Elliptic Curve Cryptography in Bitcoin

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

August 8, 2019

Group Theory Recap

Groups

Groups

Definition

A set G with a binary operation $\star$ defined on it is called a group if

Groups

Definition

A set G with a binary operation $\star$ defined on it is called a group if

- the operation $\star$ is associative,

Groups

Definition

A set G with a binary operation $\star$ defined on it is called a group if

- the operation $\star$ is associative,
- there exists an identity element $e \in G$ such that for any $a \in G$

$$a \star e = e \star a = a,$$

Groups

Definition

A set G with a binary operation $\star$ defined on it is called a group if

- the operation $\star$ is associative,
- there exists an identity element $e \in G$ such that for any $a \in G$

$$a \star e = e \star a = a,$$

- for every $a \in G$, there exists an element $b \in G$ such that

$$a \star b = b \star a = e.$$

Groups

Definition

A set G with a binary operation $\star$ defined on it is called a group if

- the operation $\star$ is associative,
- there exists an identity element $e \in G$ such that for any $a \in G$

$$a \star e = e \star a = a,$$

- for every $a \in G$, there exists an element $b \in G$ such that

$$a \star b = b \star a = e.$$

Example

- Modulo n addition on $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$

Cyclic Groups

Definition

A finite group is a group with a finite number of elements.

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \dots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

- For an integer $n \geq 1$, $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

- For an integer $n \geq 1$, $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
  - Operation is addition modulo n

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

- For an integer $n \geq 1$, $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
  - Operation is addition modulo n
  - $\mathbb{Z}_n$ is cyclic with generator 1

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

- For an integer $n \geq 1$, $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
  - Operation is addition modulo n
  - $\mathbb{Z}_n$ is cyclic with generator 1
- For an integer $n \geq 2$, $\mathbb{Z}_n^* = \{i \in \mathbb{Z}_n \setminus \{0\} \mid \gcd(i, n) = 1\}$

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

- For an integer $n \geq 1$, $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
    - Operation is addition modulo n
    - $\mathbb{Z}_n$ is cyclic with generator 1
- For an integer $n \geq 2$, $\mathbb{Z}_n^* = \{i \in \mathbb{Z}_n \setminus \{0\} \mid \gcd(i, n) = 1\}$
    - Operation is multiplication modulo n

Cyclic Groups

Definition

A finite group is a group with a finite number of elements. The order of a finite group G is its cardinality.

Definition

A cyclic group is a finite group G such that each element in G appears in the sequence

$$\{g, g \star g, g \star g \star g, \ldots\}$$

for some particular element $g \in G$, which is called a generator of G.

Examples

- For an integer $n \geq 1$, $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
  - Operation is addition modulo n
  - $\mathbb{Z}_n$ is cyclic with generator 1
- For an integer $n \geq 2$, $\mathbb{Z}_n^* = \{i \in \mathbb{Z}_n \setminus \{0\} \mid \gcd(i, n) = 1\}$
  - Operation is multiplication modulo n
  - $\mathbb{Z}_n^*$ is cyclic if n is a prime

Subgroups

Subgroups

- **Definition:** If G is a group, a nonempty subset $H \subseteq G$ is a *subgroup* of G if H itself forms a group under the same operation associated with G.

Subgroups

- **Definition:** If G is a group, a nonempty subset $H \subseteq G$ is a *subgroup* of G if H itself forms a group under the same operation associated with G.
- Example: Consider the subgroups of $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$.

Subgroups

- **Definition:** If G is a group, a nonempty subset $H \subseteq G$ is a *subgroup* of G if H itself forms a group under the same operation associated with G.
- Example: Consider the subgroups of $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$.
- **Lagrange's Theorem:** If H is a subgroup of a finite group G, then $|H|$ divides $|G|$.

Subgroups

- **Definition:** If G is a group, a nonempty subset $H \subseteq G$ is a *subgroup* of G if H itself forms a group under the same operation associated with G.
- Example: Consider the subgroups of $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$.
- **Lagrange's Theorem:** If H is a subgroup of a finite group G, then $|H|$ divides $|G|$.
- Example: Check the cardinalities of the subgroups of $\mathbb{Z}_6$.

Subgroups

- **Definition:** If G is a group, a nonempty subset $H \subseteq G$ is a *subgroup* of G if H itself forms a group under the same operation associated with G.
- Example: Consider the subgroups of $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$.
- **Lagrange's Theorem:** If H is a subgroup of a finite group G, then $|H|$ divides $|G|$.
- Example: Check the cardinalities of the subgroups of $\mathbb{Z}_6$.
- **Corollary:** If a group has prime order, then every non-identity element is a generator.

Elliptic Curves Over Real Numbers

Elliptic Curves over Reals

The set E of real solutions (x, y) of

$$y^2 = x^3 + ax + b$$

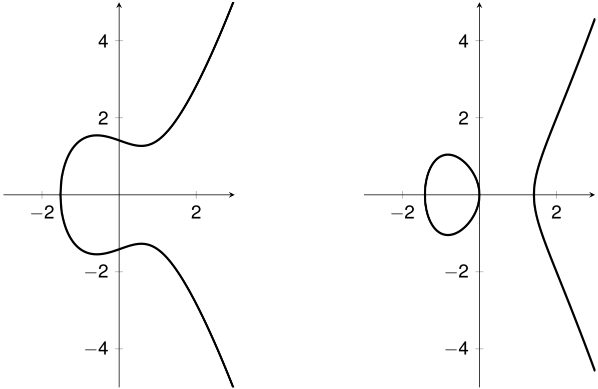
along with a "point of infinity" $\mathcal{O}$. Here $4a^3 + 27b^2 \neq 0$.

Elliptic Curves over Reals

The set E of real solutions (x, y) of

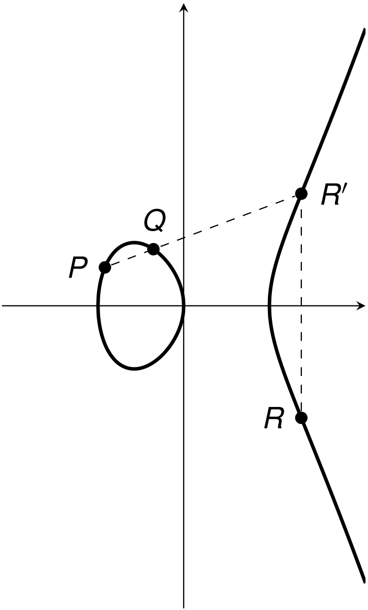
$$y^2 = x^3 + ax + b$$

along with a "point of infinity" $\mathcal{O}$. Here $4a^3 + 27b^2 \neq 0$.



$y^2 = x^3 - x + 2$ $\qquad$ $y^2 = x^3 - 2x$

Point Addition (1/3)

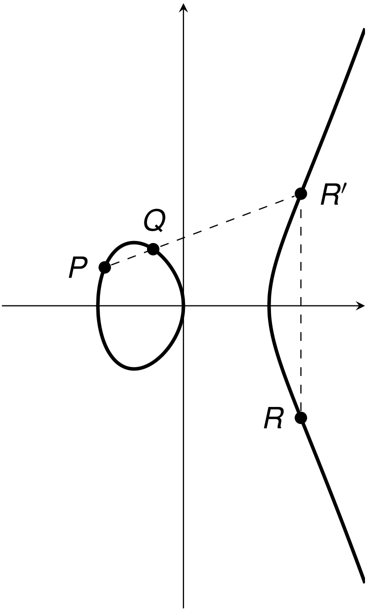


$$P = (x_1, y_1), Q = (x_2, y_2)$$

$$x_1 \neq x_2$$

$$P + Q = R$$

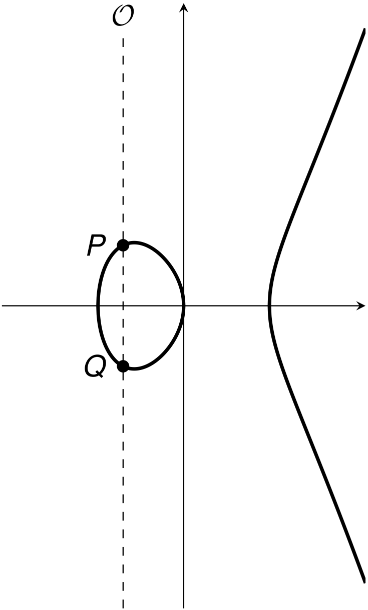
Point Addition (1/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 \neq x_2$$
$$P + Q = R$$

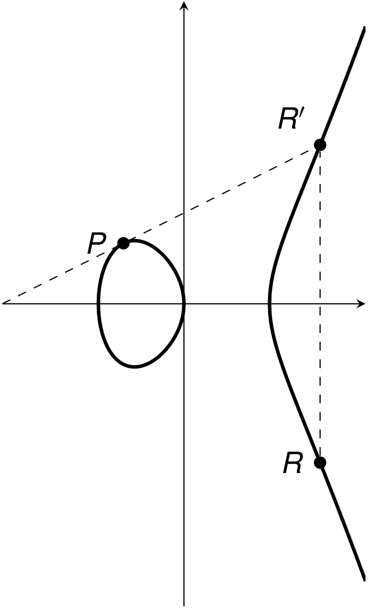
$$R = (x_3, y_3)$$
$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$
$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

Point Addition (2/3)



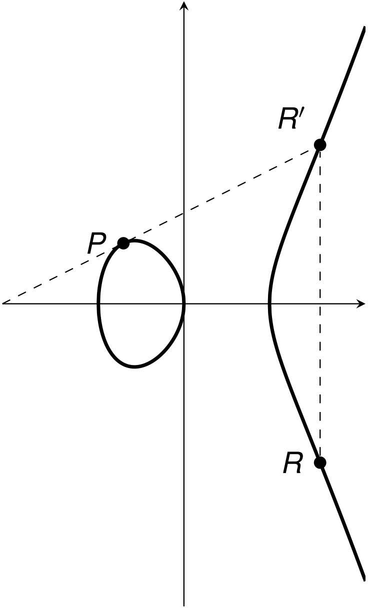
$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 = x_2, y_1 = -y_2$$
$$P + Q = \mathcal{O}$$

Point Addition (3/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 = x_2, y_1 = y_2 \neq 0$$
$$P + Q = R$$

Point Addition (3/3)



$$P = (x_1, y_1), Q = (x_2, y_2)$$
$$x_1 = x_2, y_1 = y_2 \neq 0$$
$$P + Q = R$$

$$R = (x_3, y_3)$$
$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$$
$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

Elliptic Curves Over Finite Fields

Fields

Definition

A set F together with two binary operations $+$ and $*$ is a field if

Fields

Definition

A set F together with two binary operations $+$ and $*$ is a field if

- F is an abelian group under $+$ whose identity is called 0

Fields

Definition

A set F together with two binary operations $+$ and $*$ is a field if

- F is an abelian group under $+$ whose identity is called 0
- $F^* = F \setminus \{0\}$ is an abelian group under $*$ whose identity is called 1

Fields

Definition

A set F together with two binary operations $+$ and $*$ is a field if

- F is an abelian group under $+$ whose identity is called 0
- $F^* = F \setminus \{0\}$ is an abelian group under $*$ whose identity is called 1
- For any $a, b, c \in F$

$$a * (b + c) = a * b + a * c$$

Fields

Definition

A set F together with two binary operations $+$ and $*$ is a field if

- F is an abelian group under $+$ whose identity is called 0
- $F^* = F \setminus \{0\}$ is an abelian group under $*$ whose identity is called 1
- For any $a, b, c \in F$

$$a * (b + c) = a * b + a * c$$

Definition

A finite field is a field with a finite cardinality.

Prime Fields

Prime Fields

- $\mathbb{F}_p = \{0, 1, 2, \ldots, p - 1\}$ where p is prime

Prime Fields

- $\mathbb{F}_p = \{0, 1, 2, \ldots, p-1\}$ where p is prime
- $+$ and $*$ defined on $\mathbb{F}_p$ as

$$x + y = x + y \bmod p,$$
$$x * y = xy \bmod p.$$

Prime Fields

- $\mathbb{F}_p = \{0, 1, 2, \ldots, p-1\}$ where p is prime
- $+$ and $*$ defined on $\mathbb{F}_p$ as

$$x + y = x + y \bmod p,$$
$$x * y = xy \bmod p.$$

- $\mathbb{F}_5$

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$*$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Prime Fields

- $\mathbb{F}_p = \{0, 1, 2, \ldots, p-1\}$ where p is prime
- $+$ and $*$ defined on $\mathbb{F}_p$ as

$$x + y = x + y \bmod p,$$
$$x * y = xy \bmod p.$$

- $\mathbb{F}_5$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

*	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

- In fields, division is multiplication by multiplicative inverse

$$\frac{x}{y} = x * y^{-1}$$

Characteristic of a Field

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1.

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1. The characteristic of F is the smallest integer p such that

$$\underbrace{1 + 1 + \cdots + 1 + 1}_{p \text{ times}} = 0$$

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1. The characteristic of F is the smallest integer p such that

$$\underbrace{1 + 1 + \cdots + 1 + 1}_{p \text{ times}} = 0$$

Examples

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1. The characteristic of F is the smallest integer p such that

$$\underbrace{1 + 1 + \cdots + 1 + 1}_{p \text{ times}} = 0$$

Examples

- $\mathbb{F}_2$ has characteristic 2

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1. The characteristic of F is the smallest integer p such that

$$\underbrace{1 + 1 + \cdots + 1 + 1}_{p \text{ times}} = 0$$

Examples

- $\mathbb{F}_2$ has characteristic 2
- $\mathbb{F}_5$ has characteristic 5

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1. The characteristic of F is the smallest integer p such that

$$\underbrace{1 + 1 + \cdots + 1 + 1}_{p \text{ times}} = 0$$

Examples

- $\mathbb{F}_2$ has characteristic 2
- $\mathbb{F}_5$ has characteristic 5
- $\mathbb{R}$ has characteristic 0

Characteristic of a Field

Definition

Let F be a field with multiplicative identity 1. The characteristic of F is the smallest integer p such that

$$\underbrace{1 + 1 + \cdots + 1 + 1}_{p \text{ times}} = 0$$

Examples

- $\mathbb{F}_2$ has characteristic 2
- $\mathbb{F}_5$ has characteristic 5
- $\mathbb{R}$ has characteristic 0

Theorem

The characteristic of a finite field is prime

Elliptic Curves over Finite Fields

For char$(F) \neq 2, 3$, the set E of solutions (x, y) in $\mathbb{F}^2$ of

$$y^2 = x^3 + ax + b$$

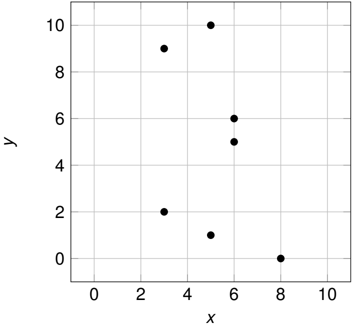
along with a "point of infinity" $\mathcal{O}$. Here $4a^3 + 27b^2 \neq 0$.

Elliptic Curves over Finite Fields

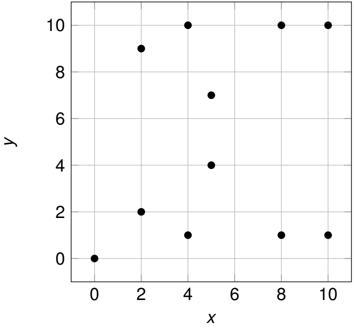
For $\mathrm{char}(F) \neq 2, 3$, the set E of solutions (x, y) in $\mathbb{F}^2$ of

$$y^2 = x^3 + ax + b$$

along with a "point of infinity" $\mathcal{O}$. Here $4a^3 + 27b^2 \neq 0$.



$y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$



$y^2 = x^3 + 9x$ over $\mathbb{F}_{11}$

Point Addition for Finite Field Curves

Point Addition for Finite Field Curves

- Point addition formulas derived for reals are used

Point Addition for Finite Field Curves

- Point addition formulas derived for reals are used
- Example: $y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$

Point Addition for Finite Field Curves

- Point addition formulas derived for reals are used
- Example: $y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$

$+$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$\mathcal{O}$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$(3,2)$	$(3,2)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(3,9)$	$(5,10)$	$(5,1)$
$(3,9)$	$(3,9)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(6,6)$	$(5,1)$	$(3,2)$	$(5,10)$
$(5,1)$	$(5,1)$	$(6,5)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(5,10)$	$(3,9)$	$(3,2)$
$(5,10)$	$(5,10)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(3,2)$	$(5,1)$	$(3,9)$
$(6,5)$	$(6,5)$	$(3,9)$	$(5,1)$	$(5,10)$	$(3,2)$	$(8,0)$	$\mathcal{O}$	$(6,6)$
$(6,6)$	$(6,6)$	$(5,10)$	$(3,2)$	$(3,9)$	$(5,1)$	$\mathcal{O}$	$(8,0)$	$(6,5)$
$(8,0)$	$(8,0)$	$(5,1)$	$(5,10)$	$(3,2)$	$(3,9)$	$(6,6)$	$(6,5)$	$\mathcal{O}$

Point Addition for Finite Field Curves

- Point addition formulas derived for reals are used
- Example: $y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$

$+$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$\mathcal{O}$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$(3,2)$	$(3,2)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(3,9)$	$(5,10)$	$(5,1)$
$(3,9)$	$(3,9)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(6,6)$	$(5,1)$	$(3,2)$	$(5,10)$
$(5,1)$	$(5,1)$	$(6,5)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(5,10)$	$(3,9)$	$(3,2)$
$(5,10)$	$(5,10)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(3,2)$	$(5,1)$	$(3,9)$
$(6,5)$	$(6,5)$	$(3,9)$	$(5,1)$	$(5,10)$	$(3,2)$	$(8,0)$	$\mathcal{O}$	$(6,6)$
$(6,6)$	$(6,6)$	$(5,10)$	$(3,2)$	$(3,9)$	$(5,1)$	$\mathcal{O}$	$(8,0)$	$(6,5)$
$(8,0)$	$(8,0)$	$(5,1)$	$(5,10)$	$(3,2)$	$(3,9)$	$(6,6)$	$(6,5)$	$\mathcal{O}$

- The set $E \cup \mathcal{O}$ is closed under addition

Point Addition for Finite Field Curves

- Point addition formulas derived for reals are used
- Example: $y^2 = x^3 + 10x + 2$ over $\mathbb{F}_{11}$

$+$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$\mathcal{O}$	$\mathcal{O}$	$(3,2)$	$(3,9)$	$(5,1)$	$(5,10)$	$(6,5)$	$(6,6)$	$(8,0)$
$(3,2)$	$(3,2)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(3,9)$	$(5,10)$	$(5,1)$
$(3,9)$	$(3,9)$	$\mathcal{O}$	$(6,5)$	$(8,0)$	$(6,6)$	$(5,1)$	$(3,2)$	$(5,10)$
$(5,1)$	$(5,1)$	$(6,5)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(5,10)$	$(3,9)$	$(3,2)$
$(5,10)$	$(5,10)$	$(8,0)$	$(6,6)$	$\mathcal{O}$	$(6,5)$	$(3,2)$	$(5,1)$	$(3,9)$
$(6,5)$	$(6,5)$	$(3,9)$	$(5,1)$	$(5,10)$	$(3,2)$	$(8,0)$	$\mathcal{O}$	$(6,6)$
$(6,6)$	$(6,6)$	$(5,10)$	$(3,2)$	$(3,9)$	$(5,1)$	$\mathcal{O}$	$(8,0)$	$(6,5)$
$(8,0)$	$(8,0)$	$(5,1)$	$(5,10)$	$(3,2)$	$(3,9)$	$(6,6)$	$(6,5)$	$\mathcal{O}$

- The set $E \cup \mathcal{O}$ is closed under addition
- In fact, its a group

Bitcoin's Elliptic Curve: `secp256k1`

Bitcoin's Elliptic Curve: `secp256k1`

- $y^2 = x^3 + 7$ over $\mathbb{F}_p$ where

$$p = \underbrace{\texttt{FFFFFFFF} \;\cdots\; \texttt{FFFFFFFF}}_{\text{48 hexadecimal digits}} \; \texttt{FFFFFFFE} \; \texttt{FFFFFC2F}$$

$$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

Bitcoin's Elliptic Curve: `secp256k1`

- $y^2 = x^3 + 7$ over $\mathbb{F}_p$ where

$$
\begin{aligned}
p &= \underbrace{\texttt{FFFFFFFF}\ \cdots\ \texttt{FFFFFFFF}}_{\text{48 hexadecimal digits}}\ \texttt{FFFFFFFE}\ \texttt{FFFFFC2F} \\
&= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1
\end{aligned}
$$

- $E \cup \mathcal{O}$ has cardinality n where

$$
\begin{aligned}
n = \ &\texttt{FFFFFFFF}\ \texttt{FFFFFFFF}\ \texttt{FFFFFFFF}\ \texttt{FFFFFFFE} \\
&\texttt{BAAEDCE6}\ \texttt{AF48A03B}\ \texttt{BFD25E8C}\ \texttt{D0364141}
\end{aligned}
$$

Bitcoin's Elliptic Curve: `secp256k1`

- $y^2 = x^3 + 7$ over $\mathbb{F}_p$ where

$$p = \underbrace{\texttt{FFFFFFFF} \;\cdots\; \texttt{FFFFFFFF}}_{\text{48 hexadecimal digits}} \;\texttt{FFFFFFFE}\;\texttt{FFFFFC2F}$$

$$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

- $E \cup \mathcal{O}$ has cardinality n where

$$n = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE}$$
$$\texttt{BAAEDCE6 AF48A03B BFD25E8C D0364141}$$

- Private key is $k \in \{1, 2, \dots, n-1\}$

Bitcoin's Elliptic Curve: `secp256k1`

- $y^2 = x^3 + 7$ over $\mathbb{F}_p$ where

$$p = \underbrace{\texttt{FFFFFFFF} \;\cdots\; \texttt{FFFFFFFF}}_{\text{48 hexadecimal digits}} \;\texttt{FFFFFFFE}\;\texttt{FFFFFC2F}$$
$$= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

- $E \cup \mathcal{O}$ has cardinality n where

$$n = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE}$$
$$\texttt{BAAEDCE6 AF48A03B BFD25E8C D0364141}$$

- Private key is $k \in \{1, 2, \ldots, n-1\}$
- Public key is kP where $P = (x, y)$

$$x = \texttt{79BE667E F9DCBBAC 55A06295 CE870B07}$$
$$\texttt{029BFCDB 2DCE28D9 59F2815B 16F81798},$$
$$y = \texttt{483ADA77 26A3C465 5DA4FBFC 0E1108A8}$$
$$\texttt{FD17B448 A6855419 9C47D08F FB10D4B8}.$$

Point Multiplication using Double-and-Add

Point Multiplication using Double-and-Add

- Point multiplication: kP calculation from k and P

Point Multiplication using Double-and-Add

- Point multiplication: kP calculation from k and P
- Let $k = k_0 + 2k_1 + 2^2k_2 + \cdots + 2^mk_m$ where $k_i \in \{0, 1\}$

Point Multiplication using Double-and-Add

- Point multiplication: kP calculation from k and P
- Let $k = k_0 + 2k_1 + 2^2k_2 + \cdots + 2^m k_m$ where $k_i \in \{0, 1\}$
- Double-and-Add algorithm

Point Multiplication using Double-and-Add

- Point multiplication: kP calculation from k and P
- Let $k = k_0 + 2k_1 + 2^2 k_2 + \cdots + 2^m k_m$ where $k_i \in \{0, 1\}$
- Double-and-Add algorithm
  - Set $N = P$ and $Q = \mathcal{O}$

Point Multiplication using Double-and-Add

- Point multiplication: kP calculation from k and P
- Let $k = k_0 + 2k_1 + 2^2k_2 + \cdots + 2^mk_m$ where $k_i \in \{0, 1\}$
- Double-and-Add algorithm
  - Set $N = P$ and $Q = \mathcal{O}$
  - for $i = 0, 1, \ldots, m$
    - if $k_i = 1$, set $Q \leftarrow Q + N$
    - Set $N \leftarrow 2N$
  - Return Q

Why ECC?

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$

$$C_{EC}(n) = 2^{n/2}$$

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$
$$C_{EC}(n) = 2^{n/2}$$
- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$

$$C_{EC}(n) = 2^{n/2}$$

- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right)$ with $0 < v < 1$

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$

$$C_{EC}(n) = 2^{n/2}$$

- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right)$ with $0 < v < 1$
- Using GNFS method, DLs can be found in $L_p(1/3, c_0)$ in $\mathbb{F}_p^*$

$$C_{CONV}(N) = \exp\left(c_0 N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}\right)$$

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$
$$C_{EC}(n) = 2^{n/2}$$
- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right)$ with $0 < v < 1$
- Using GNFS method, DLs can be found in $L_p(1/3, c_0)$ in $\mathbb{F}_p^*$

$$C_{CONV}(N) = \exp\left(c_0 N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}\right)$$

- Best algorithms for factorization have same asymptotic complexity

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$
$$C_{EC}(n) = 2^{n/2}$$
- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log \log p)^{(1-v)}\right)$ with $0 < v < 1$
- Using GNFS method, DLs can be found in $L_p(1/3, c_0)$ in $\mathbb{F}_p^*$
$$C_{CONV}(N) = \exp\left(c_0 N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}\right)$$
- Best algorithms for factorization have same asymptotic complexity
- For similar security levels
$$n = \beta N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}$$

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$

$$C_{EC}(n) = 2^{n/2}$$

- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right)$ with $0 < v < 1$
- Using GNFS method, DLs can be found in $L_p(1/3, c_0)$ in $\mathbb{F}_p^*$

$$C_{CONV}(N) = \exp\left(c_0 N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}\right)$$

- Best algorithms for factorization have same asymptotic complexity
- For similar security levels

$$n = \beta N^{1/3} \left(\log\left(N \log 2\right)\right)^{2/3}$$

- Key size in ECC grows slightly faster than cube root of conventional key size

Why ECC?

- For elliptic curves $E(\mathbb{F}_q)$, best DL algorithms are exponential in $n = \lceil \log_2 q \rceil$
$$C_{EC}(n) = 2^{n/2}$$
- In $\mathbb{F}_p^*$, best DL algorithms are sub-exponential in $N = \lceil \log_2 p \rceil$
  - $L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right)$ with $0 < v < 1$
- Using GNFS method, DLs can be found in $L_p(1/3, c_0)$ in $\mathbb{F}_p^*$
$$C_{CONV}(N) = \exp\left(c_0 N^{1/3} (\log (N \log 2))^{2/3}\right)$$
- Best algorithms for factorization have same asymptotic complexity
- For similar security levels
$$n = \beta N^{1/3} (\log (N \log 2))^{2/3}$$
- Key size in ECC grows slightly faster than cube root of conventional key size
  - 173 bits instead of 1024 bits, 373 bits instead of 4096 bits

Digital Signatures

Digital Signatures

Digital Signatures

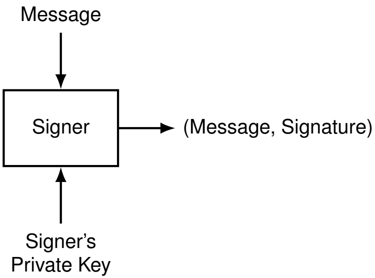
- Digital signatures prove that the signer knows private key

Digital Signatures

- Digital signatures prove that the signer knows private key
- Interactive protocols are not feasible in practice

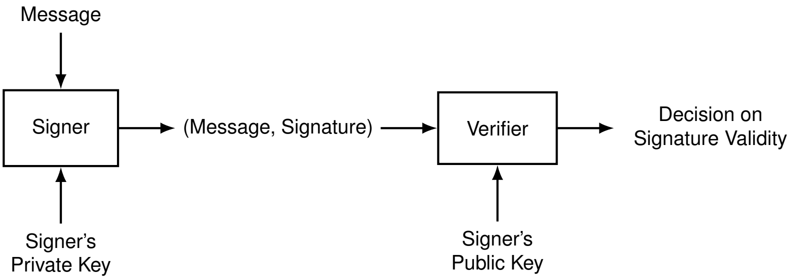
Digital Signatures

- Digital signatures prove that the signer knows private key
- Interactive protocols are not feasible in practice



Digital Signatures

- Digital signatures prove that the signer knows private key
- Interactive protocols are not feasible in practice



Schnorr Signature Algorithm

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0, 1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**
  1. On input m and (r, s)

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**
  1. On input m and (r, s)
  2. Compute $I := g^s \cdot h^{-r}$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**
  1. On input m and (r, s)
  2. Compute $I := g^s \cdot h^{-r}$
  3. Signature valid if $H(I, m) \stackrel{?}{=} r$

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**
  1. On input m and (r, s)
  2. Compute $I := g^s \cdot h^{-r}$
  3. Signature valid if $H(I, m) \stackrel{?}{=} r$
- Example of Fiat-Shamir transform

Schnorr Signature Algorithm

- Based on the Schnorr identification scheme
- Let G be a cyclic group of order q with generator g
- Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
- Signer knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
- **Signer:**
  1. On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q$
  2. Sets $I := g^k$
  3. Computes $r := H(I, m)$
  4. Computes $s = rx + k \bmod q$
  5. Outputs (r, s) as signature for m
- **Verifier**
  1. On input m and (r, s)
  2. Compute $I := g^s \cdot h^{-r}$
  3. Signature valid if $H(I, m) \stackrel{?}{=} r$
- Example of Fiat-Shamir transform
- Patented by Claus Schnorr in 1988

Digital Signature Algorithm

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{r s^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$
     3.2 Computes $s := \left[k^{-1} \cdot (H(m) + xr)\right] \bmod q$

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     - 3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$
     - 3.2 Computes $s := \left[k^{-1} \cdot (H(m) + xr)\right] \bmod q$
     - 3.3 If $r = 0$ or $s = 0$, choose k again

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$
     3.2 Computes $s := \left[k^{-1} \cdot (H(m) + xr)\right] \bmod q$
     3.3 If $r = 0$ or $s = 0$, choose k again
     3.4 Outputs (r, s) as signature for m

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and
  $$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$
- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     - 3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$
     - 3.2 Computes $s := \left[k^{-1} \cdot (H(m) + xr)\right] \bmod q$
     - 3.3 If $r = 0$ or $s = 0$, choose k again
     - 3.4 Outputs (r, s) as signature for m
  4. **Verifier**

Digital Signature Algorithm

- Part of the Digital Signature Standard issued by NIST in 1994
- Based on the following identification protocol
  1. Suppose prover knows $x \in \mathbb{Z}_q$ such that public key $h = g^x$
  2. Prover chooses $k \leftarrow \mathbb{Z}_q^*$ and sends $I := g^k$
  3. Verifier chooses uniform $\alpha, r \in \mathbb{Z}_q$ and sends them
  4. Prover sends $s := \left[k^{-1} \cdot (\alpha + xr) \bmod q\right]$ as response
  5. Verifier accepts if $s \neq 0$ and

$$g^{\alpha s^{-1}} \cdot h^{rs^{-1}} \stackrel{?}{=} I$$

- Digital Signature Algorithm
  1. Let $H : \{0,1\}^* \mapsto \mathbb{Z}_q$ be a cryptographic hash function
  2. Let $F : G \mapsto \mathbb{Z}_q$ be a function, not necessarily CHF
  3. **Signer:**
     - 3.1 On input $m \in \{0,1\}^*$, chooses $k \leftarrow \mathbb{Z}_q^*$ and sets $r := F(g^k)$
     - 3.2 Computes $s := \left[k^{-1} \cdot (H(m) + xr)\right] \bmod q$
     - 3.3 If $r = 0$ or $s = 0$, choose k again
     - 3.4 Outputs (r, s) as signature for m
  4. **Verifier**
     - 4.1 On input m and (r, s) with $r \neq 0, s \neq 0$ checks

$$F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right) \stackrel{?}{=} r$$

ECDSA in Bitcoin

ECDSA in Bitcoin

- **Signer:** Has private key k and message m

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$
  3. Calculate the point $Q = j_1 P + j_2(kP)$

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$
  3. Calculate the point $Q = j_1 P + j_2(kP)$
  4. If $Q = \mathcal{O}$, then the signature is invalid.

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$
  3. Calculate the point $Q = j_1 P + j_2(kP)$
  4. If $Q = \mathcal{O}$, then the signature is invalid.
  5. If $Q \neq \mathcal{O}$, then let $Q = (x, y) \in \mathbb{F}_p^2$. Calculate $t = x \bmod n$. If $t = r$, the signature is valid.

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$
  3. Calculate the point $Q = j_1 P + j_2(kP)$
  4. If $Q = \mathcal{O}$, then the signature is invalid.
  5. If $Q \neq \mathcal{O}$, then let $Q = (x, y) \in \mathbb{F}_p^2$. Calculate $t = x \bmod n$. If $t = r$, the signature is valid.
- As n is a 256-bit integer, signatures are 512 bits long

ECDSA in Bitcoin

- **Signer:** Has private key k and message m
  1. Compute $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Choose a random integer j from $\mathbb{Z}_n^*$
  3. Compute $jP = (x, y)$
  4. Calculate $r = x \bmod n$. If $r = 0$, go to step 2.
  5. Calculate $s = j^{-1}(e + kr) \bmod n$. If $s = 0$, go to step 2.
  6. Output (r, s) as signature for m
- **Verifier:** Has public key kP, message m, and signature (r, s)
  1. Calculate $e = \text{SHA-256}(\text{SHA-256}(m))$
  2. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$
  3. Calculate the point $Q = j_1 P + j_2(kP)$
  4. If $Q = \mathcal{O}$, then the signature is invalid.
  5. If $Q \neq \mathcal{O}$, then let $Q = (x, y) \in \mathbb{F}_p^2$. Calculate $t = x \bmod n$. If $t = r$, the signature is valid.
- As n is a 256-bit integer, signatures are 512 bits long
- As j is randomly chosen, ECDSA output is random for same m

References

- Sections 10.3, 11.4, 12.5 of *Introduction to Modern Cryptography*, J. Katz, Y. Lindell, 2nd edition
- Section 19.1 of *A Graduate Course in Applied Cryptography*, D. Boneh, V. Shoup, `www.cryptobook.us`
- Chapter 2 of *An Introduction to Bitcoin*, S. Vijayakumaran, `www.ee.iitb.ac.in/~sarva/bitcoin.html`