Cryptography in Blockchain Part I: Crypto Basics and Monero

Zhiguo Wan
Shandong University
Outline

- Crypto Basics
- Monero and RingCT
- Zero Knowledge Proof
- ZeroCash and ZK-SNARK
Group $G<\cdot>$, Ring $R<+,*>$ and Field $F<+,*>$

- **Group**: a set of “numbers”
  - closure:
  - associative law: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - identity $e$: $e \cdot a = a \cdot e = a$
  - inverses $a^{-1}$: $a \cdot a^{-1} = e$
- **Abelian group**: commutative $a \cdot b = b \cdot a$
  - e.g. $\langle \text{integer, +} \rangle$, $\langle \text{real, *} \rangle$

- **Ring**: a set of “numbers” with two operations
  - an **abelian group** with addition operation
  - multiplication:
    - has closure
    - is associative
  - distributive over addition: $a(b+c) = ab + ac$

- **Field**: a set of numbers with two operations:
  - **abelian group** for addition
  - **abelian group** for multiplication (ignoring 0)

- **Such as**:
  - rational number / real number / complex number
Galois Fields (GF)

• finite fields play a key role in cryptography

• number of elements in a finite field must be a power of a prime: $p^n$

• Galois fields (伽罗瓦域) $GF(p^n)$, in particular often use the fields:
  - $GF(p)$ (for $n=1$)
  - $GF(2^n)$ (for $p=2$)

• $GF(p)$ is the set of integers $\{0, 1, \ldots, p-1\}$ with arithmetic operations modulo prime $p$

• these form a finite field
  - since have multiplicative inverses

• hence arithmetic is “well-behaved”: addition, subtraction, multiplication, and division without leaving the field $GF(p)$
### Example GF(7)

#### (a) Addition modulo 7

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

#### (c) Additive and multiplicative inverses modulo 7

<table>
<thead>
<tr>
<th></th>
<th>w</th>
<th>-w</th>
<th>w⁻¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

#### (b) Multiplication modulo 7

<table>
<thead>
<tr>
<th>×</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Assumptions in Cryptography

• From hard problems: NP problems
• DLP: Discrete Logarithm Problem
  • Given \( X (= g^x \mod p) \), \( g \) and \( p \), compute \( x \).
  • \( x \) is an integer and \( g \) is generator of a group, \( p \) is big prime.
• CDH: Computational Diffie-Hellman
  • Given \( g^a \) and \( g^b \), compute \( g^{ab} \).
• ECDLP: Elliptic Curve DLP
  • Elliptic curve version of DLP
• ECDH: Elliptic Curve DH
  • Elliptic curve version of CDH
RSA Signature

• (Key generation) \( n = pq \)
• Choose \( e \) and \( d \), s.t. \( ed = 1 \mod \varphi(n) \)
• public key : \( \{e, n\} \), private key : \( \{d, n\} \)
• (Signing) \( \text{sig}_K(x) = x^d \mod n \)
• (Verification) \( \text{ver}_K(x, y) = \text{true} \iff x = y^e \mod n, x, y \in \mathbb{Z}_n \)

• (Problem) If an adversary know signature of \( x_1 \) and \( x_2 \) to be \( s_1 \) and \( s_2 \), he can create signature of \( x_3 = x_1x_2 \), i.e., \( s_3\{= s_1s_2 = x_1^d x_2^d = (x_1x_2)^d \} \) without knowing private key \( d \). To prevent this, hashing before signing or other means must be used.
• Note that \( E(m_1)E(m_2) = E(m_1m_2) \) in RSA
ElGamal Signature(I)

- $p$: prime, $\alpha \in Z_p^*$: primitive element,
- Choose $a$: $\beta = \alpha^a \mod p$
- public: $(p, \alpha, \beta)$, private: $a$

(signing) Secret random $k \in Z_{p-1}^*$

$$\text{sig}_K(m, k) = (\gamma, \delta) \text{ where } \gamma = \alpha^k \mod p,$$
$$\delta = (m - a\gamma)k^{-1} \mod (p - 1)$$

(verification) $m, \gamma \in Z_p^*$ and $\delta \in Z_{p-1}^*$

$$\text{ver}_K(m, \gamma, \delta) = \text{true} \iff \beta^{\gamma\delta} = \alpha^m \mod p$$

* $\beta^{\gamma\delta} = \alpha^{a\gamma} \alpha^{k\delta} = \alpha^m \mod p$
ElGamal Signature(II)

- **(Preparation)** \( p = 467, \alpha = 2, a = 127 \)
  \[ \beta = \alpha^a \mod p = 2127 \mod 467 = 132 \]

  - message \( m = 100 \)
  - random \( k = 213 \) s.t., \( \gcd(213,466) = 1. \)
    \[ 213^{-1} \mod 466 = 431 \]

- **signing**
  \[ \gamma = \alpha^k = 2213 \mod 467 = 29 \]
  \[ \delta = (m - a\gamma)k^{-1} \mod (p - 1) \]
  \[ = (100 - 127 \times 29) 431 \mod 466 = 51 \]

- **Verification on \( m, \gamma \) and \( \delta \)**
  \[ \beta^\gamma\delta = \alpha^m \mod p ? \]
  \[ \beta^\gamma\delta = 132^{29} 29^{51} = 189 \mod 467 \]
  \[ \alpha^m = 2^{100} = 189 \mod 467 \]
ElGamal Signature(III)

• Security: without knowing $a$, forgery of $x$’s signature is reducible to DLP of finding $\delta (\gamma)$ chosen $\gamma (\delta)$.

• Note
  • Keep $k$ to be secret
  • Not to use $k$ two times.

• Generalization from $\mathbb{Z}_p^*$ to any finite Abelian group is possible
ASA (I)

• After 1991 August for 3-year public debate, NIST announced DSS (Digital Signature Standard) documented FIPS-186 in 1994 December.

• RSA was not selected since its patent

• Introduce efficient operation under subgroup in ElGamal signature scheme

• Used with DHA (Digital Hash Algorithm)
**DSA(II)**

- \( p:512 \)-bit prime, \( q:160 \)-bit prime, \( q \mid p-1 \), \( g \in \mathbb{Z}_p^* \)
- \( \alpha = g^{(p-1)/q} \mod p \) \((q\text{-th root of } 1 \text{ mod } p)\),
- \( M = \mathbb{Z}_p^* \), \( K = \{(p,q,\alpha,a,\beta): \beta = \alpha^a \mod p\} \)
- public: \((p,q,\alpha,\beta)\), private: \(a\).

(signing) secret random \( k \) \((1 \leq k \leq q-1, \gcd(k,q)=1)\),
\( \text{sig}_K(m,k) = (\gamma, \delta) \) where \( \gamma = (\alpha^k \mod p) \mod q \),
\( \delta = (m + a\gamma)k^{-1} \mod q \).

(verification) \( m \in \mathbb{Z}_p^* \) and \( \gamma, \delta \in \mathbb{Z}_q \)
\( \text{ver}_K(m,\gamma,\delta) = \text{true} \iff (\alpha^{e_1}\beta^{e_2} \mod p) \mod q = \gamma \).
\( e_1 = m\delta^{-1} \mod q \), \( e_2 = \gamma\delta^{-1} \mod q \).
$$\textbf{DSA(III)}$$

(Ex.) \(q=101\), \(p=78q+1=7879\), \(g=3\), \(\alpha = 3^{78} \mod 7879 = 170\), \(a=75\), \(\beta = \alpha^a \mod 7879 = 4567\)

(signing) message \(m=1234\), random \(k=50\), \(k^{-1} \mod 101=99\).

\(\gamma = (\alpha^k \mod p) \mod q = (170^{50} \mod 7879) \mod 101 = 2518 \mod 101 = 94\).

\(\delta = (m+a\gamma)k^{-1} \mod q = (1234 + 75 \times 94) \mod 101 = 97\).

(verification) \(\text{sig}_k(m,k) = (\gamma, \delta) = (94, 97)\), \(m=1234\)

\(\delta^{-1} = 97^{-1} \mod 101 = 25\), \(e_1 = m\delta^{-1} \mod q = 1234 \times 25 \mod 101 = 45\),

\(e_2 = \gamma\delta^{-1} \mod q = 94 \times 25 \mod 101 = 27\)

\((\alpha^{e_1}\beta^{e_2} \mod p) \mod q = (170^{45} 4567^{27} \mod 7879) \mod 101 = 2518 \mod 101 = 94 \Rightarrow ? \gamma = 94 \text{ (valid)}\)
Elliptic Curve Cryptography

- Research on EC has a history of more than 150 years
- 1985
  - Neal Koblitz of Washington University
  - Victor Miller of IBM
  - Applied EC in Cryptography
- ECC and RSA are two widely used PKC systems
Elliptic curves over $R$

- **Definition**
  
  Let
  
  \[ E = \left\{ (x, y) \in R \times R \mid y^2 = x^3 + ax + b \right\} \cup \{ O \} \]

- **Example:**
  
  \[ E : y^2 = x^3 - 4x \]
Group operation $+$

- The point of infinity, $O$, will be the identity element.
  Given $P, Q \in E, P = (x_1, y_1), Q = (x_2, y_2)$
  \[ P + O = O + P \]

  If $x_1 = x_2$, and $y_1 = -y_2$, then $P + Q = O$
  (i.e. $-P = -(x_1, y_1) = (x_1, -y_1)$)

- $P + Q \neq (x_1 + x_2, y_1 + y_2)$
Group operation +

- Given \( P, Q \in E, P = (x_1, y_1), Q = (x_2, y_2) \)
  - Compute \( R = P + Q = (x_3, y_3) \)
  - Addition \( (P \neq Q) \)

\[
\begin{align*}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\
x_3 &= \lambda^2 - x_1 - x_2 \\
y_3 &= (x_1 - x_3)\lambda - y_1
\end{align*}
\]

- Doubling \( (P = Q) \)

\[
\begin{align*}
\lambda &= \frac{3x_1^2 + a}{2y_1} \\
x_3 &= \lambda^2 - 2x_1 \\
y_3 &= (x_1 - x_3)\lambda - y_1
\end{align*}
\]
Example (addition):

• Given \( E : y^2 = x^3 - 25x \)
  
  • \( P = (x_1, y_1) = (0,0), \ Q = (x_2, y_2) = (-5,0), \ P + Q = (x_3, y_3) \)

\[
\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{0 - 0}{-5 - 0} = 0
\]

\[
x_3 = \lambda^2 - x_1 - x_2 = 0^2 - 0 - (-5) = 5
\]

\[
y_3 = (x_1 - x_3) \lambda - y_1 = (0 - 5) \times 0 - 0 = 0
\]
Example (doubling):

- Given \( E : y^2 = x^3 - 25x \)
  - \( P = (x_1, y_1) = (-4, 6), \ 2P = (x_2, y_2) \)

\[
\lambda = \frac{3x_1^2 + a}{2y_1} = \frac{3(-4)^2 - 25}{2 \times 6} = \frac{23}{12}
\]

\[
x_2 = \lambda^2 - 2x_1 = \left(\frac{23}{12}\right)^2 - 2 \times (-4) = \frac{1681}{144}
\]

\[
y_2 = (x_1 - x_2)\lambda - y_1 = \left(-4 - \frac{1681}{144}\right) \times \frac{23}{12} - 6 = -\frac{62279}{1728}
\]
Elliptic Curves over GF(p)

- **Definition**
  
  Let $p > 3, a, b \in \mathbb{Z}_p$, $4a^3 + 27b^2 \neq 0 \pmod{p}$
  
  $E = \{(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p \mid y^2 \equiv x^3 + ax + b \pmod{p} \} \cup \{O\}$

- **Example:**
  
  $E : y^2 = x^3 + x$ over $\mathbb{Z}_{23}$

Elliptic curve equation: $y^2 = x^3 + x$ over $\mathbb{F}_{23}$
Example

- \( E : y^2 = x^3 + x + 6 \) over \( \mathbb{Z}_{11} \)

Find all \((x, y)\) and \(O\):
- Fix \(x\) and determine \(y\)
- \(O\) is an artificial point

12 \((x, y)\) pairs plus \(O\), and have \(\#E = 13\)

<table>
<thead>
<tr>
<th>(x)</th>
<th>(x^3 + x + 6)</th>
<th>quad res?</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>yes</td>
<td>4,7</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>yes</td>
<td>5,6</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>yes</td>
<td>2,9</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>yes</td>
<td>2,9</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>yes</td>
<td>3,8</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>yes</td>
<td>2,9</td>
</tr>
</tbody>
</table>
Example (continue):

- There are 13 points on the group $E(Z_{11})$ and so any non-identity point (i.e. not the point at infinity, noted as $O$) is a generator of $E(Z_{11})$.

Choose generator $\alpha = (2,7)$
Compute $2\alpha = (x_2, y_2)$

$$\lambda = \frac{3x_1^2 + a}{2y_1} = \frac{3(2)^2 + 1}{2 \times 7} = \frac{13}{14} = 2 \times 3^{-1} = 2 \times 4 = 8 \mod 11$$

$$x_2 = \lambda^2 - 2x_1 = (8)^2 - 2 \times (2) = 5 \mod 11$$

$$y_2 = (x_1 - x_2)\lambda - y_1 = (2 - 5) \times 8 - 7 = 2 \mod 11$$
Example (continue):

• Compute $3\alpha = (x_3, y_3)$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2 - 7}{5 - 2} = 2 \mod 11$$

$$x_3 = \lambda^2 - x_1 - x_2 = 2^2 - 2 - 5 = 8 \mod 11$$

$$y_3 = (x_1 - x_3)\lambda - y_1 = (2 - 8) \times 2 - 7 = 3 \mod 11$$

So, we can compute

$$\alpha = (2,7) \quad 2\alpha = (5,2) \quad 3\alpha = (8,3)$$

$$4\alpha = (10,2) \quad 5\alpha = (3,6) \quad 6\alpha = (7,9)$$

$$7\alpha = (7,2) \quad 8\alpha = (3,5) \quad 9\alpha = (10,9)$$

$$10\alpha = (8,8) \quad 11\alpha = (5,9) \quad 12\alpha = (2,4)$$
Example (continue):

• Let’s modify ElGamal encryption by using the elliptic curve $E(\mathbb{Z}_{11})$.
  Suppose that $\alpha = (2,7)$ and Bob’s private key is 7, so

  $$\beta = 7\alpha = (7,2)$$

  Thus the encryption operation is

  $$e_K(x, k) = (k(2,7), x + k(7,2)),$$

  where $x \in E$ and $0 \leq k \leq 12$, and the decryption operation is

  $$d_K(y_1, y_2) = y_2 - 7y_1.$$
Example (continue):

- Suppose that Alice wishes to encrypt the plaintext $x = (10,9)$ (which is a point on $E$).
  If she chooses the random value $k = 3$, then

  \[ y_1 = 3(2,7) = (8,3) \quad \text{and} \quad y_2 = (10,9) + 3(7,2) = (10,9) + (3,5) = (10,2) \]

- Hence $y = ((8,3), (10,2))$. Now, if Bob receives the ciphertext $y$, he decrypts it as follows:

  \[ x = (10,2) - 7(8,3) = (10,2) - (3,5) \]
  \[ = (10,2) + (3,6) = (10,9) \]
Elliptic Curve Discrete Logarithm Problem

• Basic computation of ECC
  • $Q = kP = P + P + \ldots + P$
  where $P$ is a curve point, $k$ is an integer

• Strength of ECC
  • Given curve, the point $P$, and $kP$
    It is hard to recover $k$
    - Elliptic Curve Discrete Logarithm Problem (ECDLP)

Solve $2^x = 8192$
\[ x = 13 \]

Solve $2^x = 927 \pmod{1453}$
\[ x = 13 \]
The Elliptic Curve Discrete Log Problem

Given points $P$ and $Q$ on an elliptic curve with $Q = k \cdot P$ for some integer $k$.

Find $k$

**Example**: On the elliptic curve $y^2 = x^3 - 5x + 12 \pmod{13}$, find $k$ such that

$k \cdot (2,6) = (4,11)$.

$7 \cdot (2,6) = (4,11)$

The elliptic curve discrete log problem is very hard.
Security of ECC versus RSA/ElGamal

- Elliptic curve cryptosystems give the most security per bit of any known public-key scheme.
- The ECDLP problem appears to be much more difficult than the integer factorisation problem and the discrete logarithm problem of $\mathbb{Z}_p$. (no index calculus algo!)
- The strength of elliptic curve cryptosystems grows much faster with the key size increases than does the strength of RSA.
## Elliptic Curve Security

<table>
<thead>
<tr>
<th>Symmetric Key Size (bits)</th>
<th>RSA and Diffie-Hellman Key Size (bits)</th>
<th>Elliptic Curve Key Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>1024</td>
<td>160</td>
</tr>
<tr>
<td>112</td>
<td>2048</td>
<td>224</td>
</tr>
<tr>
<td>128</td>
<td>3072</td>
<td>256</td>
</tr>
<tr>
<td>192</td>
<td>7680</td>
<td>384</td>
</tr>
<tr>
<td>256</td>
<td>15360</td>
<td>521</td>
</tr>
</tbody>
</table>

NIST Recommended Key Sizes
ECC Benefits

ECC is particularly beneficial for application where:

- computational power is limited (wireless devices, PC cards)
- integrated circuit space is limited (wireless devices, PC cards)
- high speed is required.
- intensive use of signing, verifying or authenticating is required.
- signed messages are required to be stored or transmitted (especially for short messages).
- bandwidth is limited (wireless communications and some computer networks).
Elliptic Curve Diffie-Hellman Key Establishment

Alice and Bob want to establish a secret encryption key.

1. Alice and Bob choose an elliptic curve mod a large prime.
2. They choose a random point $P$ on the curve.
3. Alice chooses a secret integer $a$ and computes $aP$.
4. Bob chooses a secret integer $b$ and computes $bP$.
5. Alice sends $aP$ to Bob and Bob sends $bP$ to Alice.
6. Alice computes $a(bP)$ and Bob computes $b(aP)$.
7. They use some agreed-upon method to produce a key from $abP$.

The eavesdropper sees only $P, aP, bP$.
It is hard to deduce $abP$ from this information without computing discrete logs.
Alice and Bob agree on $y^2 = x^3 - 5x + 12 \pmod{13}$ and take $P = (2,6)$.

**Alice**

- $a = 7$
- $7(2,6) = (4, 11)$
- $(4, 11)$

**Bob**

- $b = 5$
- $5(2,6) = (12, 4)$
- $(12, 4)$

$7(12, 4) = (8,9)$

$5(4,11) = (8,9)$
Exercises

1. Does the elliptic curve equation
   \[ y^2 = x^3 + 10x + 5 \] define a group over \( F_{17} \)?

2. Do the points \( P(2,0) \) and \( Q(6,3) \)
   lie on the elliptic curve \( y^2 = x^3 + x + 7 \) over \( F_{17} \)?

3. What are the negatives of the following elliptic curve points over \( F_{17} \)?
   \[ P(5,8), Q(3,0), R(0,6) \]

4. In the elliptic curve group defined by \( y^2 = x^3 + x + 7 \) over \( F_{17} \), what is \( P + Q \) if \( P = (2,0) \) and \( Q = (1,3) \)?

5. In the elliptic curve group defined by \( y^2 = x^3 + x + 7 \) over \( F_{17} \), what is \( 2P \) if \( P = (1,3) \)?
Elliptic Curve Digital Signature Algorithm (ECDSA)

- Public: $G, Q = dG$
- Private: $d$
- Signature Generation
  1. A random number $k, 1 \leq k \leq n - 1$ is chosen
  2. $kG = (x_1, y_1)$ is computed. $x_1$ is converted to its corresponding integer $x_1'$
  3. Next, $r = x_1 \mod n$ is computed
  4. We then compute $k^{-1} \mod n$
  5. $e = \text{HASH}(m)$ where $m$ is the message to be signed
  6. $s = k^{-1}(e + dr) \mod n$

- where $d$ is the private key of the sender.

We have the signature as $(r, s)$
ECDSA

Signature Verification

At the receiver’s end the signature is verified as follows:

1. Verify whether \( r \) and \( s \) belong to the interval \([1, n - 1]\) for the signature to be valid.
2. Compute \( e = HASH(m) \). The hash function should be the same as the one used for signature generation.
3. Compute \( w = s^{-1} \mod n \).
4. Compute \( u_1 = ew \mod n \) and \( u_2 = rw \mod n \).
5. Compute \( (x_1, y_1) = u_1G + u_2Q \). \((Q = dG)\)
6. The signature is valid if \( r = x_1 \mod n \), invalid otherwise.

The verification works because:

- We have \( u_1G + u_2Q = (u_1 + u_2d)G = (ew + rwd)G \)
  \(= (es^{-1} + rds^{-1})G = (e + rd)s^{-1}G \)
- Note that \( s = k^{-1} (e + dr) \mod n \)
- So \( u_1G + u_2Q = (e + rd)s^{-1}G = kG \)
Advanced Digital Signature

• Blind signature
• One-time signature
  • Lamport scheme or Bos-Chaum scheme
• Undeniable signature
  • Chaum-van Antwerpen scheme
• Fail-stop signature
  • van Heyst-Peterson scheme
• Proxy signature
• Group (Ring) signature: group member can generate signature if dispute occurs, identify member. etc.
Chaum’s Blind Signature(I)

Without B seeing the content of message M, A can get a signature of M from B.

RSA scheme, B’s public key :\(\{n,e\}\), private key: \(\{d\}\)

1. select random \(k\) s.t. \(\gcd(n,k)=1\), \(1<k<n-1\)
2. \(m^*=mk^e \mod n\)
3. \(s^*=(m^*)^d \mod n\)
4. \(s=k^{-1}s^* \mod n\) (signature of M by B : \(k^{-1}(mk^e)^d = k^{-1}m^d k^{ed} = m^d\))

\(g(S_B f(m))=S_B(m)\)
\(f:\)blinding \(ft\)
\(g:\)unblinding \(ft\) only A knows \(f(m)\) : blinded message
Chaum’s Blind Signature(II)

- (Preparation) \( p=11, q=3, \ n=33, \phi(n)= 10 \times 2=20 \)
- \( \gcd(d, \phi(n))=1 \) \( \Rightarrow \ d=3, \ ed =1 \mod \phi(n) \Rightarrow 3 \ d = 1 \mod 20 \Rightarrow e=7 \)
- B: public key :\( \{n,e\} = \{33,7\} \), private key =\( \{d\} = \{3\} \)

1. A’s blinding of \( m=5 \)
   - select \( k \) s.t. \( \gcd(k,n)=1 \). \( \gcd(k,33)=1 \Rightarrow k=2 \)
   - \( m^* = m \ k^e \mod n = 5 \ 2^7 \mod 33 = 640 \mod 33 = 13 \mod 33 \)
2. B’s signing without knowing the original \( m \)
   - \( s^* = (m^*)^d \mod n = 13^3 \mod 33 = 2197 \mod 33 = 19 \mod 33 \)
3. A’s unblinding
   - \( s=k^{-1} \ s^* \mod n \) \( (2 \ k^{-1}=1 \mod 33 \Rightarrow k=17) \)
   - \( = 17 \ 19 \mod 33 = 323=26 \mod 33 \)

* Original Signature : \( m^d \mod n = 5^3 \mod 33 = 125 = 26 \mod 33 \)
Group Signature

• Only members of the group can sign messages.
• The receiver of the signature can verify that it is a valid signature from the group.
• The receiver of the signature cannot determine which member of the group is the signer.
• In the case of a dispute, the signature can be “opened” to reveal the identity of the signer.
Group Signature with a Trusted Arbitrator

- Receiver can verifies signatures
- Cannot identify which member is signer
- Problems:
  - Too many keys
  - Master list

Trent:

- $\text{pk}_{1,1}/\text{sk}_{1,1}$
- $\text{pk}_{1,2}/\text{sk}_{1,2}$
- 
- $\text{pk}_{1,n}/\text{sk}_{1,n}$

Member 1

- $\text{pk}_{2,1}/\text{sk}_{2,1}$
- $\text{pk}_{2,2}/\text{sk}_{2,2}$
- 
- $\text{pk}_{2,n}/\text{sk}_{2,n}$

Member 2

- $\text{pk}_{m,1}/\text{sk}_{m,1}$
- $\text{pk}_{m,2}/\text{sk}_{m,2}$
- 
- $\text{pk}_{m,n}/\text{sk}_{m,n}$

Member m

Public Master list:

- $\text{pk}_{2,1}$
- $\text{pk}_{1,3}$
- $\text{pk}_{7,1}$
- $\text{pk}_{4,5}$
- 
- $\text{pk}_{3,9}$
Ring Signature

- Introduced by Ron Rivest, Adi Shamir, and Yael Tauman at ASIACRYPT 2001
- A message signed by someone in a particular group of people.
- Computationally infeasible to determine which of the group members' keys was used to produce the signature.
- Similar to group signatures but differ in two key ways:
  - There is no way to revoke the anonymity of an individual signature,
  - Any group of users can be used as a group without additional setup.
Hands-on Exercises

• Install charm-crypto
  • Install pbc (pairing-based crypto)
  • sudo apt install charm-crypto

• Using charm-crypto (/root/charm) (run with python3)
  1. RSA encryption and signature: rsa_alg_test.py
  2. DSA signature algorithm: pksig_dsa.py
  3. El Gamal encryption: pkenc_test.py
  4. ECDSA signature algorithm: pksig_test.py
  5. Group signature: grpsig_test.py
Monero Ring Confidential Transaction

• Based on CryptoNote, which only hides destination and origin using ring signature

• Multi-layered Linkable Spontaneous Anonymous Group Signature (MLSAG)

• 4 key components
  • MLSAG signature: hide address
  • Commitment scheme: hide amount
  • Range proof: ensure amount range
  • Tag-linkability: prevent double spending
Monero Ring Transaction

**MONERO: UNDER THE HOOD**

---

Monero is:
- **UNTRACEABLE**
- **PRIVATE**
- **SECURE**

Monero’s No. 1 Feature:
- **PRIVACY**

Monero fulfills:
- **UNTRACEABLE**
- **PRIVATE**
- **SECURE**

---

**MONERO has**
- Two **PRIVATE** keys
- Two **PUBLIC** keys

**THE 3 PILLARS OF MONERO**

1. **Ring Signatures**
2. **Stealth Address**
3. **Confidential Transactions**

---

**Now what happens when Alice tries to send 12.5 XMR to Bob?**

1. **RING SIGNATURE** protects the sender’s (Alice) identity
   - The sender’s signature is clouded by other decoy signatures to protect the identity of the sender.
   - **ALICE**

2. **STEALTH ADDRESS** protects the receiver’s (Bob) identity
   - **BOB**
   - **Blockchain**
   - 1. Bob’s public view key and public spend key together generates a random one-time public key.
   - 2. The one-time public key generates a one-time “stealth address”.
   - 3. Alice sends the Monero to the stealth address.
   - 4. Bob’s private spend key now traces the blockchain to find that transaction.
   - 5. Bob generates a one-time private key corresponding to the one-time public key and retrieves the Monero.

3. **CONFIDENTIAL TRANSACTIONS** or **RING CT** protects the transaction identity
   - **BEFORE RING CT**
     - 12.5
     - **TRANSACTION VISIBLE**
     - **12.5**
     - **10**
     - **2**
     - **0.5**
   - **AFTER RING CT**
     - **TRANSACTION NOT VISIBLE**
     - **12.5**
     - **TRANSACTION NOT VISIBLE**
     - **Doesn’t need to be broken down into known denominations**
     - **Doesn’t need to be broken down into known denominations**
Monero Overview

• Suppose, you pick up 4 random people from the streets. And merge your signatures with theirs to create a unique signature. Nobody can find out which one is your signature.

• In Monero, Alice has to send 1000 XMR (XMR = Monero) to Bob,
  • Firstly, she will determine her “ring size”. The ring size are random outputs taken from the blockchain. She then signs these outputs with her private spend key and sends it to the blockchain.
  • Note that Alice doesn’t need to ask the permission of the owners of these previous transactions to use the outputs.
  • Suppose Alice chooses a ring size of 5 i.e. 4 decoy outputs and her own transaction, for an outsider, this is what it will look like:
Monero Overview

• Suppose, you pick up 4 random people from the streets. And merge your signatures with theirs to create a unique signature. Nobody can find out which one is your signature.

• In Monero, Alice has to send 1000 XMR (XMR = Monero) to Bob,
  • Firstly, she will determine her “ring size”. The ring size are random outputs taken from the blockchain. She then signs these outputs with her private spend key and sends it to the blockchain.
  • Note that Alice doesn’t need to ask the permission of the owners of these previous transactions to use the outputs.
  • Suppose Alice chooses a ring size of 5 i.e. 4 decoy outputs and her own output, that it will look like:

INPUTS OF A TRANSACTION

• decoy
• decoy
• decoy
• decoy
• output
A real monero tx

MoneroExplorer.com  dvwae436pd7nt4bc.onion

Tx hash: fdc44c636cb26d01b245be4d24719b074b5ceda05867397c0cfdf5850acc04b
Tx public key: bbff056a732c155114dbc860e049a1038ebd536bc510caa70845432ad5a86e1d

Timestamp: 1528775173  Timestmap [UTC]: 2018-06-12 03:46:13  Age [y:d:h:m:s]: 00:00:00:00:36
Block: 1593111  Fee (per_kb): 0.023228820000 (0.000178036691)
Tx version: 2  Tx size: 130.4727 kb
Extra: 01bbff056a732c155114dbc860e049a1038ebd536bc510caa70845432ad5a86e1d

2 input(s) for total of 2 xmr

key image 01: 7b0e30835389b0dd92fa6da376c4cb60033d7d71653370ca21342fe4c261e82
ring member
- 00: 3b369c78b94a3d73e266951562c3e8c639ec5177a3a393e3a211fe29955f
- 01: c9a9e9ed59fe3881324e6a7da276103121f7f819aef79af0d0a5a143c66c876970
- 02: 34b3f71ccode605718014df731a7114fe3c1376781a0a10065
- 03: 1438684b32e0ca23c135695005ab7284a93756b020313a775e4b4f2cc254
- 04: dbd2f64b4e3d0f80f4f29c2ae49065c716f07e14330103ef9024366133
- 05: e6a2e5975a603bbaf800760623500143b2ced48441cf319445edc01cs3519
- 06: ac3cb4d7da9ae01cb2d25cb176f7ec0a47f5818bc5eb0d0200d268799a9

key image 01: 554667ba26f4598775f6f841630666e5992e8b97680d591789ff6a4c80931
ring member
- 00: 73880e084442ca457f7b8ee3d3e03d9a3dbd13070c421342fe4c261e82
- 01: c25fc76c1c1e369e16b59196cdff1128ca9c5aded44900351c82d2710f299208601
- 02: 1ca9af86f84f7c7df073927758f38c8c74ca19ae47c135421e2377e925d87660
- 03: a3033b3950263e3e0883b9e42c70fddcd12d7517e87d32df9d1d81e03c87914e94
- 04: 1d9b7be4b7402a0eaf24d8e6f410892adb20489db27c02a87fe74303d
- 05: 619ff496a35f98e5f4f58593a38db083eefdf11b7aaac4b2a773a7682a9fc71d

2 output(s) for total of 2 xmr

stealth address
- 00: 1d2f2b181e3263c871105ac69bf0f32823b739169b3873b6e0c3e3a6076b752
- 01: 145c54ce00b394aa099953b02da0884635e72fd878e182cd0403b2ce0d77ecca

amount
- 4776581 of 4776597
- 4776582 of 4776597
A real monero tx

MoneroExplorer.com  dvwae436pd7nt4bc.onion

| Tx hash:   | fdc44c636cb28dd01b245be4d24719b074b5cedda05667397c0dfdf5850aac04b |
| Tx public key: | bff056a732c15514d4bc860e049a1038ebd536bc510ca70f84532ad5a86e1d |

**Transaction Details**

| Timestamp: 1528775173 | UCT: 2018-06-12 03:46:13 |
| Block: 1593111 | Fee (per KB): 0.02322820000 (0.00178036691) |
| Extra: 01bff( | Tx version: 2 |

**Payment id (encrypted):** 004e16571ec6ca4b

| Tx hash: 2763c5b29d6e0777450121f1e1f2c11a9bc621de94cac96ed549b458e36c4af1 |
| Tx public key: 596c8923359d89fd7cf8c4e7c8b9daa73d9156243427f10dc3950a445e4688 |

**Transactions**

| Key image 0: 7b98369ad580c1c99a9ed580 | Ring member |
| Extra: 01bff( | Tx version: 2 |

| RingCT/Type: yes/1 |

| 2 output(s) for total of ? XMR |
| Stealth address: 73980e0d8442ca445f7f7be8ec0dec0498637646807a54fc0b4d9364df52f7f | amount |
| 562a2ac1881f1fa2767f6f2d0529425c532e9e9aeeefd476e13cd18c6cc81d56c | amount idx |

| Key image 0: 55 | Ring member |
| Extra: 0db28f6b0 | Tx version: 2 |

| 2 output(s) for total of ? XMR |
| Stealth address: 73980e0d8442ca445f7f7be8ec0dec0498637646807a54fc0b4d9364df52f7f |
| 562a2ac1881f1fa2767f6f2d0529425c532e9e9aeeefd476e13cd18c6cc81d56c |

| 2 output(s) for total of ? XMR |
| Stealth address: 1d21f2b181e3263c871105ac69b0f32825b739169b8373b6aeca3ea6076b1752 |
| e15554ce00b394aa0999953b020da084635e72f8d87e182c0d403b2ce0d77eca |
| 4776581 of 4776597 | 4776582 of 4776597 |
Monero keys

For viewing incoming transactions, cannot spend

Private view key (a)

Public view key (A = aG)

For viewing funds (R=rG is in TX). Checking P = H(aR)G + B

Private spend key (b)

Public spend key (B=bG)

One-time public key (Stealth address):
\[ P = H(aR)G + B = (H(rA)+b)G \]

Random r

Key Image:
\[ xH_p(P) \]
Where \( x = H(rA)+b \)

Spend with \( H(rA)+b = H(aR)+b \)

Each Tx with a unique key image
Monero Overview

• How to prevent double spending in Monero?
  • Every transaction in Monero comes with its own unique key image.
  • The miners can simply check it out and know whether a Monero coin is being double spent or not.
  • Linkable when double spending!

• Multiple keys
  • Public view key and a private view key.
    • The public view key is used to generate the one-time stealth public address where the funds will be sent to the receiver.
    • The private view key is used by the receiver to scan the blockchain to find the funds sent to them.
  • Spend Keys: the spend key is all about the sender.
    • The public spend key will help the sender take part in ring transactions and also verify the signature of the key image.
    • The private spend key helps in creating that key image which enables them to send transactions.
Monero Overview

• Stealth Addresses: \( P = H(rA)G + B \)
  • \( r \) = Random scalar chosen by Alice. (Alice sends fund to Bob)
  • \( A \) = Bob’s public view key.
  • \( G \) = Cryptographic constant.
  • \( B \) = Bob’s public spend key.
  • \( H() \) = The Keccak hashing algorithm used by Monero.

• Key image: \( I = xH(P) \).

• It is infeasible to derive the one time public address \( P \) from the key image “I” and hence Alice’s identity will never be exposed.

• \( P \) will always give the same value when it’s hashed, meaning \( H(P) \) will always be the same.

• What this means is, since the value of “\( x \)” is constant for Alice, she will never be able to generate multiple values of “I”, which makes the key image unique for every transaction.
Stealth address generation

• Alice does ECDH with her randomly-chosen $r$ and Bob's public view key, $A$. Let's call this point $D$. No one other than Alice or Bob can compute $D$ (see the discussion above on ECDH).

• Alice uses $D$ to generate a new scalar; we'll call it $f$. $f = H(D)$. This is the step that actually causes unlinkability between Bob's outputs.

• Alice computes $F = fG$.

• Alice computes $P = F + B$ (Bob's public spend key).

• $P$ is the stealth destination!
**Input/Output commitments**

• Why input commitments? To hide input amounts

• An Example:
  
  • Input: $C_{in} = x_c G + aH$, where $a$ is the input amount, $H = \gamma G$ for an unknown $\gamma$.
  
  • Output 1: $C_{out-1} = y_1 G + b_1 H$, $b_1$ is the output amount
  
  • Output 2: $C_{out-2} = y_2 G + b_2 H$, $b_2$ is the output amount
  
  • Benefits: amounts are masked!

• Need to ensure: $a = b_1 + b_2$
  
  • $C_{in} - C_{out-1} - C_{out-2} = (x_c - y_1 - y_2)G$ iff $a = b_1 + b_2$
  
  • Generate a ring signature using $(x_c - y_1 - y_2)$
  
  • *Cannot produce a signature if* $a \neq b_1 + b_2$
Recall Monero keys

Private view key (a)

Private spend key (b)

Public view key (A = aG)

Public spend key (B=bG)

One-time public key (Stealth address):

\[ P = H(rA)G + B = (H(rA)+b)G \]

Key Image:

\[ xH(P) \]

Where \( x=H(rA)+b \)

\[ C_{in} = x_cG + aH \]

\[ C_{out-1} = y_1G + b_1H \]

\[ C_{out-2} = y_2G + b_2H \]

\[ C_{in} - C_{out-1} - C_{out-2} = (x_c-y_1-y_2)G \]

\[ (x_c - y_1 - y_2) \]
Linkable Spontaneous Anonymous Group Signature (LSAG)

- Each of $n$ members has a key pair $\{x_j, P_j\}_{j=1}^n$, where $P_j = x_jG$.
- For the $j$-th member, the key image is $I_j = x_jH_p(P_j)$.
- $M$: Message to be signed; $\alpha, s_i$: Random numbers for $i \neq j$

$$
c_n = H(M, L_{n-1}, R_{n-1})
$$

$$
L_n = s_n G + c_n P_n
$$

$$
R_n = s_n H_p(P_n) + c_n I_j
$$

$$
c_{j+1} = H(M, L_j, R_j)
$$

$$
L_{j+1} = s_{j+1} G + c_{j+1} P_{j+1}
$$

$$
R_{j+1} = s_{j+1} H_p(P_{j+1}) + c_{j+1} I_j
$$

$$
L_j = \alpha G, R_j = \alpha H_p(P_j), \quad \text{Set } s_j = \alpha - c_j x_j
$$

$$
L_j = (s_j + c_j x_j) G, R_j = (s_j + c_j x_j) H_p(P_j),
$$
Linkable Spontaneous Anonymous Group Signature (LSAG)

• Verify \((I_j, c_1, s_1, s_2, \ldots, s_n)\)

• Calculate all \(L_i\) and \(R_i\), then compute \(c_{n+1}\)

\[
L_{k+1} = s_{k+1}G + c_{k+1}P_{k+1}
\]
\[
R_{k+1} = s_{k+1}H(P_{k+1}) + c_{k+1}I_j
\]
\[
c_{k+1} = H(M, L_k, R_k)
\]

• Finally, verify \(c_{n+1} = c_1\)

• Signature size \(O(n)\)

• *Linkable w.r.t. \(I_j\)*
Multi-layered Linkable Spontaneous Anonymous Group Signature (MLSAG)

- Each of $n$ members has $m$ keys for $\{P_i^j\}_{i=1}^{j=m}^n$.
- For the $k$-th member, its secret key associated with $\{P_k^j\}$ is $\{x_j\}$, the key image is $I_j = x_jH(P_k^j)$.
- $M$: Message to be signed; $\alpha_j, s_i^j$: Random numbers for $j = \{1 \ldots m\}, i = \{1 \ldots n\}/\{k\}$.

\[
c_n = H \left( M, L_{n-1}^1, R_{n-1}^1, \ldots, L_{n-1}^m, R_{n-1}^m \right)
\]
\[
L_n^j = s_n^j G + c_n P_n^j
\]
\[
R_n^j = s_n^j H(P_n^j) + c_n I_j
\]

\[
c_{k+1} = H \left( M, L_k^1, R_k^1, \ldots, L_k^m, R_k^m \right)
\]
\[
L_{k+1}^j = s_{k+1}^j G + c_{k+1} P_{k+1}^j
\]
\[
R_{k+1}^j = s_{k+1}^j H(P_{k+1}^j) + c_{k+1} I_j
\]

\[
L_k^j = \alpha_j G, R_k^j = \alpha_j H(P_k^j), \quad \text{Set } s_k^j = \alpha_j - c_k x_j \quad (j = 1, \ldots, m)
\]

Final Sig:
\[
(I_1, \ldots, I_m, c_1^1, s_1^1, \ldots, s_1^m, \ldots, s_n^1 \ldots, s_n^m)
\]
Multi-layered Linkable Spontaneous Anonymous Group Signature (MLSAG)

- Verify \((I_1, \ldots, I_m, c_1, s_1^1, \ldots, s_1^m, s_2^1, \ldots, s_2^m, \ldots, s_n^1, \ldots, s_n^m)\)
- Calculate all \(L_i^j\) and \(R_i^j\), then compute \(c_{n+1}\)

\[
\begin{align*}
L_{k+1}^j &= s_{k+1}^j G + c_{k+1}^j P_{k+1}^j \\
R_{k+1}^j &= s_{k+1}^j H(P_{k+1}^j) + c_{k+1}^j I_j \\
c_{k+1} &= H(M, L_k^1, R_k^1, \ldots, L_k^m, R_k^m)
\end{align*}
\]

- Finally, verify \(c_{n+1} = c_1\)
- Signature size \(O(mn)\)
- Linkable if one of the keys are reused in MLSAG signature
The general case of MLSAG in Monero

- $P_i^j$ the stealth address
- $C_i^j$ the payment commitment (masked payment)
- $(P_i^j, C_i^j)$ can be viewed as Monero address
- $m$ inputs and $q$ decoys
- Normally $q = 4$ and $m = 1$ or $2$ in Monero

$$\mathcal{A} := \left\{\left\{ (P_1^1, C_1^1), \ldots, (P_m^m, C_m^m), \left( \sum_j P_i^j + \sum_{j=1}^m C_i^j - \sum_{i} C_{i,\text{out}} \right) \right\}, \ldots, \left\{ (P_{q+1}^1, C_{q+1}^1), \ldots, (P_{q+1}^m, C_{q+1}^m), \left( \sum_j P_{q+1}^j + \sum_{j=1}^m C_{q+1}^j - \sum_{i} C_{i,\text{out}} \right) \right\} \right\}.$$
Range Proof: Aggregate Schnorr Non-linkable Ring Signature

• Why range proof?

\[ C_{in} = x_c G + 10H \quad C_{out \,-1} = y_1 G - 1H \quad C_{out \,-2} = y_2 G + 11H \]

• -1 is a very large number modulo the curve group order \( \rightarrow \) free money has been created

• Binary decomposition:
  • \( b = b_0 2^0 + b_1 2^1 + \cdots + b_n 2^n \)

\[ C_{out} = C^0_{out} + C^1_{out} + C^2_{out} + \cdots + C^n_{out} \]
  • \( C^j_{out} = y_j G + b_j \times 2^j H \)

• Private key \((y_j)\) for either of \((C^j_{out}, C^j_{out} - 2^j H)\) is known.

• A ring signature for two members is generated for \( n+1 \) commitments.
  • \( n=40 \) in Monero, proof size: 6176 bytes/output
A Monero Tx Example

"rct_signatures": {
  "type": 2,
  "txnFee": 2477860000,
  "pseudoOuts": [88c6381c4993c3074ba414bc644f6bfbb28460539eab9b7ed808637b6d330, 6652f3df249102139c8ed3ac952fbc1a8d501d1ebb080df69a14e4971b0316e],
  "ecdhInfo": {
    "mask": "aes6383d11d6f7f2a329815b04108123d0c6fbd3ca0f6e9710e6fd9b0201",
    "amount": "7bc0e39e62f431942d09776b9ba8304b9e2257d605354e393e6e5a02a3b03d"
  },
  "mask": "e117c7e1119fs9fe7f3f3fe516677ee1400b131a99930f1dab9a0866ff9ca00",
  "amount": "0f9df4c74990ab7b64408690b766226c1b68f50d0cc1027d030ca7f4f178c05"
}
},
"outPK": [d3a3b5aaeddaa384e881aa1951c6e1b4e38a4dedsa35aa97383436a8d0bfe, 68164ebdd06ad6132916896a00e6f8b3a8b9ed8b3df879381e6d31db84e48c9],
"rctsig_prunable": {
  "rctCltSigns": {
    "asig": "3e3b81c373cb8f2f13a31310954aef187ea2eef4f2e611de09ab91cc1749a89054e28b02c461531001d383d02611e181a99974a0300c09db4e25870068b02d3e34f49f4fde",
    "asig": "7487b90967ed9111a7002d5fe697b00bd5f7e1731faa0fe8b7b67e3b9a0a0b0b0f0e07b9380fa59d3baf69f0d508c301d87d4f395f9d8d15d872b93109951b2e43f4399b971278d560c0a4",
    "asig": "e12700e068a1820badb0049b3242bf7dcd9e6b9794d008a9159a59f42091750a2111a2a3b21b72957ff7df497d1e8df5bece8e932060d4daa1dfda1ad8f1a2877f89a86062b18",
    "asig": "d3a45af8fe3e9720c97cfa5b4a4dc008425d2c5b7e4b93bb7cd8f1abe55c0397ad6435060a267c7b568e42a9618c2729541332723d984f95a3f280b6e35107545b1e4f8e883f"
  },
  "MKs": {
    "m": ["8b8b545406a033740773ac1b14a0def68d575213345d5c2e9767cc462002a804", "8b8b545406a033740773ac1b14a0def68d575213345d5c2e9767cc462002a804", "8b8b545406a033740773ac1b14a0def68d575213345d5c2e9767cc462002a804", "8b8b545406a033740773ac1b14a0def68d575213345d5c2e9767cc462002a804", "8b8b545406a033740773ac1b14a0def68d575213345d5c2e9767cc462002a804"]
  }
}
Hands-on Exercises

• MiniNero (run with python)
  1. Test.py RingCT/RingCTSimple
  2. Test.py MLSAG/MLSAG2
  3. Test.py ASNL

• Study monero source code
Questions

• How the range proofs are generated in MiniNero?
• How the MLSAG signatures are generated in MiniNero?
• What is the purpose of `ecdhinfo` in monero tx?
• Why range proof can prevent payments like -1?
• What private key is used in Monero to generate a MLSAG ring signature?