# ASSIGNMENT – 3

## CYBER SECURITY AND ETHICAL HACKING

**NAME : Periyakkal.A**

**REG NO : 20BCI0189**

## Topic : Cryptography Analysis and Implementation

### Objective:

The objective of this assignment is to analyze cryptographic algorithms and implement them in a practical scenario.

### Instructions:

Research: Begin by conducting research on different cryptographic algorithms such as symmetric key algorithms (e.g., AES, DES), asymmetric key algorithms (e.g., RSA, Elliptic Curve Cryptography), and hash functions (e.g., MD5, SHA-256). Understand their properties, strengths, weaknesses, and common use cases.

### Analysis:

Choose three cryptographic algorithms (one symmetric, one asymmetric, and one hash function) and write a detailed analysis of each. Include the following points in your analysis:

Briefly explain how the algorithm works.

Discuss the key strengths and advantages of the algorithm.

Identify any known vulnerabilities or weaknesses.

Provide real-world examples of where the algorithm is commonly used.

## Implementation:

Select one of the cryptographic algorithms you analyzed and implement it in a practical scenario. You can choose any suitable programming language for the implementation. Clearly define the scenario or problem you aim to solve using cryptography.Provide step-by-step instructions on how you implemented the chosen algorithm. Include code snippets and explanations to demonstrate the implementation. Test the implementation and discuss the results.

## Security Analysis:

Perform a security analysis of your implementation, considering potential attack vectors and countermeasures. Identify potential threats or vulnerabilities that could be exploited. Propose countermeasures or best practices to enhance the security of your implementation. Discuss any limitations or trade-offs you encountered during the implementation process.

## Conclusion:

Summarize your findings and provide insights into the importance of cryptography in cybersecurity and ethical hacking.

## Submission Guidelines:

Prepare a well-structured report that includes the analysis, implementation steps, code snippets, and security analysis. Use clear and concise language, providing explanations where necessary. Include any references or sources used for research and analysis. Compile all the required files (report, code snippets, etc.) into a single zip file for submission.

Analysis :

## Symmetric Algorithm: AES

The Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm that provides secure and efficient data encryption and decryption. It was adopted by the U.S. government as a standard for encrypting sensitive information and has since become a global encryption standard. AES operates on fixed-size blocks of data and uses a secret key to perform the encryption and decryption processes. It offers a high level of security, with three key lengths available: 128, 192, and 256 bits. AES has demonstrated resistance against various cryptographic attacks and is used in various applications, including secure communications, financial transactions, and data protection.

## How AES encryption works:

AES works by performing a series of mathematical transformations on fixed-size blocks of data. The algorithm operates on 128-bit blocks and uses a secret key of either 128, 192, or 256 bits. The encryption and decryption processes are similar, with slight differences.

## Encryption:

Key Expansion: The secret key is expanded and transformed into a set of round keys, which are used in subsequent rounds of encryption.

Initial Round: The input data (plaintext) is XORed with the first round key.

Rounds: AES consists of multiple rounds (10, 12, or 14 rounds depending on the key length). Each round applies four operations to the data: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

SubBytes: Each byte in the data block is substituted with a corresponding byte from a substitution table (S-box).

ShiftRows: The bytes in each row of the data block are cyclically shifted.

MixColumns: Each column of the data block is transformed using a matrix multiplication.

AddRoundKey: The data block is XORed with the current round key.

Final Round: The final round skips the MixColumns operation.

Output: The resulting encrypted data (ciphertext) is obtained.

## Decryption:

Decryption in AES is the reverse process of encryption, with the operations applied in reverse order.

Key Expansion: The secret key is expanded to obtain the round keys.

Initial Round: The ciphertext is XORed with the last round key.

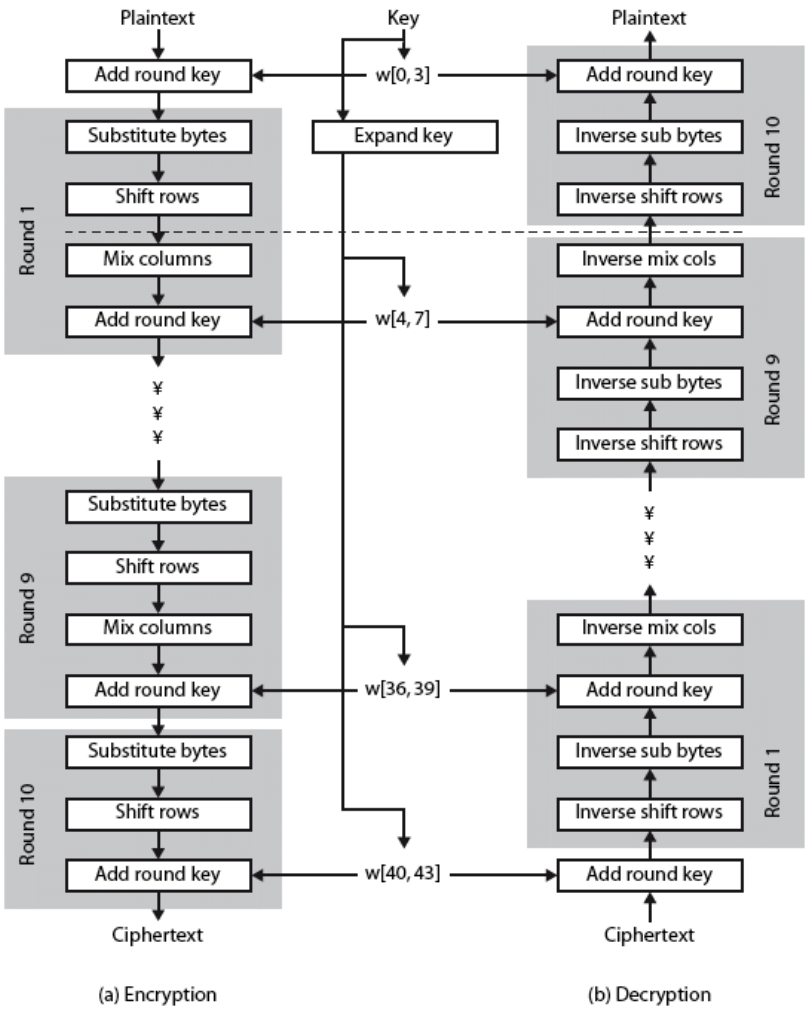
Rounds: Similar to encryption, but the inverse of each operation is applied: InvSubBytes, InvShiftRows, InvMixColumns, and AddRoundKey.

Final Round: The final round skips the InvMixColumns operation.

Output: The resulting decrypted data (plaintext) is obtained.

The strength of AES lies in the complexity of its mathematical operations and the number of rounds, making it resistant to various cryptographic attacks.

## Structure :



## Strengths of AES :

AES data encryption is a more mathematically efficient and elegant cryptographic algorithm, but its main strength rests in the option for various key lengths. AES allows you to choose a 128-bit, 192-bit or 256-bit key, making it exponentially stronger than the 56-bit key of DES.

## Advantages of AES :

- Security: AES provides a high level of security against various cryptographic attacks. It has withstood extensive scrutiny from the cryptographic community and has not been compromised since its introduction. The algorithm's mathematical operations and key scheduling contribute to its resistance against known attacks.

- Efficiency: AES is designed to be computationally efficient, allowing for fast encryption and decryption processes. It is optimized for modern computer architectures and can be implemented in hardware, software, and even embedded systems with limited resources.

- Versatility: AES supports key lengths of 128, 192, and 256 bits, offering flexibility in choosing the level of security required for different applications. It can handle a wide range of data sizes and is suitable for various platforms and environments.

- Standardization: AES has been adopted as a global encryption standard by governments, organizations, and industries worldwide. Its standardization ensures interoperability and compatibility across different systems and implementations.

- Transparent and Open: AES was selected through an open and transparent competition, with submissions from leading cryptographers worldwide. The algorithm's design and specification are publicly available, allowing for independent analysis and verification of its security properties.

- Wide Usage: AES is extensively used in numerous applications, including secure communication protocols (e.g., SSL/TLS), disk encryption, wireless networks, VPNs, and financial systems. Its widespread usage and acceptance in various domains validate its reliability and trustworthiness.

## Drawbacks of AES :

One limitation is the challenge of key management, as securely distributing and managing the encryption keys can be complex, especially in large-scale systems or distributed environments. Another drawback is the symmetric nature of AES, which requires the same key for encryption and decryption, making key distribution among multiple parties a potential challenge. AES also lacks built-in authentication and integrity checking mechanisms, necessitating the use of additional measures like digital signatures or message authentication codes to ensure data integrity and verify authenticity. Additionally, AES can be computationally intensive, especially with larger key sizes, which can impact performance in resource-constrained devices. There is also the potential vulnerability of AES to side-channel attacks, which exploit information leakage through unintended channels.

## Weakness:

The biggest problem with AES symmetric key encryption is that you need to have a way to get the key to the party with whom you are sharing data. Symmetric encryption keys are often encrypted with an asymmetric algorithm like RSA and sent separately.

Examples where AES technology is used: VPN Implementations. File transfer protocols ( FTPS, HTTPS, SFTP, OFTP, AS2, WebDAVS ) Wi-Fi security protocols (WPA-PSK, WPA2-PSK)

## Real time example :

Its usage in securing network communications, particularly in secure web browsing through the HTTPS protocol.

When you access a website that uses HTTPS, your web browser establishes a secure connection with the server using TLS (Transport Layer Security) or SSL (Secure Sockets Layer) protocols. AES is commonly used as the symmetric encryption algorithm within these protocols to encrypt the data transmitted between your browser and the web server.

Here's how AES is used in securing web browsing:

- Handshake: The browser and the server initiate a TLS handshake, where they negotiate the encryption parameters, including the choice of

symmetric encryption algorithm. AES is often one of the options offered during this negotiation.

- Key Exchange: The browser and server securely exchange a symmetric encryption key, known as the session key, using asymmetric encryption algorithms like RSA or Diffie-Hellman. This key will be used for encrypting and decrypting the data transmitted during the session.
- Data Encryption: Once the session key is established, AES is used to encrypt the actual data exchanged between the browser and server. AES operates in a mode like CBC (Cipher Block Chaining) or GCM (Galois/Counter Mode) to provide confidentiality, ensuring that the data is securely encrypted.
- Data Integrity: AES can also be used in conjunction with a message authentication code (MAC) algorithm to ensure data integrity. The MAC generates a unique code based on the encrypted data, which is then verified by the recipient to ensure that the data has not been tampered with during transmission.

By utilizing AES in the HTTPS protocol, web browsing sessions are protected from eavesdropping, data interception, and tampering. AES provides strong encryption, making it computationally infeasible for an attacker to decrypt the transmitted data without the session key.

It is important to note that AES is just one component of the overall security provided by HTTPS. The protocol also involves certificate validation, secure key exchange, and other cryptographic mechanisms to ensure a secure and trustworthy browsing experience.

## Asymmetric Algorithm - EIGamal Encryption :

Asymmetric encryption refers to a cryptographic system that uses a pair of mathematically related keys for encryption and decryption. One well-known example of asymmetric encryption is the ElGamal encryption scheme. ElGamal encryption was developed by Taher ElGamal in the 1980s and is based on the computational difficulty of solving certain mathematical problems, such as the discrete logarithm problem. It provides a secure method for encrypting data and is widely used in various cryptographic applications.

ElGamal cryptosystem can be defined as the cryptography algorithm that uses the public and private key concepts to secure communication between two systems. It can be considered the asymmetric algorithm where the encryption and decryption happen by using public and private keys. In order to encrypt the

message, the public key is used by the client, while the message could be decrypted using the private key on the server end. This is considered an efficient algorithm to perform encryption and decryption as the keys are extremely tough to predict. The sole purpose of introducing the message transaction's signature is to protect it against MITM, which this algorithm could very effectively achieve.

## Working :

The ElGamal encryption scheme operates based on the principles of modular exponentiation and the computational difficulty of the discrete logarithm problem. The encryption process involves the following steps:

## Key Generation:

The recipient generates a key pair consisting of a private key and a corresponding public key. The private key is kept secret, while the public key is shared openly. The keys are typically generated using large prime numbers and other mathematical parameters.

## Encryption:

The sender wants to encrypt a plaintext message (M) to send to the recipient. The sender selects a random value, known as the ephemeral key (k). The sender computes the ciphertext by performing mathematical operations with the recipient's public key and the chosen ephemeral key. First, the sender computes the ephemeral public key (K) by raising the recipient's public key (Y) to the power of the ephemeral key (k): K = Y^k mod p, where p is the prime number used in key generation. Next, the sender computes the shared secret (S) by raising the recipient's public key (Y) to the power of the ephemeral key (k): S = Y^k mod p.Finally, the sender converts the plaintext message (M) to a numerical representation and combines it with the shared secret (S) to produce the ciphertext.

## Decryption:

The recipient receives the ciphertext (C) and wants to decrypt it to obtain the original plaintext message.The recipient uses their private key (x) to perform the decryption process. The recipient computes the shared secret (S) by raising the received ephemeral public key (K) to the power of their private key (x): S = K^x mod p. Finally, the recipient recovers the plaintext message by dividing the received ciphertext (C) by the shared secret (S).

The security of the ElGamal encryption scheme relies on the computational difficulty of the discrete logarithm problem, which makes it computationally infeasible to derive the private key from the public key. It is important to note that the ElGamal encryption scheme requires appropriate key management, random key selection, and secure implementation to ensure its security. ElGamal encryption is widely used in various cryptographic protocols, such as secure communication, digital signatures, and key exchange, providing a secure method for encrypting data and protecting confidentiality.

**Example :**

Example: Alice chooses pA = 107, αA = 2, dA = 67, and she computes βA = 267 ≡ 94 (mod 107). Her public key is (pA, αA, βA) = (2,67,94), and her private key is dA = 67. sends the encrypted message (28, 9) to Alice. −dA = 9·28 −67 ≡ 9·28106−67 ≡ 9·43 ≡ 66 (mod 107).

The EIGamal Encryption algorithm method's sole concept is to make it nearly impossible to calculate the encryption approach even if certain important information is known to the attacker. It is mainly concerned about the difficulty of leveraging the cyclic group to find the discrete logarithm.

It will be very easy to understand, using a simple example. Suppose that even if the value like g^a and g^b are the values known to the attacker, the attacker will find it extremely difficult to find out the value of g^ab which is nothing but the cracked value.In order to understand the entire scenario, we need to go in a stepwise manner on how the encryption and decryption of messages happen actually. We will be considering the example of two peers who are willing to exchange data in a secure manner by leveraging the ElGamal algorithm. Let's suppose user1 and user2 want to exchange the information secretly in that case, the following procedure will be followed.

**Step 1: Generation of the public and private keys.**

The user1 will try to select a very long or large number x, and meanwhile, he will also choose a cyclic group Fx. From this cyclic group, he will be further choosing another component b and one more element c. The values will be selected in the manner that if passed through a particular function, the outcome will be equivalent to 1. Once the value selection phase is over, a value will be calculated that will be further used to generate the private key. By applying the formula fm=b^c, the value will be calculated. In the current scenario, user1 will select F,

fm = b^c, a, b as their public key, while the values of a will be saved as the private key, which will be further used as the private key.

**Step 2: User2 will encrypt the data using the public key of User1.**

In order to begin the encryption of the message, there are certain values that user2 needs to pick. The user2 will also require to pick one of the values p from the cyclic group. The cyclic group will be the same as it was for the user1. The value should be picked in a manner so that Inc passes with a in the particular function will generate the outcome 1. Know the user2 will generate some other values that will be used to encrypt the message using the public key. The value generates will be Pm=b^p. The other revalue b^c will be equal to b^ap. The outcome of this computation will be multiplied to the other value Z in order to get closer to the encryption method. Eventually, the value will be sent using the outcome of computations on b^p,Z*b^ap.

**Step 3: Decryption of the message at user1 end.**

The user1 will then use the computation of the values picked in the first and second phase to identify the appropriate number, which will be used to decrypt the encrypted message. The User1 will be processing b^ap, and then the outcome will be used to divide the by Z in order to get the decrypted value. The decrypted value is something that is that was encrypted in the second phase. In the above scenario, the user1 has initiated the process by calculating the private and public key, which is the algorithm's soul. The key is further used by user2 in the second step in order to encrypt the method. The message is encrypted so that they value computed in that initial phase could be everaged to decrypt the message. In the third step, it could be witnessed that after diving the entire value with the number that is computed in the third step itself totally decrypts the message making it readable for the end-user. The same approach is followed every when the urge to pass the message securely occurs

**Advantage :**

- Key Exchange: Diffie-Hellman enables secure key exchange over an insecure channel. Two parties can independently generate a shared secret key without ever directly transmitting the key. This eliminates the need for secure key distribution, which can be a challenging task, especially in large-scale systems.
- Forward Secrecy: Diffie-Hellman provides forward secrecy, also known as perfect forward secrecy. Even if an attacker compromises the long-term private key of a participant, they cannot derive past session keys or decrypt previously intercepted communications. This ensures that the

confidentiality of past communications remains intact, even in the presence of future attacks.

- Public/Private Key Pair Generation: Diffie-Hellman utilizes the concept of public and private key pairs. Each party generates its own private key and computes a corresponding public key. The private key remains secret, while the public key can be freely distributed.
- Computational Security: The security of Diffie-Hellman is based on the computational difficulty of solving the discrete logarithm problem. In the case of large prime numbers and appropriate parameters, the problem is believed to be computationally infeasible to solve, even for powerful adversaries. This provides a high level of security against various attacks.
- Wide Applicability: Diffie-Hellman can be used in various cryptographic applications, including secure key exchange, establishing secure communication channels (e.g., SSL/TLS), and generating session keys for symmetric encryption. Its versatility and broad applicability make it a foundational component of many cryptographic protocols and systems.
- Efficiency: The Diffie-Hellman key exchange protocol is computationally efficient compared to other asymmetric encryption algorithms, such as RSA. The computational complexity primarily lies in modular exponentiation operations, which can be efficiently performed using algorithms like exponentiation by squaring. This efficiency enables practical deployment in various computing environments.

**Drawback :**

The main disadvantage of El-Gamal is he need for randomness, and its slower speed(especially for signing). Another potential disadvantage of the El-Gamal algorithm is that the message expansion by a factor of two takes place during encryption.

**Strength** :

forward secrecy, computational security, versatility, and efficiency. Its widespread adoption and usage in secure communication protocols demonstrate its effectiveness and importance in modern cryptography.

**Real time example :**

Its usage in securing network communications, particularly in secure web browsing through the HTTPS protocol.

When you access a website that uses HTTPS, your web browser establishes a secure connection with the server using TLS (Transport Layer Security) or SSL (Secure Sockets Layer) protocols. AES is commonly used as the symmetric encryption algorithm within these protocols to encrypt the data transmitted between your browser and the web server.

Here's how AES is used in securing web browsing:

- Handshake: The browser and the server initiate a TLS handshake, where they negotiate the encryption parameters, including the choice of symmetric encryption algorithm. AES is often one of the options offered during this negotiation.
- Key Exchange: The browser and server securely exchange a symmetric encryption key, known as the session key, using asymmetric encryption algorithms like RSA or Diffie-Hellman. This key will be used for encrypting and decrypting the data transmitted during the session.
- Data Encryption: Once the session key is established, AES is used to encrypt the actual data exchanged between the browser and server. AES operates in a mode like CBC (Cipher Block Chaining) or GCM (Galois/Counter Mode) to provide confidentiality, ensuring that the data is securely encrypted.
- Data Integrity: AES can also be used in conjunction with a message authentication code (MAC) algorithm to ensure data integrity. The MAC generates a unique code based on the encrypted data, which is then verified by the recipient to ensure that the data has not been tampered with during transmission.

By utilizing AES in the HTTPS protocol, web browsing sessions are protected from eavesdropping, data interception, and tampering. AES provides strong encryption, making it computationally infeasible for an attacker to decrypt the transmitted data without the session key.

It is important to note that AES is just one component of the overall security provided by HTTPS. The protocol also involves certificate validation, secure key exchange, and other cryptographic mechanisms to ensure a secure and trustworthy browsing experience

## Hash function: MD5

MD5 is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message of 16 bytes. MD5 algorithm stands for the message-digest algorithm. MD5 was developed as an improvement of MD4, with advanced security purposes. The output of MD5 (Digest size) is always 128 bits. MD5 was developed in 1991 by Ronald Rivest. MD5 (Message Digest Algorithm 5) is a widely used cryptographic hash function. It takes an input (message) of any length and produces a fixed-size 128-bit hash value. The primary purpose of MD5 is to verify the integrity of data by generating a unique hash value that represents the input data.

## Characteristics of MD5 :

- One-Way Function: MD5 is a one-way hash function, meaning it is computationally infeasible to retrieve the original message from its hash value. It is designed to be irreversible, making it suitable for tasks like password storage, digital signatures, and checksum verification.
- Fixed Output Size: MD5 always generates a 128-bit hash value, regardless of the input size. This fixed output size makes it convenient for applications that require a consistent-sized hash value.
- Collision Vulnerabilities: MD5 is considered cryptographically weak due to its vulnerability to collision attacks. A collision occurs when two different inputs produce the same hash value. Over time, researchers have discovered collision vulnerabilities in MD5, making it unsuitable for security-sensitive applications.
- Fast Computation: MD5 is relatively fast and efficient, making it useful in applications where speed is important. However, its computational efficiency comes at the cost of reduced security.

## Working :

The working process of MD5 (Message Digest Algorithm 5) involves the following steps:

- Padding: If the input message length is not a multiple of 512 bits (64 bytes), MD5 pads the message to make it a multiple of 512 bits. The padding includes a 1-bit followed by a series of zeros, and then the length of the original message is appended as a 64-bit representation.
- Initialization: MD5 initializes a 128-bit buffer known as the chaining variables with predetermined values. These variables will be updated throughout the algorithm's execution.

- Message Processing: The padded message is divided into blocks of 512 bits. Each block is processed individually.
- Round Function: MD5 applies a series of four rounds to each 512-bit block. Each round consists of multiple operations, including bitwise logical functions (AND, OR, XOR), modular addition, and bitwise rotations.
- Finalization: After processing all blocks, MD5 produces a 128-bit hash value. This hash value represents a unique fingerprint of the input message.

The specific details of the round functions and operations within MD5 are complex and involve bit manipulation, logical operations, and modular arithmetic. These operations create a non-reversible transformation of the input message, resulting in the fixed-length hash value.

It's important to note that MD5 has been found to have security vulnerabilities, including the possibility of collision attacks. As a result, it is no longer considered secure for cryptographic purposes and is not recommended for applications requiring strong data integrity or security. It is advised to use more secure hash functions, such as SHA-256 or SHA-3, for cryptographic applications.

## Strengths of MD5:

MD5 (Message Digest Method 5) is a cryptographic hash algorithm used to generate a 128-bit digest from a string of any length. It represents the digests as 32 digit hexadecimal numbers. Ronald Rivest designed this algorithm in 1991 to provide the means for digital signature verification

## Advantage :

While MD5 (Message Digest Algorithm 5) is no longer considered secure for cryptographic purposes, it still has some advantages in certain non-security-sensitive applications:

- Speed: MD5 is a relatively fast hash function compared to more secure hash algorithms. It can quickly generate hash values for large volumes of data, making it useful in applications where speed is a primary concern.
- Efficiency: MD5 has a fixed output size of 128 bits, regardless of the input size. This fixed-length output makes it convenient for applications that require a consistent-sized hash value.
- Compatibility: MD5 has been widely used and implemented in various systems and applications. As a result, there is a large body of existing code and infrastructure that relies on MD5. In certain legacy systems or non-

security-critical scenarios, MD5 may still be used for compatibility reasons.

- Non-cryptographic Applications: Despite its security vulnerabilities, MD5 can still be useful in non-cryptographic applications. For example, it can be used for checksum verification, data integrity checks, or as a basic hash function for non-security-related tasks.

**Disadvantage :**

MD5 (Message Digest Algorithm 5) has several significant drawbacks, which have rendered it unsuitable for security-sensitive applications:

- Vulnerability to Collision Attacks: MD5 is vulnerable to collision attacks, where two different inputs can produce the same hash value. This weakness allows attackers to create deliberate collisions, which undermines the integrity and security of the hash function. The ability to generate collisions makes MD5 unsuitable for applications that require data integrity guarantees.
- Weakened Security: Over the years, significant vulnerabilities have been discovered in MD5, leading to its classification as cryptographically broken. These vulnerabilities make it susceptible to various attacks, such as pre-image attacks and birthday attacks. As a result, MD5 is no longer considered secure for cryptographic purposes.
- Wide-scale Adoption of More Secure Hash Functions: Due to the known vulnerabilities of MD5, the cryptographic community has widely recommended the use of stronger hash functions, such as SHA-256 (Secure Hash Algorithm 256-bit) or SHA-3 (Secure Hash Algorithm 3rd generation). These hash functions offer improved security and resistance against known attacks, making them more suitable for secure applications.
- Limited Hash Length: MD5 produces a fixed-length output of 128 bits. In modern security contexts, this length is considered relatively short and provides limited entropy, which can make it easier for attackers to perform exhaustive search or brute-force attacks.

## IMPLEMENTATIONS :

### Real time example :

## OPTIMIZED ELGAMAL ENCRYPTION ENHANCE SECURITY

Now a days, everyone using the internet for many reasons like transfer their money, data and etc. So, to improve the security of the current system we implement the cryptosystem which uses the Discrete Logarithm problem(DLP) for encryption that makes the encryption algorithm more secure. That is called Elgamal encryption. It is a public key cryptosystem. It uses asymmetric key for encryption and decryption. Existing elgamal cryptosystem uses only one key for encryption. These days there are such a large number of situations where unapproved clients access to imperative information.

The security of the ElGamal Cryptosystem is based on the success of the Discrete Logarithm Problem (DLP). As long as the DLP remain hard, ElGamal will remain efficient. The security of the DLP is dependent on the size of the Prime modulo and the private keys used by the sender and the receiver. In the case of the Prime Modulo, the longer it is, the DLPbecomes more complicated, and the implementation of the ElGamal Cryptosystem would be more secure. When these keys are kept in secret, the ciphertext is going to be very hard to break, and when there are severalmany private keys, the level of security proportionally increases also.

The most common application of elgamal cryptosystem is Pretty Good Privacy(PGP)which is hybrid cryptosystem that uses both symmetric and asymmetric techniques. Otherapplications are internet voting, hybrid cryptosystem, part of digitalsignature and free GNU privacy guard.

**Features added in it:**

- ➢ Using more than one key to encrypt the message
- ➢ Using a random number of keys to encrypt the message every time
- ➢ Generating new keys after every communication
- ➢ Padding message with random characters to hide length of message
- ➢ Mapping the message to Unicode characters (optimize the file size)

## Existing method :

Existing elgamal encryption consists of three parts. They are key generation, encryption and decryption.

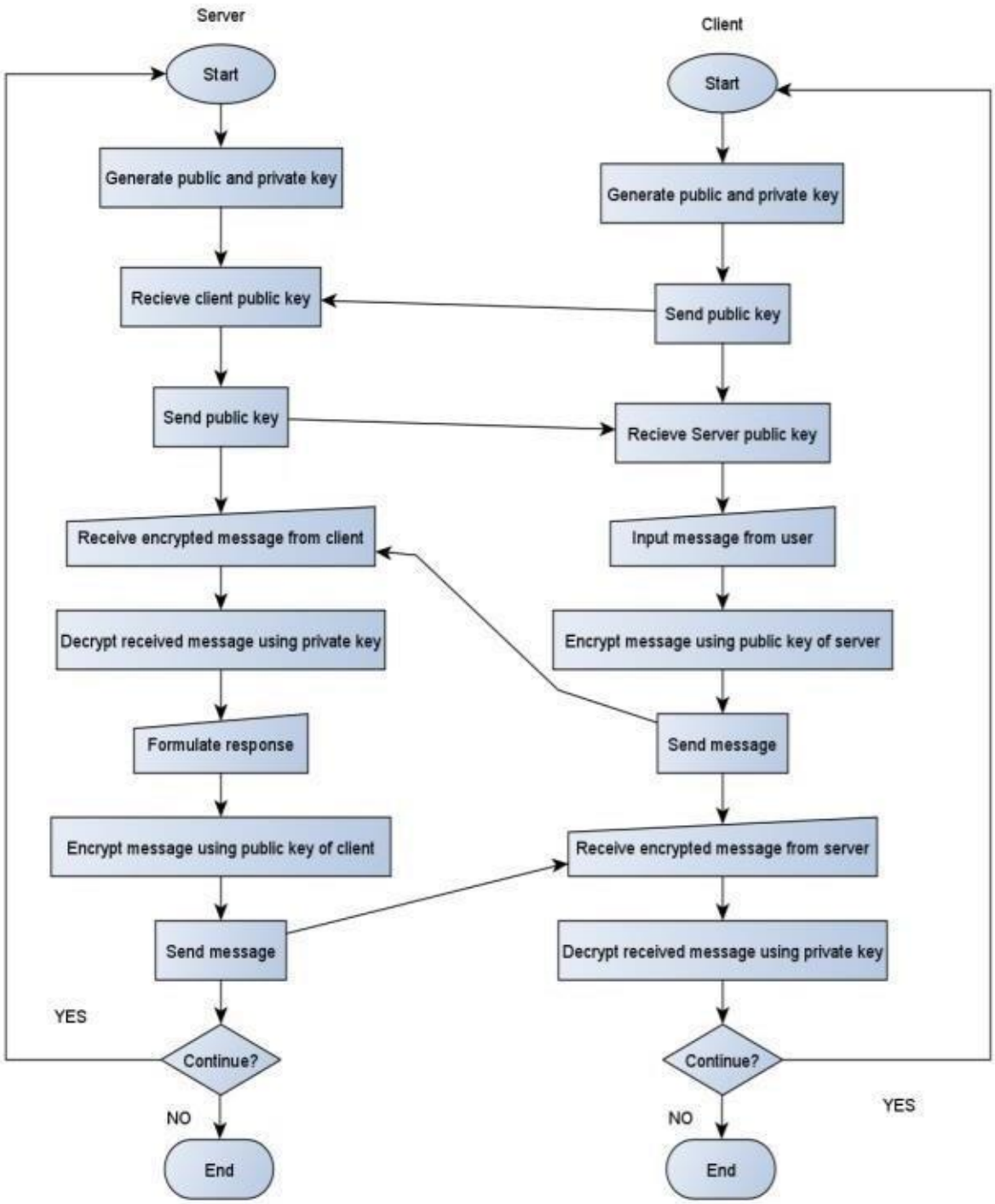
1. Bob generates public and private key:
   - ➢ Bob chooses a very large prime number p.
   - ➢ From the p, he finds g which is the primitive root of p.
   - ➢ Then he computes y = gx mod p.
   - ➢ Bob publishes p, g and y as his public key and retains x as private key.

2. Alice encrypts data using Bob's public key:
   - ➢ Alice selects a random integer k < p.
   - ➢ Then she computes c1 = gk mod p.
   - ➢ She multiples yk with M that is consider as c2.
   - ➢ Then she sends (c1,c2).

3. Bob decrypts the message:
   - ➢ Bob calculates s′ = (c1x)-1.
   - ➢ Then he multiplies c2 by s′ to obtain M.

## PROPOSED SYSTEM :

Existing algorithm uses discrete logarithmic problem it is very hard to crack the key using brute force attack. With modified and dedicated hardware we can crack the key. But our goal is to develop an enhanced Elgamal encryption system which system is not crack easily. Existing encryption method is using integer as a cipher text. But, in our proposed system we use character string as a cipher text so it reduce the file size. In this proposed algorithm it is impossible to break it via brute-force attack. And also Cipher text attack is not possible since attacker has no idea about the keys and length of the message.


Server
Start
Generate public and private key
Recieve client public key
Send public key
Receive encrypted message from client
Decrypt received message using private key
Formulate response
Encrypt message using public key of client
Send message
YES
Continue?
NO
End
Client
Start
Generate public and private key
Send public key
Recieve Server public key
Input message from user
Encrypt message using public key of server
Send message
Receive encrypted message from server
Decrypt received message using private key
Continue?
YES
NO
End

## Module wise description:

Figure shows the how our proposed system works. Herewe explain it detailed.

### 1. Key generation (Server):

➢ Choose a random prime number (p) and choose a random primitive root(g) of the prime number.

➢ Choose a random integer (n) as the number of keys (rounds) in our private key

➢ Generate n random integers (xn) which will act as our private key,applyyn = gx mod p to each of the integers to generate Y (list of integers).

➢ Send p, g, Y to client which acts as our public key.

### 2. Encryption (Client):

➢ Choose n length(Y) random numbers (kn)

➢ Compute Yk mod p using all numbers in Y and a and multiply them and store in one variable c and then compute c=c mod p.

➢ Compute a list A = gk mod p using all integers in a kn

➢ Pad the message with c random characters in the beginning and c/2 in the end.

➢ Encrypt message by multiplying c with message and then

➢ Convert them to characters resulting in encrypted message B

➢ Send A,B to Server

### 3. Decryption (Server):

➢ Receive A,B from client

➢ Compute c by applying Ax mod p on all the integers in A corresponds to x then multiply them together and mod them with p.

➢ Begin processing message from position c as the characters before it are just padding and end the decryption at c/2.

➢ Divide the Unicode value of each character in the message by cand then convert them back to a character.

➢ This is our decrypted message

## Implementation Details and Analysis :

Implementing this using python language

### Elgamal.py

```
import math,random
import string
primel=[]
for i in range(76342,652423):
f=1
for j in range(2,int(math.sqrt(i))+1):
if i%j == 0:
f=0
break
if f:
primel.append(i)
def primitive_root(p):
if p == 2:
return 1
p1 = 2
p2 = (p-1) // p1
while(1):
g = random.randint( 2, (p-1) )
if not (pow( g, (p-1)//p1, p ) == 1):
if not pow( g, (p-1)//p2, p ) == 1:
return g
def genkey():
p=primel[random.randint(0,len(primel)-1)]
```

```
g = primitive_root(p)
n = random.randint(4,9)
b=[]
B=[]
while len(b)!=n:
x = random.randint(2,p-2)
if x not in b:
b.append(x)
for i in range(n):
B.append(pow(g,b[i],p))
return [p,g,B,b]
def encrypt(z,n,p,g,B):
a=[]
while len(a)!=n:
x=random.randint(2,p-2)
if x not in a:
a.append(x)
c=1
A=[]
for i in range(n):
sec = pow(B[i],a[i],p)
A.append( pow(g,a[i],p))
c*=sec
c%=p
if(c==1):
c=5
while(c>255):
```

```
c=c//2
mes=random.choices(string.ascii_letters + string.digits, k = c)
for i in z:
mes.append(i)
mes+=random.choices(string.ascii_letters + string.digits, k = c//2)
for i in range(0,len(mes)):
w=(c*ord(mes[i]))
mes[i]=chr(w)
return [A,".join(mes)]
def decrypt(n,A,b,p,l):
s=[];l2=[]
for i in range(n):
l2.append(pow(A[i],b[i],p))
c=1
#B.append(pow(g,b[i],p))
for i in l2:
c*=i
c%=p
if(c==1):
c=5
while(c>255):
c=c//2
s=""
ll = len(l)-c//2
for i in range(c,ll):
w=(ord(l[i])//c)
s+=chr(w)
```

```
return s
```

**Client.py**

```
import socket
import string
import random
import math as m
import pickle
from elgamal import *
cs = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host = socket.gethostname()
port = 12345
cs.connect((host,port))
while True:
k = genkey()
recv = cs.recv(10000)
serverkey = pickle.loads(recv)
key=pickle.dumps(k[0:3])
cs.send(key)
# Key exchange done
print("\nEnter message:")
mes = input()
x = encrypt(mes,len(serverkey[2]),serverkey[0],serverkey[1],serverkey[2])
encmes=pickle.dumps(x)
cs.send(encmes)
recv = cs.recv(10000)
encmes = pickle.loads(recv)
print("\nEncrypted message: ",encmes[1])
```

```
decmes = decrypt(len(k[3]),encmes[0],k[3],k[0],encmes[1])
print("\nDecrypted message: ",decmes)
```

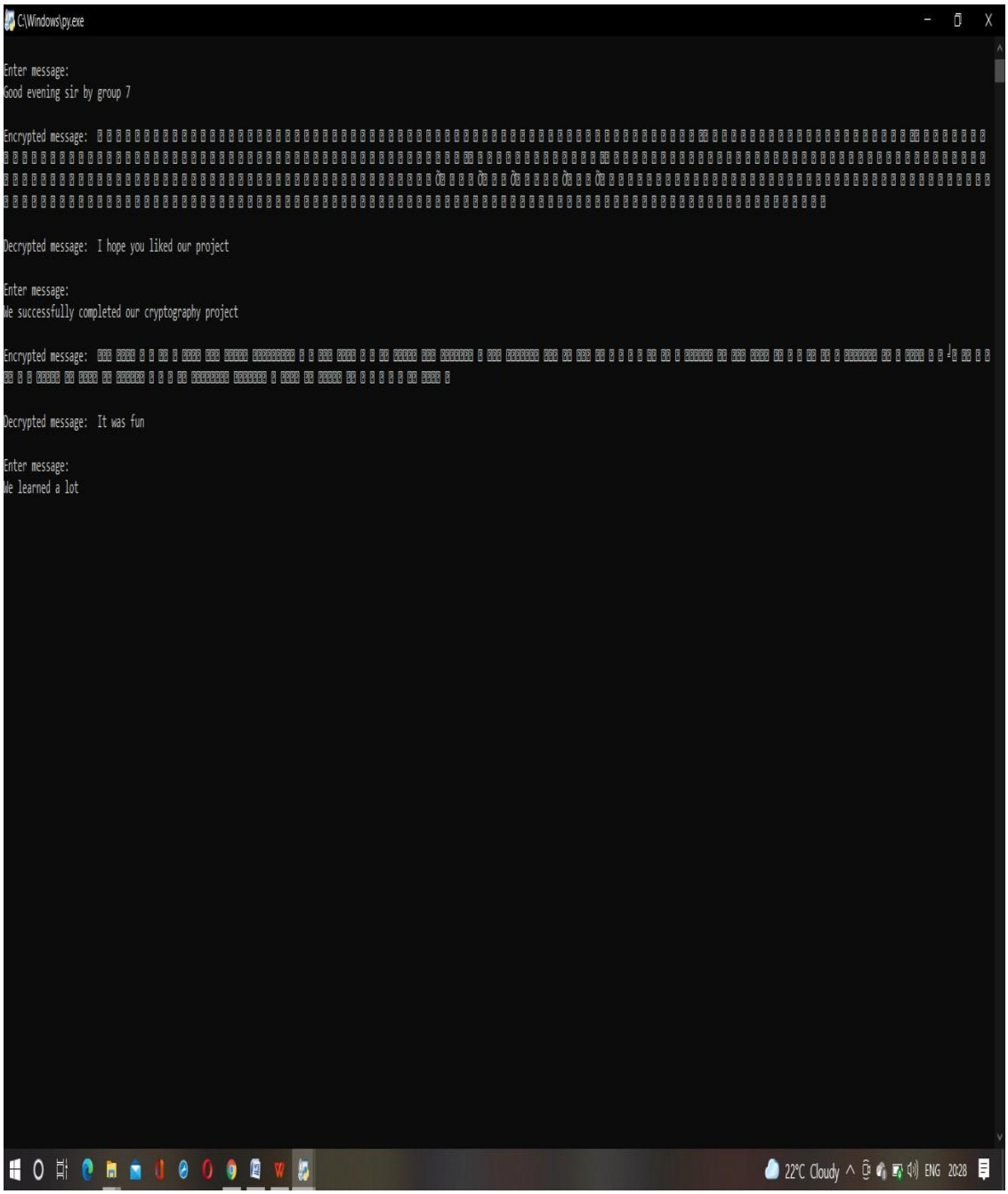
**Server.py**

```
import socket
import string
import random
import math as m
import pickle
from elgamal import *
ss = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host = socket.gethostname()
port = 12345
ss.bind((host,port))
ss.listen(1)
c,addr = ss.accept()
while True:
k = genkey()
key=pickle.dumps(k[0:3])
c.send(key)
recv = c.recv(10000)
clientkey = pickle.loads(recv)
#key exchange done
recv = c.recv(10000)
encmes = pickle.loads(recv)
print("\nEncrypted message: ",encmes[1])
decmes = decrypt(len(k[2]),encmes[0],k[3],k[0],encmes[1])
print("\nDecrypted message: ",decmes)
```

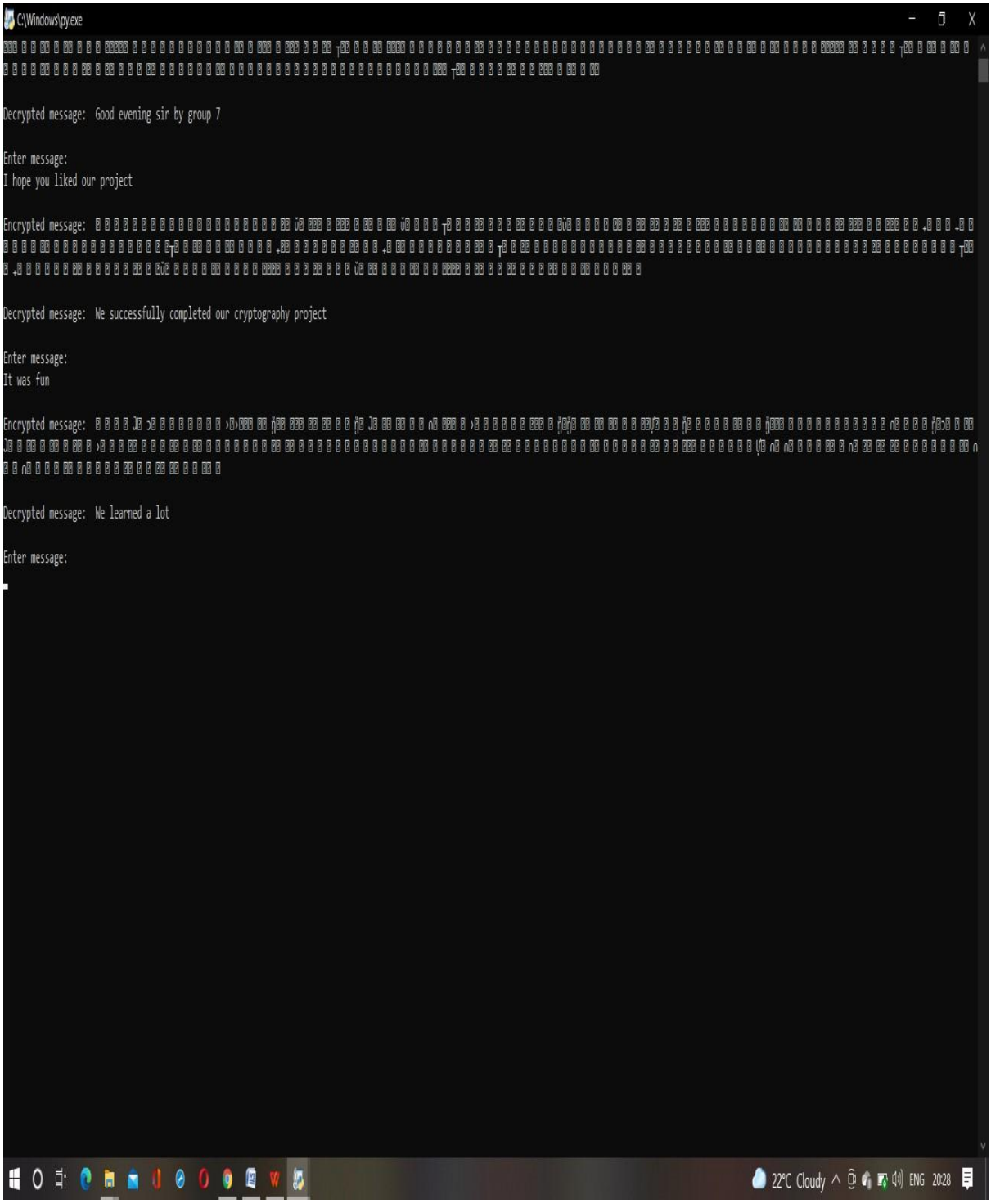
```
print("\nEnter message:")
mes = input()
x = encrypt(mes,len(clientkey[2]),clientkey[0],clientkey[1],clientkey[2])
encmes=pickle.dumps(x)
c.send(encmes)
```

**OUTPUT :**

**Client.py :**



## Server.py :


C:\Windows\py.exe
Decrypted message: Good evening sir by group 7
Enter message:
I hope you liked our project
Decrypted message: We successfully completed our cryptography project
Enter message:
It was fun
Decrypted message: We learned a lot
Enter message:
22°C Cloudy
ENG
2028

**Pros :**

- Multiple rounds of encryption make the algorithm hard to break via brute force attack.
- We randomize the initial prime number picked, the number of integers in the private key (number of rounds) whichmakes it hard for the hacker to crack the system.
- A new key is generated for each of the participants of theconversation after each round of communication, whichprevents hackers from gaining access to future messageseven if they intercept the keys of one communication.
- Since we have modified an existing algorithm in a novelway, hackers who intercept the public key will be confuse das to what algorithm we have used to encrypt our messagessince the structure of our key system is different.
- By padding the message with a random number of randomcharacters during encryption we ensure that the attacker cant guess the length of the original message.
- By padding the message, we also make it useless for the attacker to try and change the message in anymeaningful way as he doesn't know which part of themessage is a pad and which is the message.
- Since we map the characters of the encrypted message back toUnicode, the encrypted message is not displayed in many Englishdevices as the characters aren't supported by the device. This may make it seem like the messages are getting corrupted during transmit but they actually aren't.
- Since this is an asymmetric public-key encryption system the attacker cant decode the message without the private key of the recipient of the message.

**References :**

1. Wu, Jiahui, Xiaofeng Liao, and Bo Yang. "Color image encryption based on chaotic systems and elliptic curve ElGamal scheme." SignalProcessing 141 (2017): 109-124.

2. Minfeng, Fu, and Chen Wei. "Elliptic curve cryptosystem ElGamalencryption and transmission scheme." 2010 International Conference on Computer Application and System Modeling (ICCASM 2010). Vol.6. IEEE, 2010.

3. Siahaan, Andysah Putera Utama. "Comparative analysis ofrsa and elgamal cryptographic public-key algorithms." (2018)