INTRODUCTION TO CRYPTOGRAPHY

Math 490 - Graduation Project

2025-2026 Fall Semester

Author
Bengisu EĞİN

Supervisor
Asst. Prof. Dr. Erkan
Murat TÜRKAN

Department of Mathematics

Çankaya University

1 INTRODUCTION

Cryptography is the study of secure communication techniques that protect information from unauthorized access. It has evolved from simple classical methods, like the Caesar cipher which uses basic shifts, to modern public-key systems that secure global digital data. The security of modern systems, such as RSA and ElGamal, relies on the computational difficulty of specific mathematical problems. While RSA is based on the difficulty of factoring large prime numbers, ElGamal depends on the Discrete Logarithm Problem. These systems use a public key for encryption and a private key for decryption, ensuring that only the intended recipient can access the message.

The purpose of selecting this topic stems from a strong personal interest in the intersection of mathematics and information security. Although I did not have the opportunity to take a formal elective course in this area during my undergraduate studies, I became curious about the critical importance of cryptography in the modern digital world. Therefore, I identified this project as a stepping stone to explore the intricacies of the field through independent study and to elevate my knowledge and understanding in this area to an academic level.

This project examines the mathematical foundations of cryptography, including modular arithmetic, Fermat's Little Theorem, and Euler's Totient Function. It further explores the transition from classical ciphers to modern algorithms like RSA and ElGamal, providing a comprehensive analysis of their structures and practical applications.

2 MATHEMATICAL PREREQUISITES

2.1 Modular Arithmetic and Congruence Relations

Modular arithmetic, which is also known as congruence arithmetic, is one of the key topics in number theory and has a direct impact on cryptography. Instead of allowing numbers to grow without limit, this approach studies numerical operations within a fixed range that repeats periodically. This range is defined by a positive integer called the modulus.

Within this system, integers are grouped according to the remainders they produce when divided by the modulus. If two integers leave the same remainder after division by a given integer n, they are considered congruent modulo n. This idea is written using the notation:

$$a \equiv b \pmod{n}$$

What this expression tells us is: the difference between a and b can be divided by n. In other words, the integer n is a divisor of the value $(a - b)$. Even though the definition itself is straightforward, it forms the basis of many algorithms used in modern cryptographic systems.

2.2 Fermat's Little Theorem

Fermat's Little Theorem was introduced in the seventeenth century by the French mathematician Pierre de Fermat. It describes an important relationship between prime numbers and modular arithmetic and has become a fundamental tool in both number theory and cryptography. The theorem is usually stated in two different but closely related forms.

If p is a prime number and a is any integer, then the following congruence holds:

$$a^p \equiv a \pmod{p}$$

A second version, which is more commonly used in cryptographic applications, applies when p is a prime and a is coprime to p:

$$a^{p-1} \equiv 1 \pmod{p}$$

In the example below, this theorem is verified for the values $a = 7$ and $p = 19$ (a prime number).

The value of 7^2:

$$7^2 = 49 = (2 \times 19) + 11 \equiv 11 \pmod{19}$$

The value of 7^4:

$$(7^2)^2 \equiv 11^2 = 121 = (6 \times 19) + 7 \equiv 7 \pmod{19}$$

The value of 7^8:

$$(7^4)^2 \equiv 7^2 = 49 \equiv 11 \pmod{19}$$

The value of 7^{16}:

$$(7^8)^2 \equiv 11^2 = 121 \equiv 7 \pmod{19}$$

Final Result (a^{p-1}):

$$7^{18} \equiv 7^{16} \times 7^2 = 7 \times 11 = 77 = (4 \times 19) + 1 \equiv 1 \pmod{19}$$

This result is especially useful in practice because it allows large exponentiation operations to be reduced efficiently using modular arithmetic, which is essential for encryption systems.

As an example, consider $a = 7$ and $p = 19$. Since 19 is a prime number and $\gcd(7, 19) = 1$, Fermat's Little Theorem predicts that:

$$7^{18} \equiv 1 \pmod{19}$$

By computing powers of 7 step by step and reducing each result modulo 19, the final value indeed becomes 1. This confirms the validity of the theorem for the chosen values and demonstrates how it simplifies large numerical calculations.

2.3 Euler's Totient Function and Euler's Theorem

Leonhard Euler extended Fermat's ideas by developing a more general theorem that applies to a wider set of integers. Euler's Theorem plays a central role in modular arithmetic and is one of the main mathematical foundations of modern public-key cryptography, especially the RSA algorithm.

Euler's Totient Function

Euler's Totient Function, denoted by $\varphi(n)$, gives the number of positive integers smaller than n that are relatively prime to n. Two integers are said to be relatively prime if their greatest common divisor is equal to 1.

In simpler terms, $\varphi(n)$ counts how many integers k satisfy:

$$1 \leq k < n \quad \text{and} \quad \gcd(k, n) = 1$$

Euler's Theorem

If a and n are positive integers such that $\gcd(a, n) = 1$, then the following congruence always holds:

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Example 1:

Prime number case ($n = 37$):

Since 37 is a prime number, all integers from 1 to 36 are relatively prime to it. Therefore, $\varphi(37) = 36$.

Example 2:

Composite number case ($n = 35$):

When n is a composite number, all numbers less than n are not coprime to it. List of numbers less than 35 that are coprime to 35:

$$\{1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34\}$$

We obtain:

$$\varphi(35) = 24$$

2.4 Primitive Roots and Their Cryptographic Importance

Primitive roots are used in cryptographic systems such as Diffie–Hellman and ElGamal for two main reasons.
First, a primitive root produces the maximum possible cycle length when raised to successive powers modulo a prime. This means that every nonzero residue modulo p appears exactly once.
Second, although computing powers such as $g^x \pmod{p}$ is easy, reversing the process to find x is extremely difficult. This difficulty is known as the discrete logarithm problem and forms the basis of security for several modern cryptosystems.
Let n be an integer and let a be an integer such that $\gcd(a, n) = 1$. If the powers of a modulo n generate all integers that are relatively prime to n, then a is called a primitive root modulo n.

More formally, a is a primitive root modulo n if:

$$\mathrm{ord}_n(a) = \varphi(n)$$

Let us examine primitive roots for the prime $p = 7$ ($\varphi(7) = 6$):
For $a = 2$:

$$2^1 \equiv 2, \quad 2^2 \equiv 4, \quad 2^3 \equiv 1 \pmod{7}$$

The cycle ends after 3 steps ($\mathrm{ord}_7(2) = 3 < 6$). Therefore, 2 is not a primitive root modulo 7.

For $a = 3$:

$$\begin{aligned} 3^1 &\equiv 3 \\ 3^2 &\equiv 2 \\ 3^3 &\equiv 6 \\ 3^4 &\equiv 4 \\ 3^5 &\equiv 5 \\ 3^6 &\equiv 1 \pmod 7 \end{aligned}$$

All values $\{1, 2, 3, 4, 5, 6\}$ are generated ($\text{ord}_7(3) = 6$).
Thus, 3 is a primitive root modulo 7.

3 CLASSICAL ENCRYPTION TECHNIQUES

3.1 Monoalphabetic Ciphers (Caesar Cipher)

The Caesar cipher is one of the earliest known encryption methods and is named after Julius Caesar. It works by shifting each letter of the plaintext by three positions in the alphabet. for example, during encryption, A would become D, and during decryption, D would revert to A).

Although it has historical importance, this cipher provides very weak security. For example, using a shift of three transforms the message "CAESAR WAS GREAT" into "FDHVDU ZDV JUHDW".

3.2 Polyalphabetic Ciphers (Vigenère Cipher)

The Vigenère cipher improves upon simple substitution by using a keyword that applies different shifts to different letters. This approach was an important development in classical cryptography, as it reduced the effectiveness of basic frequency analysis attacks.

3.3 Hill Cipher

The Hill cipher, developed by Lester S. Hill, was one of the first encryption schemes to operate on blocks of letters rather than individual characters. It uses matrix multiplication in modular arithmetic, converting letters into numbers (A=0, B=1, ..., Z=25) and encrypting them using an invertible key matrix.

4 MODERN PUBLIC-KEY CRYPTOSYSTEMS

4.1 Merkle–Hellman Knapsack Cryptosystem

The Merkle–Hellman cryptosystem was introduced in 1978 and is one of the earliest public-key encryption schemes. It is based on the computational difficulty of the knapsack (subset sum) problem.

The private key consists of a super-increasing sequence, which is easy to solve. The public key is obtained by transforming this sequence using modular arithmetic, producing a sequence that appears random.

Encryption is performed by summing selected elements of the public key, while decryption uses modular inverses and the structure of the super-increasing sequence. Despite its early success, the system was broken by Adi Shamir in 1982 and is no longer considered secure.

Super-Increasing Sequence

In a super-increasing sequence, each element is greater than the sum of all previous elements.

EXAMPLE: ({1, 2, 4, 8, 16, ...}) (powers of 2) or ({2, 3, 7, 15, 31, ...})

Key Generation and Encryption Process

The super-increasing sequence is kept secret and is called the private key.

Using modular arithmetic, the private key is transformed by the sender into a sequence that appears complex and difficult to solve; this sequence becomes the public key.

The message is encrypted using the public key.

The resulting ciphertext is the sum of a selected subset of the elements of the public key.

The receiver, who knows the private key, applies an inverse operation in modular arithmetic to transform the encrypted sum back into a sum over the easy-to-solve super-increasing sequence.

In this way, the original message is successfully recovered.

EXAMPLE:

First, we must choose a super-increasing sequence. Each element must be greater than the sum of all previous elements.

Secret Sequence (S): $\{2, 5, 11, 22, 45\}$

Modulus (m): A number greater than the sum of the elements in the sequence must be chosen. $(2 + 5 + 11 + 22 + 45 = 85)$. Therefore, let us choose $m = 90$.

Multiplier (w): A number that is relatively prime to m must be selected ($\gcd(w, m) = 1$). Let us choose $w = 31$, which is suitable since $\gcd(31, 90) = 1$.

The public key is generated using the formula "$b_i = w \cdot s_i \bmod m$":

$$31 \cdot 2 = 62 \pmod{90}$$

$$31 \cdot 5 = 155 \equiv 65 \pmod{90}$$

$$31 \cdot 11 = 341 \equiv 71 \pmod{90}$$

$$31 \cdot 22 = 682 \equiv 52 \pmod{90}$$

$$31 \cdot 45 = 1395 \equiv 45 \pmod{90}$$

Public Key (B): $(\{62, 65, 71, 52, 45\})$

Step 2: Encryption

Suppose the message we want to encrypt is "11010" (5 bits).

Plaintext (X): $(x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 0)$

The ciphertext (C) is the sum of the products of the message bits and the elements of the public key:

$$C = (1 \cdot 62) + (1 \cdot 65) + (0 \cdot 71) + (1 \cdot 52) + (0 \cdot 45)$$

$$C = 62 + 65 + 52 = 179$$

Our encrypted message, 179, is sent to the receiver.

Step 3: Decryption

The receiver uses w, m, and the super-increasing sequence from the private key to decrypt the message.

Finding the Modular Inverse:

First, calculate the modular inverse of w modulo m. From the equation:

$$31 \cdot d \equiv 1 \pmod{90}$$

We find $w^{-1} = 61$.

(Check: $31 \cdot 61 = 1891$, $1891 = (21 \cdot 90) + 1$ so the remainder is 1, confirming that 61 is indeed the modular inverse.)

Calculating the New Target Sum:

The ciphertext (C) is multiplied by the modular inverse:

$$C' = C \cdot w^{-1} \bmod m$$

$$C' = 179 \cdot 61 \bmod 90$$

$$C' = 10919 \bmod 90$$

$$10919 \div 90 = 121 \text{ remainder } 29$$

So, $C' = 29$.

Finding the Subset Sum in the Super-Increasing Sequence:

Now, we need to represent the number 29 using the elements of our secret super-increasing sequence $\{2, 5, 11, 22, 45\}$. Using the "greedy" algorithm, we start from the largest element:

$$\begin{aligned} 29 \geq 45? \text{ No} &\rightarrow \text{Bit } 5 = 0 \\ 29 \geq 22? \text{ Yes} &\rightarrow 29 - 22 = 7 \rightarrow \text{Bit } 4 = 1 \\ 7 \geq 11? \text{ No} &\rightarrow \text{Bit } 3 = 0 \\ 7 \geq 5? \text{ Yes} &\rightarrow 7 - 5 = 2 \rightarrow \text{Bit } 2 = 1 \\ 2 \geq 2? \text{ Yes} &\rightarrow 2 - 2 = 0 \rightarrow \text{Bit } 1 = 1 \end{aligned}$$

So the recovered plaintext is 11010.

Arranging the recovered bits in order (1, 2, 3, 4, 5): 11010.

Result: The original message has been successfully recovered.

4.2 RSA Cryptosystem

RSA is one of the most widely used public-key cryptosystems and was introduced in 1977 by Rivest, Shamir, and Adleman. Security is ensured by the difficulty of factoring two large prime numbers and by keeping these prime numbers secret. When a sufficiently large key is used, there is no feasible method to break it.

RSA operates in three main stages: key generation, encryption, and decryption. The correctness of the algorithm relies on Euler's Theorem and the condition

$$e \cdot d \equiv 1 \pmod{\varphi(n)}$$

Start
Input X, Y
Sum = X + Y
Display SUM
End

As long as this relationship holds, the original plaintext can be recovered correctly after decryption.

RSA algorithm processes data in blocks. Each encryption block (that is, the numerical representation of the plaintext M to be encrypted) must not exceed a certain limit. This limit is the modulus n used in the RSA system.

For the system to work correctly and securely from a mathematical perspective, each plaintext block to be encrypted must satisfy

$$M < n.$$

The condition

$$2^i < n \leq 2^{i+1}$$

is used to select a secure block size. That is, if the RSA key is k-bit (for example, 2048-bit), each data block is configured to be smaller than 2^{2048}. Since the blocks are smaller than n, the encryption process remains within modular arithmetic.

Exponential Encryption Method and Functional Structure

The fundamental operation of the RSA algorithm is based on two complementary mathematical processes.

Encryption Process

The plaintext data block M is transformed into ciphertext using the recipient's public key exponent e. This operation is performed using the following modular arithmetic formula:

$$C \equiv M^e \pmod{n}.$$

Decryption Process

When the encrypted data block C reaches the recipient, the recipient uses their private key exponent d to recover the original data. This reverse operation is defined by the following mathematical expression:

$$M \equiv C^d \pmod{n}.$$

For a data block satisfying $M < n$, the encryption and decryption processes should yield the original data at the end. This can be expressed as

$$M^{ed} \equiv M \pmod{n}.$$

When this congruence is satisfied, the operation of the algorithm is considered correct.

Computational Efficiency

For the practical usability of this system, both the encryption process $M^e \pmod{n}$ and the decryption process $C^d \pmod{n}$ must be completed by computers in a reasonable amount of time, regardless of how large the numbers involved are.

Irreversibility (Security)

Although the public key parameters e and n are publicly known, computing the private key d from these values must be computationally infeasible.

Functional Relationship Between the Keys

In RSA, the link between the public and private keys is formed through Euler's Totient Function $\varphi(n)$. Let

$$n = p \cdot q,$$

where p and q are prime numbers. The selection of the keys is based on the following criteria.

Modular Inverse Relationship

The encryption exponent e and the decryption exponent d are multiplicative inverses modulo $\varphi(n)$. This relationship is expressed by the congruence

$$ed \equiv 1 \pmod{\varphi(n)}.$$

This also implies that

$$d \equiv e^{-1} \pmod{\varphi(n)}.$$

Coprimality Condition

For this inverse to exist, the chosen value e must be coprime with $\varphi(n)$; that is, it must satisfy

$$\gcd(\varphi(n), e) = 1.$$

Parameter	Definition	Status	Source
p, q	Two large prime numbers	private	Randomly selected
$n = p \cdot q$	Common modulus	public	Computed
$\Phi(n)$	$(p-1)(q-1)$	private	Computed
e	Public exponent	public	Selected $(1 < e < \Phi(n))$
d	Private exponent	private	$d = e^{-1} \pmod{\Phi(n)}$

Table 1: RSA Parameters

Encryption and Decryption Stages

Once the necessary numerical conditions are satisfied, the communication process between users proceeds as follows:

Encryption: The sender (User B) encrypts the message using the recipient's (User A's) publicly announced (e, n) key pair. The operation is performed using the formula:

$$C = M^e \pmod{n}$$

Decryption: The recipient (User A) converts the received ciphertext C back into the original message using their private key d, which is known only to them, by performing:

$$M = C^d \pmod{n}$$

EXAMPLE :

Step 1: Key Generation

To set up the system, the private and public key components are calculated as follows:

- Prime Number Selection: Two prime numbers are chosen: $p = 17$ and $q = 11$.
- Modulus Calculation (n): $n = p \times q = 17 \times 11 = 187$.
- Euler's Totient Function $(\phi(n))$: $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$.

- Public Exponent (e) Selection: A value e is chosen such that $\gcd(\phi(n), e) = 1$. In this example, $e = 7$ ($\gcd(160, 7) = 1$).
- Private Exponent (d) Calculation: The private key exponent (d) is determined such that $d \equiv e^{-1} \pmod{\phi(n)}$. Since $23 \times 7 = 161 \equiv 1 \pmod{160}$, the private exponent is $d = 23$.

Resulting Keys:

- Public Key: $PU = \{7, 187\}$
- Private Key: $PR = \{23, 187\}$

Step 2: Encryption Process

Suppose the plaintext message to be sent is $M = 88$ ($M < 187$ condition is satisfied).

The encryption is performed using the formula:

$$C = M^e \pmod{n}$$

$$C = 88^7 \pmod{187}$$

To handle large exponents, modular arithmetic properties are applied:

$$88^1 \equiv 88 \pmod{187}$$

$$88^2 \pmod{187} = 7744 \equiv 77 \pmod{187}$$

$$88^4 = 88^2 \times 88^2 \equiv 77^2 \equiv 5929 \equiv 132 \pmod{187}$$

Then combining the results:

$$88^7 \equiv (88^4 \times 88^2 \times 88^1) \equiv (132 \times 77 \times 88) \equiv 11 \pmod{187}$$

The ciphertext $C = 11$ is then sent to the receiver.

Step 3: Decryption Process

The receiver uses their private key ($d = 23$) to perform:

$$M = C^d \pmod{n}$$

$$M = 11^{23} \pmod{187}$$

Breaking down the exponentiation using modular arithmetic:

$$11^1 \equiv 11 \pmod{187}$$

$$11^2 \equiv 121 \pmod{187}$$

$$11^4 \equiv 55 \pmod{187}$$

$$11^8 \equiv 33 \pmod{187}$$

Then combining the necessary powers to get 11^{23}:

$$11^{23} \equiv (11^1 \times 11^2 \times 11^4 \times 11^8 \times 11^8) \equiv (11 \times 121 \times 55 \times 33 \times 33) \equiv 88 \pmod{187}$$

Thus, the original message $M = 88$ is successfully recovered.

4.3 ElGamal Cryptosystem

The ElGamal cryptosystem, proposed in 1985, is an alternative to RSA and is based on the discrete logarithm problem rather than integer factorization. It uses a prime modulus and a primitive root to generate public and private keys.

Encryption produces a pair of values, while decryption relies on modular inverses. The mathematical correctness of the system follows from Fermat's Little Theorem and basic properties of modular arithmetic.

Within the framework of number theory, let p be a prime number and g a primitive root modulo p. Given the values of g, a, and p, the process of finding the integer exponent k in the equation:

$$g^k \equiv a \pmod{p}$$

is called the Discrete Logarithm Problem. In particular, in scenarios where the modulus p is chosen to be very large, such as 2048 bits, computing k with current algorithms is considered technically infeasible.

The system essentially consists of the following three stages:

A. Key Generation Process

A sufficiently large prime number p is selected.

A value $g \in \{1, 2, \ldots, p-1\}$, which is a primitive root of p, is chosen.

A secret integer k (private key) is selected within the range $1 < k < p-1$.

Compute:

$$y \equiv g^k \pmod{p}$$

Public Key: $\{p, g, y\}$

Private Key: k

Encryption Stage

For a message M to be sent ($0 \leq M < p$), a random integer j is generated for each communication session (random j ensures different outputs). The ciphertext consists of two components (C_1, C_2):

$$C_1 \equiv g^j \pmod{p}$$

$$C_2 \equiv M \cdot y^j \pmod{p}$$

C. Decryption Process

The recipient uses their private key k to recover the original message using the mathematical relation:

$$M \equiv C_2 \cdot (C_1^k)^{-1} \pmod{p}$$

Proof of Correctness:

The reason why the decryption produces the correct result can be demonstrated using properties of modular arithmetic and Fermat's Little Theorem:

$$C_2 \cdot (C_1^k)^{-1} \equiv (M \cdot y^j) \cdot (g^j)^{-k} \pmod{p}$$

Since $y \equiv g^k$, substituting gives:

$$M \cdot (g^k)^j \cdot (g^{jk})^{-1} \equiv M \cdot g^{kj} \cdot g^{-kj} \equiv M \pmod{p}$$

This proof demonstrates the consistency of the system and confirms that it operates correctly according to the laws of number theory.

Prime: $p = 19$

Primitive root: $g = 2$ (2 is a primitive root modulo 19)

Private key: $k = 10$

Compute public key:

$$y \equiv 2^{10} \pmod{19}$$

$$2^{10} = 1024 \equiv 17 \pmod{19}$$

Public Key: $\{19, 2, 17\}$
Private Key: $k = 10$

Encryption:
Message: $M = 14$
Random integer: $j = 5$
Compute the ciphertext pair (C_1, C_2):

$$C_1 \equiv 2^5 = 32 \equiv 13 \pmod{19}$$

$$C_2 \equiv 14 \cdot 17^5 \pmod{19}$$

Since $17 \equiv -2 \pmod{19}$, we can simplify:

$$(-2)^5 = -32 \equiv 6 \pmod{19}$$

$$C_2 \equiv 14 \cdot 6 = 84 \equiv 8 \pmod{19}$$

Ciphertext: $(13, 8)$

Decryption:
Compute:

$$C_1^k = 13^{10} \pmod{19}$$

$$13^{10} \equiv 6 \pmod{19}$$

Find the modular inverse of 6 modulo 19, which is 16 (because $6 \cdot 16 = 96 \equiv 1 \pmod{19}$).
Recover the message:

$$M \equiv C_2 \cdot (C_1^k)^{-1} \equiv 8 \cdot 16 = 128 \equiv 14 \pmod{19}$$

Decrypted message: $M = 14$, which matches the original message.

5 SECURITY ANALYSIS AND COMPARISON

5.1 Comparative Analysis

Feature	**RSA**	**ElGamal**
Mathematical Problem	Integer Factorization	Discrete Logarithm Problem
Speed	Fast encryption, slower decryption process	Similar computational load for both processes
Ciphertext Size	Limited by modulus n, similar to plaintext size	Consists of two components (C_1, C_2), doubling the size
Nature of Algorithm	Deterministic (same input produces same output)	Probabilistic (random j ensures different outputs)

Table 2: Comparison of RSA and ElGamal Cryptosystems

5.2 Cryptanalytic Attacks

Analysis of Attacks and Threats Against Cryptographic Systems

The security of both classical and modern encryption methods is evaluated based on their resilience to various attack vectors. Below are the primary attack types and the current standing of cryptographic systems against these threats:

1. Brute-Force Attacks

This method is like trying every possible combination to open a locked safe. Modern encryption keys (like 2048-bit keys) are so long that it would take current computers thousands of years to try every possibility. Therefore, it is considered computationally "impossible" for now.

2. Frequency Analysis and Statistical Approaches

In traditional methods (like the Caesar cipher), messages were cracked by looking at which letters appear most often (for example, 'e' is the most common letter in English).

Modern "block ciphers" scramble data so thoroughly (through processes called diffusion and confusion) that no statistical patterns or clues are left behind for attackers to use.

3. Mathematical Vulnerabilities (The Knapsack Case)

Some encryption systems are built on mathematical problems that are assumed to be difficult to solve. If someone discovers a "shortcut" or a mathematical weakness in that underlying problem, the system fails. A famous example is the Merkle–Hellman Knapsack system; it was abandoned after researchers found a mathematical way to bypass its structure in 1982.

4. The Discrete Logarithm Problem (DLP)

The security of systems like ElGamal is rooted in a specific math problem: finding x in the congruence

$$g^x \equiv y \pmod{p}.$$

Finding the value of x (the private key) is like looking for a needle in a haystack when dealing with massive prime numbers. As long as there is no fast way to solve this discrete logarithm, the system remains secure.

5. The Quantum Threat (Shor's Algorithm)

The biggest future risk comes from quantum computing. An approach called Shor's Algorithm, running on a powerful quantum computer, could solve the math problems behind RSA and ElGamal almost instantly. This is why experts are currently developing quantum-resistant algorithms to prepare for the future.

6 Conclusion

Modern cryptography is deeply rooted in number theory. Concepts such as modular arithmetic, prime numbers, Euler's Totient Function, and primitive roots form the mathematical foundation of secure communication systems. While classical ciphers illustrate the historical development of encryption, modern algorithms like RSA and ElGamal demonstrate how these mathematical ideas are applied in real-world security systems. As a continuation of this work, I aim to gain knowledge on quantum-resistant algorithms against emerging quantum technologies that threaten current encryption systems and to pursue my learning process in this field.

References

Burton, D. M. (2011). *Elementary Number Theory* (7th ed.). McGraw-Hill Education.

Stallings, W. (2003). *Cryptography and Network Security: Principles and Practices* (3rd ed.). Prentice Hall.

Stallings, W. (2017). *Cryptography and Network Security.* Pearson Education.

Rivest, R., Shamir, A., & Adleman, L. (1977). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM.*

Wikipedia contributors. ...RSA Cryptosystem.

Wikipedia contributors. *Caesar Cipher.*

Wikipedia contributors. *Hill Cipher.*