Mathematics for Cryptography II: Security Analysis, Polynomials, Number Theory

Distributed Lab

July 25, 2024

Plan

What will we learn today?

- How to define the security formally.
- How to read... This...

Definition 4 (Hiding Commitment)**.** *A commitment scheme is said to be hiding if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mu(\lambda)$ such that.*

$$\left| \mathrm{P}\left[b = b' \;\middle|\; \begin{array}{l} \mathrm{pp} \leftarrow \mathrm{Setup}(1^\lambda); \\ (x_0, x_1) \in \mathsf{M}_{\mathrm{pp}}^2 \leftarrow \mathcal{A}(\mathrm{pp}), b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} \mathsf{R}_{\mathrm{pp}}, \\ \mathbf{com} = \mathrm{Com}(x_b; r), b' \leftarrow \mathcal{A}(\mathrm{pp}, \mathbf{com}) \end{array} \right] - \frac{1}{2} \right| \leqslant \mu(\lambda)$$

where the probability is over b, r, Setup and $\mathcal{A}$. If $\mu(\lambda) = 0$ then we say the scheme is perfectly hiding.

Definition 5 (Binding Commitment)**.** *A commitment scheme is said to be binding if for all PPT adversaries $\mathcal{A}$ there exists a negligible function μ such that.*

$$\mathrm{P}\left[\mathrm{Com}(x_0; r_0) = \mathrm{Com}(x_1; r_1) \wedge x_0 \neq x_1 \;\middle|\; \begin{array}{l} \mathrm{pp} \leftarrow \mathrm{Setup}(1^\lambda), \\ x_0, x_1, r_0, r_1 \leftarrow \mathcal{A}(\mathrm{pp}) \end{array} \right] \leqslant \mu(\lambda)$$

where the probability is over Setup *and $\mathcal{A}$. If $\mu(\lambda) = 0$ then we say the scheme is* perfectly binding.

Figure: This is not that hard as it seems. Figure from "*Bulletproofs: Short Proofs for Confidential Transactions and More*"

Quick Recap

Quick Recap

1. We know how to read formal statements, like

$$(\forall n \in \mathbb{N})\,(\exists k \in \mathbb{Z}) : \{n = 2k + 1 \vee n = 2k\}$$

2. Group $\mathbb{G}$ is a set with a binary operation that satisfies certain rules. In this lecture, we will use the **multiplicative** notation: for example, g^α means g multiplied by itself α times.
3. Probability of event E is denoted by $\Pr[E]$ – we will need it further.

Security Analysis

Introducing the Cipher: a bit of Formalism

We will consider an example of a cipher to demonstrate the notion of security.

Let us introduce three sets:

- $\mathcal{K}$ – a set of all possible keys.
- $\mathcal{M}$ – a set of all possible messages. For example, $\mathcal{M} = \{0,1\}^n$ – all binary strings of length n.
- $\mathcal{C}$ – a set of all possible ciphertexts.

The cipher is defined over a tuple $(\mathcal{K}, \mathcal{M}, \mathcal{C})$.

Tiny Note

Cryptography is a very formalized field, but everything considered is well-known to you.

Introducing the Cipher

Definition

Cipher scheme $\mathcal{E} = (E, D)$ over the space $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ consists of two efficiently computable methods:

- $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ – encryption method, that based on the provided message $m \in \mathcal{M}$ and key $k \in \mathcal{K}$ outputs the cipher $c = E(k, m) \in \mathcal{C}$.
- $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ – decryption method, that based on the provided cipher $c \in \mathcal{C}$ and key $k \in \mathcal{K}$ outputs the message $m = D(k, c) \in \mathcal{M}$.

We require the **correctness**:

$$\forall k \in \mathcal{K}, \forall m \in \mathcal{M} : D(k, E(k, m)) = m.$$

Question

What else cipher must have to be practical?

Defining Security

Typically, the security is defined as a game between the adversary ($\mathcal{A}$) and the challenger ($\mathcal{C}h$).

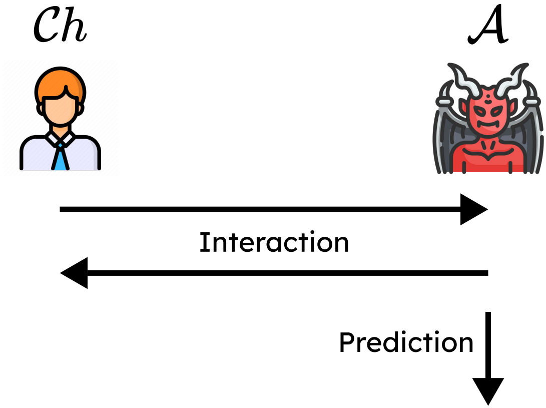
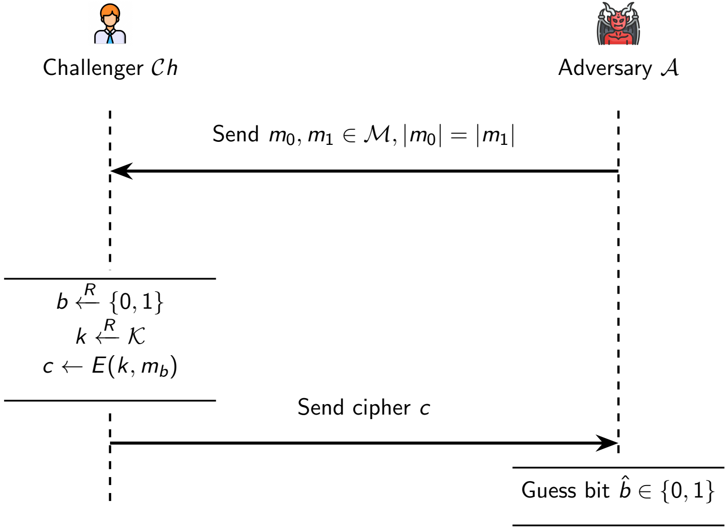


Figure: Challenger $\mathcal{C}h$ follows a straightforward protocol, while the adversary $\mathcal{A}$ might take any strategy to win the game.

Semantic Security: Bit Guessing Game



Semantic Security: Bit Guessing Game

Question #1

Suppose our cipher is perfect. What is the probability that the adversary guesses the bit b correctly? (that is, $b = \hat{b}$)

Definition

Advantage in the Cipher Big Guessing Game of the adversary $\mathcal{A}$ given cipher $\mathcal{E}$ is defined as:

$$\text{SSadv}[\mathcal{E}, \mathcal{A}] := \left|\Pr[\hat{b} = b] - \frac{1}{2}\right|$$

Definition

The cipher $\mathcal{E}$ is called **semantically secure** if for any adversary $\mathcal{A}$ the advantage $\text{SSadv}[\mathcal{E}, \mathcal{A}]$ is **negligible**.

Advantage

Question #2

If adversary can guess the bit with probability 0.000000001, is the cipher semantically secure?

Question #3

If adversary has the advantage 0.0001, is the cipher semantically secure?

Note

Advantage is just a measure of how many information is leaked to the adversary. The smaller the advantage, the better the cryptographic system. Formally, we want advantage to be **negligible**.

But what does it mean to be **negligible**?

Neglibility

In practice, neglibible means below 2^{-128} (called 128 bits of security). In theory, however...

Definition

Security parameter, denoted by $\lambda \in \mathbb{N}$, is just a variable that measures the input size of some computational program.

Example

The security of the group of points on the elliptic curve (say, $\mathbb{G}$) is defined by the number of bits in the order of the group. So if $|\mathbb{G}|$ is 256 bits long, then we can define $\lambda = 256$.

Neglible Functions

Now, the probability of advesary winning the game depends on λ and we want this dependence to decrease rapidly.

Definition

A function $f : \mathbb{N} \to \mathbb{R}$ is called **negligible** if for all $c \in \mathbb{R}_{>0}$ there exists $n_c \in \mathbb{N}$ such that for any $n \geq n_c$ we have $|f(n)| < 1/n^c$.

The alternative definition, which is problably easier to interpret, is the following.

Theorem

A function $f : \mathbb{N} \to \mathbb{R}$ is **negligible** *if and only if for any $c \in \mathbb{R}_{>0}$, we have*

$$\lim_{n \to \infty} f(n)n^c = 0$$

Neglible Functions Examples

Example

The function $f(\lambda) = 2^{-\lambda}$ is negligible since for any $c \in \mathbb{R}_{>0}$ we have $\lim_{\lambda\to\infty} 2^{-\lambda}\lambda^c = 0$.
The function $g(\lambda) = \frac{1}{\lambda!}$ is also negligible for similar reasons.

Example

The function $h(\lambda) = \frac{1}{\lambda}$ is not negligible since for $c = 1$ we have

$$\lim_{\lambda\to\infty} \frac{1}{\lambda} \times \lambda = 1 \neq 0$$

Question

Is the function $u(\lambda) = \lambda^{-10000}$ negligible?

Discrete Logarithm Problem (DL)

Definition

Assume that $\mathbb{G}$ is a cyclic group of prime order r generated by $g \in \mathbb{G}$. Define the following game:

1. Both challenger $\mathcal{C}h$ and adversary $\mathcal{A}$ take a description $\mathbb{G}$ as an input: order r and generator $g \in \mathbb{G}$.
2. $\mathcal{C}h$ computes $\alpha \xleftarrow{R} \mathbb{Z}_r$, $u \leftarrow g^\alpha$ and sends $u \in \mathbb{G}$ to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ outputs $\hat{\alpha} \in \mathbb{Z}_r$.

We define $\mathcal{A}$'s **advantage in solving the discrete logarithm problem in** $\mathbb{G}$, denoted as $\text{DLadv}[\mathcal{A}, \mathbb{G}]$, as the probability that $\hat{\alpha} = \alpha$.

Definition

The **Discrete Logarithm Assumption** holds in the group $\mathbb{G}$ if for any efficient adversary $\mathcal{A}$ the advantage $\text{DLadv}[\mathcal{A}, \mathbb{G}] \leq \text{negl}(\lambda)$.

Computational Diffie-Hellman Assumption (CDH)

Definition

Let $\mathbb{G}$ be a cyclic group of prime order r generated by $g \in \mathbb{G}$. Define the following game:

1. $\mathcal{C}h$ computes $\alpha, \beta \xleftarrow{R} \mathbb{Z}_r, u \leftarrow g^\alpha, v \leftarrow g^\beta, w \leftarrow g^{\alpha\beta}$ and sends $u, v \in \mathbb{G}$ to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs $\hat{w} \in \mathbb{G}$.

We define $\mathcal{A}$'s **advantage in solving the computational Diffie-Hellman problem in** $\mathbb{G}$, denoted as $\mathrm{CDHadv}[\mathcal{A}, \mathbb{G}]$, as the probability that $\hat{w} = w$.

Definition

The **Computational Diffie-Hellman Assumption** holds in the group $\mathbb{G}$ if for any efficient adversary $\mathcal{A}$ the advantage $\mathrm{CDHadv}[\mathcal{A}, \mathbb{G}]$ is negligible.

Decisional Diffie-Hellman Assumption (DDH)

Definition

Let $\mathbb{G}$ be a cyclic group of prime order r generated by $g \in \mathbb{G}$. Define the following game:

1. $\mathcal{C}h$ computes $\alpha, \beta, \gamma \xleftarrow{R} \mathbb{Z}_r$, $u \leftarrow g^{\alpha}$, $v \leftarrow g^{\beta}$, $w_0 \leftarrow g^{\alpha\beta}$, $w_1 \leftarrow g^{\gamma}$. Then, $\mathcal{C}h$ flips a coin $b \xleftarrow{R} \{0,1\}$ and sends (u, v, w_b) to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs the predicted bit $\hat{b} \in \{0,1\}$.

We define $\mathcal{A}$'s **advantage in solving the Decisional Diffie-Hellman problem in** $\mathbb{G}$, denoted as $\text{DDHadv}[\mathcal{A}, \mathbb{G}]$, as

$$\text{DDHadv}[\mathcal{A}, \mathbb{G}] := \left| \Pr[b = \hat{b}] - \frac{1}{2} \right|$$

Let us show when DDH does not hold!

Even-ordered Cyclic Group

Theorem

Suppose that $\mathbb{G}$ is a cyclic group of an even order. Then, the Decision Diffie-Hellman Assumption does not hold in $\mathbb{G}$. In fact, there is an efficient adversary $\mathcal{A}$ with an advantage $1/4$.

Idea of proof. We first prove the following statement:

Lemma

Based on $u = g^\alpha \in \mathbb{G}$, it is possible to determine the parity of α.

Lemma proof. Notice that if $\alpha = 2\alpha'$, then

$$u^n = g^{\alpha n} = g^{2\alpha' n} = (g^{2n})^{\alpha'} = 1^{\alpha'} = 1$$

Therefore, if $u = g^\alpha, v = g^\beta, w = g^\gamma$, adversary knows the parity of α, β, γ. He then checks if parity of $\alpha\beta$ equals parity of γ.

Polynomials

Definition

Definition

A **polynomial** $f(x)$ is a function of the form

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n = \sum_{k=0}^{n} c_k x^k,$$

where $c_0, c_1, \dots, c_n$ are coefficients of the polynomial.

Definition

A set of polynomials depending on x with coefficients in a field $\mathbb{F}$ is denoted as $\mathbb{F}[x]$, that is

$$\mathbb{F}[x] = \left\{ p(x) = \sum_{k=0}^{n} c_k x^k : c_k \in \mathbb{F},\ k = 0, \dots, n \right\}.$$

Examples of Polynomials

Example

Consider the finite field $\mathbb{F}_3$. Then, some examples of polynomials from $\mathbb{F}_3[x]$ are listed below:

1. $p(x) = 1 + x + 2x^2$.
2. $q(x) = 1 + x^2 + x^3$.
3. $r(x) = 2x^3$.

If we were to evaluate these polynomials at $1 \in \mathbb{F}_3$, we would get:

1. $p(1) = 1 + 1 + 2 \cdot 1 \bmod 3 = 1$.
2. $q(1) = 1 + 1 + 1 \bmod 3 = 0$.
3. $r(1) = 2 \cdot 1 = 2$.

More about polynomials

Definition

The **degree** of a polynomial $p(x) = c_0 + c_1x + c_2x^2 + \dots$ is the largest $k \in \mathbb{Z}_{\geq 0}$ such that $c_k \neq 0$. We denote the degree of a polynomial as $\deg p$. We also denote by $\mathbb{F}^{(\leq m)}[x]$ a set of polynomials of degree at most m.

Example

The degree of the polynomial $p(x) = 1 + 2x + 3x^2$ is 2, so $p(x) \in \mathbb{F}_3^{(\leq 2)}[x]$.

Theorem

For any two polynomials $p, q \in \mathbb{F}[x]$ and $n = \deg p$, $m = \deg q$, the following two statements are true:

1. $\deg(pq) = n + m$.
2. $\deg(p + q) = \max\{n, m\}$ *if* $n \neq m$ *and* $\deg(p + q) \leq m$ *for* $m = n$.

Roots of Polynomials

Definition

Let $p(x) \in \mathbb{F}[x]$ be a polynomial of degree $\deg p \geq 1$. A field element $x_0 \in \mathbb{F}$ is called a root of $p(x)$ if $p(x_0) = 0$.

Example

Consider the polynomial $p(x) = 1 + x + x^2 \in \mathbb{F}_3[x]$. Then, $x_0 = 1$ is a root of $p(x)$ since $p(x_0) = 1 + 1 + 1 \bmod 3 = 0$.

Theorem

Let $p(x) \in \mathbb{F}[x], \deg p \geq 1$. Then, $x_0 \in \mathbb{F}$ is a root of $p(x)$ if and only if there exists a polynomial $q(x)$ (with $\deg q = n - 1$) such that

$$p(x) = (x - x_0)q(x)$$

Polynomial Division

Theorem

Given $f, g \in \mathbb{F}[x]$ *with* $g \neq 0$, *there are unique polynomials* $p, q \in \mathbb{F}[x]$ *such that*

$$f = q \cdot g + r,\ 0 \leq \deg r < \deg g$$

Example

Consider $f(x) = x^3 + 2$ and $g(x) = x + 1$ over $\mathbb{R}$. Then, we can write $f(x) = (x^2 - x + 1)g(x) + 1$, so the remainder of the division is $r \equiv 1$. Typically, we denote this as:

$$f \operatorname{div} g = x^2 - x + 1, \quad f \bmod g = 1.$$

The notation is pretty similar to one used in integer division.

Polynomial Divisibility

Definition

A polynomial $f(x) \in \mathbb{F}[x]$ is called **divisible** by $g(x) \in \mathbb{F}[x]$ (or, g **divides** f, written as $g \mid f$) if there exists a polynomial $h(x) \in \mathbb{F}[x]$ such that $f = gh$.

Theorem

If $x_0 \in \mathbb{F}$ is a root of $p(x) \in \mathbb{F}[x]$, then $(x - x_0) \mid p(x)$.

Definition

A polynomial $f(x) \in \mathbb{F}[x]$ is said to be **irreducible** in $\mathbb{F}$ if there are no polynomials $g, h \in \mathbb{F}[x]$ both of degree more than 1 such that $f = gh$.

Polynomial Divisibility

Example

A polynomial $f(x) = x^2 + 16$ is irreducible in $\mathbb{R}$. Also $f(x) = x^2 - 2$ is irreducible over $\mathbb{Q}$, yet it is reducible over $\mathbb{R}$: $f(x) = (x - \sqrt{2})(x + \sqrt{2})$.

Example

There are no polynomials over complex numbers $\mathbb{C}$ with degree more than 2 that are irreducible. This follows from the *fundamental theorem of algebra*. For example, $x^2 + 16 = (x - 4i)(x + 4i)$.

Interpolation

Question

How can we define the polynomial?

The most obvious way is to specify coefficients $(c_0, c_1, \ldots, c_n)$. Can we do it in a different way?

Theorem

Given $n+1$ distinct points $(x_0, y_0), \ldots, (x_n, y_n)$, there exists a unique polynomial $p(x)$ of degree at most n such that $p(x_i) = y_i$ for all $i = 0, \ldots, n$.

Illustration with two points

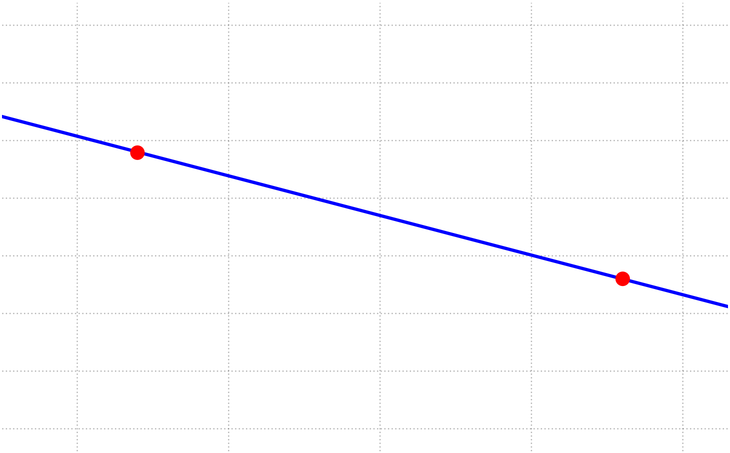


Figure: 2 points on the plane uniquely define the polynomial of degree 1 (linear function).

Illustration with five points

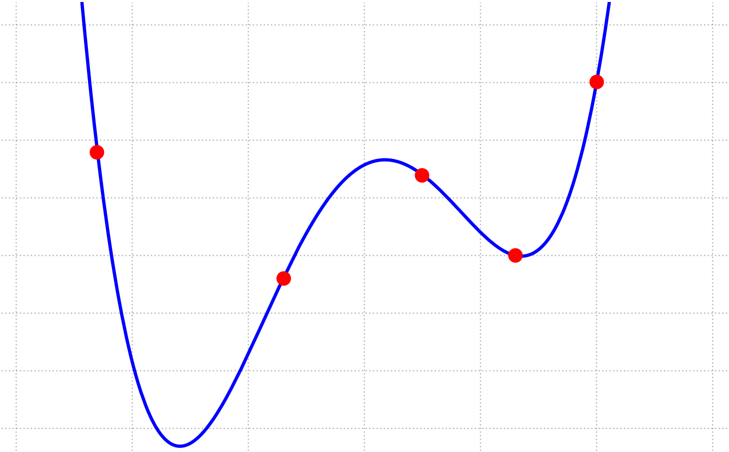


Figure: 5 points on the plane uniquely define the polynomial of degree 4.

Illustration with three points

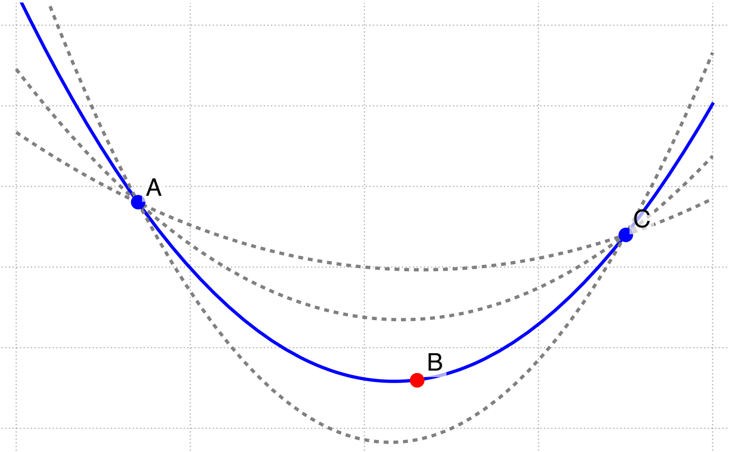


Figure: 2 points are not enough to define the quadratic polynomial $(c_2x^2 + c_1x + c_0)$.

Lagrange Interpolation

One of the ways to interpolate the polynomial is to use the Lagrange interpolation.

Theorem

Given $n+1$ distinct points $(x_0, y_0), \ldots, (x_n, y_n)$, the polynomial $p(x)$ that passes through these points is given by

$$p(x) = \sum_{i=0}^{n} y_i \ell_i(x), \quad \ell_i(x) = \prod_{i=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

Application: Shamir Secret Sharing

Motivation

How to share a secret α among n people in such a way that any t of them can reconstruct the secret, but any $t-1$ cannot?

Definition

Secret Sharing scheme is a pair of efficient algorithms $(\mathrm{Gen}, \mathrm{Comb})$ which work as follows:

- $\mathrm{Gen}(\alpha, t, n)$: probabilistic sharing algorithm that yields n shards $(\alpha_1, \dots, \alpha_t)$ for which t shards are needed to reconstruct the secret α.
- $\mathrm{Comb}(\mathcal{I}, \{\alpha_i\}_{i \in \mathcal{I}})$: deterministic reconstruction algorithm that reconstructs the secret α from the shards $\mathcal{I} \subset \{1, \dots, n\}$ of size t.

Note

Here, we require the **correctness**: for every $\alpha \in F$, for every possible output $(\alpha_1, \dots, \alpha_n) \leftarrow \mathsf{Gen}(\alpha, t, n)$, and any t-size subset $\mathcal{I}$ of $\{1, \dots, n\}$ we have

$$\mathsf{Comb}(\mathcal{I}, \{\alpha_i\}_{i \in \mathcal{I}}) = \alpha. \tag{1}$$

Definition

Now, **Shamir's protocol** works as follows: $F = \mathbb{F}_q$ and

- $\mathsf{Gen}(\alpha, k, n)$: choose random $k_1, \dots, k_{t-1} \xleftarrow{R} \mathbb{F}_q$ and define the polynomial

$$\omega(x) := \alpha + k_1 x + k_2 x^2 + \dots + k_{t-1} x^{t-1} \in \mathbb{F}_q^{\leq (t-1)}[x], \tag{2}$$

  and then compute $\alpha_i \leftarrow \omega(i) \in \mathbb{F}_q, \ i = 1, \dots, n$.

Shamir's Protocol

Definition

- $\text{Comb}(\mathcal{I}, \{\alpha_i\}_{i\in\mathcal{I}})$: interpolate the polynomial $\omega(x)$ using the Lagrange interpolation and output $\omega(0) = \alpha$.

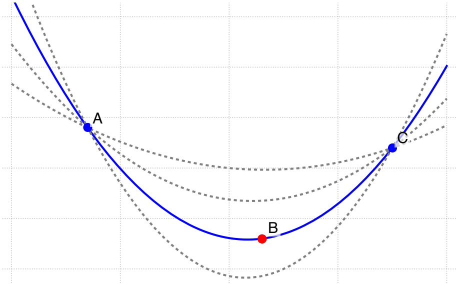


Figure: There are infinitely many quadratic polynomials passing through two blue points (gray dashed lines). However, knowing the red point allows us to uniquely determine the polynomial and thus get its value at 0.

Reed-Solomon Codes

Definition

- Reed-Solomon codes is an error-correcting algorithm based on polynomials. It allows to restore lost or corrupted data, implement threshold secret sharing and it is used in some ZK protocols.

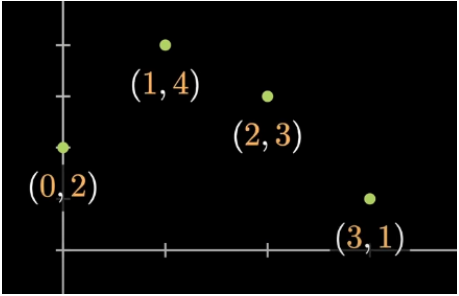


Figure: Polynomial with degree n can be uniquely defined using $(n+1)$ unique points. Defining more points on the same polynomial adds a redundancy, which can be used to restore the polynomial even if some points are missing.

Reed-Solomon Codes

The error-correcting ability of a Reed-Solomon code is $n - k$, the measure of redundancy in the block. If the locations of the error symbols are not known in advance, then a Reed-Solomon code can correct up to $n - k/2$ erroneous symbols.

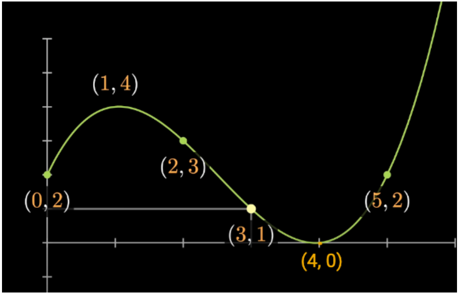
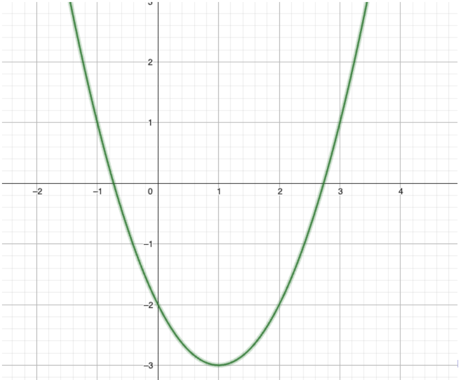


Figure: Polynomial with degree n can be uniquely defined using $(n + 1)$ unique points. Defining more points on the same polynomial adds a redundancy, which can be used to restore the polynomial even if some points are missing.

Schwartz-Zippel Lemma

Definition

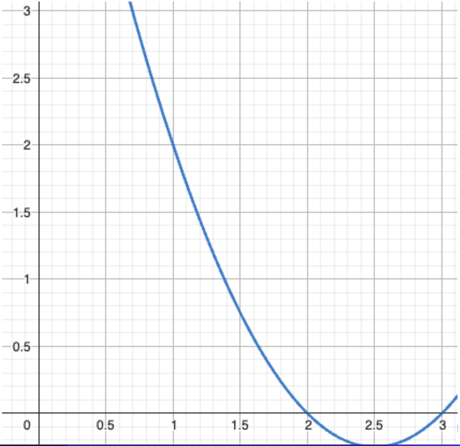
Let $\mathbb{F}$ be a field. Let $f(x_1, x_2, ..., x_n)$ be a polynomial of total degree d. Suppose that f is not the zero polynomial. Let S be a finite subset of $\mathbb{F}$. Let $r_1, r_2, ... r_n$ be chosen at random uniformly and independently from S. Then the probability that $f(r_1, r_2, ..., r_n) = 0$ is $\leq \frac{d}{|S|}$.



Schwartz-Zippel Lemma

Definition

Let $F = \mathbb{F}_3$, $f(x) = x^2 - 5x + 6$, $S = F$, $r \xleftarrow{R} \mathbb{F}_3$. Schwartz-Zippel lemma says that the probability that $f(r) = 0$ is $\leq \frac{2}{3}$.



Given two polynomials P, Q with degree d in a field $\mathbb{F}_p$, for $r \xleftarrow{R} \mathbb{F}_3$: $\Pr[P(r) == Q(r)] \leq \frac{d}{p}$. For large fields, where $\frac{d}{p}$ is negligible, this property allows to succinctly check the equality of polynomials.

Proof.

Let $H(x) := P(x) - Q(x)$. Than for each $P(x) = Q(x) \rightarrow H(x) = 0$.

Applying Schwartz-Zippel lemma, the probability of $H(x) = 0$ for $x \xleftarrow{R} \mathbb{F}$ is $\leq \frac{d}{|S|}$. □

Basic Number Theory

Primes

Primes are often used when doing almost any cryptographic computation. A prime number is a natural number ($\mathbb{N}$) that is not a product of two smaller natural number. In other words, the prime number is divisible only by itself and 1. The first primes are: 2, 3, 5, 7, 11...

Deterministic prime tests

A primality test is deterministic if it outputs True when the number is a prime and False when the input is composite with probability 1. Here is an example implementation in Rust:

```
fn is_prime(n: u32) -> bool {
    let square_root = (n as f64).sqrt() as u32;

    for i in 2.. = square_root {
        if n % i == 0 {
            return false;
        }
    }

    true
}
```

Probabilistic prime tests

A primality test is probabilistic if it outputs `True` when the number is a prime and `False` when the input is composite with probability less than 1. Fermat Primality and Miller-Rabin Primality Tests are examples of probabilistic primality test.

Theorem

Let p be a prime number and a be an integer not divisible by p. Then $a^{p-1} - 1$ is always divisible by p: $a^{p-1} \equiv 1 \pmod{p}$

Probabilistic prime tests

The key idea behind the Fermat Primality Test is that if for some a not divisible by n we have $a^{n-1} \not\equiv 1 \pmod{n}$ then n is definitely NOT prime. Athough, false positives are possible.

For example, consider $n = 15$ and $a = 4$.
$4^{15-1} \equiv 1 \pmod{15}$, but $n = 15 = 3 \cdot 5$ is composite.

Solution: a is picked many times. The probability that a composite number is mistakenly called prime for k iterations is $2^{-k} = \dfrac{1}{2^k}$.

Probabilistic prime tests

There exists a problem with such an algorithm in the form of **Carmichael numbers**, which are numbers that are Fermat pseudoprime to all bases. Asymptotic complexity $O(\log^3 n)$.

```
# n = number to be tested for primality
# k = number of times the test will be repeated
def is_prime(n, k):
    i = 1
    while i <= k:
        a = rand(2, n - 1)

        if a^(n - 1) != 1 (mod n):
            return False

        i++

    return True
```

Greatest Common Divisor

Greatest common divisor (GCD) of two or more integers, which are not all zero, is the largest positive integer that divides each of the integers.

Example

$gcd(8, 12) = 4$, $gcd(3, 15) = 3$, $gcd(15, 10) = 5$.

Computing GCD using Euclid's algorithm.

The is based on the fact that, given two positive integers a and b such that $a > b$, the common divisors of a and b are the same as the common divisors of $a - b$ and b.

It can be observed, that it can be further optimized, by using $a \pmod b$, instead of $a - b$.

For example, $gcd(26, 8) = gcd(18, 8) = gcd(10, 8) = gcd(2, 8)$ can be optimized to $gcd(26, 8) = gcd(26 \pmod 8, 8) => gcd(2, 8)$

Computing *GCD* using Euclid's algorithm.

```
int gcd(a, b):
 if (b == 0):
     return a
 return gcd(b, a % b)
```

Algorithm can be implemented using recursion. Base of the recursion is $gcd(a, 0) = a$. Provided algorithm work with $O(log(N))$ asymptotic complexity.

Least Common Multiple

Least common multiple (LCM) of two integers a and b, is the smallest positive integer that is divisible by both a and b.
The least common multiple can be computed from the greatest common divisor with the formula:
$lcm(a, b) = \frac{|ab|}{gcd(a,b)}$

```
    int lcm(a, b):
     return a * (b / gcd(a, b))
```

Modular inverse

Modular multiplicative inverse of an integer a is an integer b such that $a \cdot b \equiv 1 \pmod{m}$.

One of the ways to compute the modular inverse is by using Euler‘s theorem:
$a^{\phi(m)} \equiv 1 \pmod{m}$, where ϕ is Euler’s totient function.

For prime numbers, where $\phi(m) = m - 1$:
$a^{m-2} \equiv a^{-1} \pmod{m}$.

```
a_inverse = powmod(a, m-2, m)
```

Thanks for your attention!