ASSIGNMENT-3

NAME : KALAMEGAM.V

REG NO : 20BIT0302

Course : Cyber Security and Ethical Hacking

Assignment-3: 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 symmetric block cipher chosen by the U.S. government to protect classified information. AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.

How AES encryption works:

AES includes three block ciphers:

1.AES-128 uses a 128-bit key length to encrypt and decrypt a block of messages.

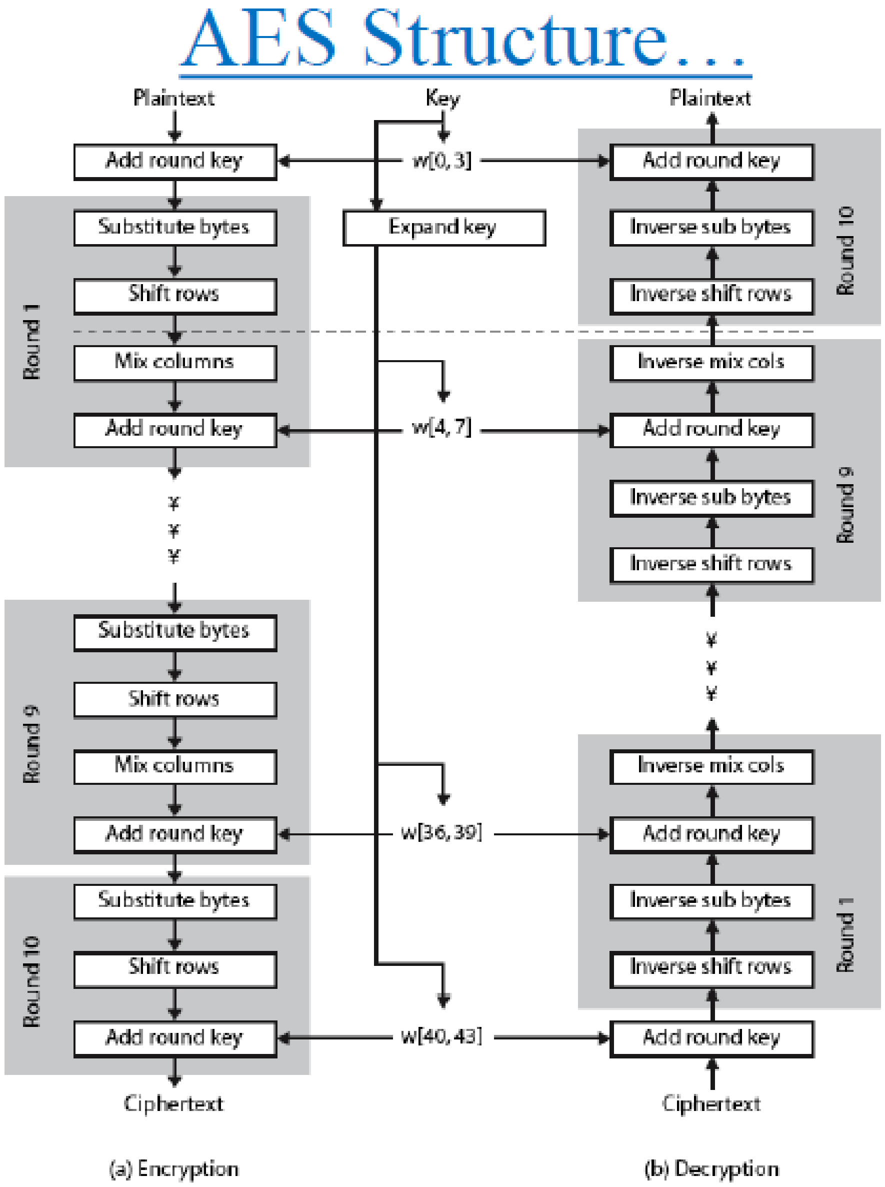
2.AES-192 uses a 192-bit key length to encrypt and decrypt a block of messages.

3.AES-256 uses a 256-bit key length to encrypt and decrypt a block of messages.

Each cipher encrypts and decrypts data in blocks of 128 bits using cryptographic keys of 128, 192 and 256 bits, respectively.Symmetric, also known as secret key, ciphers use the same key for encrypting and decrypting. The sender and the receiver must both know -- and use -- the same secret key.

The government classifies information in three categories: Confidential, Secret or Top Secret. All key lengths can be used to protect the Confidential and Secret level. Top Secret information requires either 192- or 256-bit key lengths.

There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. A round consists of several processing steps that include substitution, transposition and mixing of the input plaintext to transform it into the final output of ciphertext.



The AES encryption algorithm defines numerous transformations that are to be performed on data stored in an array. The first step of the cipher is to put the data into an array, after which the cipher transformations are repeated over multiple encryption rounds.

The first transformation in the AES encryption cipher is substitution of data using a substitution table. The second transformation shifts data rows. The third mixes columns. The last transformation is performed on each column using a different part of the encryption key. Longer keys need more rounds to complete.

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.

Benefits or advantages of AES

- As it is implemented in both hardware and software, it is most robust security protocol.
- It uses higher length key sizes such as 128, 192 and 256 bits for encryption. Hence it makes AES algorithm more robust against hacking.
- It is most common security protocol used for wide variety of applications such as wireless communication, financial transactions, e-business, encrypted data storage etc.
- It is one of the most widely used commercial and open source solutions across the world.
- No one can hack your personal information.
- For 128 bit, about 2128 attempts are needed to break. This makes it very difficult to hack it as a result it is very safe protocol.

Drawbacks or disadvantages of AES

- It uses too simple algebraic structure.
- Every block is always encrypted in the same way.
- Hard to implement with software.
- AES in counter mode is complex to implement in software taking both performance and security into considerations.

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)

Asymmetric Algorithm: EIGamal Encryption

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.

ElGamal Encryption Algorithm with Example

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 leveraged 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 of ElGamal algorithm:

The advantage of the ElGamal algorithm is the generation of keys using discrete logarithms. Encryption and decryption techniques use a large computing process so that the encryption results are twice the size of the original size.

Disadvantage of El-Gamal

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.

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).

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.

How does MD5 work?

MD5 runs entire files through a mathematical hashing algorithm to generate a signature that can be matched with an original file. That way, a received file can be authenticated as matching the original file that was sent, ensuring that the right files get where they need to go.

The MD5 hashing algorithm converts data into a string of 32 characters. For example, the word "frog" always generates this hash: 938c2cc0dcc05f2b68c4287040cfcf71. Similarly, a file of 1.2 GB also generates a hash with the same number of characters. When you send that file to someone, their computer authenticates its hash to ensure it matches the one you sent.

If you change just one bit in a file, no matter how large the file is, the hash output will be completely and irreversibly changed. Nothing less than an exact copy will pass the MD5 test.

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.

Advantages of the MD5 algorithm

- It's easier to compare and store smaller hashes using MD5 Algorithms than it is to store a large variable-length text.
- By using MD5, passwords are stored in 128-bit format.
- You may check for file corruption by comparing the hash values before and after transmission. To prevent data corruption, file integrity tests are valid once the hashes match.

- A message digest can easily be created from an original message using MD5.

Disadvantages of the MD5 algorithm

- When compared to other algorithms like the SHA algorithm, MD5 is comparatively slow.
- It is possible to construct the same hash function for two distinct inputs using MD5.
- MD5 is less secure when compared to the SHA algorithm since MD5 is more vulnerable to collision attacks.

Real-World Example of Hashing: Online Passwords

Every time you attempt to log in to your email account, your email provider hashes the password YOU enter and compares this hash to the hash it has saved. Only when the two hashes match are you authorized to access your email.

IMPLEMENTATIONS:

Aim and abstract:

Security is everyone's top concern in the modern era. Everyone uses the internet these days for a variety of purposes, including data and money transfers. Therefore, we build a cryptosystem that leverages the Discrete Logarithm Problem (DLP) for encryption, which makes the encryption method more safe, in order to increase the security of the current system. It's known as Elgamal encryption. A public key cryptosystem is used. Both the encryption and decoding processes require asymmetric keys. The current Elgamal cryptosystem encrypts data using just one key. These days, there are so many instances where unauthorised clients get access to crucial information. In order to add an additional layer of security to the system, we suggest that the Elgamal Cryptosystem be modified in this work. With this update, the user is able to encrypt the message with numerous private keys. The text will be converted into integer values by an existing algorithm, increasing the file size by 2*n. By translating the int values to the corresponding characters, we were able to reduce the file size.

Key words: Security, cryptosystem, Discrete Logarithm Problem, encryption, Elgamal, private keys.

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 = g^x 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 = g^k mod p.
- She multiples y^k with M that is consider as c2.
- Then she sends (c1,c2).

3. Bob decrypts the message:

- Bob calculates s′= ($c1x^{-}1$)mod p.
- 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.

Module and Description:

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, apply yn = 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 c and then convert them back to a character.
- This is our decrypted message

Implementation Details and Analysis:

We are implementing this using **python** language.

elgamal.py

```
import math,random
import string

primel=[]
for i in range(76432,652423):
    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>225):
        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
    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
```

Bob.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)
```

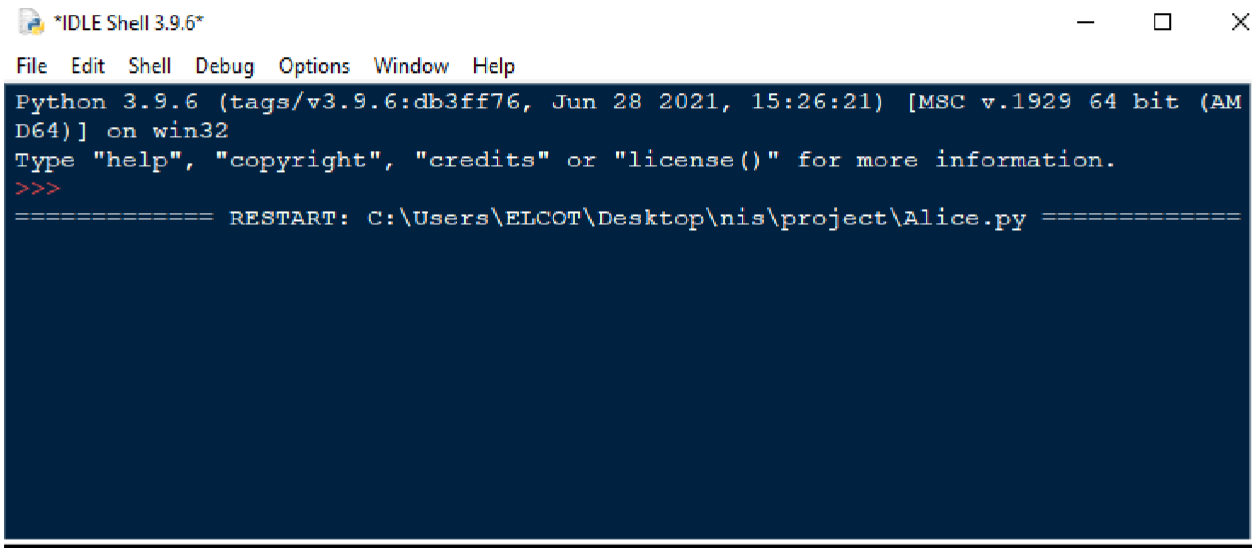
Alice.py:

```
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("\nEncryted 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:

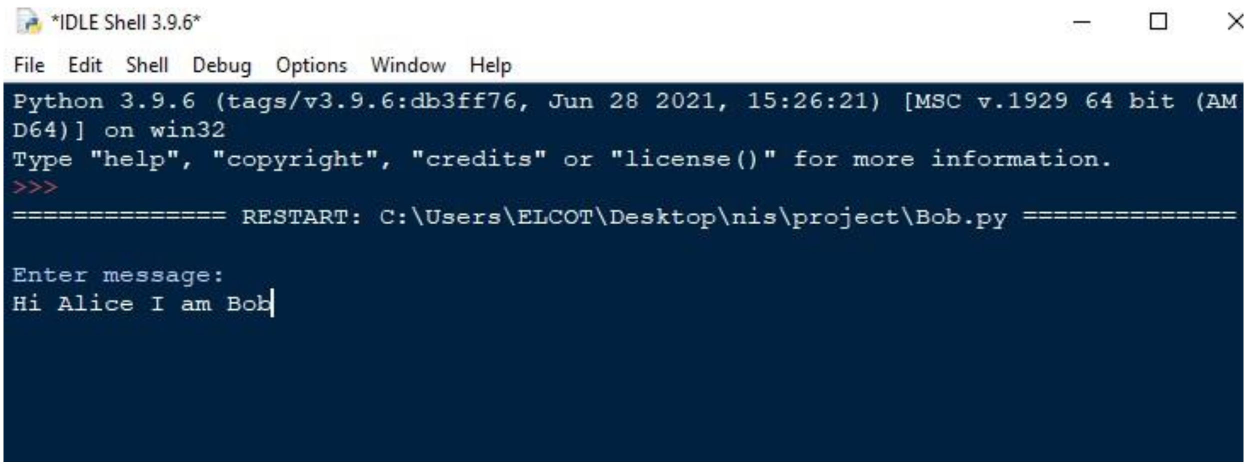
Alice.py:



```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:\Users\ELCOT\Desktop\nis\project\Alice.py ============
```

Bob.py:

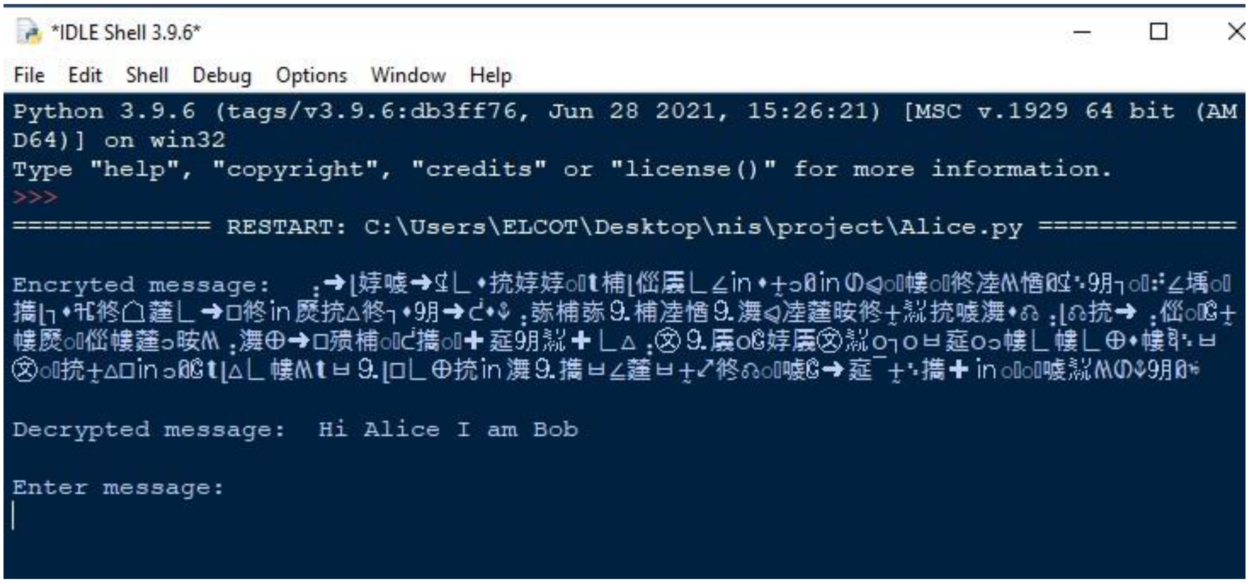
Bob sending message to Alice:



```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\ELCOT\Desktop\nis\project\Bob.py ==============

Enter message:
Hi Alice I am Bob
```

Encrypted & decrypted message received by Alice from Bob:



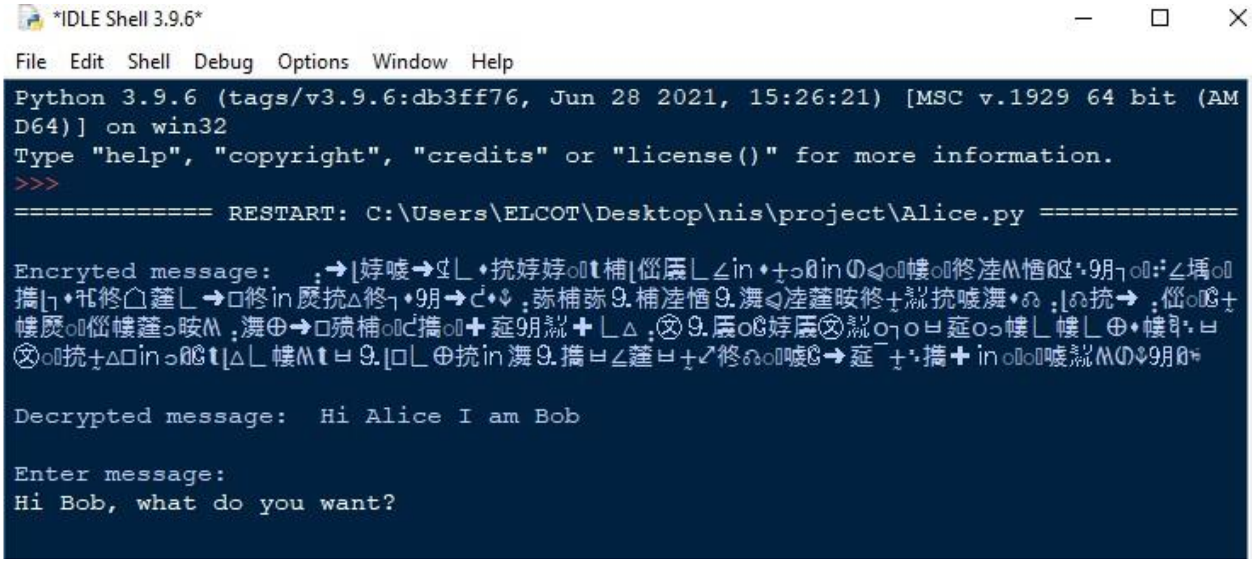
```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:\Users\ELCOT\Desktop\nis\project\Alice.py =============

Encryted message:   ;➔⌊婷唛➔ʤ∟⬩捥婷婷ₒ▯ŧ補⌊偿屡∟∠in⬩ƚɔß in⏀⊲ₒ▯帻ₒ▯佟涬ʍ惜ßʤ⁖9月┐ₒ▯⁝∠墒ₒ▯
攜⌊┐⬩礼佟⌂蓬∟➔□佟in厤捥△佟┐⬩9月➔ḋ⬩⇩ ;菸補菸9.補涬惜9.湃⊲涬蓬晐佟ƚ紭捥唛湃⬩ɷ ;⌊ɷ捥➔ ;偿ₒ▯ß ƚ
帻厤ₒ▯偿帻蓬ɔ晐ʍ ;湃⊕➔□殰補ₒ▯ḋ攜ₒ▯✚ 莚9月紭✚∟△ ;㊫9.屡oß婷屡㊫紭o┐o⊟莚oɔ帻∟帻∟⊕⬩帻ᾑ⁖⊟
㊫ₒ▯捥ƚ△□in ɔßß ŧ⌊△∟帻ʍŧ⊟9.⌊□∟⊕捥in湃9.攜⊟∠蓬⊟ƚ⤢佟ɷₒ▯唛ß➔莚¯ƚ⁖攜✚in ₒ▯ₒ▯唛紭ʍ⏀⇩9月ß⇸

Decrypted message:  Hi Alice I am Bob

Enter message:
|
```

Alice sending reply to Bob:



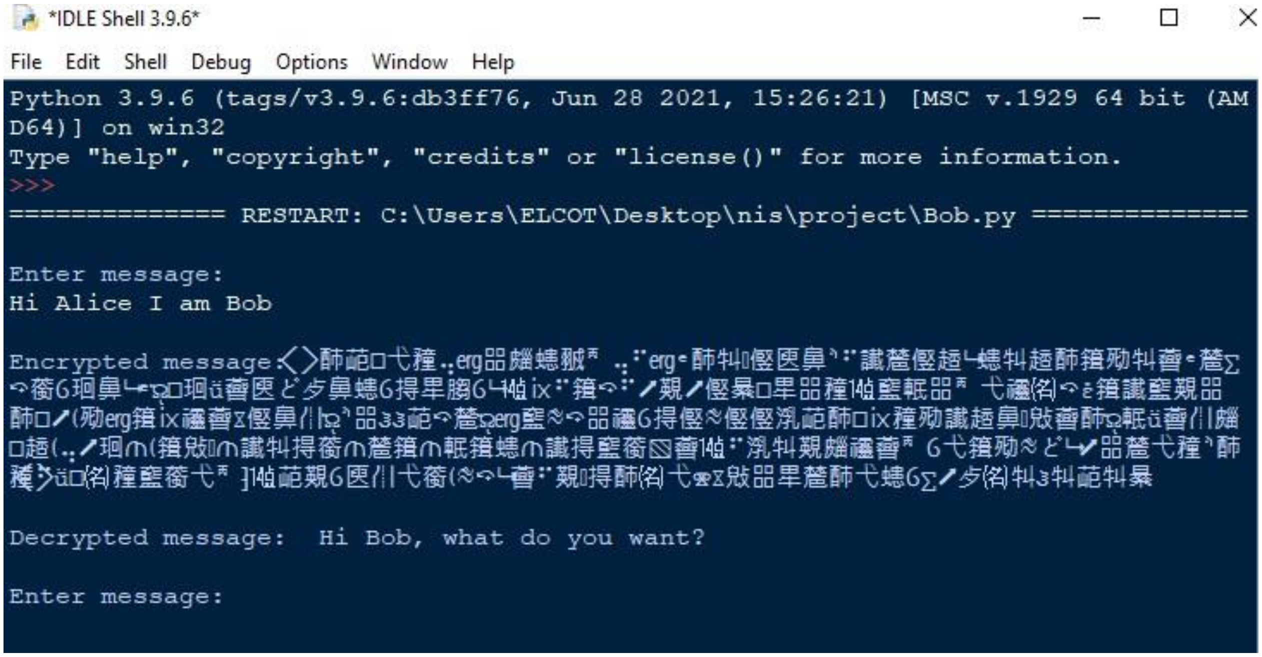
```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:\Users\ELCOT\Desktop\nis\project\Alice.py =============

Encryted message:   ;➔⌊婷唛➔ʤ∟⬩捥婷婷ₒ▯ŧ補⌊偿屡∟∠in⬩ƚɔß in⏀⊲ₒ▯帻ₒ▯佟涬ʍ惜ßʤ⁖9月┐ₒ▯⁝∠墒ₒ▯
攜⌊┐⬩礼佟⌂蓬∟➔□佟in厤捥△佟┐⬩9月➔ḋ⬩⇩ ;菸補菸9.補涬惜9.湃⊲涬蓬晐佟ƚ紭捥唛湃⬩ɷ ;⌊ɷ捥➔ ;偿ₒ▯ß ƚ
帻厤ₒ▯偿帻蓬ɔ晐ʍ ;湃⊕➔□殰補ₒ▯ḋ攜ₒ▯✚ 莚9月紭✚∟△ ;㊫9.屡oß婷屡㊫紭o┐o⊟莚oɔ帻∟帻∟⊕⬩帻ᾑ⁖⊟
㊫ₒ▯捥ƚ△□in ɔßß ŧ⌊△∟帻ʍŧ⊟9.⌊□∟⊕捥in湃9.攜⊟∠蓬⊟ƚ⤢佟ɷₒ▯唛ß➔莚¯ƚ⁖攜✚in ₒ▯ₒ▯唛紭ʍ⏀⇩9月ß⇸

Decrypted message:  Hi Alice I am Bob

Enter message:
Hi Bob, what do you want?
```

Bob receiving reply from the Alice:



```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
============= RESTART: C:\Users\ELCOT\Desktop\nis\project\Bob.py =============

Enter message:
Hi Alice I am Bob

Encrypted message:〈〉酾葩□弋穜⁞erg噐嬺蟪豗〽 ⁞⁚erg﹡酾糾▯慳叜鼻〻⁚讌薼慳趏└蟪糾趏酾鍎殓糾薈﹡薼∑
↶蕎6珋鼻↳ꭥ□珋ü薈叜ど歺鼻蟪6捋畢腵6└恉ix⁚鍎↶⁚⁄覫⁄慳綦□畢噐穜恉壂粻噐〽 弋禵㈴↶ế鍎讌壂覫噐
酾□⁄(殓erg鍎ix禵薈ꞩ慳鼻⺢ꭥ〻噐ɜɜ葩↶薼ꭥerg壂≈↶噐禵6捋慳≈慳慳洴葩酾□ix穜殓讌趏鼻▯敐薈酾ꭥ粻ü薈⺢嬺
□趏(⁞⁄珋ന(鍎敐▯ന讌糾捋蕎ന薼鍎ന粻鍎蟪ന讌捋壂蕎▧薈恉⁚洴糾覫嬺禵薈〽 6弋鍎殓≈ど↳✓噐薼弋穜〻酾
穳⋟ü□㈴穜壂蕎弋〽 ]恉葩覫6叜⺢弋蕎(≈↶└薈⁚覫▯捋酾㈴弋ꝏꞩ敐噐畢薼酾弋蟪6∑⁄歺㈴糾ɜ糾葩糾槑

Decrypted message:  Hi Bob, what do you want?

Enter message:
```

Security Analysis:

(In)security of ElGamal in OpenPGP

IBM cryptographers in Zurich report two new vulnerabilities they discovered in OpenPGP. The vulnerabilities make emails easily decryptable by any mathematically skilled hacker with modest resources.

IBM cryptographers in Zurich report two new vulnerabilities they discovered in OpenPGP. The vulnerabilities make emails easily decryptable by any mathematically skilled hacker with modest resources.

OpenPGP is a popular standard for end-to-end encrypted email, supported by many email applications for both PCs and mobile devices including Outlook, Thunderbird and Apple Mail. While popular, turns out that it is also insecure.

Limitations:

1. Its need for randomness, and its slower speed(especially for signing).
2. The potential disadvantage of the ElGamal system is that message ex- pansion by a factor of two takes place during encryption(means the ciphertext is twice as long as the plaintext.)

Conclusion:

We obtained an encrypted cypher text version of the communication that is difficult for brute force attacks to crack. We safely exchange messages between thesender and receiver using our improved Elgamal technique. In the currently used encryption method, integers are used as cypher text. However, the file size is reduced in our suggested approach by using character strings as cypher text. It is difficult to defeat this proposed method with a brute-force attack. Additionally, since the attacker is unaware of the message's length and encryption keys, a cyphertext attack is not feasible. Therefore, compared to the Elgamal algorithm, it reducesoverall processing times due to these characteristics.