

Modern Cryptography: ElGamal Encryption Algorithm

KH4015CMD Foundations of Computer Science

Submitted by: Hala Ahmed Sayed | 202300277

Submitted to: Marwa Refaie

Contents

[Problem Statement: 3](#_Toc184530524)

[Introduction: 3](#_Toc184530525)

[Elgamal Encryption Algorithm: 4](#_Toc184530526)

[Time Complexity: 7](#_Toc184530527)

[Data storage & processing: 8](#_Toc184530528)

[Application 10](#_Toc184530529)

[Source Code: 12](#_Toc184530530)

[12](#_Toc184530531)

[Test cases: 13](#_Toc184530532)

[GitHub Link: 14](#_Toc184530533)

[Comparison with AES: 14](#_Toc184530534)

[References: 15](#_Toc184530535)

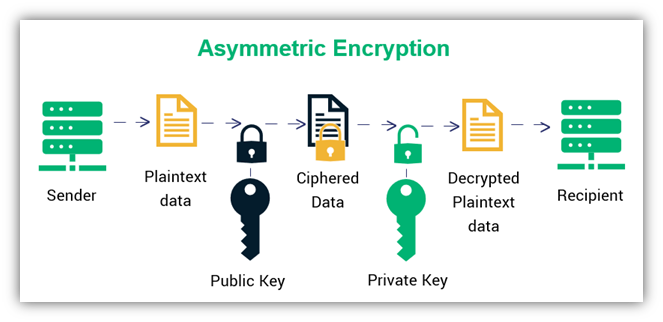
# Problem Statement:

As technology, online communication, and data exchange continue to grow, cyber risks and attacks are escalating drastically, with an average of 2,200 cyberattacks occurring every day: one every 39 seconds (Sanskriti Jain, 2024). According to Terranova Security (2024), organizations faced 1,636 cyberattacks on average per week in 2024, marking a 30% increase from 2023. These rising threats result in significant losses, particularly in sensitive industries such as business, banking, and healthcare. To mitigate these risks, data must be encrypted or concealed, ensuring its security even in the event of a breach. However, the chosen encryption algorithm must be capable of efficiently handling large inputs of data while maintaining security. Consequently, traditional cryptographic methods need to evolve to meet the growing demands of modern cyber threats. Thus, this report will be tackling Elgamal’s encryption algorithm, highlighting its modern usage, security, key steps, time complexity and data storage & processing.

# Introduction:

The study of cryptography is the science underlying communication and data security. It uses a variety of methods, including digital signatures, hashing, encryption, and authentication. To guarantee the security of the data and its CIA (confidentiality, integrity, and availability), encryption is a technique under cryptography that is used to exchange plain text with cipher text utilizing an encryption and decryption key. It has 2 types: symmetrical and asymmetrical encryption. In a symmetric encryption method, the data is encrypted and decrypted using the same key. This implies that the secret key, which must be kept confidential, must be available to both the sender and the recipient. In asymmetric encryption, two different keys are used: one for encryption and a different one for decryption, known as the public key and private key.

# Elgamal Encryption Algorithm:

As Әмірханова & Мамырбаев (2024) stated, Elgamal Encryption Algorithm is an asymmetrical method that’s based on The Discrete Logarithm Problem (DLP). Introduced by Taher Elgamal- an Egyptian cryptographer- in 1985, the algorithm is used for data encryption and digital signatures. It is divided into 3 sections: key generation, encryption, and decryption. It uses modular exponentiation as a core operation to encrypt data, ensuring the data’s confidentiality and integrity. (Note: the length of the text should be fixed and known by the sender and the recipient.)

**Algorithm:**

1. Receiver and sender generate their keys.
2. Receiver sends his/her public key to the sender.
3. The sender encrypts the message using the receiver’s public key.
4. The sender sends the encrypted message.
5. The receiver decrypts the encrypted message using his/her private key.
6. **Key generation: done by receiver**

The public key consists of 3 variables: (P, G, Y), **private key=** **x**

**P**= a large prime number (at least 2048 bits long) for better security

**G**= primitive root: a number that can create all the numbers coprime to n (numbers less than n and not divisible by any factors of n) by raising it to different powers a number, generating all the needed values for encryption and decryption using this rule: G^n mod P.

* G represents the “generator” number, n represents the power value and P represents prime number.
* The result of the root can’t be repeated.
* G should be greater than 1 and smaller than p (1<g<p).
* N should be from 1 to p-1 (1≤n<p−1).

**Y**=G^x mod P

**private key:** a random integer that’s used secretly for decryption; it should be larger than 1 and smaller than p-2 (1<g<p-2).

1. **Encryption: done by sender**
2. Sender receives the receiver’s public key: (P, G, Y)
3. Name the unencrypted text “M” which stands for message; it should be smaller than P
4. Choose a random number B: the sender picks a new random number B (should be larger than 1 and smaller than p-1) each time they send a message, making it more random.
5. Calculate C1=G^B mod P
6. Calculate C2= (M \* Y^B mod P) mod P: Here, M= message
7. Ciphertext/encryption: (C1​, C2​) is sent to the receiver
8. **Decryption: done by receiver**
9. Receiver receives the encrypted message: (C1​, C2​)
10. Calculates Z=(C1) ^x mod P: the receiver uses his/her private key x to compute this value.
11. Recovers the plaintext message: the original message is recovered by computing: M=(C2\*(Z^(P-2) mod P)) mod P.
12. Plaintext: the result is the decrypted message M.

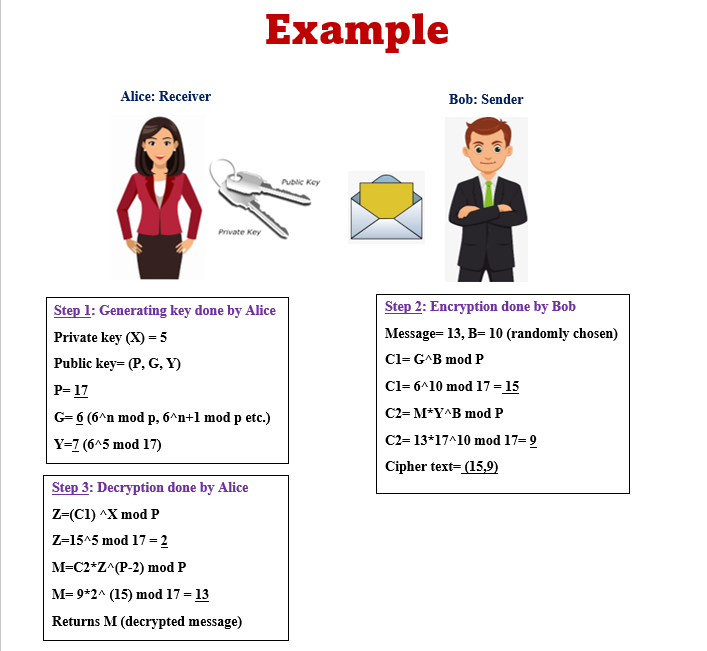


Illustration 1: Example of ElGamal Encryption Algorithm Process

# Time Complexity:

To analyze the algorithm’s time complexity, each step of the algorithm will be analyzed alone then summed up together at the end. Starting with the generating the keys process, it has a complexity of O(logp) as the public key is computed using a modular exponentiation in finding the primitive root. Moving on to the encryption and decryption processes, each one of them has a time complexity of O(logp) also because of the usage of modular exponentiation in the operations: C1, C2, Z, M. This makes the complexity of the whole algorithm O(logp).

**In short:**

All because of the modular exponentiation operations

Generating the keys: O(logp) for the public key generator

Encryption process: O(logp) for the computation of C1 and C2

Decryption process: O(logp) for the computation of Z and M

Final time complexity: O(logp) taking the dominant or worst case

**Why It’s Secure**:

1. As it’s based on The Discrete Logarithm Problem (DLP), it is very hard to compute because of the large prime numbers used, resulting in a very low possibility of cracking the private key which is the main source of decrypting the message.
2. Randomness: the ciphertext for the identical message varies each time it is encrypted since the encryption uses a random element (B) that is selected by the sender. By doing this, attackers are unable to identify any patterns (frequency analysis) in the encrypted messages.

# Data storage & processing:

**Public key:**

From its name, the public key is publicly shared for anyone who wants to send a message to the key’s owner. Although it is shared for everyone, it must be authenticated to avoid any privacy concerns. The public key could be stored on key servers, in X.509 certificates, or publicly on websites. Key serves are servers where users can fetch each other's public keys. X.509 certificates are used to package a public key with its associated information and are signed by a trusted certificate authority to verify their authenticity.

**Private key:**

The private key should be kept a secret, only available for the owner’s use. Therefore, its storage should be very important as the message only gets encrypted with this code. Thus, using large numbers in the calculations is necessary. Another way to secure the private key is to encrypt it and hide it in an image or a text file that only allows the owner of the key to access using passwords or biometrics. It should also have backups, so storing it on a flash drive or smart cards with only the owner's access increases security.

**Real-World Scenario:**

* **Usage examples**: In a secure messaging application (WhatsApp for instance), each end user, sender, and receiver, generate their ElGamal key pair. Then, the public key is shared publicly (through the methods mentioned above), while the private key is kept a secret. When a user receives a message (either a text:str or a number:int), the message is encrypted with their public key. To extract the plain text, the receiver decrypts it using their private key which confirms that only the intended receiver can decrypt and read the message.
* **Challenges**:
  + **Key Management**: Individuals must securely store their private keys. If the private key is misplaced or revealed, it puts the security of the algorithm at risk. To reduce this issue, the utilization of secure storage hardware and key backups is necessary.
  + **Complexity:** ElGamal encryption is inefficient, when dealing with long messages. “Its computational complexity can be quite high, especially when working with large numbers, which can lead to an increase in the time it takes to encrypt and decrypt data” (Әмірханова & Мамырбаев, 2024). As a solution, hybrid encryption could be used beside ElGamal algorithm for key exchange. Hybrid encryption combines fast, convenient public asymmetric encryption with effective private symmetric encryption to encode and decode data.

**Possible attacks:**

1. Man-in-the-Middle Attacks (MITM): The communication between the sender and the recipient is intercepted by an attacker, who can change the encryption or decryption procedure if they are able to intercept the public keys, which are typically exchanged.

2. Key Recovery Attacks: If the private’s key value is small or the implementation is weak, attackers might try to break into the calculation (DLP).

3. Chosen Ciphertext Attack (CCA): By acquiring a decryption oracle—a machine that can decode ciphertexts while the attacker does not have the private key—the attacker attempts to decrypt any ciphertext needed.

Application:

This application is a small-scale implementation of El Gamal Encryption Algorithm. As mentioned previously, this algorithm is all based on calculations and numbers. Consequently, the application starts by taking an integer message and a prime number as an input from the user. Then the integer message gets checked for specific requirements: 0<=m<prime number, also handling exceptions like invalid inputs (text: str, bool: True/False). Moving on to the prime number, the application will check if the number is prime or not. If the number is not a prime number, the user will be asked to input the number again and again till it’s prime. Then, the first step of the algorithm is generating its key. The generating key function will return the public and private key. Then the encryption function will take place and it returns the encrypted message in a tuple of 2 values:(C1, C2). Finally, the decryption function will take the encrypted message and returns it to its original form. Clarifying that the process is done, the application will print an ending statement.

Algorithm:

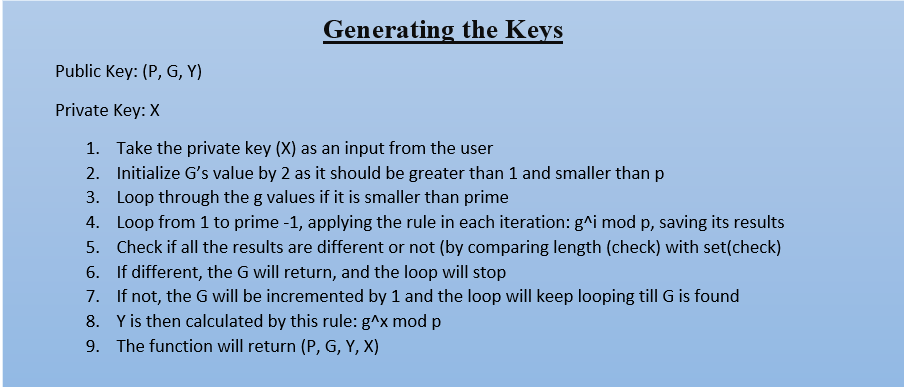
1. Take an input from the user: integer message and the prime number
2. Check if the number is prime or not

* If the number is prime, the application will continue.
* If the number is not prime, the application will keep asking the user to re-input the number till it’s prime.

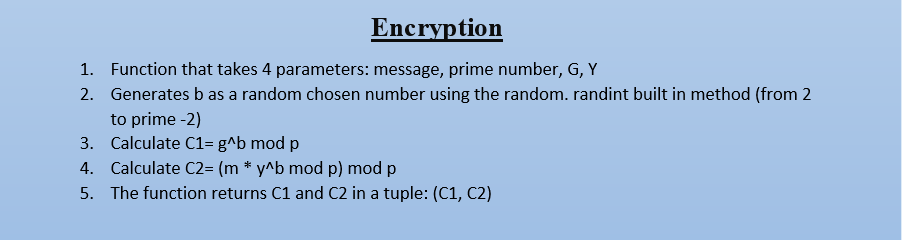
1. Ensure the integer message’s limitations

* Checks if the message is greater than the prime number: if yes, the user will be asked to re-input it
* It will raise an error if anything except a number is placed in the input

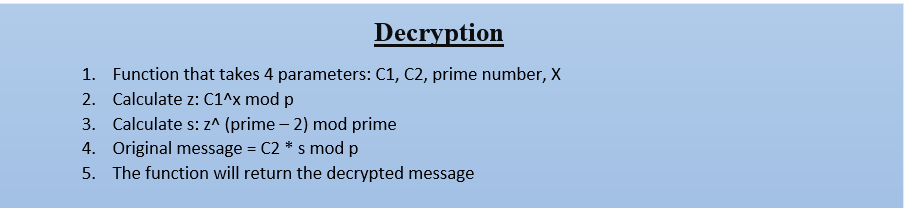
1. Generate the keys
2. Encrypt the message
3. Decrypt the message
4. End the process



Illustrations 2: explaining the algorithm(pseudocode) of generating the keys section



Illustrations 3: explaining the algorithm(pseudocode) of the encryption section

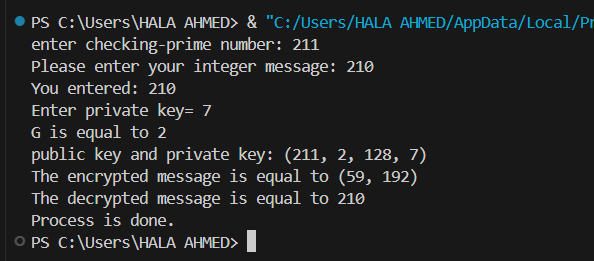
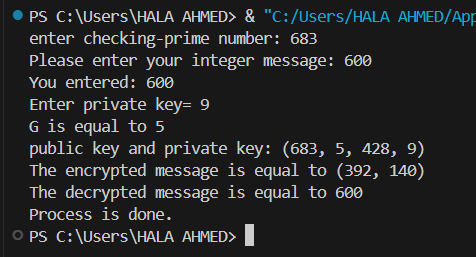


Illustrations 4: explaining the algorithm(pseudocode) of the decryption section

# Source Code:

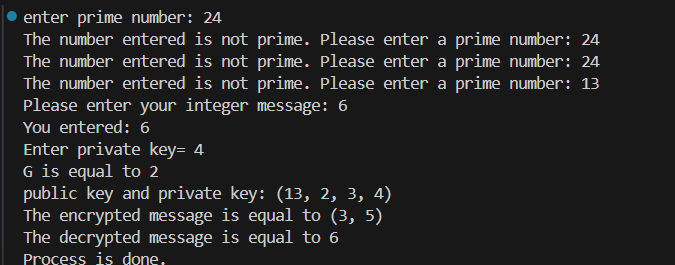
# 

# Test cases:

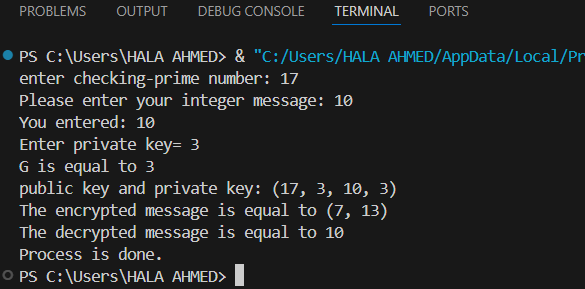


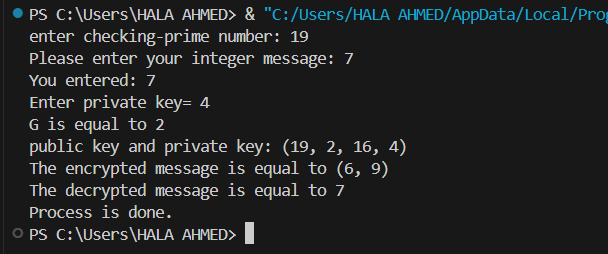
Illustrations 5,6: testing the application with slightly big numbers.

A screenshot of a computer

Description automatically generated

Illustrations 7,8: testing the handling exceptions functions in the application





Illustrations 8,9: testing the application with small numbers.

# GitHub Link:

<https://github.com/hala-ahmedd/CS-CW/commits/main/>

# Comparison with AES:

According to the NIST cryptographic workshop (June,2017), AES and El Gamal are both encrypting algorithm that serves for providing security of the given data by encrypting it. However, there are differences between both algorithms. As mentioned previously, El Gamal algorithm is an asymmetric encryption (2 keys) method that’s based on the Discrete Logarithmic Problem. On the other hand, AES is a symmetric (one key) encryption method that’s based on substitution-permutation network (SPN): the usage of substitution (replacing each byte of data with another), permutation (rearranging data), and mixing (combining data) to securely encrypt the data. It also encrypts data in standard blocks like 128,192,256 unlike El Gamal algorithm that relies on the size of the prime number. As for complexity and efficiency, AES is faster and more efficient with handling large number than El Gamal algorithm (because of the mods). However, AES has a vulnerability which is the security of the shared key. To conclude, both encryption algorithms serve the same goal; consequently, their choice of usage depends on the application’s needs and inputs

# References:

1. 130 cyber security statistics: 2024 trends and Data. 130 Cybersecurity Statistics: 2024 Trends and Data. (n.d.). <https://www.terranovasecurity.com/blog/cyber-security-statistics>
2. El-Gamal’s cryptographic algorithm: Mathematical Foundations, applications and analysis. Известия НАН РК. Серия физико-математическая. (2024, September 29). <https://journals.nauka-nanrk.kz/physics-mathematics/article/view/6290>
3. Jain, S., & Sanskriti Jain                              Sanskriti is a technical writer at Astra who believes in writing with purpose and for a purpose. When she is not busy exploring the world of cybersecurity. (2024, November 15). 160 cybersecurity statistics: Updated report 2024. Astra Security. <https://www.getastra.com/blog/security-audit/cyber-security-statistics/>
4. (PDF) the NIST cryptographic workshop on hash functions. (n.d.-a). https://www.researchgate.net/publication/3437772\_The\_NIST\_Cryptographic\_Workshop\_on\_Hash\_Functions
5. YouTube. (n.d.). YouTube. <https://www.youtube.com/watch?v=oQqr8d5s3Uk>
6. What is hybrid encryption? - definition from Techopedia. (n.d.). <https://www.techopedia.com/definition/1779/hybrid-encryption>