Monero RingCT explained

Oleg Fomenko
Distributed Lab

June 2024

Special thanks to Alex Kurbatov for his kind review.

1 Introduction

The evolution of blockchain technology includes a huge variety of approaches to maximize privacy and anonymity during the assets transfer. Clearly, true privacy must rely on strong time-proven cryptography and decentralization to achieve zero information sharing about the sender, receiver, and the amount of assets transferred. These requirements, combined with the default transparency of the blockchain, produces a real challenge for developers to create a system that users can trust their money. So, it is not surprisingly that a lot of privacy solution exists.

Monero – a cryptocurrency focused on privacy, employs a technology called **Ring Confidential Transactions** (RingCT) [Noe15] to enhance the anonymity of its users. Built on top of Confidential assets and Bitcoin UTXO model, RingCT ensures that transaction amounts in UTXOs are hidden and sender-receiver identities remain untraceable, setting Monero apart from other cryptocurrencies like Bitcoin. By using a combination of ring signatures and stealth addresses, RingCT allows private and secure transfers without revealing any transaction details to the outside observers.

In this paper, we describe the primitives on which the RingCT protocol relies. It covers the Pedersen commitments, Confidential assets, stealth addresses, ring signatures and the form of RingCT transaction.

2 Preliminaries

Notation. We denote by $\mathbb{G}$ a cyclic group of prime order p written additively. We write the elements in $\mathbb{G}$ with capital letters $(G, H, ...)$ and scalars in $\mathbb{F}_p$ with lower case letters $(a, b, c, ...)$.

Discrete Logarithm Relation Problem. The discrete logarithm relation problem in $\mathbb{G}$ is *hard* if for a randomly chosen $a \in \mathbb{F}_p$, such that $A = aG$, given the point A, the probability of finding a from A is negligible.

2.1 Commitment

Commitment scheme is an approach to encode some data that will be stored hidden until user decides to unhide it. Commitment scheme in general is defined as follows:

- $Com(x, r) \to C$ accepts on the input value x and user randomness (blinding) r and outputs value C that is referred to *commitment*.

A commitment scheme is called secure if it satisfies the following properties:

- *Blinding*: For the commitment scheme it is hard to find such two pairs x_1, r_1 and x_2, r_2 that $Com(x_1, r_1) = Com(x_2, r_2)$.
- *Hiding*: While r is generated with a proper randomness source, a commitment C does to reveal any information about x (even if x can be guessed with high probability).

The simplest example of the cryptographic commitment scheme is a hash function $\mathbf{H}$:

$$Com(x, r) = \mathbf{H}(x||r)$$

2.2 Pedersen commitment

The **Pedersen commitment** [Ped91] schemes is a cryptographic primitive that allows to commit to the scalar value in the additive group element. It is defined with respect to the two fixed additive group elements $G, H \in \mathbb{G}$ as follows:

$$Com(x, r) = xH + rG \in \mathbb{G}$$

where $r \in \mathbb{F}_p$ is a commitment blinding (big random scalar), $x \in \mathbb{F}_p$ – committed value, $G, H \in \mathbb{G}$ – group generators.

While defined over the additive group, Pedersen commitments allow addition between committed values that holds binding and hiding properties:

$$Com(x_1, r_1) + Com(x_2, r_2) = Com(x_1 + x_2, r_1 + r_2)$$

2.3 Pedersen commitment security

Pedersen commitment satisfies both the security properties of binding and hiding for the robust additive group $\mathbb{G}$ where the discrete logarithm relation problem is hard.

Also, while utilizing two different generators to construct commitments it is required that nobody knows the relation between generators H and G (such k that $kG = H$). Otherwise, it can be used to manipulate committed value as follows:

$$xH + rG = x \cdot kG + rG = (x + 1)kG + (r - k)G = (x + 1)H + (r - k)G$$

So, for any selected G point, each protocol using Pedersen commitment must define H and prove that it was chosen independently (aka *Nothing-up-my-sleeve number*). For example in Monero, authors utilizes the following equation:

$$H = 8 * to\_point(\mathbf{H}(G))$$

where $\mathbf{H}$ is a special hash-function defined as $\mathbf{H} : \mathbb{G} \to \mathbb{F}_p$ and *to_point* is a special function that maps a scalar into the curve point (for example by setting the scalar as x-coordinate and calculating y-coordinate). Multiplication on 8 (size of smaller curve subgroup) is used to ensure that resulting point lies in the bigger subgroup of the Ed25519 [Ber+08] curve.

3 Confidential assets

The Confidential assets [Poe+16] introduced a method enabling users to conceal the amounts of inputs and outputs in their UTXOs. By combining Pedersen commitments and signature aggregation, it becomes feasible to hide balances while still verifying that the value in transaction inputs matches that of the outputs.

3.1 UTXO

Launched in 2009, Bitcoin, as the pioneer of blockchain systems, introduces the concept of Unspent Transaction Outputs (UTXO) for the storage organization. Under this approach, each UTXO stores the output of a transaction that remains unspent, comprising a certain asset amount and the conditions for spending. When a transaction occurs, it consumes one or more UTXOs as inputs and creates new UTXOs as outputs, effectively transferring ownership of tokens. Also, during the transaction, it is important to verify that the following two requirements are met:

1. The sum of UTXO at the output is the same (or less) than at the input.
2. All input UTXOs are authorized for spending and have not been spent previously.

3.2 Hidden amounts

In general, for the given token amount a user owns a commitment stored in UTXO of form:

$$Com(a, r) = aH + rG = C \in \mathbb{G}$$

Because of the additive nature of Pedersen commitments it becomes possible to perform addition and subtraction operations on them. This allows us to prove the correctness of the transaction without revealing of any information about amounts. In particular, we can check that sum of the input amounts in transaction is equal to the sum of outputs by calculating

$$\sum_{C_i \in in} C_i - \sum_{C_j \in out} C_j = (a_{in} - a_{out})H + (r_{in} - r_{out})G$$

If the transaction is correct, then

$$(a_{in} - a_{out})H + (r_{in} - r_{out})G = 0H + (r_{in} - r_{out})G$$

Now, it becomes possible for the sender and receiver to prove the knowledge of r_{in} and r_{out} by generation of the aggregated signature for $(r_{in} - r_{out})G$ public key. Note, that this signature can be produced only in case of correct amounts and knowledge of the openings for the commitments, otherwise the probability to broke the protocol is equal to the discrete logarithm problem.

3.3 Range proofs

Unfortunately, it is not enough to produce only an aggregated signature to prove the transaction correctness. Let's observe a simple example: for the group with order $p = 10$ let's take $C_1 = 5H + r_1G$ for the input and $C_2 = 9H + r_2G, C_3 = 6H + r_3G$ for the outputs. It's easy to check that difference between input and output commitments produces $0H$ term because of the group law and then it is possible to generate an aggregated signature while the transaction is still incorrect.

To solve this problem sender have to generate a **zero-knowledge range-proofs** that all amounts in the commitments lies in some range. *Zero-knowledge* means that no information about amounts will be revealed. The range for proving should be selected in a such way that it will not be possible to generate such transaction where $\sum_{a_i \in amounts} a_i >= p$. For example, while working in the 2^{256} field in often enough to prove that all amount lies in $[0; 2^{64})$ range.

The original Confidential assets used to implement Back-Maxwell range-proof protocol, but for the modern systems it is preferred to use more effective protocols such as Bulletproofs protocols family.

4 Ring signatures

The **ring signature** is a signature protocol in which, using a set of public keys (ring) R, some of its members can sign the message m without revealing any information about who exactly it was. It is hard to underestimate the importance of such a protocol in providing anonymity to users when transferring assets.

The simplest ring signature scheme[RST01] uses encryption protocols (RSA[RSA77] for example) to prove that bitwise xor over all encryption of provided scalars array is equal to zero (according to this protocol, one of the element in an array for the corresponding known private key will be calculated to satisfy our xor equation). More complex and flexible ring signatures can be built as a modification of Schnorr identification protocol[Sch89].

Let's describe the form of Schnorr signature that will be later modified:

Function 1 Schnorr signature protocol

Input: Signer private key k, public key $K = kG$, message m.

Proving:

Select random key $a \leftarrow \mathbb{F}_p$ and public $A = aG$

Calculate challenge $c = \mathbf{H}(m, A)$

Put the response $r = a - kc$

Put the signature as (c, r)

Verification:

Recover challenge $c' = \mathbf{H}(m, [rG + cK])$

Check $c = c'$

4.1 SAG signature

The most trivial ring signature protocol built on top of Schnorr identification protocol is a **Spontaneous Anonymous Group (SAG) signature** [LWW04]. Firstly, let's define the ring as $R = \{K_1, K_2, ..., K_d\}$ where we know the private key k_π of the key with secret position π. Then, to generate the signature for the message m where the signer belongs to the ring R we will go through the following protocol:

Function 2 SAG signature protocol

Input: Ring $R = \{K_1, K_2, ..., K_d\}$, private key k_π, public key $K_\pi = k_\pi G$ on the secret position π in the ring, message m.

Proving:

Select random key $a \leftarrow \mathbb{F}_p$ and $r_i \leftarrow \mathbb{F}_p$ for $\forall i \neq \pi$

Put $c_{\pi+1} = \mathbf{H}(R, m, [aG])$

$\forall i = \pi + 1, \pi + 2, ..., d, 1, ..., \pi - 1$ (replacing $d + 1 \to 1$)
calculate $c_{i+1} = \mathbf{H}(R, m, [r_i G + c_i K_i])$

Put the response $r_\pi = a - c_\pi k_\pi$

Put the signature $(c_1, r_1, ..., r_d)$ and the ring R

Verification:

$\forall i = 1, 2, ..., d$ (replacing $d + 1 \to 1$)
calculate $c'_{i+1} = \mathbf{H}(R, m, [r_i G + c_i K_i])$

Check $c_1 = c'_1$.

4.2 bLSAG signature

Linkability is a property that describes relation between two signatures. For the protocols with such property it becomes possible to check that two different signatures have been signed with the same public keys. Linkability in a couple with *anonymity* property gives verifier an opportunity to check this relation without revealing any information about signer.

Back's Linkable Spontaneous Anonymous Group (bLSAG) signature protocol, as a modification of the described in previous post SAG protocol, introduces the ring signature witch follows

anonymity and linkability properties. Before going thought the protocol let's describe the special hash function $\mathbf{H_p}(x) \to \mathbb{G}$ that gives as the result a point in curve (Discrete-Log problem can not be solved with overwhelming probability). Then, for the given ring $R = \{K_1, K_2, ..., K_d\}$ where we know the private key k_π of the key with secret position π, bLSAG protocol can be defined as follows:

Function 3 bLSAG signature protocol

Input: Ring $R = \{K_1, K_2, ..., K_d\}$, private key k_π, public key $K_\pi = k_\pi G$ on the secret position π in the ring, message m.

Proving:

Calculate key image $\hat{K} = k_\pi \cdot \mathbf{H_p}(K_\pi)$.

Select random key $a \leftarrow \mathbb{F}_p$ and $r_i \leftarrow \mathbb{F}_p$ for $\forall i \neq \pi$

Put $c_{\pi+1} = \mathbf{H}(m, [aG], [a\mathbf{H_p}(K_\pi)])$

$\forall i = \pi + 1, \pi + 2, ..., d, 1, ..., \pi - 1$ (replacing $d + 1 \to 1$)
calculate $c_{i+1} = \mathbf{H}(m, [r_i G + c_i K_i], [r_i \mathbf{H_p}(K_i) + c_i \hat{K}])$

Put the response $r_\pi = a - c_\pi k_\pi$

Put the signature $(c_1, r_1, ..., r_d)$, ring R and key image $\hat{K}$

Verification:

Check $p \cdot \hat{K} = 0$, where $p = |\mathbb{G}|$

$\forall i = 1, 2, ..., d$ (replacing $d + 1 \to 1$)
calculate $c'_{i+1} = \mathbf{H}(m, [r_i G + c_i K_i], [r_i \mathbf{H_p}(K_i) + c_i \hat{K}])$

Check $c_1 = c'_1$.

Finally, if the two different signatures (even with different rings) have been produced by the same signer then the both will have the same key images $\hat{K}$. Note, that verification of $p \cdot \hat{K} = 0$ is necessary to ensure that point belongs to $\mathbb{G}$ group. If this relation is not satisfied it means that passed point belongs to the other curve subgroup, which may affect the linkability property.

4.3 MLSAG signature

The **Multilayer Linkable Spontaneous Anonymous Group (MLSAG) signature** protocol [Noe15] provides an opportunity to generate signature with several signers while they are all still hidden in the ring. So, given the ring $R = \{K_{i,j}\}$ for $i \in \{1, 2, ...n\}$ and $j \in \{1, 2, ...m\}$ where we know the private keys $k_{\pi,j}$ for corresponding public keys $K_{\pi,j}$ with secret position π for $j \in \{1, 2, ...m\}$. Then, the MLSAG protocol has a lot in common with bLSAG protocol:

Function 4 MLSAG signing protocol

Input: $R = \{K_{i,j}\}$ for $i \in \{1, 2, ...n\}$ and $j \in \{1, 2, ...m\}$, private keys $k_{\pi,j}$, public keys $K_{\pi,j} = k_{\pi,j}G$.

Proving:

Calculate key image $\hat{K}_j = k_{\pi,j}\mathbf{H_p}(K_{\pi,j})$ for $j \in \{1, 2, ...m\}$

Select random key $a_j \leftarrow \mathbb{F}_p$ for $\forall j \in \{1, 2, ...m\}$

For $i \in \{1, 2, ...n\} \setminus \pi$ and $j \in \{1, 2, ...m\}$ select random $r_{i,j} \leftarrow \mathbb{F}_p$

Put $c_{\pi+1} = \mathbf{H}(m, [a_1 G], [a_1 \mathbf{H_p}(K_{\pi,1})], [a_2 G], [a_2 \mathbf{H_p}(K_{\pi,2})], ...)$

$\forall i = \pi + 1, \pi + 2, ..., d, 1, ..., \pi - 1$ (replacing $d + 1 \to 1$)
calculate $c_{i+1} = \mathbf{H}(m, [r_{i,1}G + c_i K_{i,1}], [r_{i,1}\mathbf{H_p}(K_{i,1}) + c_i \hat{K}_1], ...)$

Put response $r_{\pi,j} = a - c_\pi k_{\pi,j}$ for $j \in \{1, 2, ...m\}$

Put signature $(c_1, r_{i,j})$, ring R and key images $\hat{K}_j$

Verification:

Check $\forall j \in \{1, 2...m\} : p \cdot \hat{K}_j = 0$, where $p = |\mathbb{G}|$

$\forall i = 1, 2, ..., d$ (replacing $d + 1 \to 1$)
calculate $c'_{i+1} = \mathbf{H}(m, [r_{i,1}G + c_i K_{i,1}], [r_{i,1}\mathbf{H_p}(K_{i,1}) + c_i \hat{K}_1], ...)$

Check $c_1 = c'_1$.

Finally, if for the two different signatures S_1 and S_2 such indexes i, j exist that $\hat{K}_{S_1,i} = \hat{K}_{S_2,j}$ then these signatures are linked by signing with the same key.

In addition, using signatures linking and one-time stealth addresses Monero blockchain solves a double-spending problem: if transaction contains the key image that already has been included into the any block before then in seems to be a double-spending attack.

5 Stealth addresses

Stealth addresses [NMT16] is an approach to hide transaction receiver by calculating the one-time receiver address. In general, receiver can share with sender two public keys ($K_v = k_v G, K_s = k_s G$) – view key and spend key. Then, sender during the process of transaction creation will select a random value r that generates a one-time recipient address by:

$$K_O = \mathbf{Hash}(rK_v)G + K_s$$

Additionally, rG will be added to the transaction extra data and referred to the *transaction public key*. Receiver, has to monitor all transactions and by utilizing the public transaction data (K_O and rG) check if:

$$K_s =? K_O - \mathbf{Hash}(k_v rG)G$$

For the transactions that satisfied this relation user calculates the private key for the coins spending:

$$k_O = \mathbf{Hash}(k_v rG) + k_s$$

The security of this protocol lies on the discrete logarithm problem hardness as in the Diffie-Hellman protocol, so nobody even with the knowledge of the sender and receiver public keys can match the transaction between them.

6 RingCT transaction

In the following section we mention C^a the Pedersen commitment referred to the amount a:

$$C^a = aH + xG$$

RingCT [Noe15] transaction utilized the model where the user's balances is stored in pair of public key and commitment $\langle K^a, C^a \rangle$ over UTXOs. So, imagine user wants to transfer some coins using output from other transaction $\langle K_i^a, C_i^a \rangle$ where $i \in \{1, ..., m\}$ for the input and $\langle K_j^b, C_j^b \rangle$ where $j \in \{1, ..., p\}$ for the output (K_i^b is a receiver one-time address and $C_j^b = b_j H + y_j G$).

To achieve additional confidentiality, user generates m *pseudo-output commitments* $\hat{C}_i^a$ with same amounts but different blinding in a such way that

$$\sum \hat{x}_i - \sum y_j = 0$$

It's obvious that using such construction $\sum \hat{C}_i^a - \sum C_j^b = 0$, so we can convince the verifier that sum of input coins equals to the output. Also, note that for every i sender knows the private key for zero-value commitment $C_i^a - \hat{C}_i^a = (x_i - \hat{x}_i)G = z_i G$. Then, for every input i sender selects a random ring of size $v + 1$ of form:

$$\begin{gathered} R_i = \{\{K_{1,i}, [C_{1,i} - \hat{C}_{\pi,i}^a]\}, \\ ..., \\ \{K_{\pi,i}, [C_{\pi,i}^a - \hat{C}_{\pi,i}^a]\}, \\ ..., \\ \{K_{v+1,i}, [C_{v+1,i} - \hat{C}_{\pi,i}^a]\}\} \end{gathered}$$

User can generate a MLSAG signature using secrets for π position: $k_{\pi,j}$ for public key $K_{\pi,j}$ and z_j for zero-value commitment $C_{\pi,i}^a - \hat{C}_{\pi,i}^a = (x_{\pi,i} - \hat{x}_{\pi,i})G$. Also, sender attaches the key image for his key $K_{\pi,i}$. Because this key should be a one-time address we can consider that it can be used only once. So, if there is any included into the block transaction exist with same key image then we faced the try of double-spending of some output. Finally, the RingCT transaction can be built as follows:

- **type**: *RCTTypeBulletproof2*
- **inputs**: *for each input* $i \in \{1...m\}$*:*
  Ring R members
  MLSAG signature
  Key image
  Pseudo output $\hat{C}_i^a$
- **outputs**: *for each output* $j \in \{1...p\}$*:*
  One time address K_j^O
  Output commitment C_j^b
  Encrypted amount (see docs for more info)
  Range proof that committed amount lies in $[0..2^{64})$ *range*
- **fee**
- **extra**: *Transaction public key rG, etc.*

In conclusion, to generate a RingCT transaction, for every input that sender spends, he creates a random ring and selects the pseudo-output commitment with the same amount. Because of the knowledge of randomness for the input and pseudo-output commitments, sender can generate a separate ring signature for every ring and also, prove the range for the outputs. *Encrypted amount* in transaction outputs is used by sender to share the commitment blinding with recipient in such way that only with knowledge of view key and amount it can be decrypted.

References

[RSA77] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. *The Original RSA Patent as filed with the U.S. Patent Office*. 1977. URL: https://patents.google.com/patent/US4405829.

[Sch89] Claus P. Schnorr. *The Original Schnorr authentication protocol Patent as filed with the U.S. Patent Office*. 1989. URL: https://patents.google.com/patent/US4995082.

[Ped91] Torben Pryds Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. 1991. URL: https://link.springer.com/content/pdf/10.1007/3-540-46766-1_9.pdf.

[RST01] Ronald L. Rivest, Adi Shamir, and Y. Tauman. "How to leak a secret". In: *Advances in Cryptology — ASIACRYPT 2001. Lecture Notes in Computer Science. Vol. 2248. pp. 552–565* (2001).

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. *Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups*. 2004. URL: https://eprint.iacr.org/2004/027.pdf.

[Ber+08] Daniel J. Bernstein et al. *Twisted Edwards Curves*. 2008. URL: https://eprint.iacr.org/2008/013.pdf.

[Noe15] Shen Noether. *Ring Confidential Transactions*. 2015. URL: https://eprint.iacr.org/2015/1098.pdf.

[NMT16] Shen Noether, Adam Mackenzie, and Monero Core Team. *Ring Confidential Transactions*. 2016. URL: https://www.getmonero.org/es/resources/research-lab/pubs/MRL-0005.pdf.

[Poe+16] Andrew Poelstra et al. *Confidential Assets*. 2016. URL: https://blockstream.com/bitcoin17-final41.pdf.