

Topics in Algebra: Cryptography

Univ.-Prof. Dr. Goulmira ARZHANTSEVA

WS 2019



Cryptography: Overview

Cryptography

- I **Past:** Diffie–Hellman (1976) and **Rivest-Shamir-Adleman** (1977)
- II Nowadays: Blockchain ([1991], 2008)
- III Future: Quantum ([1927, 1982], 1983) and Post-quantum cryptography (1994, 1996)

RSA cryptosystem

Definition: RSA cryptosystem

Let $n = pq$, where p, q are primes. Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}/n\mathbb{Z}$ and

$$\mathcal{K} = \{(n, p, q, d, e) : de = 1 \pmod{\phi(n)}\}$$

For $k = (n, p, q, d, e)$, we define

$$E_k(x) = x^e \pmod{n} \text{ and } D_k(c) = c^d \pmod{n}.$$

Public-key is (n, e) and private-key is (p, q, d) .

Here, x is a plaintext.

Euler's function $\phi(n)$ = the number of positive integers less than n and relatively prime to n .

RSA cryptosystem

Encryption and decryption are inverse operations.

$$n = pq \Rightarrow \phi(n) = (p - 1)(q - 1)$$

We have that $de = 1 \pmod{\phi(n)}$, i.e. $de = t\phi(n) + 1$ for some $t \in \mathbb{Z}$.

RSA cryptosystem

Encryption and decryption are inverse operations.

$$n = pq \Rightarrow \phi(n) = (p - 1)(q - 1)$$

We have that $de = 1 \pmod{\phi(n)}$, i.e. $de = t\phi(n) + 1$ for some $t \in \mathbb{Z}$.

(1) Suppose that $x \in (\mathbb{Z}/n\mathbb{Z})^\times$, then

$$(x^e)^d = x^{t\phi(n)+1} \pmod{n} = (x^{\phi(n)})^t x \pmod{n} = 1^t x \pmod{n} = x \pmod{n}.$$

RSA cryptosystem

Encryption and decryption are inverse operations.

$$n = pq \Rightarrow \phi(n) = (p - 1)(q - 1)$$

We have that $de = 1 \pmod{\phi(n)}$, i.e. $de = t\phi(n) + 1$ for some $t \in \mathbb{Z}$.

(1) Suppose that $x \in (\mathbb{Z}/n\mathbb{Z})^\times$, then

$$(x^e)^d = x^{t\phi(n)+1} \pmod{n} = (x^{\phi(n)})^t x \pmod{n} = 1^t x \pmod{n} = x \pmod{n}.$$

(2) If $x \notin (\mathbb{Z}/n\mathbb{Z})^\times$, then $x = 0 \pmod{p}$ or $x = 0 \pmod{q}$.

If $x = 0 \pmod{p}$, then $(x^e)^d = 0 \pmod{p}$ as well. If the same holds for $\pmod{q}$ we are done by the Chinese remainder theorem.

RSA cryptosystem

Encryption and decryption are inverse operations.

$$n = pq \Rightarrow \phi(n) = (p-1)(q-1)$$

We have that $de = 1 \pmod{\phi(n)}$, i.e. $de = t\phi(n) + 1$ for some $t \in \mathbb{Z}$.

(1) Suppose that $x \in (\mathbb{Z}/n\mathbb{Z})^\times$, then

$$(x^e)^d = x^{t\phi(n)+1} \pmod{n} = (x^{\phi(n)})^t x \pmod{n} = 1^t x \pmod{n} = x \pmod{n}.$$

(2) If $x \notin (\mathbb{Z}/n\mathbb{Z})^\times$, then $x = 0 \pmod{p}$ or $x = 0 \pmod{q}$.

If $x = 0 \pmod{p}$, then $(x^e)^d = 0 \pmod{p}$ as well. If the same holds for $\pmod{q}$ we are done by the Chinese remainder theorem.

Otherwise, $x \not\equiv 0 \pmod{q}$. Then, by [Fermat's little theorem](#),
 $(x^e)^d = x^{ed-1} x = x^{t(p-1)(q-1)} x = (x^{q-1})^{t(p-1)} x \equiv 1^{t(p-1)} x \pmod{q} = x \pmod{q}$. We conclude by the Chinese remainder theorem.

Reminder: Cryptosystem: basic model for secrecy

Definition: Cryptosystem is a 5 -tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ satisfying:

- $\mathcal{P}$ is a finite set of possible **plaintexts**;
- $\mathcal{C}$ is a finite set of possible **ciphertexts**;
- $\mathcal{K}$, the **keyspace**, is a finite set of possible **keys**;
- $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ consists of **encryption functions** $E_k : \mathcal{P} \rightarrow \mathcal{C}$;
- $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ consists of **decryption functions** $D_k : \mathcal{C} \rightarrow \mathcal{P}$;
- For all $e \in \mathcal{K}$ there exists $d \in \mathcal{K}$ such that for all plaintexts $p \in \mathcal{P}$ we have:

$$D_d(E_e(p)) = p$$

- Symmetric cryptosystem: $d = e$
- **Public-key cryptosystem**: d cannot be derived from e in a computationally feasible way

RSA cryptosystem parameters

Algorithm: RSA parameter generation

1. Generate two large primes, p and q , such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a random e with $1 < e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$
4. $d \leftarrow e^{-1} \pmod{\phi(n)}$
5. The public key is (n, e) and the private key is (p, q, d) .

Reminder: Breaking encryption algorithms

- A practical method of determining the **decryption key** is found.
- A weakness in the encryption algorithm leads to a **plaintext**.

Reminder: Breaking encryption algorithms

- A practical method of determining the **decryption key** is found.

RSA: Find the private key (p, q, d) , knowing the public key (n, e)

- A weakness in the encryption algorithm leads to a **plaintext**.

RSA: Invert the RSA encryption function

One-way function

A function that is easy to compute on every input, but **almost always** hard to invert: a **polynomial-time** interceptor will fail to invert the function, except with **negligible** probability.

Definition: Properties of an algorithm

An algorithm is **deterministic** if the output only depends on the input. Otherwise, it is called **probabilistic** or **randomized**.

An algorithm is a **polynomial** algorithm if the number of operations when executed by a multitape Turing machine is $O(n^k)$ for some $k \in \mathbb{N}$ on input of size n .

Turing machine

Turing (1936)

A **tape** is an infinite one-directional line of cells

Definition: the **k -tape Turing machine** is a triple $M = (A, Q, \tau)$ satisfying:

- A is a finite **alphabet** that the k tapes contain: $A = \{\square, \triangleright, 0, 1\}$;
- Q is a finite set of **states**: $q_{\text{start}}, q_{\text{halt}} \in Q$;
- $\tau: Q \times A^k \rightarrow Q \times A^{k-1} \times \{L, S, R\}^k$ is the **transition function** of M .

1st **input** tape, $k - 1$ **work** tapes, k th **output** tape

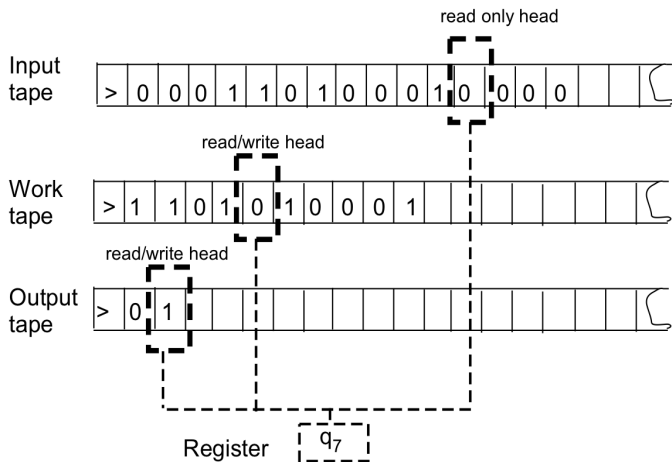
$\square$ the **blank** symbol, $\triangleright$ the **start** symbol

$q_{\text{start}}, q_{\text{halt}}$ the **start** and **halting** states

A **register** contains a current state

A **tape head** reads/write symbols, moves Left, Right or Stays

Turing machine



A 3-tape Turing machine [image: Arora & Barak's book on Complexity]

Computing with Turing machine

- **Start** configuration:

Input tape: $\triangleright$, the input (a **finite** non-blank string), the blank symbols

$k - 1$ tapes: $\triangleright$, the blank symbols

All heads start at the left

- **Step** of computation: apply $\tau: Q \times A^k \rightarrow Q \times A^{k-1} \times \{L, S, R\}^k$

- **Halting**: M halts when it enters q_{halt}

Computing with Turing machine

- **Start** configuration:

Input tape: $\triangleright$, the input (a **finite** non-blank string), the blank symbols

$k - 1$ tapes: $\triangleright$, the blank symbols

All heads start at the left

- **Step** of computation: apply $\tau: Q \times A^k \rightarrow Q \times A^{k-1} \times \{L, S, R\}^k$

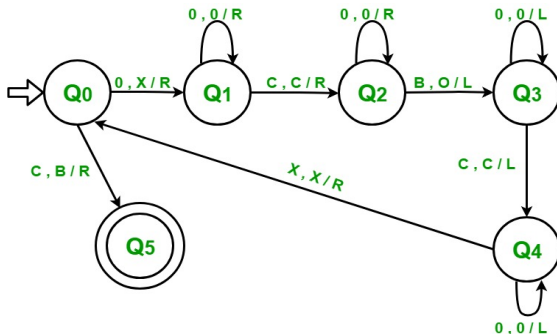
- **Halting**: M halts when it enters q_{halt}

$T: \mathbb{N} \rightarrow \mathbb{N}$ is the running time

Definition: M computes $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ in $T(n)$ -time

For every input $x \in \{0, 1\}^*$, M halts after at most $T(|x|)$ steps, with output $f(x)$.

Example: a Turing machine for the unary addition



Transition function for addition [image: geeksforgeeks.org]

Unary format using 0's: $4 = 0\ 0\ 0\ 0$, $3 = 0\ 0\ 0$

$B = \square$ and $c, x \in A$, $Q_0 = q_{\text{start}}$, $Q_5 = q_{\text{halt}}$.

Turing machine: variants

- **deterministic**: see above
- **non-deterministic**: can be > 1 move for a given state and tape
- **probabilistic**: a non-deterministic Turing machine which chooses moves according to a probability distribution

Turing machine: decision

L is a **language**, i.e. $L \subseteq A^*$ for a finite alphabet A

Definition: M **decides** whether an input $x \in L$

There is a deterministic Turing machine computing

$$\chi_L(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$$

L is **decidable** or **recursive**.

L is computable by M or the decision problem is solvable by M

Turing machine: verification

Definition: M **verifies** whether an input $x \in L$

There is a deterministic Turing machine computing ν_L such that for each given **certificate** $c \in A^*$,

$$\nu_L(x, c) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \text{ or } x \in L \text{ and } c \text{ is wrong} \end{cases}$$

L is **recognizable** or **recursively enumerable**.

M verifies whether c is a valid proof of $x \in L$

Complexity classes

A problem instance x lies in the complexity class

- P if x is solvable by a polynomial **deterministic** algorithm.
- BPP if x is solvable by a polynomial **probabilistic** algorithm.
- BQP if x is solvable by a polynomial deterministic algorithm on a **quantum** computer.
- NP if x is **verifiable** by a polynomial deterministic algorithm.

Complexity classes

A problem instance x lies in the complexity class

- P if x is solvable by a polynomial **deterministic** algorithm.
- BPP if x is solvable by a polynomial **probabilistic** algorithm.
- BQP if x is solvable by a polynomial deterministic algorithm on a **quantum** computer.
- NP if x is **verifiable** by a polynomial deterministic algorithm.

Known: $P \subseteq NP$, $P \subseteq BPP$, Factorisation and Discrete logarithm problem are in $NP \cap BQP$.

Conjectures: $P=BPP$, Factorisation and Discrete logarithm problem are not in $NP \cap BPP$.

Open Problem: Is there $x \in NP \setminus BQP$?

One-way function

A function that is easy to compute on every input, but **almost always** hard to invert: a **polynomial-time** interceptor will fail to invert the function, except with **negligible** probability.

Definition: Negligible function

A function $f: \mathbb{N} \rightarrow \mathbb{R}$ is **negligible** if for each positive polynomial p , $\exists n_0 \in \mathbb{N}$ such that $|f(n)| < \frac{1}{p(n)}$ for all $n \geq n_0$

Example: $f = 2^{-n}$ Non-example: $f = n^{-4}$

One-way function

A function that is easy to compute on every input, but **almost always** hard to invert: a **polynomial-time** interceptor will fail to invert the function, except with **negligible** probability.

Definition: Negligible function

A function $f: \mathbb{N} \rightarrow \mathbb{R}$ is **negligible** if for each positive polynomial p , $\exists n_0 \in \mathbb{N}$ such that $|f(n)| < \frac{1}{p(n)}$ for all $n \geq n_0$

Example: $f = 2^{-n}$ Non-example: $f = n^{-4}$

Notation: $\{0, 1\}^*$ = set of all finite binary strings
($\{0, 1\}^*$, concatenation) is a semi-group

One-way function

Definition: One-way function

A function $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a **one-way function** if

- 1 for all input $x \in \{0, 1\}^*$ there is a polynomial deterministic algorithm that outputs $f(x)$;
- 2 for all polynomial probabilistic algorithm $A: \{0, 1\}^* \rightarrow \{0, 1\}^*$ there is a negligible function negl such that

$$\Pr[A(f(x)) \in f^{-1}(f(x))] \leq \text{negl}(n),$$

where the probability is over the choice of x according to the uniform distribution on $\{0, 1\}^n$, and the randomness of A .

One-way function

Definition: One-way function

A function $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a **one-way function** if

- 1 for all input $x \in \{0, 1\}^*$ there is a polynomial deterministic algorithm that outputs $f(x)$;
- 2 for all polynomial probabilistic algorithm $A: \{0, 1\}^* \rightarrow \{0, 1\}^*$ there is a negligible function negl such that

$$\Pr[A(f(x)) \in f^{-1}(f(x))] \leq \text{negl}(n),$$

where the probability is over the choice of x according to the uniform distribution on $\{0, 1\}^n$, and the randomness of A .

Hard to invert when the input is uniformly distributed. In particular, hard to invert in the **average-case** (not in the worst-case sense =NP-hard).

Hard to invert for **long enough inputs**.

One-way function

We are interested in existence of injective **trapdoor** one-way functions, i.e. those easy to invert with the knowledge of a trapdoor (e.g. with a private-key).

One-way function

We are interested in existence of injective **trapdoor** one-way functions, i.e. those easy to invert with the knowledge of a trapdoor (e.g. with a private-key).

Open problem: Do one-way functions exist?

One-way function

We are interested in existence of injective **trapdoor** one-way functions, i.e. those easy to invert with the knowledge of a trapdoor (e.g. with a private-key).

Open problem: Do one-way functions exist?

Open problem: Is breaking RSA as hard as factoring integers?

RSA keys vs Factoring

Theorem: RSA keys vs Factoring

If the Factoring is not in BPP, then the Asymmetry of RSA is not in BPP.

Asymmetry problem = compute the private key from the public key
Here: compute d (and not, in addition p and q), knowing (n, e) .

RSA keys vs Factoring

Theorem: RSA keys vs Factoring

If the Factoring is not in BPP, then the Asymmetry of RSA is not in BPP.

Asymmetry problem = compute the private key from the public key
Here: compute d (and not, in addition p and q), knowing (n, e) .

Theorem: One-way $\Leftrightarrow$ Pseudorandom

The existence of one-way functions is a **minimal assumption** that is both necessary and sufficient for constructions of **pseudorandom generators** and functions.

Cryptanalysis of RSA: Weakness of the RSA primitive

If the interceptor can factor the modulus n in polynomial-time, then the private key can be efficiently calculated.

Integer factorisation methods

- Trial division
- Pollard's $p - 1$ method
- Elliptic curve factoring method
- Quadratic sieve and Number field sieve
- ...

A 768-bit number factored, two years of computations in 2007-2009.
So, 512-bit keys are 'sufficient'.

In practice, RSA keys are typically 1024- to 2048-bits long.

Cryptanalysis of RSA: Factoring

If n is composite, then it has a prime factor $p \leq \sqrt{n}$

Trial division

Exhaustive search over all successive primes until $\sqrt{n}$.

Complexity of such attacks allow to derive lower bounds on RSA parameters (key size etc.)

Test questions

Question 6

What is the complexity of the RSA parameter generation?

Question 7

Let f be a one-way function. Is $f(f(x))$ necessarily a one-way function?

Question 8

What is the worst-case / average-case complexities of trial division?

Question 9

Design an algorithm computing the square root of a positive integer. What about its complexity? What about its modular variant and its complexity?

Cryptanalysis of RSA: The RSA parameters

Attacks on the RSA function

- Low e or d attack
- Partial d exposure attack

Cryptanalysis of RSA: Implementation attacks

Side-channel attacks

- Time analysis: a correlation between e and the runtime of the cryptographic operation (Solution: delay / blinding)
- Power analysis: monitoring power consumption (Solution: engineering)
- Fault analysis: exploiting errors in cryptographic operations (Solution: verify with e)
- ...

Remainder: RSA cryptosystem parameters

Algorithm: RSA parameter generation

1. Generate two large primes, p and q , such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a **random** e with $1 < e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$
4. $d \leftarrow e^{-1} \pmod{\phi(n)}$
5. The public key is (n, e) and the private key is (p, q, d) .

Randomness of the encryption key is to resist an **informed exhaustive plaintext search attack**. This contrasts the symmetric key encryption.

Cryptanalysis of RSA: practice

Key choice

1024- to 2048-bits long

Strong primes vs Random primes

Multi-prime RSA

More than two primes p and q , hence, primes are smaller for a big n , the encryption is faster.

Cryptanalysis of RSA: practice

Encoding of plaintext = Padding schemes

Plaintext is preprocessed using a probabilistic encoding: same plaintext with the same e gives a different ciphertext.

Goal: resist to the **informed exhaustive plaintext search attack**.

Definition: Hash function

A one-way function $h: \{0, 1\}^* \rightarrow \{0, 1\}^k$ for some $k \in \mathbb{N}$.

Hash: data of arbitrary size is mapped to data of fixed size.

Optimal asymmetric encryption padding (OAEP)

A simplified variant: ignoring the lengths of input / outputs.

Given: x and (n, e) , two hashes h_1 and h_2 , and a **random** number r .

RSA-OAEP encoding

1. Hash r using h_1 and XOR to x :

$$A = h_1(r) \oplus x$$

2. Hash A using h_2 and XOR to r :

$$B = h_2(A) \oplus r$$

3. Apply RSA to the concatenation $A \| B$:

$$c = (A \| B)^e \mod n$$

XOR = the exclusive disjunction = $\oplus = + \mod 2$

Optimal asymmetric encryption padding (OAEP)

Bob can decrypt without knowing r .

RSA-OAEP decoding

1. Decrypt c using d , get $A || B$
2. Hash A using h_2 and XOR to B , get r :

$$h_2(A) \oplus B = h_2(A) \oplus (h_2(A) \oplus r) = r$$

3. Hash r using h_1 and XOR to A , get x :

$$h_1(r) \oplus A = h_1(r) \oplus (h_1(r) \oplus x) = x$$

Hashes: with trapdoor, given by a polynomial algorithm.

Discrete Logarithm problem

Let G be a finite group, $g \in G$ an element, $\langle g \rangle \leq G$ a cyclic subgroup it generates, n its order.

Discrete Logarithm Problem = DLP

Given n, g and $y \in \langle g \rangle$, find the unique integer d , $0 \leq d \leq n - 1$, such that

$$g^d = y.$$

$d := \log_g y$ is called the **discrete logarithm of y to base g** .

Example: $G = (\mathbb{Z}/p\mathbb{Z})^\times$, g is a primitive element mod p , $n = p - 1$

A **primitive** element mod p is an element of $(\mathbb{Z}/p\mathbb{Z})^\times$ of order $p - 1$.

Cryptosystems based on the DLP: ElGamal cryptosystem

ElGamal'1985 cryptosystem is a cryptosystem based on the Discrete Logarithm problem in $(\mathbb{Z}/p\mathbb{Z})^\times$

DLP assumption

1. The DLP in $(\mathbb{Z}/p\mathbb{Z})^\times$ is not in BPP.
2. The Factoring is not in BPP.

ElGamal cryptosystem: Parameter generation

ElGamal'1985 cryptosystem is a cryptosystem based on the Discrete Logarithm problem in $(\mathbb{Z}/p\mathbb{Z})^\times$

Algorithm: ElGamal parameter generation

1. Generate a large prime p .
2. Choose a primitive element $g \in (\mathbb{Z}/p\mathbb{Z})^\times$.
3. Choose a random d with $1 < d < p - 1$.
4. $y \leftarrow g^d \bmod p$
5. The public key is (p, g, y) and the private key is d .

ElGamal cryptosystem: informally

The encryption of a plaintext x is randomised using a random value r chosen by Alice in $\mathbb{Z}/(p-1)\mathbb{Z}$:

There are $p-1$ ciphertexts c that are encryptions of the same x .

ElGamal cryptosystem: informally

The encryption of a plaintext x is randomised using a random value r chosen by Alice in $\mathbb{Z}/(p-1)\mathbb{Z}$:

There are $p-1$ ciphertexts c that are encryptions of the same x .

Randomization of x :

the plaintext x is 'masked' by multiplying it by y^r yielding c_2 .

ElGamal cryptosystem: informally

The encryption of a plaintext x is randomised using a random value r chosen by Alice in $\mathbb{Z}/(p-1)\mathbb{Z}$:

There are $p-1$ ciphertexts c that are encryptions of the same x .

Randomization of x :

the plaintext x is 'masked' by multiplying it by y^r yielding c_2 .

The value g^r is also transmitted giving part c_1 of the ciphertext (c_1, c_2) .

ElGamal cryptosystem: informally

The encryption of a plaintext x is randomised using a random value r chosen by Alice in $\mathbb{Z}/(p-1)\mathbb{Z}$:

There are $p-1$ ciphertexts c that are encryptions of the same x .

Randomization of x :

the plaintext x is 'masked' by multiplying it by y^r yielding c_2 .

The value g^r is also transmitted giving part c_1 of the ciphertext (c_1, c_2) .

Bob, knowing the private key d , can compute y^r from g^r .

Then he can remove the 'mask' by dividing c_2 by y^r to obtain x .

ElGamal cryptosystem

Definition: ElGamal cryptosystem

Let p be a prime and g a primitive element $\bmod p$.

Let $\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^\times$, $\mathcal{C} = (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/p\mathbb{Z})^\times$ and define

$$\mathcal{K} = \{(p, g, d, y) : y = g^d \bmod p\}.$$

For $k = (p, g, d, y)$, and for a secret random number $r \in \mathbb{Z}/(p-1)\mathbb{Z}$, define

$$E_k(x; r) = (c_1, c_2), \text{ where}$$

$$c_1 = g^r \bmod p, \quad \text{and} \quad c_2 = xy^r \bmod p.$$

For $c_1, c_2 \in (\mathbb{Z}/p\mathbb{Z})^\times$, define

$$D_k(c_1, c_2) = c_2(c_1^d)^{-1} \bmod p$$

Public key is (p, g, y) and private key is d .

ElGamal cryptosystem

Encryption and decryption are inverse operations

$$c_2(c_1^d)^{-1} = xy^r \cdot ((g^r)^d)^{-1} \mod p = x \cdot (g^d)^r \cdot ((g^r)^d)^{-1} \mod p = x \mod p$$

Theorem: ElGamal keys vs DLP

If the DLP in $(\mathbb{Z}/p\mathbb{Z})^\times$ is not in BPP, then the Asymmetry of ElGamal is not in BPP.

ElGamal Cryptosystem: finite fields etc.

ElGamal is for an arbitrary finite group G , we had $(\mathbb{Z}/p\mathbb{Z})^\times$. Other groups:

1. The multiplicative group of the finite field $\mathbb{F}_{p^k}$
2. The group of an elliptic curve defined over a finite field.

The group has to satisfy the DLP assumption.

Weierstrass equation

Let $\mathbf{k}$ be a field.

Weierstrass equations

The **affine** Weierstrass equation:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, a_i \in \mathbf{k}.$$

The **homogeneous** Weierstrass equation:

$$E^*: y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz^2 + a_6z^3, a_i \in \mathbf{k}.$$

The **vanishing set**:

$$E(\mathbf{k}) = \{(x : y : z) \in \mathbb{P}^2 \text{ so that } x, y, z \in \mathbf{k} \text{ is a solution of } E^*\} \subseteq \mathbb{P}^2$$

Singular points and curves

The **defining polynomial**:

$$F^* : y^2z + a_1xyz + a_3yz^2 - (x^3 + a_2x^2z + a_4xz^2 + a_6z^3), a_i \in \mathbf{k}.$$

Definition: Singular points and curves

Let $P = (x_0 : y_0 : z_0) \in E(\mathbf{k})$.

1. P is a **singular** point of E if

$$\frac{\partial F^*}{\partial x}(x_0, y_0, z_0) = \frac{\partial F^*}{\partial y}(x_0, y_0, z_0) = \frac{\partial F^*}{\partial z}(x_0, y_0, z_0) = 0.$$

2. E is **singular** if there is a singular point $P \in E(\mathbf{k})$, otherwise E is **nonsingular** or **smooth**.

Example: $(0 : 1 : 0)$ is the only point of E at infinity, i.e. at $z = 0$. It has multiplicity 3 as $E^*(x, y, 0) : x^3 = 0$.

It is not singular: $\frac{\partial F^*}{\partial z}(0, 1, 0) = 1 \neq 0$.

Elliptic curves

Definition: Elliptic curve

E is **elliptic** if E is smooth.

Normal forms

1. If $\text{char } \mathbf{k} \neq 2$ then in E substitute $y \mapsto y - \frac{a_1x+a_3}{2}$ obtaining

$$y^2 = x^3 + a'_2x^2 + a'_4x + a'_6$$

2. If $\text{char } \mathbf{k} \neq 2, 3$ then substitute $x \mapsto x - \frac{1}{3}a'_2$, $a'_2 = a_2 + \frac{a_1^2}{4}$ obtaining

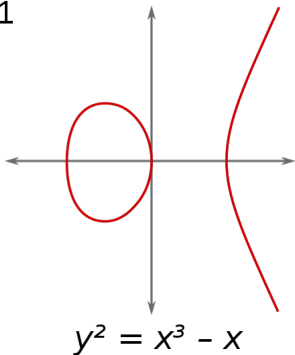
$$y^2 = x^3 + ax + b$$

$\text{char } \mathbf{k} \neq 2, 3 : \text{disc}(x^3 + ax + b) = -16(4a^3 + 27b^2)$

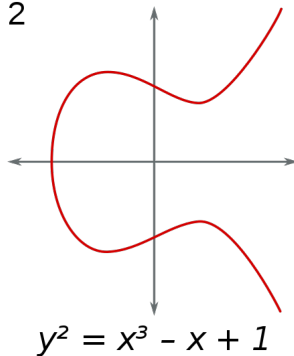
$\text{char } \mathbf{k} \neq 2, y^2 = f(x) = x^3 + a'_2x^2 + a'_4x + a'_6$ is singular $\iff \text{disc } f = 0$

Elliptic curves

1



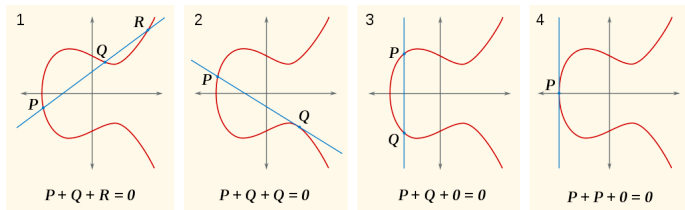
2



Elliptic curves in normal form [image: Wikipedia]

Elliptic curve: The group structure $(E(\overline{\mathbf{k}}), +)$

$\mathbf{k}$ a field, $\overline{\mathbf{k}}$ its algebraic closure



Group structure [image: Wikipedia]

Test questions

Question 10

Which of the following statements are true?

- 1 If the RSA cryptosystem is breakable, then large numbers can be factored.
- 2 Breaking the ECC cryptosystem is equivalent to solving the discrete logarithm problem.
- 3 There is no message expansion in the ECC cryptosystem.

Question 11

Why in practice public-key cryptosystems have longer key lengths than symmetric cryptosystems?