BUNNYTN 7 2016

Monero vs Bitcoin

Francesco Romeo

Università degli Studi di Messina

16 November 2016, Trento

Summary of Presentation

Summary of Presentation

- Reasons and Reviews

Summary of Presentation

- Reasons and Reviews
- Monero

Summary of Presentation

- Reasons and Reviews
- Monero
- Monero vs Bitcoin

Reasons and Reviews

Reasons and Reviews

Digital Signatures

Digital Signatures

The document is signed through the Signing Algorithm.

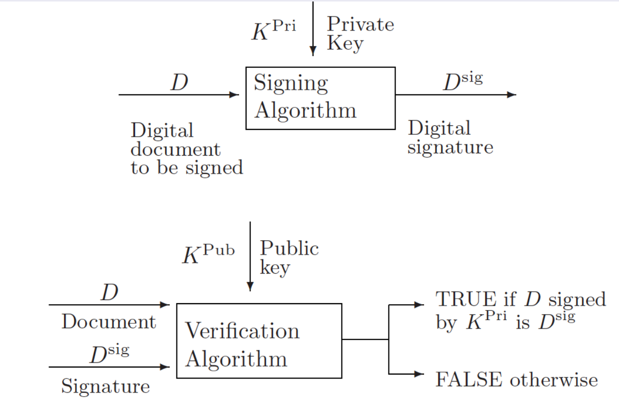
Reasons and Reviews

Digital Signatures

The document is signed through the Signing Algorithm.
The Signature is verified through the Verification Algorithm

Digital Signatures

The document is signed through the Signing Algorithm.
The Signature is verified through the Verification Algorithm



Reasons and Reviews

Hash Functions

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \to \{0,1\}^k \qquad \text{k fixed}$$

Reasons and Reviews

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D)$

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D) \Rightarrow$ **One-Way Function**

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D) \Rightarrow$ **One-Way Function**
- If $D \neq D'$,

Reasons and Reviews

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D) \Rightarrow$ **One-Way Function**
- If $D \neq D'$, then

Reasons and Reviews

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D)$ $\Rightarrow$ **One-Way Function**
- If $D \neq D^{'}$, then $H(D) \neq H(D^{'})$

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H : \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D)$ $\Rightarrow$ **One-Way Function**
- If $D \neq D'$, then $H(D) \neq H(D')$

There are some Hash Algorithms that take as input an Elliptic Curve Point and return another point. (i. e. a Keccak Algorithm)

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D) \Rightarrow$ **One-Way Function**
- If $D \neq D'$, then $H(D) \neq H(D')$

There are some Hash Algorithms that take as input an Elliptic Curve Point and return another point. (i. e. a Keccak Algorithm)

Kinds of Signatures

Reasons and Reviews

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D)$ $\Rightarrow$ **One-Way Function**
- If $D \neq D'$, then $H(D) \neq H(D')$

There are some Hash Algorithms that take as input an Elliptic Curve Point and return another point. (i. e. a Keccak Algorithm)

Kinds of Signatures

- **Elliptic Curve Digital Signature Algorithm (ECDSA)**:

Reasons and Reviews

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D) \Rightarrow$ **One-Way Function**
- If $D \neq D'$, then $H(D) \neq H(D')$

There are some Hash Algorithms that take as input an Elliptic Curve Point and return another point. (i. e. a Keccak Algorithm)

Kinds of Signatures

- **Elliptic Curve Digital Signature Algorithm (ECDSA)**: important in Cryptocurrencies transactions

Reasons and Reviews

Hash Functions

Let $\mathcal{D}$ be the set of the documents,

$$H: \ \mathcal{D} \rightarrow \{0,1\}^k \quad \text{k fixed}$$

- Not possible to recover D from $H(D) \Rightarrow$ **One-Way Function**
- If $D \neq D'$, then $H(D) \neq H(D')$

There are some Hash Algorithms that take as input an Elliptic Curve Point and return another point. (i. e. a Keccak Algorithm)

Kinds of Signatures

- **Elliptic Curve Digital Signature Algorithm (ECDSA)**: important in Cryptocurrencies transactions
- **Ring Signatures**

Reasons and Reviews

Ring Signatures

Ring Signatures

- Performed by a group of people

Reasons and Reviews

Ring Signatures

- Performed by a group of people
- n members

Ring Signatures

- Performed by a group of people
- n members $\Rightarrow$ n pairs (P_i, S_i)

Ring Signatures

- Performed by a group of people
- n members $\Rightarrow$ n pairs (P_i, S_i) Public and Secret Keys

Reasons and Reviews

Ring Signatures

- Performed by a group of people
- n members $\Rightarrow$ n pairs (P_i, S_i) Public and Secret Keys
- Only one member produces the signature on the document,

Ring Signatures

- Performed by a group of people
- n members $\Rightarrow$ n pairs (P_i, S_i) Public and Secret Keys
- Only one member produces the signature on the document, but it contains all the public parameters P_i.

Ring Signatures

- Performed by a group of people
- n members $\Rightarrow$ n pairs (P_i, S_i) Public and Secret Keys
- Only one member produces the signature on the document, but it contains all the public parameters P_i.

In application, i.e. in Monero, we use **One-Time** keys to perform Ring Signatures:

Ring Signatures

- Performed by a group of people
- n members $\Rightarrow$ n pairs (P_i, S_i) Public and Secret Keys
- Only one member produces the signature on the document, but it contains all the public parameters P_i.

In application, i.e. in Monero, we use **One-Time** keys to perform Ring Signatures: the private key used to sign the transaction generates a **Residue Image**, that is unique.

Reasons and Reviews

Centralization Problem

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block
-**Mining**: activity of Nodes that add transactions solving Hash Algorithms

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block
-**Mining**: activity of Nodes that add transactions solving Hash Algorithms
-**Wallet**: where money are saved.

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block
-**Mining**: activity of Nodes that add transactions solving Hash Algorithms
-**Wallet**: where money are saved.
Each Cryptocurrency has a specific Elliptic Curve to perform ECDSA on Transactions.

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block
-**Mining**: activity of Nodes that add transactions solving Hash Algorithms
-**Wallet**: where money are saved.
Each Cryptocurrency has a specific Elliptic Curve to perform ECDSA on Transactions. In general we need a curve over a Finite Field

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block
-**Mining**: activity of Nodes that add transactions solving Hash Algorithms
-**Wallet**: where money are saved.

Each Cryptocurrency has a specific Elliptic Curve to perform ECDSA on Transactions. In general we need a curve over a Finite Field and a **base point** G

Reasons and Reviews

Centralization Problem

The necessity of a central authority that controls money is a potential weakness of this system.
The authority could double existing cash, halving everyone's wealth.
It would lead to runaway inflation.

New Frontier: Crytpocurrencies

-They are decentralized: **Peer-to-Peer** system
-**Blockchain**: transactions in blocks linked each other
-**Blocksize**: "weight" in byte of block
-**Mining**: activity of Nodes that add transactions solving Hash Algorithms
-**Wallet**: where money are saved.
Each Cryptocurrency has a specific Elliptic Curve to perform ECDSA on Transactions. In general we need a curve over a Finite Field and a **base point** G of **prime** order l.

Monero

Definition

Monero

Definition

In Esperanto, **mono** (money)

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit),

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519**

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:
-Defined on $\mathbb{F}_q$,

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:
-Defined on $\mathbb{F}_q$, with $q = 2^{255} - 19$;

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:
-Defined on $\mathbb{F}_q$, with $q = 2^{255} - 19$;
-It has Equation $y^2 = x^3 + 486662x^2 + x$

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:
-Defined on $\mathbb{F}_q$, with $q = 2^{255} - 19$;
-It has Equation $y^2 = x^3 + 486662x^2 + x$ so it's a Montgomery Elliptic Curve;

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:
-Defined on $\mathbb{F}_q$, with $q = 2^{255} - 19$;
-It has Equation $y^2 = x^3 + 486662x^2 + x$ so it's a Montgomery Elliptic Curve;
-Base Point $G = (9, 1478161944758954479102059356840998688$
$7264606134616475288964881837755586237401)$;

Monero

Definition

In Esperanto, **mono** (money) and **ero** (bit), it is a Cryptocurrency born in 2014.
It is based on CryptoNote a digital protocol that powers decentralized privacy oriented digital currencies.

Monero Parameters

Monero uses **Curve25519** with the following parameters:
-Defined on $\mathbb{F}_q$, with $q = 2^{255} - 19$;
-It has Equation $y^2 = x^3 + 486662x^2 + x$ so it's a Montgomery Elliptic Curve;
-Base Point $G = (9, 14781619447589544791020593568409986887264606134616475288964881837755586237401)$;
-Order of G, $l = 2^{252} + 27742317777372353535851937790883648493$.

Monero

Transactions in Monero

Transactions in Monero

It uses **Ring Confidential Transactions**

Monero

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature,

Monero

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG),

Monero

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses *n* vectors of *m* Public Keys instead of *n* Public Keys (for LSAG), where *n* are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**:

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;
-**SIGN**:

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;
-**SIGN**: producing the signature;

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;
-**SIGN**: producing the signature;
-**VER**:

Monero

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses *n* vectors of *m* Public Keys instead of *n* Public Keys (for LSAG), where *n* are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;
-**SIGN**: producing the signature;
-**VER**: verification of the signature;

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;
-**SIGN**: producing the signature;
-**VER**: verification of the signature;
-**LNK**:

Monero

Transactions in Monero

It uses **Ring Confidential Transactions** and we want the amount of transactions to be hidden.
This is achieved by using the **Multilayered Linkable Spontaneous Anonimous Group** (MLSAG) Ring Signature, based on the previous LSAG.
The main difference is that MLSAG uses n vectors of m Public Keys instead of n Public Keys (for LSAG), where n are the users of the group.

MLSAG Protocol is divided in 4 algorithms:
-**KEYGEN**: generation of the keys and computation of the Keys Images;
-**SIGN**: producing the signature;
-**VER**: verification of the signature;
-**LNK**: check of the uniqueness of the Keys Images.

Monero

Monero

MLSAG

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

MLSAG

Let $\{P_i^j\}_{i=1\ldots n}^{j=1\ldots m}$ the group public keys.

- **KEYGEN**:

MLSAG

Let $\{P_i^j\}_{i=1\ldots n}^{j=1\ldots m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

Monero

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m$$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod\ q)$$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod\ q)$$

and compute the Keys Images

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod\ q)$$

and compute the Keys Images

$$\forall j = 1...m$$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod\ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j)$$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod\ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

  and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

  Let $\mathfrak{m}$ be the message.

- **SIGN**:

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

- **SIGN**: $\forall i = 1..n$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

- **SIGN**: $\forall i = 1..n \quad i \neq \pi$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

- **SIGN**: $\forall i = 1..n \quad i \neq \pi$ and $\forall j = 1..m$

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

- **SIGN**: $\forall i = 1..n \quad i \neq \pi$ and $\forall j = 1..m$ select random scalars (in Z_q) s_i^j

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod\ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

- **SIGN**: $\forall i = 1..n \quad i \neq \pi$ and $\forall j = 1..m$ select random scalars (in Z_q) s_i^j and a_j.

MLSAG

Let $\{P_i^j\}_{i=1...n}^{j=1...m}$ the group public keys.

- **KEYGEN**: Let π the secret index such that

$$\forall j = 1...m \quad x_j G = P_\pi^j \ (mod \ q)$$

and compute the Keys Images

$$\forall j = 1...m \quad \mathrm{I}_j = x_j H_P(P_\pi^j) \ \ with \ \ H_P(P) = Keccak(P)$$

Let $\mathfrak{m}$ be the message.

- **SIGN**: $\forall i = 1..n \quad i \neq \pi$ and $\forall j = 1..m$ select random scalars (in Z_q) s_i^j and a_j. Let h be the hash function.

MLSAG

Compute:

MLSAG

Compute:

$$L_{\pi}^{j} = a_j G$$

MLSAG

Compute:

$$L_{\pi}^{j} = a_j G$$

$$R_{\pi}^{j} = a_j H(P_{\pi}^{j})$$

MLSAG

Compute:

$$L_\pi^j = a_j G$$

$$R_\pi^j = a_j H(P_\pi^j)$$

$$c_{\pi+1} = h(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m)$$

MLSAG

Compute:

$$L_\pi^j = a_j G$$

$$R_\pi^j = a_j H(P_\pi^j)$$

$$c_{\pi+1} = h(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m)$$

and $\forall i = \pi + 1 ... \pi - 1 \ mod \ n$

MLSAG

Compute:

$$L_\pi^j = a_j G$$

$$R_\pi^j = a_j H(P_\pi^j)$$

$$c_{\pi+1} = h(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m)$$

and $\forall i = \pi + 1 ... \pi - 1 \; mod \; n$

$$L_i^j = s_i^j G + c_i P_i^j$$

MLSAG

Compute:

$$L_\pi^j = a_j G$$

$$R_\pi^j = a_j H(P_\pi^j)$$

$$c_{\pi+1} = h(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m)$$

and $\forall i = \pi + 1 ... \pi - 1 \ mod \ n$

$$L_i^j = s_i^j G + c_i P_i^j$$

$$R_i^j = s_i^j H(P_i^j) + c_i \mathrm{I}_j$$

MLSAG

Compute:

$$L_\pi^j = a_j G$$

$$R_\pi^j = a_j H(P_\pi^j)$$

$$c_{\pi+1} = h(\mathfrak{m}, L_\pi^1, R_\pi^1, ..., L_\pi^m, R_\pi^m)$$

and $\forall i = \pi + 1 ... \pi - 1 \ mod \ n$

$$L_i^j = s_i^j G + c_i P_i^j$$

$$R_i^j = s_i^j H(P_i^j) + c_i \mathrm{I}_j$$

$$c_{i+1} = h(\mathfrak{m}, L_i^1, R_i^1, ..., L_i^m, R_i^m)$$

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \quad mod \ l$.

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \mod l$. A signature for the message $\mathfrak{m}$ is

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \ \ mod\ l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \;\; mod\ l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

Monero

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \quad mod\ l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

- **VER**:

Monero

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \mod l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

- **VER**: Everyone could regenerate all L^j_i, R^j_i

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \quad mod\ l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

- **VER**: Everyone could regenerate all L^j_i, R^j_i and verify the hash

$$c_{n+1} = c_1$$

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \quad mod\ l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

- **VER**: Everyone could regenerate all L^j_i, R^j_i and verify the hash

$$c_{n+1} = c_1$$

- **LNK**:

Monero

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \mod l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

- **VER**: Everyone could regenerate all L^j_i, R^j_i and verify the hash

$$c_{n+1} = c_1$$

- **LNK**: If any of the I_j was already used,

MLSAG

Where

$$c_\pi = h(\mathfrak{m}, L^1_{\pi-1}, R^1_{\pi-1}, ..., L^m_{\pi-1}, R^m_{\pi-1})$$

Then we define $s^j_\pi = a_j - c_\pi x_j \;\; mod \; l$. A signature for the message $\mathfrak{m}$ is

$$\sigma = (\mathrm{I}_1, ..., \mathrm{I}_m, c_1, s^1_1, ..., s^m_1, ..., s^m_n)$$

The complexity is $\mathcal{O}(m \cdot (n+1))$.

- **VER**: Everyone could regenerate all L^j_i, R^j_i and verify the hash

$$c_{n+1} = c_1$$

- **LNK**: If any of the I_j was already used, the signature is rejected.

Monero

Theorem (MLSAG Unforgeability)

Theorem (MLSAG Unforgeability)

Let $\mathcal{A}$ be a **Probabilistic Polynomial Time** *(PPT) Adversary(Algorithm).*

Theorem (MLSAG Unforgeability)

Let $\mathcal{A}$ be a **Probabilistic Polynomial Time** *(PPT) Adversary(Algorithm). Then the probability that $\mathcal{A}$ forges a verifying MLSAG Signature is* **Negligible** *under the (EC)DLOG Assumption.*

Theorem (MLSAG Unforgeability)

Let $\mathcal{A}$ be a **Probabilistic Polynomial Time** *(PPT) Adversary(Algorithm). Then the probability that $\mathcal{A}$ forges a verifying MLSAG Signature is* **Negligible** *under the (EC)DLOG Assumption.*

Theorem (MLSAG Linkability)

Theorem (MLSAG Unforgeability)

Let $\mathcal{A}$ be a **Probabilistic Polynomial Time** *(PPT) Adversary(Algorithm). Then the probability that $\mathcal{A}$ forges a verifying MLSAG Signature is* **Negligible** *under the (EC)DLOG Assumption.*

Theorem (MLSAG Linkability)

The probability that a PPT Algorithm $\mathcal{A}$ can create two verifying signatures σ and σ' signed with the vectors $\bar{y}$ and $\bar{y}'$ such that there exists the same public key y in both $\bar{y}$ and $\bar{y}'$ is **Negligible**

Theorem (MLSAG Unforgeability)

Let $\mathcal{A}$ be a **Probabilistic Polynomial Time** *(PPT) Adversary(Algorithm). Then the probability that $\mathcal{A}$ forges a verifying MLSAG Signature is* **Negligible** *under the (EC)DLOG Assumption.*

Theorem (MLSAG Linkability)

The probability that a PPT Algorithm $\mathcal{A}$ can create two verifying signatures σ and σ' signed with the vectors $\bar{y}$ and $\bar{y}'$ such that there exists the same public key y in both $\bar{y}$ and $\bar{y}'$ is **Negligible**

Theorem (MLSAG Anonimity)

Theorem (MLSAG Unforgeability)

Let $\mathcal{A}$ be a **Probabilistic Polynomial Time** *(PPT) Adversary(Algorithm). Then the probability that $\mathcal{A}$ forges a verifying MLSAG Signature is* **Negligible** *under the (EC)DLOG Assumption.*

Theorem (MLSAG Linkability)

The probability that a PPT Algorithm $\mathcal{A}$ can create two verifying signatures σ and σ' signed with the vectors $\bar{y}$ and $\bar{y}'$ such that there exists the same public key y in both $\bar{y}$ and $\bar{y}'$ is **Negligible**

Theorem (MLSAG Anonimity)

The MLSAG protocol is Signer Ambiguous under the Decisional Diffie Hellman Assumption.

Monero

Monero

Commitments

Monero

Commitments

Let G be the Curve25519 Base Point

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown.

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitment to the value a with mask x.

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitment to the value a with mask x.
If $a = 0$, C is a commitment to 0

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitment to the value a with mask x.
If $a = 0$, C is a commitment to 0 such that $x = \log_G C$ and one can sign with the pair $(x, C(0, x))$.

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitment to the value a with mask x.
If $a = 0$, C is a commitment to 0 such that $x = \log_G C$ and one can sign with the pair $(x, C(0, x))$.In Bitcoin: $\sum C_{in} - \sum C_{out} = 0$

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitment to the value a with mask x.
If $a = 0$, C is a commitment to 0 such that $x = \log_G C$ and one can sign with the pair $(x, C(0, x))$.In Bitcoin: $\sum C_{in} - \sum C_{out} = 0$ while in Monero: $\sum C_{in} - \sum C_{out} = C(0, z)$.

Monero

Commitments

Let G be the Curve25519 Base Point and H a hash such that $H = \gamma G$, with γ unknown. Let's define

$$C(a, x) = xG + aH$$

Commitment to the value a with mask x.
If $a = 0$, C is a commitment to 0 such that $x = \log_G C$ and one can sign with the pair $(x, C(0, x))$.In Bitcoin: $\sum C_{in} - \sum C_{out} = 0$ while in Monero: $\sum C_{in} - \sum C_{out} = C(0, z)$.
If i.e. there are 1 input and 2 outputs:

$$C_{in} = x_C G + aH$$

$$C_{out-1} = y_1 G + b_1 H$$

$$C_{out-2} = y_2 G + b_2 H$$

Monero

Commitments

with:

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

In practice C_i $i = 1...n$ are the input commitments.

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

In practice C_i $i = 1...n$ are the input commitments. With the pairs (P_i, C_i)

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

In practice C_i $i = 1...n$ are the input commitments. With the pairs (P_i, C_i) we create a Ring Signature of the form:

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

In practice C_i $i = 1...n$ are the input commitments. With the pairs (P_i, C_i) we create a Ring Signature of the form:

$$\left\{ P_1 + C_1 - \sum_j C_{j,out}, ..., P_s + C_s - \sum_j C_{j,out}, ..., P_n + C_n - \sum_j C_{j,out} \right\}$$

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

In practice C_i $i = 1...n$ are the input commitments. With the pairs (P_i, C_i) we create a Ring Signature of the form:

$$\left\{P_1 + C_1 - \sum_j C_{j,out}, ..., P_s + C_s - \sum_j C_{j,out}, ..., P_n + C_n - \sum_j C_{j,out}\right\}$$

with private key $z + x'$

Monero

Commitments

with:

- $x_C - y_1 - y_2 = z$
- $a = b_1 + b_2$

so

$C_{in} - C_{out-1} - C_{out-2} = zG = C(0, z)$ with z unknown.

Ring Confidential Transactions

In practice C_i $i = 1...n$ are the input commitments. With the pairs (P_i, C_i) we create a Ring Signature of the form:

$$\left\{ P_1 + C_1 - \sum_j C_{j,out}, ..., P_s + C_s - \sum_j C_{j,out}, ..., P_n + C_n - \sum_j C_{j,out} \right\}$$

with private key $z + x'$ with $x'G = P_s$

Monero

Monero

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let $\left\{(P_\pi^1, C_\pi^1), \ldots, (P_\pi^m, C_\pi^m)\right\}$ be pairs of PubKeys/Commitments

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let $\left\{(P_\pi^1, C_\pi^1), ..., (P_\pi^m, C_\pi^m)\right\}$ be pairs of PubKeys/Commitments with private keys $x_j \quad j = 1...m$.

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let $\left\{(P_\pi^1, C_\pi^1), ..., (P_\pi^m, C_\pi^m)\right\}$ be pairs of PubKeys/Commitments with private keys $x_j \;\; j = 1...m$.
- Find $q+1$ collections $\left\{(P_i^1, C_i^1), ..., (P_i^m, C_i^m)\right\}$, $i = 1..q+1$

Monero

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let $\left\{(P_\pi^1, C_\pi^1), ..., (P_\pi^m, C_\pi^m)\right\}$ be pairs of PubKeys/Commitments with private keys $x_j \quad j = 1...m$.
- Find $q+1$ collections $\left\{(P_i^1, C_i^1), ..., (P_i^m, C_i^m)\right\}$, $i = 1..q+1$ not already **Tag-Linked**.

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let $\left\{(P_\pi^1, C_\pi^1), ..., (P_\pi^m, C_\pi^m)\right\}$ be pairs of PubKeys/Commitments with private keys $x_j \;\; j = 1...m$.
- Find $q + 1$ collections $\left\{(P_i^1, C_i^1), ..., (P_i^m, C_i^m)\right\}$, $i = 1..q + 1$ not already **Tag-Linked**.
- Choose a set of outputs $(Q_i, C_{i,out})$ such that $\sum_{j=1}^{m} C_\pi^j - \sum_i C_{i,out}$

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let $\left\{(P_\pi^1, C_\pi^1), ..., (P_\pi^m, C_\pi^m)\right\}$ be pairs of PubKeys/Commitments with private keys $x_j \;\; j = 1...m$.
- Find $q + 1$ collections $\left\{(P_i^1, C_i^1), ..., (P_i^m, C_i^m)\right\}$, $i = 1..q + 1$ not already **Tag-Linked**.
- Choose a set of outputs $(Q_i, C_{i,out})$ such that $\sum_{j=1}^{m} C_\pi^j - \sum_i C_{i,out}$ is a commitment to 0.

Monero

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

$$\mathfrak{R} := \left\{ \left\{ (P_1^1, C_1^1), ..., (P_1^m, C_1^m), \left(\sum_j P_1^j + \sum_{j=1}^m C_1^j - \sum_i C_{i,out} \right) \right\}, \right.$$

$$...,$$

$$\left. \left\{ (P_{q+1}^1, C_{q+1}^1), .., (P_{q+1}^m, C_{q+1}^m), \left(\sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_i C_{i,out} \right) \right\} \right\}$$

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

$$\mathfrak{R} := \Bigg\{ \Big\{ (P_1^1, C_1^1), ..., (P_1^m, C_1^m), \Big(\sum_j P_1^j + \sum_{j=1}^{m} C_1^j - \sum_i C_{i,out} \Big) \Big\},$$

$$...,$$

$$\Big\{ (P_{q+1}^1, C_{q+1}^1), .., (P_{q+1}^m, C_{q+1}^m), \Big(\sum_j P_{q+1}^j + \sum_{j=1}^{m} C_{q+1}^j - \sum_i C_{i,out} \Big) \Big\} \Bigg\}$$

be the Generalized Ring which we wish to sign.

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

$$\mathfrak{R} := \Bigg\{ \Big\{ (P_1^1, C_1^1), \ldots, (P_1^m, C_1^m), \Big(\sum_j P_1^j + \sum_{j=1}^m C_1^j - \sum_i C_{i,out} \Big) \Big\},$$

$$\ldots,$$

$$\Big\{ (P_{q+1}^1, C_{q+1}^1), \ldots, (P_{q+1}^m, C_{q+1}^m), \Big(\sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_i C_{i,out} \Big) \Big\} \Bigg\}$$

be the Generalized Ring which we wish to sign.

- Compute MLSAG signature Σ on $\mathfrak{R}$

Monero

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

$$\mathfrak{R} := \Bigg\{ \Big\{ (P_1^1, C_1^1), ..., (P_1^m, C_1^m), \Big(\sum_j P_1^j + \sum_{j=1}^m C_1^j - \sum_i C_{i,out} \Big) \Big\},$$

$$...,$$

$$\Big\{ (P_{q+1}^1, C_{q+1}^1), .., (P_{q+1}^m, C_{q+1}^m), \Big(\sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_i C_{i,out} \Big) \Big\} \Bigg\}$$

be the Generalized Ring which we wish to sign.

- Compute MLSAG signature Σ on $\mathfrak{R}$

Conclusions on RCT

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

$$\mathfrak{R} := \Bigg\{ \Big\{ (P_1^1, C_1^1), ..., (P_1^m, C_1^m), \Big(\sum_j P_1^j + \sum_{j=1}^m C_1^j - \sum_i C_{i,out} \Big) \Big\},$$

$$...,$$

$$\Big\{ (P_{q+1}^1, C_{q+1}^1), .., (P_{q+1}^m, C_{q+1}^m), \Big(\sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_i C_{i,out} \Big) \Big\} \Bigg\}$$

be the Generalized Ring which we wish to sign.

- Compute MLSAG signature Σ on $\mathfrak{R}$

Conclusions on RCT

RCTs ensure hiding of amount,origins and destination.

Monero

Tag-Linkable Ring-CT with Multiple Inputs and OneTime Keys

- Let

$$\mathfrak{R} := \left\{ \left\{ (P_1^1, C_1^1), ..., (P_1^m, C_1^m), \left(\sum_j P_1^j + \sum_{j=1}^m C_1^j - \sum_i C_{i,out} \right) \right\}, \right.$$

$$...,$$

$$\left. \left\{ (P_{q+1}^1, C_{q+1}^1), .., (P_{q+1}^m, C_{q+1}^m), \left(\sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_i C_{i,out} \right) \right\} \right\}$$

be the Generalized Ring which we wish to sign.

- Compute MLSAG signature Σ on $\mathfrak{R}$

Conclusions on RCT

RCTs ensure hiding of amount,origins and destination. In additon coin generation is trustless and verifyable secure.

Monero vs Bitcoin

Monero vs Bitcoin

Monero and Bitcoin

Monero vs Bitcoin

Monero and Bitcoin

Bitcoin and Monero are just similar as they are different.

Monero vs Bitcoin

Monero and Bitcoin

Bitcoin and Monero are just similar as they are different.
Main differences:

Monero and Bitcoin

Bitcoin and Monero are just similar as they are different.

Main differences:

-**Blocksize Limit**

Monero vs Bitcoin

Monero and Bitcoin

Bitcoin and Monero are just similar as they are different.

Main differences:

-**Blocksize Limit**

-**Transaction Time**

Monero vs Bitcoin

Monero and Bitcoin

Bitcoin and Monero are just similar as they are different.

Main differences:

-**Blocksize Limit**

-**Transaction Time**

-**Untraceability**

Monero vs Bitcoin

Monero and Bitcoin

Bitcoin and Monero are just similar as they are different.

Main differences:

-**Blocksize Limit**

-**Transaction Time**

-**Untraceability**

-**Safe Elliptic Curves**

Monero vs Bitcoin

Blocksize Limit

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Bitcoin:

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Bitcoin: has limit at 1MB.

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Bitcoin: has limit at 1MB. Some people agree to remove the Limit, but it could overload the nodes.

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Bitcoin: has limit at 1MB. Some people agree to remove the Limit, but it could overload the nodes.
Monero:

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Bitcoin: has limit at 1MB. Some people agree to remove the Limit, but it could overload the nodes.
Monero: has **Scalability**,

Monero vs Bitcoin

Blocksize Limit

Crypto	Monero	Bitcoin
Blocksize limit	None	1 MB

Bitcoin: has limit at 1MB. Some people agree to remove the Limit, but it could overload the nodes.
Monero: has **Scalability**, it modify its size in scale with respect to memory requested.

Monero vs Bitcoin

Transaction Time

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Bitcoin:

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Bitcoin: about 10 min.;

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Bitcoin: about 10 min.; Hash Algorithm is CPU-bound.

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Bitcoin: about 10 min.; Hash Algorithm is CPU-bound.
Monero:

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Bitcoin: about 10 min.; Hash Algorithm is CPU-bound.
Monero: about 1 min;

Monero vs Bitcoin

Transaction Time

Crypto	Monero	Bitcoin
Transaction time	1 minute	10 minutes

Bitcoin: about 10 min.; Hash Algorithm is CPU-bound.
Monero: about 1 min; Hash Algorithm is Memory-bound.

Monero vs Bitcoin

Monero vs Bitcoin

Untraceability

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Bitcoin:

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Bitcoin: most trasparent currency,

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Bitcoin: most trasparent currency, all transactions are public

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Bitcoin: most trasparent currency, all transactions are public
Monero:

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Bitcoin: most trasparent currency, all transactions are public
Monero: Untraceable thanks to Ring Confidential Transactions.

Monero vs Bitcoin

Untraceability

Crypto	Monero	Bitcoin
Untraceable	Yes	No

Bitcoin: most trasparent currency, all transactions are public
Monero: Untraceable thanks to Ring Confidential Transactions. It is optionally transparent.

Monero vs Bitcoin

Monero vs Bitcoin

Safe Elliptic Curves

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve:

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve: **Secp256k1**

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve: **Secp256k1** Unsafe

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve: **Secp256k1** Unsafe
Monero Curve:

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve: **Secp256k1** Unsafe
Monero Curve: **Curve25519**

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve: **Secp256k1** Unsafe
Monero Curve: **Curve25519** Safe

Monero vs Bitcoin

Safe Elliptic Curves

Crypto	Monero	Bitcoin
Safe elliptic curve	Yes (Curve25519)	No (Secp256k1)

Bitcoin Curve: **Secp256k1** Unsafe
Monero Curve: **Curve25519** Safe

Curve	Field	Equation	Base	ρ	Transfer	CM Discr.	Rigid.	Ladder	Twist	Complete	Indistin.	Safe?
Curve25519	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Secp256k1	✓	✓	✓	✓	✓	x	✓	x	✓	x	x	x

GRAZIE PER L'ATTENZIONE!!