# Notes for An Introduction to Mathematical Cryptography

Mohamed-Amine Azzouz

2024-05-04

# Contents

# Lecture 1: An Introduction to Cryptography

# Introduction

**Note: these are a set of notes to help me summarize my casual reading of the textbook of the same name.**
The book presents some mathematical background that is necessary to understand cryptography's algorithms, starting with algebra and basic number theory. I will only list those that I forgot for my own learning.
For the sake of notation, Bob encrypts a message for Alice to read. Over an insecure channel, Eve is there to try to decrypt it.

## 1.1 Simple substitution ciphers

- Idea: to encrypt a message, replace each letter with another. Someone intercepting the message would need to try 26! combinations using brute force.
- In particular, one could shift each letter by a fixed amount (modulo 26), giving us the formula:

$$(\text{Ciphertext Letter}) \equiv_{26} (\text{Plaintext Letter}) + (\text{Secret Key})$$

## 1.3 Modular Arithmetic

More notation: $a \equiv_m b$ is the same as saying $a \equiv b \pmod{m}$.

### 1.3.2 The Fast Powering Algorithm

**Proposition**

**Step 1.** Compute the binary expansion of $A$ as

$$A = A_0 + A_1 \cdot 2 + A_2 \cdot 2^2 + A_3 \cdot 2^3 + \ldots + A_r \cdot 2^r$$

with $A_0, \ldots, A_r \in \{0, 1\}$, where we may assume that $A_r = 1$.

**Step 2.** Compute the powers $g^{2i} \pmod N$ for $0 \le i \le r$ by successive squaring,

$$\begin{aligned}
a_0 &\equiv g \pmod N \\
a_1 &\equiv a_0^2 \pmod N \\
a_2 &\equiv a_1^2 \equiv g^2 \pmod N \\
a_3 &\equiv a_2^2 \equiv g^4 \pmod N \\
&\vdots \\
a_r &\equiv a_{r-1}^2 \equiv g^{2^r} \pmod N.
\end{aligned}$$

Each term is the square of the previous one, so this requires $r$ multiplications.
**Step 3.** Compute $g^A \pmod N$ using the formula

$$\begin{aligned}
g^A &= g^{A_0 + A_1 \cdot 2 + A_2 \cdot 2^2 + A_3 \cdot 2^3 + \ldots + A_r \cdot 2^r} \\
&= g^{A_0} \cdot (g^2)^{A_1} \cdot (g^2)^{A_2} \cdot (g^2)^{A_3} \cdot \ldots \cdot (g^2)^{A_r} \\
&\equiv a_{A_0}^0 \cdot a_{A_1}^1 \cdot a_{A_2}^2 \cdot a_{A_3}^3 \cdot \ldots \cdot a_{A_r}^r \pmod N.
\end{aligned}$$

Note that the quantities $a_0, a_1, \ldots, a_r$ were computed in Step 2. Thus the product can be computed by looking up the values of the $a_i$'s whose exponent $A_i$ is 1 and then multiplying them together. This requires at most another $r$ multiplications.

## 1.4 Prime Numbers, Unique Factorization, and Finite Fields

**Proposition: Method to compute $a^{-1}$ mod $p$**

Use the Euclidean Algorithm to compute $u, v \in \mathbb{Z}$ such that:

$$au + pv = 1$$

Then, $a^{-1} \equiv_p u$.

## 1.5 Powers and Primitive Roots in Finite Fields

**Proposition: Fermat's Little Theorem**

Let $p$ be a prime number and let $a$ be any integer. Then,

$$a^{p-1} \equiv \begin{cases} 1 \pmod{p} & \text{if } p \nmid a, \\ 0 \pmod{p} & \text{if } p \mid a. \end{cases}$$

As a corollary, $a^{-1} \equiv_p a^{p-2}$ for prime $p$, $p \nmid a$.

# 1.7 Symmetric and Asymmetric Ciphers

## 0.6.1 1.7.1 Symmetric/Private Ciphers

For each key $k$, we get a pair of functions $e_k : \mathcal{M} \to \mathcal{C}$ and $d_k : \mathcal{C} \to \mathcal{M}$ satisfying the decryption property $d_k(e_k(m)) = m$ for all $m \in \mathcal{M}$.
In other words, for every key $k$, the function $d_k$ is the inverse function of the function $e_k$.
In particular, this means that $e_k$ must be one-to-one, since if $e_k(m) = e_k(m')$, then

$$m = d_k(e_k(m)) = d_k(e_k(m')) = m'.$$

It is safest for Alice and Bob to assume that Eve knows the encryption method that is being employed, i.e. she knows the functions $e$ and $d$.

**Definition: Kerckhoff's Principle**

The security of a cryptosystem should depend only on the secrecy of the key, and not on the secrecy of the encryption algorithm itself.

If $(K, M, C, e, d)$ is to be a successful cipher, it must have the following properties:

1. For any key $k \in K$ and plaintext $m \in M$, it must be easy to compute the ciphertext $e_k(m)$.
2. For any key $k \in K$ and ciphertext $c \in C$, it must be easy to compute the plaintext $d_k(c)$.
3. It must be hard to decrypt any set of ciphertexts without the key $k$.

Another desirable but hard property to get is:

4. Security against a known plaintext attack: if Eve is given any pair of plaintexts and their encryptions, she can't use that to find the key. (The simple substitution cipher does not follow that)

An even more secure property is:

5. Security against a chosen plaintext attack: if Eve chooses to have any pair of plaintexts and their encryptions available, she can't use that to find the key.

### 0.6.2 1.7.5 Random Bit Sequences and Symmetric Ciphers

We would like to construct a mapping $R : \mathcal{K} \times \mathbb{Z} \to \{0, 1\}$ such that with a relatively small key $k \in \mathcal{K}$, one can encrypt arbitrarily large messages $j \in \mathbb{Z}$. This is the goal of a pseudo-random number generator, for $R$ to be one it needs to fulfill:

1. For all $k \in \mathcal{K}$ and all $j \in \mathbb{Z}$, it is easy to compute $R(k, j)$.
2. Given an arbitrarily long sequence of integers $j_1, j_2, \ldots, j_n$ and given all of the values $R(k, j_1), R(k, j_2), \ldots, R(k, j_n)$, it is hard to determine $k$.
3. Given any list of integers $j_1, j_2, \ldots, j_n$ and given all of the values $R(k, j_1), R(k, j_2), \ldots, R(k, j_n)$, it is hard to guess the value of $R(k, j)$ with better than a 50% chance of success for any value of $j$ not already in the list.

If we could find a function $R$ with these three properties, then we could use it to turn an initial key $k$ into a sequence of bits $R(k, 1), R(k, 2), R(k, 3), R(k, 4), \ldots$, which would be the key to a one-time pad. Indeed, the one-time pad, which consists in encrypting a message by XORing it with a one-time use key of the same length, is provably secure but hard to use because the keys need to be the same size as the message.
No function has been proven to be a proper pseudorandom number generator. Some good candidates that have been working so far are algorithms that use some kind of "mixup" operations, similarly to what operations in modular arithmetic can induce. This is the basis of encryption protocols like the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES), which are widely used today. Another method is to choose $R$ such that its inverse is known to be very difficult to compute. However, this method is not as efficient for practical use.

### 0.6.3 1.7.6 Public/Asymmetric Ciphers Make a First Appearance

Public key cryptography is a way for Alice and Bob to communicate over an insecure channel without having secretly exchanged keys beforehand. If in private key cryptography, Alice and Bob first safely exchanged private keys before communicating, the revolutionary idea of public key cryptography is to use a public key, made by Alice, with which absolutely anyone can encrypt messages to and send to Alice. However, only Alice had the private key to decrypt the messages.
A real life analogue to this would be Alice installing a safe in a public space, with a slot (the public key) into which anyone can insert messages. Then, only Alice would have the key to the safe (the private key) to open it and read the messages.

**Definition: Public key cryptography**

An element $k = (k_{\text{priv}}, k_{\text{pub}})$ of the key space $\mathcal{K}$ is such that any public key has an encryption $e_{k_{\text{pub}}} : \mathcal{M} \to \mathcal{C}$, and each private key has a decryption $d_{k_{\text{priv}}} : \mathcal{C} \to \mathcal{M}$. Those maps are such that for any $k \in \mathcal{K}$, $d_{k_{\text{priv}}}(e_{k_{\text{pub}}}(m)) = m$ for all $m \in \mathcal{M}$.

For an asymmetric cipher to be secure, it must be difficult for Eve to compute the decryption function $d_{k_{\text{priv}}}(c)$, even if she knows the public key $k_{\text{pub}}$.

In practice, public key algorithms are much slower than private key cryptography, so what usually happens is that an asymmetric cipher is used to share a key for a symmetric cipher, which is how data is actually sent.

# Lecture 2: Discrete Logarithms and Diffie–Hellman

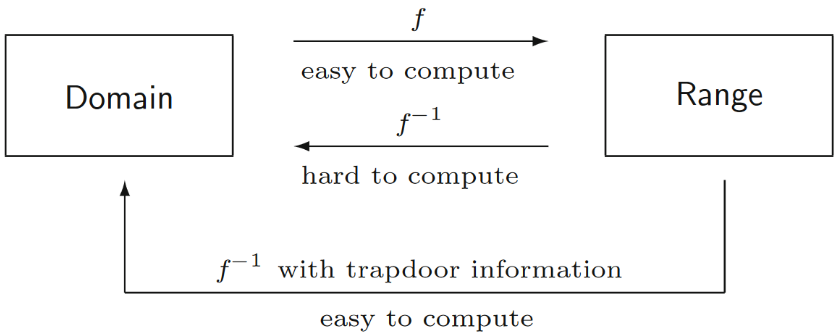


Figure 2.1: Illustration of a one-way trapdoor function

Idea: $f$ is the encryption (sometimes using a public key), $f^{-1}$ is the decryption that Eve tries to compute, with a trapdoor for those who have the key.

## 2.2 The Discrete Logarithm Problem

**Definition: The Discrete Logarithm (DLP)**

$g^x \equiv_p h$ always has a solution if $p$ is prime, since in that case, $\mathbb{F}_p^* = \langle g \rangle$. When a solution exists, one can define the multivalued solution to be $x \equiv_{p-1} \log_g(h)$. The multivalue is from Fermat's Little Theorem.

**Proposition**

Notice that:

$$\log_p(ab) = \log_p(a) + \log_p(b).$$

Otherwise, the discrete logarithm is very hard to compute, since unlike the continuous logarithm, it has no monotone properties. In general, note that the problem can be stated in a general group.

## 2.3 Diffie–Hellman Key Exchange

XXX

## 2.4 The Elgamal Public Key Cryptosystem

XXX

## 2.5 An Overview of the Theory of Groups

XXX

**Definition**

Let $G$ be a group and let $a \in G$ be an element of the group. Suppose there exists a positive integer $d$ with the property that $a^d = e$. The smallest such $d$ is called the order of $a$. If there is no such $d$, then $a$ is said to have infinite order.

**Proposition**

Let $G$ be a finite group. Then every element of $G$ has finite order. Further, if $a \in G$ has order $d$ and if $a^k = e$, then $d \mid k$.

**Proposition: Lagrange's Theorem**

Let $G$ be a finite group and let $a \in G$. Then the order of $a$ divides the order (or size) of $G$. More precisely, let $n = |G|$ be the order of $G$ and let $d$ be the order of $a$, i.e., $a^d$ is the smallest positive power of $a$ that is equal to $e$. Then $a^n = e$ and $d \mid n$.

## 2.6 How Hard Is the Discrete Logarithm Problem?

XXX

## 2.7 A Collision Algorithm for the DLP

XXX

## 2.8 The Chinese Remainder Theorem

XXX

## 2.9 The Pohlig–Hellman Algorithm

XXX

## 2.10 Rings, Quotient Rings, Polynomial Rings, and Finite Fields

This is not needed until Chapter 6 and 7. XXX

# Lecture 3: Integer Factorization and RSA

XXXXXXXXXXDDDD

## 3.1 Euler's Formula and Roots Modulo $pq$

XXX

## 3.2 The RSA Public Key Cryptosystem

XXX

## 3.3 Implementation and Security Issues

XXX

## 3.4 Primality Testing

XXX

## 3.5 Pollard's $p-1$ Factorization Algorithm

XXX

## 3.6 Factorization via Difference of Squares

XXX

## 3.7 Smooth Numbers and Sieves

XXX

## 3.8 The Index Calculus and Discrete Logarithms

XXX

## 3.9 Quadratic Residues and Quadratic Reciprocity

XXX

## 3.10 Probabilistic Encryption

XXX

# Lecture 4: Digital Signatures

:3 XXXXXXXXXXDDDD

## 4.1 What Is a Digital Signature?

XXX

## 4.2 RSA Digital Signatures

XXX

## 4.3 Elgamal Digital Signatures and DSA

XXX

# Lecture 5: Combinatorics, Probability, and Information Theory

:3 XXXXXXXXXXDDDD

## 5.1 Basic Principles of Counting

XXX

## 5.2 The Vigenere Cipher

XXX

## 5.3 Probability Theory

XXX

## 5.4 Collision Algorithms and Meet-in-the-Middle Attacks

XXX

## 5.5 Pollard's $\rho$ Method

XXX

## 5.6 Information Theory

XXX

## 5.7 Complexity Theory and $\mathcal{P}$ Versus $\mathcal{NP}$

XXX

# Lecture 6: Elliptic Curves and Cryptography

:3 X

## 6.1 Elliptic Curves

XXX

## 6.2 Elliptic Curves over Finite Fields

XXX

## 6.3 The Elliptic Curve Discrete Logarithm Problem

XXX

## 6.4 Elliptic Curve Cryptography

XXX

## 6.5 The Evolution of Public Key Cryptography

XXX

## 6.6 Lenstra's Elliptic Curve Factorization Algorithm

XXX

## 6.7 Elliptic Curves over $\mathbb{F}_2$ and over $\mathbb{F}_{2^k}$

XXX

## 6.8 Bilinear Pairings on Elliptic Curves

XXX

## 6.9 The Weil Pairing over Fields of Prime Power Order

XXX

## 6.10 Applications of the Weil Pairing

XXX

# Lecture 7: Lattices and Cryptography

:3 X

## 7.1 A Congruential Public Key Cryptosystem

XXX

## 7.2 Subset-Sum Problems and Knapsack Cryptosystems

XXX

## 7.3 A Brief Review of Vector Spaces

XXX

## 7.4 Lattices: Basic Definitions and Properties

XXX

## 7.5 Short Vectors in Lattices

XXX

## 7.6 Babai's Algorithm

XXX

## 7.7 Cryptosystems Based on Hard Lattice Problems

XXX

## 7.8 The GGH Public Key Cryptosystem

XXX

## 7.9 Convolution Polynomial Rings

XXX

## 7.10 The NTRU Public Key Cryptosystem

XXX

## 7.11 NTRUEncrypt as a Lattice Cryptosystem

XXX

## 7.12 Lattice-Based Digital Signature Schemes

XXX

## 7.13 Lattice Reduction Algorithms

XXX

## 7.14 Applications of LLL to Cryptanalysis

XXX

# Lecture 8: Additional Topics in Cryptography

:3 X

## 8.1 Hash Functions

XXX

## 8.2 Random Numbers and Pseudorandom Number

XXX

## 8.3 Zero-Knowledge Proofs

XXX

## 8.4 Secret Sharing Schemes

XXX

## 8.5 Identification Schemes

XXX

## 8.6 Padding Schemes and the Random Oracle Model

XXX

## 8.7 Building Protocols from Cryptographic Primitives

XXX

## 8.8 Blind Digital Signatures, Digital Cash, and Bitcoin

XXX

## 8.9 Homomorphic Encryption

XXX

## 8.10 Hyperelliptic Curve Cryptography

XXX

## 8.11 Quantum Computing

XXX

## 8.12 Modern Symmetric Cryptosystems: DES and AES

XXX

# Bibliography

[1] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*, 2nd ed. New York, NY: Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-1-4939-1711-2