Number Theory and Cryptography

Simple Cryptosystems

Piyush Beegala Bipplav Tiwari

IIT Kanpur

Week 5
June 23, 2022

Contents

Introduction

The idea behind cryptography is to have a secure communication between two parties so that only the only the sender and intended recipient of a message can view its contents.

Let's start with some basic notations.

Plaintext

The message that has to be sent. For example, *"MTH sucks".*

Ciphertext

The message in its disguised form. For example, *"GNB mowem"*.

In general, we say that the *plaintext* and *ciphertext* are written in some *alphabet* with (say) N letters, where every letter corresponds to an integer in the set $\{0, 1, 2, \ldots, N-1\}$.

Note that these letters could be anything: alphabets, numbers, symbols, blanks, etc.

Introduction

Encryption (*Enciphering*)

The process of converting the *plaintext* to *ciphertext*. If f is the enciphering transformation, then it can be represented as,

$$P \to f(P), \text{ where } P \text{ is the } \textit{plaintext}.$$

Note that f must be a bijective tranformation.

Decryption (*Deciphering*)

The process of retrieving the *ciphertext* from the *plaintext*. If f is defined as above, then it can be represented as,

$$C \to f^{-1}(C), \text{ where } C \text{ is the } \textit{ciphertext}.$$

The *plaintext* P may be divided into units $P = P_1 P_2 \cdots P_k$, where each unit is transformed independently as $C_i = f(P_i)$. The final *ciphertext* is then generated by concatenating the transformed units as $C = C_1 C_2 \cdots C_k$.

Permutation Map

Lets start with single lettered units. Suppose we have an alphabet with N letters and every letter corresponds to a number in $\{0, 1, 2, \ldots, N-1\}$.

Then any *permutation* σ of this set will be a valid *enciphering*.

Example (*Caeser Cipher*)

For the Latin alphabet, $N = 26$. For $P \in \{0, 1, 2, \ldots, 25\}$ Consider the permutation,

$$\sigma(P) = \begin{cases} P - 6 & \text{if } P \geq 6 \\ P + 20 & \text{if } P < 6 \end{cases}$$

For enciphering "*MTH*", note that M, T, H correspond to $12, 19, 7$ respectively. Under σ, these numbers transform to $6, 13, 1$ which correspond to G, N, B respectively. Hence, the ciphertext will be *"GNB"*.

Note that σ may not be a simple *shifting map* as shown in the above example. We can use any permutation.

Linear Transform

Like before, we have single lettered units and an alphabet with N letters, where every letter corresponds to a number in $\{0, 1, 2, \ldots, N-1\}$.

Affine Map

Let a, b be fixed integers such that $\gcd(a, N) = 1$. For $P \in \{0, 1, \ldots, N-1\}$, an *affine map* $f : [N] \to [N]$ can be defined as,

$$f(P) = aP + b \pmod{N}$$

The *deciphering* transformation for the above map can be evaluated as,

$$C \equiv aP + b \implies P \equiv a^{-1}(C - b) \pmod{N}$$

$$\implies f^{-1}(C) = a'C + b' \pmod{N}$$

where $a' = a^{-1} \pmod{N}$, and $b' = -a^{-1}b \pmod{N}$.

Linear Transform

Can you *break* this cryptosystem? Consider the following example.

Example (*Frequency Analysis*)

Suppose an *affine map cryptosystem* uses a 28−letter alphabet consisting of $(A - Z)$, a *blank* and ?, where $(A - Z)$ have numerical equivalents $(0 - 25)$, *blank* $= 26$ and $? = 27$. On examining the *ciphertexts* we find that the top two most commonly used letters of *ciphertext* are *B*, *?* respectively and the top two commonly used letters in the English are *blank* and *E* respectively.

We can safely assume that *blank* $\rightarrow B$, and $E \rightarrow$?. This gives the following set of congruences in (mod 28),

$$a' + b' \equiv 26 \text{ and } 27a' + b' = 4$$

$$\implies -26a' \equiv 22 \implies 2a' \equiv 22 \implies a' \equiv 11 \text{ or } 25$$

$$\implies a' \equiv 11, b' \equiv 15 \text{ or, } a' \equiv 25, b' \equiv 1$$

We then *decipher* using both the possibilities and keep the sensible one.

Linear Transform

Now, say we have double lettered units in both the *plaintexts* and *ciphertexts*. Like before, every letter corresponds to a number in $[N]$.

Diagraph Tranformation

For every two letter block XY with $X, Y \in \{0, 1, 2, \ldots, N-1\}$, consider a numerical equivalent,

$$P(XY) = XN + Y \in \{0, 1, \ldots, N^2 - 1\}$$

To this, we can then apply any *enciphering* transformation $f : [N^2] \to [N^2]$. The transformed value can then be converted again under the inverse transformation P^{-1} to obtain the *ciphered* unit.

$$XY \xrightarrow{P} P(XY) \xrightarrow{f} f(P(XY)) \xrightarrow{P^{-1}} X'Y'$$

Exercise

Let f above be an *affine map*. Can the resulting cryptosystem be broken using a *frequency analysis* as shown in the previous example?

Public Key Cryptography

The fixed parameters used to define a cryptosystem are called *keys* of that cryptosystem.

Public Key

The *key* used by others to *encrypt* messages intended for you will be your *public key*. This may also be referred to as your *enciphering key*. For example, (a, b) in the *affine map*. This *key* is visible to everyone.

Private Key

The *key* used by you to *decrypt* messages intended for you will be your *private key*. This may also be referred to as your *deciphering key*. For example, (a', b') in the *affine map*. This *key* is only visible to you.

In the simple cryptosystems we discussed earlier, the knowledge of the *public key* is equivalent to the knowledge of the *public key*. This would mean that anyone can *decrypt* the messages intended for you, using your *public key*.

One-way Functions

The idea is to use *enciphering functions*, which are *easy* to compute on any input, but it is *hard* to compute the inverse, given an image of an input. Such a function is called a *one-way function*.

Hence, most cryptographic algorithms are based on a certain *hard* problem. Like *Integer Factorization, Discrete Logarithm,* etc.

Here, the terms *"easy"* and *"hard"* are used in the sense of *computational complexity*.

But then, shouldn't it be equally *hard* for the intended recipient to *decipher* the *ciphertext*?

Trapdoor

Additional secret information, which allows *easy* computation of *inverse* of a *one-way function*.

In our scenario, we can think of the *private key* as a *trapdoor* to a *enciphering function*.

RSA (*Rivest–Shamir–Adleman*)

The *RSA* public key cryptosystem is based on the computational difficulty of *factoring a big integer*. It is one of the oldest public key cryptosystem.

To make sure that we are dealing with numbers, assume that you have broken the message into units and we are talking about transformations of numerical equivalents P of these units. Let your keys be,

$$\textbf{Public Key} = (n, e) \in \mathbb{Z}^2 \qquad \textbf{Private Key} = (n, d) \in \mathbb{Z}^2$$

Encryption Scheme

Any message P intended for you will be *encrypted* to C using your *public key* as follows.

$$C = f(P) = P^e \pmod{n}$$

Decryption Scheme

You will use your *private key* to *decrypt* any message C as follows.

$$P = f^{-1}(C) = C^d \pmod{n} = P^{ed} \pmod{n} = P \pmod{n}$$

RSA - Key Generation

The *keys* for the *RSA* algorithm are generated as follows.

1. Choose two distinct prime numbers q_1 and q_2 and set $n = q_1 q_2$. Prime numbers can be efficiently found using a *primality test*. These prime numbers are randomly found and are similar in magnitude. This is done in order to make it harder to factor n.
2. Compute $\phi(n) = \phi(q_1)\phi(q_2) = (q_1 - 1)(q_2 - 1)$.
3. Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.

   This is because we need e to have an inverse modulo $\phi(n)$.
4. Choose d such that $d \equiv e^{-1} \bmod (\phi(n))$.
5. Use (n, e, d) to form *public*, *private* keys and discard all other intermediate values $q_1, q_2, \phi(n)$, etc.

RSA - Proof

Since $ed \equiv 1 \pmod{\phi(n)}$,

$$ed = \lambda\phi(n) + 1, \text{ for some integer } \lambda$$

For any integer P, we have the following in modulo q_1,

$$P^{q_1-1} \equiv \begin{cases} 0 & \text{if } q_1 | P \\ 1 & \text{else} \end{cases} \implies P^{\lambda\phi(n)} \equiv (P^{q_1-1})^{\lambda(q_2-1)} \equiv \begin{cases} 0 & \text{if } q_1 | P \\ 1 & \text{else} \end{cases}$$

$$\implies P^{ed} \equiv P \cdot P^{\lambda\phi(n)} \equiv \begin{cases} 0 & \text{if } q_1 | P \\ P & \text{else} \end{cases} \equiv P$$

We can do the same for q_2 to finally get,

$$P^{ed} \equiv P \pmod{q_1} \text{ and } P^{ed} \equiv P \pmod{q_2}$$

Using *Chinese Remainder Theorem*, we have

$$P^{ed} \equiv P \pmod{n}$$

This proves the *decryption scheme*.

Key Exchange

In certain cryptosystems (unlike RSA), it may be required for the two parties to *secretly exchange* cryptographic keys beforehand for establishing a secure communication channel.

For example in *affinemap*, (a, b) cannot be made public since they can be used to easily compute (a', b').

Hence, there must be a way to establish a *shared secret* among two parties so that no one else can obtain a copy of this *secret*.

In cryptography, this process is know as *Key Exchange*.

Diffie Hellman Key Exchange

Suppose *Alice* and *Bob* want to agree upon a secret key. They *publicly* agree upon a *prime* p and a generator g modulo p.

Alice's Keys

Alice *secretly* chooses an integer a.

Private Key $= a$ **Public Key** $= A = g^a \pmod{p}$

Bob's Keys

Bob *secretly* chooses an integer b.

Private Key $= b$ **Public Key** $= B = g^b \pmod{p}$

Shared Secret

Alice *privately* evaluates: $B^a \pmod{p} = (g^b)^a \pmod{p} = g^{ba} \pmod{p}$.
Bob *privately* evaluates: $A^b \pmod{p} = (g^a)^b \pmod{p} = g^{ab} \pmod{p}$.

Shared Secret $= S = g^{ab} \pmod{p}$