Cryptography and Elliptic curves

Inna Lukyanenko

March 26, 2007

Outline

Terminology

Cryptography (from Greek "hidden" and "write") is the study of message secrecy

Modern meaning: a branch of the mathematical study of information and especially its transmission from place to place

Terminology

Cryptography (from Greek "hidden" and "write") is the study of message secrecy

Modern meaning: a branch of the mathematical study of information and especially its transmission from place to place

The primary purpose was **confidentiality**

Terminology

Cryptography (from Greek "hidden" and "write") is the study of message secrecy
Modern meaning: a branch of the mathematical study of information and especially its transmission from place to place
The primary purpose was **confidentiality**

The cipher consists of two algorithms:

- **Encryption** is the process of converting ordinary information (plaintext) into a ciphertext
- **Decryption** is the reverse process

Terminology

Cryptography (from Greek "hidden" and "write") is the study of message secrecy
Modern meaning: a branch of the mathematical study of information and especially its transmission from place to place
The primary purpose was **confidentiality**

The cipher consists of two algorithms:

- **Encryption** is the process of converting ordinary information (plaintext) into a ciphertext
- **Decryption** is the reverse process

A **key** is a secret parameter for the cipher algorithm

Modern cryptography

Modern cryptography

Symmetric-key cryptography (until 1976):

- a single (secret) key for both encryption and decryption

Modern cryptography

Symmetric-key cryptography (until 1976):

- a single (secret) key for both encryption and decryption
- a significant **drawback** : it requires the prior agreement about the key, using a secure channel

Modern cryptography

Symmetric-key cryptography (until 1976):

- a single (secret) key for both encryption and decryption
- a significant **drawback** : it requires the prior agreement about the key, using a secure channel

Public-key cryptography (invented in 1976 by W.Diffie and M.Hellman):

Modern cryptography

Symmetric-key cryptography (until 1976):

- a single (secret) key for both encryption and decryption
- a significant **drawback** : it requires the prior agreement about the key, using a secure channel

Public-key cryptography (invented in 1976 by W.Diffie and M.Hellman):

- two different, but mathematically related keys
- public - for encryption, private - for decryption

Modern cryptography

Symmetric-key cryptography (until 1976):

- a single (secret) key for both encryption and decryption
- a significant **drawback** : it requires the prior agreement about the key, using a secure channel

Public-key cryptography (invented in 1976 by W.Diffie and M.Hellman):

- two different, but mathematically related keys
- public - for encryption, private - for decryption
- private key cannot be practically derived from public key

Public-key cryptography

Public-key cryptography

- **Public-key encryption**: a message is encrypted with users public key, but cannot be decrypted without the corresponding private key $\Rightarrow$ **Confidentiality**

Public-key cryptography

- **Public-key encryption**: a message is encrypted with users public key, but cannot be decrypted without the corresponding private key $\Rightarrow$ **Confidentiality**
- **Digital signatures**: a message is signed with users private key, but can be verified by anyone who has access to the users public key $\Rightarrow$ **Authenticity**

Public-key cryptography

- **Public-key encryption**: a message is encrypted with users public key, but cannot be decrypted without the corresponding private key $\Rightarrow$ **Confidentiality**
- **Digital signatures**: a message is signed with users private key, but can be verified by anyone who has access to the users public key $\Rightarrow$ **Authenticity**

The main idea: security is based on the computational complexity of "hard" problems:

Public-key cryptography

- **Public-key encryption**: a message is encrypted with users public key, but cannot be decrypted without the corresponding private key $\Rightarrow$ **Confidentiality**
- **Digital signatures**: a message is signed with users private key, but can be verified by anyone who has access to the users public key $\Rightarrow$ **Authenticity**

The main idea: security is based on the computational complexity of "hard" problems:

- integer factorization problem
- discrete logarithm problem

Overview of Digital Signatures

Overview of Digital Signatures

- **RSA** was invented in 1978 by Ronald Rivest, Adi Shamir and Leonard Adleman and its security is based on the integer factorization problem.

Overview of Digital Signatures

- **RSA** was invented in 1978 by Ronald Rivest, Adi Shamir and Leonard Adleman and its security is based on the integer factorization problem.
- **DSA (Digital Signature Algorithm)** was developed in 1991 and is related to the discrete logarithm problem

Overview of Digital Signatures

- **RSA** was invented in 1978 by Ronald Rivest, Adi Shamir and Leonard Adleman and its security is based on the integer factorization problem.
- **DSA (Digital Signature Algorithm)** was developed in 1991 and is related to the discrete logarithm problem
- **ECDSA (Elliptic Curve Digital Signature Algorithm)** is a modification of DSA involving elliptic curve groups, which was proposed in 1992 by Scott Vanstone. It provides smaller key sizes for the same security level and that's why it has become the most popular digital signature.

Description

Description

- A **key generation** algorithm $G \Rightarrow$ a key pair (PK, SK), where PK is a public key and SK is a secret key

Description

- A **key generation** algorithm $G \Rightarrow$ a key pair (PK, SK), where PK is a public key and SK is a secret key
- A **signing** algorithm $S \Rightarrow$ a signature $\sigma = S(m, SK)$, where m is the original message

Description

- A **key generation** algorithm $G \Rightarrow$ a key pair (PK, SK), where PK is a public key and SK is a secret key
- A **signing** algorithm $S \Rightarrow$ a signature $\sigma = S(m, SK)$, where m is the original message
- A **verifying** algorithm $V \Rightarrow V(m, PK, \sigma) = yes/no$

Description

- A **key generation** algorithm $G \Rightarrow$ a key pair (PK, SK), where PK is a public key and SK is a secret key
- A **signing** algorithm $S \Rightarrow$ a signature $\sigma = S(m, SK)$, where m is the original message
- A **verifying** algorithm $V \Rightarrow V(m, PK, \sigma) = yes/no$

Remark: Public-key systems are computationally expensive $\Rightarrow$ in practice, a message is hashed (using a **cryptographic hash function**) and the smaller "hash value" is signed $\Rightarrow$ a receiver computes the hash of the message himself and verifies it.

One-way function

One-way function

Definition

A **one-way function** is a function that is easy to compute, but hard to invert ("easy" $=$ in probabilistic polynomial time)

One-way function

Definition

A **one-way function** is a function that is easy to compute, but hard to invert ("easy" = in probabilistic polynomial time)

Definition

A **trapdoor one-way function** is hard to invert without knowing some secret information - a **trapdoor**

One-way function

Definition

A **one-way function** is a function that is easy to compute, but hard to invert ("easy" = in probabilistic polynomial time)

Definition

A **trapdoor one-way function** is hard to invert without knowing some secret information - a **trapdoor**

Remark: The existence of such functions is an open question!

One-way function

Definition

A **one-way function** is a function that is easy to compute, but hard to invert ("easy" = in probabilistic polynomial time)

Definition

A **trapdoor one-way function** is hard to invert without knowing some secret information - a **trapdoor**

Remark: The existence of such functions is an open question!

Candidates:

- a product of two large primes (RSA)
- an exponentiation in the finite field (DSA, ECDSA)

Discrete Logarithm

Discrete Logarithm

$(G, \cdot)$ is a **finite multiplicative group** (For DSA: $G = \mathbb{Z}_p^*$)

Discrete Logarithm

$(G, \cdot)$ is a **finite multiplicative group** (For DSA: $G = \mathbb{Z}_p^*$)
$g \in G$ of order $n \quad \Rightarrow \quad \langle g \rangle = \{g^i : 0 \leq i \leq n-1\}$
is a **cyclic subgroup** of order n

Discrete Logarithm

$(G, \cdot)$ is a **finite multiplicative group** (For DSA: $G = \mathbb{Z}_p^*$)
$g \in G$ of order $n \quad \Rightarrow \quad \langle g \rangle = \{g^i : 0 \leq i \leq n-1\}$
is a **cyclic subgroup** of order n

Definition

Discrete logarithm problem (DLP): $y \in \langle g \rangle$
Find a unique integer x, $0 \leq x \leq n-1 : y = g^x$

Discrete Logarithm

$(G, \cdot)$ is a **finite multiplicative group** (For DSA: $G = \mathbb{Z}_p^*$)
$g \in G$ of order $n \quad \Rightarrow \quad \langle g \rangle = \{g^i : 0 \leq i \leq n-1\}$
is a **cyclic subgroup** of order n

Definition

Discrete logarithm problem (DLP): $y \in \langle g \rangle$
Find a unique integer x, $0 \leq x \leq n-1 : y = g^x \quad \Rightarrow \quad \mathbf{x = \log_g y}$
It is the inverse operation to **discrete exponentiation**

Discrete Logarithm

$(G, \cdot)$ is a **finite multiplicative group** (For DSA: $G = \mathbb{Z}_p^*$)
$g \in G$ of order $n \quad \Rightarrow \quad \langle g \rangle = \{g^i : 0 \leq i \leq n-1\}$
is a **cyclic subgroup** of order n

Definition

Discrete logarithm problem (DLP): $y \in \langle g \rangle$
Find a unique integer x, $0 \leq x \leq n-1 : y = g^x \quad \Rightarrow \quad \mathbf{x = \log_g y}$
It is the inverse operation to **discrete exponentiation**

Remark: No efficient algorithm for computing discrete logarithms is known $\Rightarrow$ discrete exponentiation is a candidate for **one-way function**

DSA (Digital Signature Algorithm)

DSA (Digital Signature Algorithm)

Domain parameters generation:

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$
- Select $h \in \mathbb{Z}_p^*$

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$
- Select $h \in \mathbb{Z}_p^*$, compute $g = h^{(p-1)/q} \bmod p$ (until $g \neq 1$)

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$
- Select $h \in \mathbb{Z}_p^*$, compute $g = h^{(p-1)/q} \bmod p$ (until $g \neq 1$)
  $\Rightarrow$ g is a generator of a **cyclic subgroup** of order q in $\mathbb{Z}_p^*$

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q | p - 1$
- Select $h \in \mathbb{Z}_p^*$, compute $g = h^{(p-1)/q} \bmod p$ (until $g \neq 1$)
  $\Rightarrow$ g is a generator of a **cyclic subgroup** of order q in $\mathbb{Z}_p^*$
- (p, q, g) are **domain parameters**

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$
- Select $h \in \mathbb{Z}_p^*$, compute $g = h^{(p-1)/q} \bmod p$ (until $g \neq 1$)
  $\Rightarrow g$ is a generator of a **cyclic subgroup** of order q in $\mathbb{Z}_p^*$
- (p, q, g) are **domain parameters**

Key generation:

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$
- Select $h \in \mathbb{Z}_p^*$, compute $g = h^{(p-1)/q} \mod p$ (until $g \neq 1$)
  $\Rightarrow g$ is a generator of a **cyclic subgroup** of order q in $\mathbb{Z}_p^*$
- (p, q, g) are **domain parameters**

Key generation:

- Select a random integer $1 \leq x \leq q-1 \Rightarrow x$ is a **private** key

DSA (Digital Signature Algorithm)

Domain parameters generation:

- Select a 160-bit prime q and 1024-bit prime p: $q|p-1$
- Select $h \in \mathbb{Z}_p^*$, compute $g = h^{(p-1)/q} \bmod p$ (until $g \neq 1$)
  $\Rightarrow$ g is a generator of a **cyclic subgroup** of order q in $\mathbb{Z}_p^*$
- (p, q, g) are **domain parameters**

Key generation:

- Select a random integer $1 \leq x \leq q-1 \Rightarrow x$ is a **private** key
- Compute $y = g^x \bmod p \Rightarrow y$ is a **public** key

DSA Signature generation

DSA Signature generation

- Select a random integer $1 \leq k \leq q - 1$

DSA Signature generation

- Select a random integer $1 \leq k \leq q - 1$
- Compute $e = HASH(m)$, where $HASH$ is a **cryptographic hash function**, such as $SHA - 1$

DSA Signature generation

- Select a random integer $1 \leq k \leq q - 1$
- Compute $e = HASH(m)$, where $HASH$ is a **cryptographic hash function**, such as $SHA - 1$
- Compute $r = (g^k \bmod p) \bmod q$
- Compute $s = (k^{-1}(e + xr)) \bmod q$,

DSA Signature generation

- Select a random integer $1 \leq k \leq q - 1$
- Compute $e = HASH(m)$, where $HASH$ is a **cryptographic hash function**, such as $SHA - 1$
- Compute $r = (g^k \bmod p) \bmod q$
- Compute $s = (k^{-1}(e + xr)) \bmod q$,
- Go to step 1 if $r = 0$ or $s = 0$

DSA Signature generation

- Select a random integer $1 \leq k \leq q - 1$
- Compute $e = HASH(m)$, where $HASH$ is a **cryptographic hash function**, such as $SHA - 1$
- Compute $r = (g^k \bmod p) \bmod q$
- Compute $s = (k^{-1}(e + xr)) \bmod q$,
- Go to step 1 if $r = 0$ or $s = 0$
- (r, s) is a **signature** for the message m

DSA Signature verification:

DSA Signature verification:

- Verify that $1 \leq r, s \leq q - 1$

DSA Signature verification:

- Verify that $1 \leq r, s \leq q - 1$
- Compute $e = HASH(m)$

DSA Signature verification:

- Verify that $1 \leq r, s \leq q - 1$
- Compute $e = HASH(m)$
- Compute $w = s^{-1} \bmod q$

DSA Signature verification:

- Verify that $1 \leq r, s \leq q - 1$
- Compute $e = HASH(m)$
- Compute $w = s^{-1} \bmod q$
- Compute $u_1 = (ew) \bmod q$
- Compute $u_2 = (rw) \bmod q$
- Compute $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$

DSA Signature verification:

- Verify that $1 \leq r, s \leq q - 1$
- Compute $e = HASH(m)$
- Compute $w = s^{-1} \mod q$
- Compute $u_1 = (ew) \mod q$
- Compute $u_2 = (rw) \mod q$
- Compute $v = (g^{u_1} y^{u_2} \mod p) \mod q$
- Accept the signature $\Leftrightarrow v = r$

DSA Correctness

DSA Correctness

$$g = h^{(p-1)/q} \bmod p$$

DSA Correctness

$$g = h^{(p-1)/q} \bmod p \quad \Rightarrow \quad g^q \equiv h^{(p-1)} \equiv 1 \bmod p$$

(Fermat's little theorem)

DSA Correctness

$$g = h^{(p-1)/q} \bmod p \quad \Rightarrow \quad g^q \equiv h^{(p-1)} \equiv 1 \bmod p$$

(Fermat's little theorem) $\Rightarrow$ g has **order** q

DSA Correctness

$$g = h^{(p-1)/q} \bmod p \quad \Rightarrow \quad g^q \equiv h^{(p-1)} \equiv 1 \bmod p$$

(Fermat's little theorem) $\Rightarrow$ g has **order** q

$$s = k^{-1}(e + xr) \bmod q$$

DSA Correctness

$$g = h^{(p-1)/q} \bmod p \quad \Rightarrow \quad g^q \equiv h^{(p-1)} \equiv 1 \bmod p$$

(Fermat's little theorem) $\Rightarrow$ g has **order** q

$$s = k^{-1}(e + xr) \bmod q \ \Rightarrow$$

$$k \equiv (e + xr)s^{-1} \equiv (e + xr)w \mod q$$

DSA Correctness

$$g = h^{(p-1)/q} \bmod p \quad \Rightarrow \quad g^q \equiv h^{(p-1)} \equiv 1 \bmod p$$

(Fermat's little theorem) $\Rightarrow$ g has **order** q

$$s = k^{-1}(e + xr) \bmod q \;\; \Rightarrow$$

$$k \equiv (e + xr)s^{-1} \equiv (e + xr)w \mod q \;\; \Rightarrow$$

$$g^k \equiv g^{ew} g^{xrw} \equiv g^{ew} y^{rw} \equiv g^{u_1} y^{u_2} \bmod p$$

DSA Correctness

$$g = h^{(p-1)/q} \bmod p \quad \Rightarrow \quad g^q \equiv h^{(p-1)} \equiv 1 \bmod p$$

(Fermat's little theorem) $\Rightarrow$ g has **order** q

$$s = k^{-1}(e + xr) \bmod q \;\; \Rightarrow$$

$$k \equiv (e + xr)s^{-1} \equiv (e + xr)w \mod q \;\; \Rightarrow$$

$$g^k \equiv g^{ew} g^{xrw} \equiv g^{ew} y^{rw} \equiv g^{u_1} y^{u_2} \bmod p \;\; \Rightarrow$$

$$\boxed{r = (g^k \ mod \ p) \ mod \ q = (g^{u_1} y^{u_2} \ mod \ p) \ mod \ q = v}$$

The algorithm always accepts the **true** signatures

Finite fields

Finite fields

Definition

A **finite field** is a finite set of elements F with "$+$" and "$\cdot$"
The **order** of F is the number of its elements

Finite fields

Definition

A **finite field** is a finite set of elements F with " $+$ " and " $\cdot$ "
The **order** of F is the number of its elements

Theorem

$\exists$ *a finite field of order* $q \iff q = p^m$ *, where* p *is* **prime**,
and this field is essentially **unique** $\Rightarrow$ *it is denoted by* $\mathbb{F}_q$

Finite fields

Definition

A **finite field** is a finite set of elements F with " $+$ " and " $\cdot$ "
The **order** of F is the number of its elements

Theorem

$\exists$ a finite field of order q $\Leftrightarrow$ $q = p^m$, where p is **prime**,
and this field is essentially **unique** $\Rightarrow$ *it is denoted by $\mathbb{F}_q$*

p is called a **characteristic** of $\mathbb{F}_q$
m is called the **extension degree** of $\mathbb{F}_q$

Finite fields

Definition

A **finite field** is a finite set of elements F with " $+$ " and " $\cdot$ "
The **order** of F is the number of its elements

Theorem

*$\exists$ a finite field of order q $\Leftrightarrow$ $q = p^m$, where p is **prime**,
and this field is essentially **unique** $\Rightarrow$ it is denoted by $\mathbb{F}_q$*

p is called a **characteristic** of $\mathbb{F}_q$
m is called the **extension degree** of $\mathbb{F}_q$

For **elliptic curve cryptography** we need one of two cases:
$q = p$, where p is an odd prime, or $q = 2^m$

The finite field $\mathbb{F}_p$

The finite field $\mathbb{F}_p$, called a **prime field**, is the set of integers $\{0, 1, 2, ..., p-1\}$ with the following **operations**:

The finite field $\mathbb{F}_p$

The finite field $\mathbb{F}_p$, called a **prime field**, is the set of integers $\{0, 1, 2, ..., p-1\}$ with the following **operations**:

- Addition: $a, b \in \mathbb{F}_p \Rightarrow a + b = r$, where $r = (a + b) \bmod p$, $0 \leq r \leq p - 1$
- Multiplication: $a, b \in \mathbb{F}_p \Rightarrow a \cdot b = s$, where $s = a \cdot b \bmod p$, $0 \leq s \leq p - 1$
- Inversion: $a \in \mathbb{F}_p,\ a \neq 0 \Rightarrow \exists\ a^{-1} \in \mathbb{F}_p : a \cdot a^{-1} = 1$

The finite field $\mathbb{F}_{2^m}$

The finite field $\mathbb{F}_{2^m}$, called a **binary finite field**, is a vector space of dimension m over the field $\mathbb{F}_2 = \{0, 1\}$

The finite field $\mathbb{F}_{2^m}$

The finite field $\mathbb{F}_{2^m}$, called a **binary finite field**, is a vector space of dimension m over the field $\mathbb{F}_2 = \{0, 1\}$

$\Rightarrow \exists$ **a basis** $\{\alpha_0, \alpha_1, ..., \alpha_{m-1}\} \in \mathbb{F}_{2^m}$: $\forall \alpha \in \mathbb{F}_{2^m}$

$\alpha = a_0\alpha_0 + a_1\alpha_1 + ... + a_{m-1}\alpha_{m-1}$, where $a_i \in \{0, 1\}$

The finite field $\mathbb{F}_{2^m}$

The finite field $\mathbb{F}_{2^m}$, called a **binary finite field**, is a vector space of dimension m over the field $\mathbb{F}_2 = \{0, 1\}$

$\Rightarrow \exists$ **a basis** $\{\alpha_0, \alpha_1, ..., \alpha_{m-1}\} \in \mathbb{F}_{2^m}$: $\forall \alpha \in \mathbb{F}_{2^m}$

$\alpha = a_0\alpha_0 + a_1\alpha_1 + ... + a_{m-1}\alpha_{m-1}$, where $a_i \in \{0, 1\}$

$$\boxed{\alpha \leftrightarrow (a_0 a_1 ... a_{m-1})}$$

The finite field $\mathbb{F}_{2^m}$

The finite field $\mathbb{F}_{2^m}$, called a **binary finite field**, is a vector space of dimension m over the field $\mathbb{F}_2 = \{0, 1\}$

$\Rightarrow \exists$ **a basis** $\{\alpha_0, \alpha_1, ..., \alpha_{m-1}\} \in \mathbb{F}_{2^m}$: $\forall \alpha \in \mathbb{F}_{2^m}$

$\alpha = a_0\alpha_0 + a_1\alpha_1 + ... + a_{m-1}\alpha_{m-1}$, where $a_i \in \{0, 1\}$

$$\boxed{\alpha \leftrightarrow (a_0 a_1 ... a_{m-1})}$$

Remark: We are interested in **two kinds** of bases: **polynomial** bases and **normal** bases

Polynomial basis representation

Definition

The **reduction polynomial** is an irreducible polynomial of deg m over $\mathbb{F}_2$: $f(x) = x^m + f_{m-1}x^{m-1} + ... + f_2x^2 + f_1x + f_0$, where $f_i \in \{0, 1\}$ for $i = \overline{0, m-1}$

Polynomial basis representation

Definition

The **reduction polynomial** is an irreducible polynomial of deg m over $\mathbb{F}_2$: $f(x) = x^m + f_{m-1}x^{m-1} + ... + f_2x^2 + f_1x + f_0$, where $f_i \in \{0, 1\}$ for $i = \overline{0, m-1}$

Field elements:

$$\mathbb{F}_{2^m} = \{a(x) = a_{m-1}x^{m-1} + ... + a_1x + a_0 : a_i \in \{0, 1\}\}$$

Polynomial basis representation

Definition

The **reduction polynomial** is an irreducible polynomial of deg m over $\mathbb{F}_2$: $f(x) = x^m + f_{m-1}x^{m-1} + ... + f_2x^2 + f_1x + f_0$, where $f_i \in \{0,1\}$ for $i = \overline{0, m-1}$

Field elements:

$$\mathbb{F}_{2^m} = \{a(x) = a_{m-1}x^{m-1} + ... + a_1x + a_0 : a_i \in \{0,1\}\}$$

$$\mathbb{F}_{2^m} = \{(a_{m-1}...a_1a_0) : a_i \in \{0,1\}\}$$

Polynomial basis representation

Definition

The **reduction polynomial** is an irreducible polynomial of deg m over $\mathbb{F}_2$: $f(x) = x^m + f_{m-1}x^{m-1} + ... + f_2x^2 + f_1x + f_0$, where $f_i \in \{0,1\}$ for $i = \overline{0, m-1}$

Field elements:

$$\mathbb{F}_{2^m} = \{a(x) = a_{m-1}x^{m-1} + ... + a_1x + a_0 : a_i \in \{0,1\}\}$$

$$\mathbb{F}_{2^m} = \{(a_{m-1}...a_1a_0) : a_i \in \{0,1\}\}$$

Identities: $(1) = (00...01)$ $(0) = (00...00)$

Field operations

Field operations

- Addition: $a = (a_{m-1} \dots a_1 a_0), b = (b_{m-1} \dots b_1 b_0) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_{m-1} \dots c_1 c_0)$, where $c_i = (a_i + b_i) \bmod 2$

Field operations

- Addition: $a = (a_{m-1} \ldots a_1 a_0), b = (b_{m-1} \ldots b_1 b_0) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_{m-1} \ldots c_1 c_0)$, where $c_i = (a_i + b_i) \bmod 2$

- Multiplication: $a = (a_{m-1} \ldots a_1 a_0), b = (b_{m-1} \ldots b_1 b_0) \in \mathbb{F}_{2^m}$
  $\Rightarrow a \cdot b = r = (r_{m-1} \ldots r_1 r_0)$,
  where $r(x) = a(x) \cdot b(x) \bmod f(x)$ over $\mathbb{F}_2$

Field operations

- Addition: $a = (a_{m-1} \ldots a_1 a_0), b = (b_{m-1} \ldots b_1 b_0) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_{m-1} \ldots c_1 c_0)$, where $c_i = (a_i + b_i) \bmod 2$

- Multiplication: $a = (a_{m-1} \ldots a_1 a_0), b = (b_{m-1} \ldots b_1 b_0) \in \mathbb{F}_{2^m}$
  $\Rightarrow a \cdot b = r = (r_{m-1} \ldots r_1 r_0)$,
  where $r(x) = a(x) \cdot b(x) \bmod f(x)$ over $\mathbb{F}_2$

- Inversion: $a = (a_{m-1} \ldots a_1 a_0) \in \mathbb{F}_{2^m}$, $a \neq 0$
  $\Rightarrow \exists \, ! \; a^{-1} \in \mathbb{F}_{2^m}$: $a \cdot a^{-1} = 1$

Normal basis representation

Definition

A **normal basis** of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ is a basis of the form $\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$, where $\beta \in \mathbb{F}_{2^m}$

Normal basis representation

Definition

A **normal basis** of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ is a basis of the form $\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$, where $\beta \in \mathbb{F}_{2^m}$

Field elements:

$$\mathbb{F}_{2^m} = \{a = \sum_{i=0}^{m-1} a_i \beta^{2^i} : a_i \in \{0,1\}\}$$

Normal basis representation

Definition

A **normal basis** of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ is a basis of the form $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$, where $\beta \in \mathbb{F}_{2^m}$

Field elements:

$$\mathbb{F}_{2^m} = \{a = \sum_{i=0}^{m-1} a_i \beta^{2^i} : a_i \in \{0, 1\}\}$$

$$\mathbb{F}_{2^m} = \{(a_0 a_1 \ldots a_{m-1}) : a_i \in \{0, 1\}\}$$

Normal basis representation

Definition

A **normal basis** of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ is a basis of the form $\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$, where $\beta \in \mathbb{F}_{2^m}$

Field elements:

$$\mathbb{F}_{2^m} = \{a = \sum_{i=0}^{m-1} a_i \beta^{2^i} : a_i \in \{0, 1\}\}$$

$$\mathbb{F}_{2^m} = \{(a_0 a_1 ... a_{m-1}) : a_i \in \{0, 1\}\}$$

Identities: $(1) = (11...11)$ $(0) = (00...00)$

Field operations

- Addition: $a = (a_0 a_1 ... a_{m-1}), b = (b_0 b_1 ... b_{m-1}) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_0 c_1 ... c_{m-1})$, where $c_i = (a_i + b_i)$ mod 2

Field operations

- Addition: $a = (a_0 a_1 ... a_{m-1}), b = (b_0 b_1 ... b_{m-1}) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_0 c_1 ... c_{m-1})$, where $c_i = (a_i + b_i) \bmod 2$
- Squaring: $a = (a_0 a_1 ... a_{m-1}) \in \mathbb{F}_{2^m}$

Field operations

- Addition: $a = (a_0 a_1 \ldots a_{m-1}), b = (b_0 b_1 \ldots b_{m-1}) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_0 c_1 \ldots c_{m-1})$, where $c_i = (a_i + b_i)$ mod 2

- Squaring: $a = (a_0 a_1 \ldots a_{m-1}) \in \mathbb{F}_{2^m}$

$$a^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = (a_{m-1} a_0 a_1 \ldots a_{m-2})$$

Field operations

- Addition: $a = (a_0 a_1 \ldots a_{m-1}), b = (b_0 b_1 \ldots b_{m-1}) \in \mathbb{F}_{2^m}$
  $\Rightarrow a + b = c = (c_0 c_1 \ldots c_{m-1})$, where $c_i = (a_i + b_i) \bmod 2$

- Squaring: $a = (a_0 a_1 \ldots a_{m-1}) \in \mathbb{F}_{2^m}$

$$a^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = (a_{m-1} a_0 a_1 \ldots a_{m-2})$$

- Multiplication: with use of **Gaussian normal basis (GNB)**

- Inversion: $a \in \mathbb{F}_{2^m},\ a \neq 0 \Rightarrow \exists\ !\ a^{-1} \in \mathbb{F}_{2^m}\colon\ a \cdot a^{-1} = 1$

Elliptic curves over $\mathbb{F}_p$

Elliptic curves over $\mathbb{F}_p$

Definition

Let $p > 3$ be a **prime** number, $a, b \in \mathbb{F}_p$: $4a^3 + 27b^2 \neq 0 \mod p$
$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : \; y^2 = x^3 + ax + b \;\} \cup$
$\cup \{\mathcal{O} - point \; at \; infinity\}$ is an **elliptic curve** over $\mathbb{F}_p$

Elliptic curves over $\mathbb{F}_p$

Definition

Let $p > 3$ be a **prime** number, $a, b \in \mathbb{F}_p$: $4a^3 + 27b^2 \neq 0 \mod p$
$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : \ y^2 = x^3 + ax + b\ \} \cup$
$\cup \{\mathcal{O} - point\ at\ infinity\}$ is an **elliptic curve** over $\mathbb{F}_p$

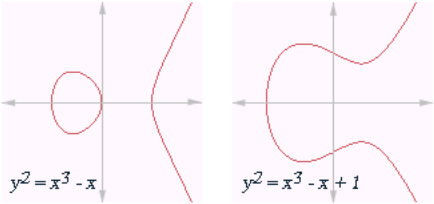
Remark: The requirement $4a^3 + 27b^2 \neq 0$ means that the curve is **non-singular**, i.e. has no cusps and self-intersections

Elliptic curves over $\mathbb{F}_p$

Definition

Let $p > 3$ be a **prime** number, $a, b \in \mathbb{F}_p$: $4a^3 + 27b^2 \neq 0 \mod p$
$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : \; y^2 = x^3 + ax + b \} \cup$
$\cup \{\mathcal{O} - \textit{point at infinity}\}$ is an **elliptic curve** over $\mathbb{F}_p$

Remark: The requirement $4a^3 + 27b^2 \neq 0$ means that the curve is **non-singular**, i.e. has no cusps and self-intersections

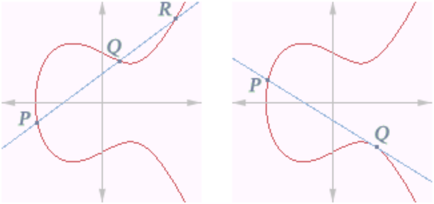


Group law (Geometric description)

- Point addition $P + Q + R = \mathcal{O}$
- Point doubling $P + 2Q = \mathcal{O}$

Group law (Geometric description)

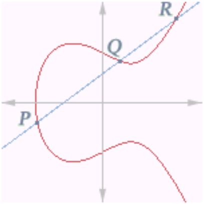
- Point addition $P + Q + R = \mathcal{O}$
- Point doubling $P + 2Q = \mathcal{O}$



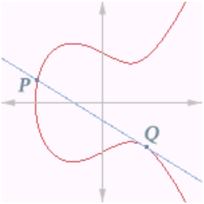
$P + Q + R = 0$ $\quad$ $P + Q + Q = 0$

Group law (Geometric description)

- Point addition $P + Q + R = \mathcal{O}$
- Point doubling $P + 2Q = \mathcal{O}$



$P + Q + R = 0$



$P + Q + Q = 0$

Remark: Together with this addition operation a set $E(\mathbb{F}_p)$ forms a **group** with $\mathcal{O}$ serving as **identity**

Group law (Analitic description)

- $P + \mathcal{O} = \mathcal{O} + P = P \quad \forall P \in E(\mathbb{F}_p)$
- $P = (x, y) \in E(\mathbb{F}_p) \Rightarrow -P = (x, -y) \in E(\mathbb{F}_p)$
- $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbb{F}_p)\colon\ P \neq \pm Q$
  $\Rightarrow P + Q = (x_3, y_3)$:

$$\begin{cases} x_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \\ y_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1 \end{cases}$$

- $P = (x_1, y_1) \in E(\mathbb{F}_p)\colon\ P \neq -P \Rightarrow 2P = (x_3, y_3)$:

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

Elliptic curves over $\mathbb{F}_{2^m}$

Definition

$a, b \in \mathbb{F}_{2^m},\ b \neq 0 \ \Rightarrow$ an **elliptic curve** over $\mathbb{F}_{2^m}$

$$E(\mathbb{F}_{2^m}) = \{(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} : \ y^2 + xy = x^3 + ax^2 + b\ \} \cup$$
$$\cup \{\mathcal{O} - point\ \ at\ \ infinity\}$$

Elliptic curves over $\mathbb{F}_{2^m}$

Definition

$a, b \in \mathbb{F}_{2^m},\ b \neq 0 \ \Rightarrow$ an **elliptic curve** over $\mathbb{F}_{2^m}$

$E(\mathbb{F}_{2^m}) = \{(x, y) \in \mathbb{F}_{2^m} \times \mathbb{F}_{2^m} :\ y^2 + xy = x^3 + ax^2 + b\ \} \cup$

$\cup \{\mathcal{O} - point\ at\ infinity\}$

Remark: The **geometric description** of an addition operation is similar to the case of $E(\mathbb{F}_p)$

Together with this addition operation a set $E(\mathbb{F}_{2^m})$ forms a **group** with $\mathcal{O}$ serving as **identity**

Group law (Analitic description)

- $P + \mathcal{O} = \mathcal{O} + P = P \quad \forall P \in E(\mathbb{F}_{2^m})$
- $P = (x, y) \in E(\mathbb{F}_{2^m}) \Rightarrow -P = (x, x + y) \in E(\mathbb{F}_{2^m})$
- $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbb{F}_{2^m})$: $P \neq \pm Q$
  $\Rightarrow P + Q = (x_3, y_3)$:

$$\begin{cases} x_3 = \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, \\ y_3 = \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1 \end{cases}$$

- $P = (x_1, y_1) \in E(\mathbb{F}_{2^m})$: $P \neq -P \Rightarrow 2P = (x_3, y_3)$:

$$x_3 = x_1^2 + \frac{b}{x_1^2}, \qquad y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3$$

Basic facts

Basic facts

- $E(\mathbb{F}_q)$ with the defined addition operation forms a **group** with point at infinity $\mathcal{O}$ serving as **identity**

Basic facts

- $E(\mathbb{F}_q)$ with the defined addition operation forms a **group** with point at infinity $\mathcal{O}$ serving as **identity**
- **Hasse's theorem:**

$$\boxed{\#E(\mathbb{F}_q) = q + 1 - t, \;\; where \; |t| \leq 2\sqrt{q}}$$

$\#E(\mathbb{F}_q)$ is called the **order**, t is the **trace** of an elliptic curve

Basic facts

- $E(\mathbb{F}_q)$ with the defined addition operation forms a **group** with point at infinity $\mathcal{O}$ serving as **identity**
- **Hasse's theorem:**

$$\boxed{\#E(\mathbb{F}_q) = q + 1 - t, \ \ where \ |t| \leq 2\sqrt{q}}$$

$\#E(\mathbb{F}_q)$ is called the **order**, t is the **trace** of an elliptic curve

- $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, where $n_2|n_1$ and $n_2|q-1$

Basic facts

- $E(\mathbb{F}_q)$ with the defined addition operation forms a **group** with point at infinity $\mathcal{O}$ serving as **identity**
- **Hasse's theorem:**

$$\boxed{\#E(\mathbb{F}_q) = q + 1 - t, \ \ where \ |t| \leq 2\sqrt{q}}$$

$\#E(\mathbb{F}_q)$ is called the **order**, t is the **trace** of an elliptic curve

- $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, where $n_2 | n_1$ and $n_2 | q - 1$
- If $n_2 = 1 \Rightarrow E(\mathbb{F}_q) \cong \mathbb{Z}_n$ is **cyclic** of order $n = n_1$

Basic facts

- $E(\mathbb{F}_q)$ with the defined addition operation forms a **group** with point at infinity $\mathcal{O}$ serving as **identity**
- **Hasse's theorem:**

$$\boxed{\#E(\mathbb{F}_q) = q + 1 - t, \ \ where \ |t| \leq 2\sqrt{q}}$$

$\#E(\mathbb{F}_q)$ is called the **order**, t is the **trace** of an elliptic curve

- $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, where $n_2 | n_1$ and $n_2 | q - 1$
- If $n_2 = 1 \Rightarrow E(\mathbb{F}_q) \cong \mathbb{Z}_n$ is **cyclic** of order $n = n_1 \ \Rightarrow$

$\exists$ a **generator** $G \in E(\mathbb{F}_q)$: $E(\mathbb{F}_q) = \{kG : 0 \leq k \leq n - 1\}$

ECDLP

$\forall G \in E(\mathbb{F}_q)$ of prime order n generates a **cyclic subgroup**

$$(\mathcal{O}, G, 2G, 3G, ..., (n-1)G) \;\Leftrightarrow\; (e, g, g^2, g^3, ..., g^{(n-1)})$$

where g is an integer modulo prime n

ECDLP

$\forall G \in E(\mathbb{F}_q)$ of prime order n generates a **cyclic subgroup**

$$(\mathcal{O}, G, 2G, 3G, ..., (n-1)G) \Leftrightarrow (e, g, g^2, g^3, ..., g^{(n-1)})$$

where g is an integer modulo prime n

Definition

Elliptic curve discrete logarithm problem (ECDLP):
Find k for given points G and kG, where $0 \leq k \leq n-1$

ECDSA

ECDSA

- The **discrete logarithm problem** on elliptic curve groups is believed to be more difficult than the corresponding problem in the multiplicative group of the underlying finite field.

ECDSA

- The **discrete logarithm problem** on elliptic curve groups is believed to be more difficult than the corresponding problem in the multiplicative group of the underlying finite field.
- The keys can be chosen much shorter for a comparable level of security

ECDSA

- The **discrete logarithm problem** on elliptic curve groups is believed to be more difficult than the corresponding problem in the multiplicative group of the underlying finite field.
- The keys can be chosen much shorter for a comparable level of security
- As for other popular public-key cryptosystems, no mathematical proof of difficulty has been published

ECDSA

- The **discrete logarithm problem** on elliptic curve groups is believed to be more difficult than the corresponding problem in the multiplicative group of the underlying finite field.
- The keys can be chosen much shorter for a comparable level of security
- As for other popular public-key cryptosystems,
  no mathematical proof of difficulty has been published
- It was accepted in 1999 as an **ANSI** (American National Standarts Institute) standard

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

- a field size $q = p \; or \; 2^m$

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

- a field size $q = p\ or\ 2^m$
- the representation FR of $\mathbb{F}_q$

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

- a field size $q = p \text{ } or \text{ } 2^m$
- the representation FR of $\mathbb{F}_q$
- $a, b \in \mathbb{F}_q$, which define the **equation** of EC:

$$\begin{cases} y^2 = x^3 + ax + b & \text{in the case } p > 3 \\ y^2 + xy = x^3 + ax^2 + b & \text{in the case } p = 2 \end{cases}$$

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

- a field size $q = p \; or \; 2^m$
- the representation FR of $\mathbb{F}_q$
- $a, b \in \mathbb{F}_q$, which define the **equation** of EC:

$$\begin{cases} y^2 = x^3 + ax + b & \text{in the case } p > 3 \\ y^2 + xy = x^3 + ax^2 + b & \text{in the case } p = 2 \end{cases}$$

- a generator $G = (x_G, y_G) \in E(\mathbb{F}_q)$ of prime order

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

- a field size $q = p \; or \; 2^m$
- the representation FR of $\mathbb{F}_q$
- $a, b \in \mathbb{F}_q$, which define the **equation** of EC:

$$\begin{cases} y^2 = x^3 + ax + b & \text{in the case } p > 3 \\ y^2 + xy = x^3 + ax^2 + b & \text{in the case } p = 2 \end{cases}$$

- a generator $G = (x_G, y_G) \in E(\mathbb{F}_q)$ of prime order
- the order n of G is a **large prime** and $n > 4\sqrt{q}$

Domain parameters

The domain parameters are: an **elliptic curve** $E(\mathbb{F}_q)$ of characteristic p and a **base point** $G \in E(\mathbb{F}_q)$

- a field size $q = p$ *or* 2^m
- the representation FR of $\mathbb{F}_q$
- $a, b \in \mathbb{F}_q$, which define the **equation** of EC:

$$\begin{cases} y^2 = x^3 + ax + b & \text{in the case } p > 3 \\ y^2 + xy = x^3 + ax^2 + b & \text{in the case } p = 2 \end{cases}$$

- a generator $G = (x_G, y_G) \in E(\mathbb{F}_q)$ of prime order
- the order n of G is a **large prime** and $n > 4\sqrt{q}$
- the **cofactor** $h = \#E(\mathbb{F}_q)/n \ \ (h \leq 4)$

Domain parameters generation

Domain parameters generation

- The order of the curve should be divisible by a large prime n

Domain parameters generation

- The order of the curve should be divisible by a large prime n
- One should avoid "weak" curves: for example, $E(\mathbb{F}_{2^m})$ with **non prime** m or $\#E(\mathbb{F}_q) = q$

Domain parameters generation

- The order of the curve should be divisible by a large prime n
- One should avoid "weak" curves: for example, $E(\mathbb{F}_{2^m})$ with **non prime** m or $\#E(\mathbb{F}_q) = q$
- A curve can be selected **verifiably at random**:
  The parameters of the curve a and b are the outputs of the one-way cryptographic hash function, such as $SHA - 1$

Domain parameters generation

- The order of the curve should be divisible by a large prime n
- One should avoid "weak" curves: for example, $E(\mathbb{F}_{2^m})$ with **non prime** m or $\#E(\mathbb{F}_q) = q$
- A curve can be selected **verifiably at random**: The parameters of the curve a and b are the outputs of the one-way cryptographic hash function, such as $SHA-1$
- **Other methods**:
  - Complex multiplication method
  - Koblitz curves

Domain parameters validation

One should always **validate domain parameters** before use

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters
- to detect coding and transmission errors

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters
- to detect coding and transmission errors

Hasse's theorem: $(\sqrt{q}-1)^2 \leq \#E(\mathbb{F}_q) \leq (\sqrt{q}+1)^2$

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters
- to detect coding and transmission errors

Hasse's theorem: $(\sqrt{q}-1)^2 \leq \#E(\mathbb{F}_q) \leq (\sqrt{q}+1)^2$

$n > 4\sqrt{q}$

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters
- to detect coding and transmission errors

Hasse's theorem: $(\sqrt{q}-1)^2 \leq \#E(\mathbb{F}_q) \leq (\sqrt{q}+1)^2$

$n > 4\sqrt{q} \Rightarrow E(\mathbb{F}_q)$ has a **unique** subgroup of order n

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters
- to detect coding and transmission errors

Hasse's theorem: $(\sqrt{q}-1)^2 \leq \#E(\mathbb{F}_q) \leq (\sqrt{q}+1)^2$

$n > 4\sqrt{q} \;\Rightarrow E(\mathbb{F}_q)$ has a **unique** subgroup of order n

$(\sqrt{q}+1)^2 - (\sqrt{q}-1)^2 = 4\sqrt{q}$

Domain parameters validation

One should always **validate domain parameters** before use

- to prevent the insertion of invalid domain parameters
- to detect coding and transmission errors

Hasse's theorem: $(\sqrt{q}-1)^2 \leq \#E(\mathbb{F}_q) \leq (\sqrt{q}+1)^2$

$n > 4\sqrt{q} \;\Rightarrow$ $E(\mathbb{F}_q)$ has a **unique** subgroup of order n

$(\sqrt{q}+1)^2 - (\sqrt{q}-1)^2 = 4\sqrt{q} \;\Rightarrow$

$\exists$ a unique h : $(\sqrt{q}-1)^2 \leq nh \leq (\sqrt{q}+1)^2$

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation:

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation:

- Select a random integer $1 \leq d \leq n-1 \Rightarrow d$ is a **private** key

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation:

- Select a random integer $1 \leq d \leq n-1 \Rightarrow d$ is a **private** key
- Compute $Q = dG \Rightarrow Q$ is a **public** key

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation:

- Select a random integer $1 \leq d \leq n-1 \Rightarrow d$ is a **private** key
- Compute $Q = dG \Rightarrow Q$ is a **public** key

Public key validation:

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation:

- Select a random integer $1 \leq d \leq n-1 \Rightarrow d$ is a **private** key
- Compute $Q = dG \Rightarrow Q$ is a **public** key

Public key validation:

- $Q \neq \mathcal{O}$
- $x_Q, y_Q \in \mathbb{F}_q$ with corresponding representation
- $Q \in E(\mathbb{F}_q)$, where $E(\mathbb{F}_q)$ is defined by a and b
- $nQ = \mathcal{O}$

Key generation

(q, FR, a, b, G, n, h) are the **domain parameters**

Key generation:

- Select a random integer $1 \leq d \leq n-1 \Rightarrow d$ is a **private** key
- Compute $Q = dG \Rightarrow Q$ is a **public** key

Public key validation:

- $Q \neq \mathcal{O}$
- $x_Q, y_Q \in \mathbb{F}_q$ with corresponding representation
- $Q \in E(\mathbb{F}_q)$, where $E(\mathbb{F}_q)$ is defined by a and b
- $nQ = \mathcal{O}$
- $\Rightarrow Q$ is **valid**, otherwise - **invalid**

Signature generation

d is a **private** key, Q is a **public** key

Signature generation

d is a **private** key, Q is a **public** key

- Select a random integer $1 \leq k \leq n - 1$

Signature generation

d is a **private** key, Q is a **public** key

- Select a random integer $1 \leq k \leq n - 1$
- Compute $e = HASH(m)$

Signature generation

d is a **private** key, Q is a **public** key

- Select a random integer $1 \leq k \leq n - 1$
- Compute $e = HASH(m)$
- Compute $r = x_1 \bmod n$, where $(x_1, y_1) = kG$
- Compute $s = k^{-1}(e + dr) \bmod n$

Signature generation

d is a **private** key, Q is a **public** key

- Select a random integer $1 \leq k \leq n-1$
- Compute $e = HASH(m)$
- Compute $r = x_1 \bmod n$, where $(x_1, y_1) = kG$
- Compute $s = k^{-1}(e + dr) \bmod n$
- Go to step 1 if $r = 0$ or $s = 0$

Signature generation

d is a **private** key, Q is a **public** key

- Select a random integer $1 \leq k \leq n - 1$
- Compute $e = HASH(m)$
- Compute $r = x_1 \bmod n$, where $(x_1, y_1) = kG$
- Compute $s = k^{-1}(e + dr) \bmod n$
- Go to step 1 if $r = 0$ or $s = 0$
- (r, s) is a **signature** for the message m

Signature verification

Signature verification

- Verify that $1 \leq r, s \leq n - 1$

Signature verification

- Verify that $1 \leq r, s \leq n - 1$
- Compute $e = HASH(m)$

Signature verification

- Verify that $1 \leq r, s \leq n-1$
- Compute $e = HASH(m)$
- Compute $w = s^{-1} \bmod n$

Signature verification

- Verify that $1 \leq r, s \leq n-1$
- Compute $e = HASH(m)$
- Compute $w = s^{-1} \mod n$
- Compute $u_1 = (ew) \mod n$
- Compute $u_2 = (rw) \mod n$
- Compute $(x_1, y_1) = u_1 G + u_2 Q$

Signature verification

- Verify that $1 \leq r, s \leq n - 1$
- Compute $e = HASH(m)$
- Compute $w = s^{-1} \mod n$
- Compute $u_1 = (ew) \mod n$
- Compute $u_2 = (rw) \mod n$
- Compute $(x_1, y_1) = u_1 G + u_2 Q$
- Accept the signature $\Leftrightarrow x_1 = r \mod n$

ECDSA Correctness

The algorithm always accepts the **true** signatures:

ECDSA Correctness

The algorithm always accepts the **true** signatures:

If a signature (r, s) on a message m was **indeed** generated

using a secret key $d \;\Rightarrow\; s = k^{-1}(e + dr) \bmod n$

ECDSA Correctness

The algorithm always accepts the **true** signatures:

If a signature (r, s) on a message m was **indeed** generated

using a secret key $d \;\Rightarrow\; s = k^{-1}(e + dr) \bmod n$

$\Rightarrow k \equiv (e + dr)s^{-1} \equiv (e + dr)w \equiv u_1 + du_2 \bmod n$

ECDSA Correctness

The algorithm always accepts the **true** signatures:

If a signature (r, s) on a message m was **indeed** generated

using a secret key $d \;\Rightarrow\; s = k^{-1}(e + dr) \bmod n$

$\Rightarrow k \equiv (e + dr)s^{-1} \equiv (e + dr)w \equiv u_1 + du_2 \bmod n$

$\Rightarrow u_1 G + u_2 Q = (u_1 + du_2)G = kG \;\Rightarrow\; r = v$

Comparing DSA and ECDSA

Comparing DSA and ECDSA

- **The basic multiplicative group**:

  The subgroup of order q of $\mathbb{Z}_p^*$ generated by g $\longrightarrow$

  The subgroup of order n of $E(\mathbb{F}_q)$ generated by G

Comparing DSA and ECDSA

- **The basic multiplicative group**:

  The subgroup of order q of $\mathbb{Z}_p^*$ generated by g $\longrightarrow$

  The subgroup of order n of $E(\mathbb{F}_q)$ generated by G

- **The construction of r**:

Comparing DSA and ECDSA

- **The basic multiplicative group**:

  The subgroup of order q of $\mathbb{Z}_p^*$ generated by $g \longrightarrow$

  The subgroup of order n of $E(\mathbb{F}_q)$ generated by G

- **The construction of r**:

  $r = (g^k \bmod p) \bmod q$

Comparing DSA and ECDSA

- **The basic multiplicative group**:

  The subgroup of order q of $\mathbb{Z}_p^*$ generated by $g \longrightarrow$

  The subgroup of order n of $E(\mathbb{F}_q)$ generated by G

- **The construction of r**:

  $r = (g^k \mod p) \mod q \longrightarrow$

  $r = x_1 \mod n$, where $(x_1, y_1) = kG$

Not included

- Security
- Known attacks
- Implementation
- Interoperability
- ECDSA standarts
- Recommended elliptic curves

Thank you for your attention!

D. Johnson, A. Menezes, S. Vanstone: The Elliptic Curve Digital Signature Algorithm (ECDSA). 2001