Comparison with regular ring signature calculation

$$\begin{aligned} L_{j+1} &= s_{j+1}G + c_{j+1}P_{j+1} \\ R_{j+1} &= s_{j+1}H_p(P_{j+1}) + c_{j+1}I \\ c_{j+2} &= H_s(m, L_{j+1}, R_{j+1}) \\ &\vdots \\ L_{j-1} &= s_{j-1}G + c_{j-1}P_{j-1} \\ R_{j-1} &= s_{j-1}H_p(P_{j-1}) + c_{j-1}I \\ c_j &= H_s(m, L_{j-1}, R_{j-1}) \end{aligned}$$

$$\begin{aligned} L_{j+1} &= s_{j+1}G + c_{j+1}P_{j+1} \\ c_{j+2} &= H_s(m, L_{j+1}) \\ &\vdots \\ L_{j-1} &= s_{j-1}G + c_{j-1}P_{j-1} \\ c_j &= H_s(m, L_{j-1}) \end{aligned}$$

Multilayered LSAG Signatures

- Consider a transaction which unlocks funds in m one-time addresses
  - Each LSAG signature is of the form $\sigma = (I, c_0, s_0, s_1, \ldots, s_{n-1})$ where n is the ring size
  - m LSAG signatures will take space $\mathcal{O}(m(n+2))$
- MLSAG signatures occupy space $\mathcal{O}(m(n+1))$
- MLSAG signatures are ring signatures over a set of n key-vectors
- Consider an $m \times n$ matrix of public keys

$$\begin{bmatrix} P_0^1 & P_1^1 & \cdots & P_\pi^1 & \cdots & P_{n-1}^1 \\ P_0^2 & P_1^2 & \cdots & P_\pi^2 & \cdots & P_{n-1}^2 \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ P_0^m & P_1^m & \cdots & P_\pi^m & \cdots & P_{n-1}^m \end{bmatrix}$$

  where the signer knows x_π^j such that $P_\pi^j = x_\pi^j G$ for $j = 1, 2, \ldots, m$
- For $I_j = x_\pi^j H_p(P_\pi^j)$, the MLSAG signature has the form $\sigma = (I_1, \ldots, I_m, c_0, s_0^1, \ldots, s_0^m, s_1^1, \ldots, s_1^m, s_{n-1}^1, \ldots, s_{n-1}^m)$

Deanonymization using Commitments

- Consider a confidential transaction which has two inputs and two outputs
- Suppose the sender uses a ring of size 5

$$\text{Public key matrix} = \begin{bmatrix} P_0^1 & P_1^1 & P_2^1 & P_3^1 & P_4^1 \\ P_0^2 & P_1^2 & P_2^2 & P_3^2 & P_4^2 \end{bmatrix}$$

  and knows private keys for P_2^1, P_2^2
- Let input commitments be

$$\begin{bmatrix} (C_{in})_0^1 & (C_{in})_1^1 & (C_{in})_2^1 & (C_{in})_3^1 & (C_{in})_4^1 \\ (C_{in})_0^2 & (C_{in})_1^2 & (C_{in})_2^2 & (C_{in})_3^2 & (C_{in})_4^2 \end{bmatrix}$$

- Let the output commitment be C_{out} and fees be f
- Observer can identify the sender column by checking for each $k = 0, 1, 2, 3, 4$ if

$$(C_{in})_k^1 + (C_{in})_k^2 = C_{out} + fH$$

- We need to prove that the commitments in a column add up without revealing the column

Monero RingCT

- Previously, to ensure $(C_{in})_\pi^1 + (C_{in})_\pi^2 = C_{out} + fH$, blinding factors need to be balanced
$$(x_{in})_\pi^1 + (x_{in})_\pi^2 = x_{out}$$
- Balancing needed only for third party verification of transactions
- For anonymization, we can set $(x_{in})_\pi^1 + (x_{in})_\pi^2 = x_{out} + z$ and communicate z to receiver using the shared secret
- How to enable third party verification?
- **Solution:** MLSAG using following public key matrix

$$\begin{bmatrix} P_0^1 & P_1^1 & P_2^1 & P_3^1 & P_4^1 \\ P_0^2 & P_1^2 & P_2^2 & P_3^2 & P_4^2 \\ \sum_{j=1}^2 (C_{in})_0^j - C_{out} - fH & & \cdots & & \sum_{j=1}^2 (C_{in})_4^j - C_{out} - fH \end{bmatrix}$$

- A signature verifiable using a public key in the last row implies knowledge of corresponding z

References

- Ring Confidential Transactions http://www.ledgerjournal.org/ojs/index.php/ledger/article/view/34
- LSAG, Part 6 of Monero's Building Blocks Articles https://delfr.com/wp-content/uploads/2018/04/Monero_Building_Blocks_Part6.pdf
- MLSAG, Part 7 of Monero's Building Blocks Articles https://delfr.com/wp-content/uploads/2018/05/Monero_Building_Blocks_Part7.pdf
- RingCT, Part 9 of Monero's Building Blocks Articles https://delfr.com/wp-content/uploads/2018/04/Monero_Building_Blocks_Part9.pdf