VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

Mini Project Report

On

COMPUTER NETWORK SECURITY (18CS52)

"Hybrid Technique for Data Encryption"

Submitted By

USN	NAME
----------	---------
1BI20CS117	NISHIT KHAMESRA
1B120CS132	PRIYAM
1BI20CS065	GAURAV SARAIWALA
1BI20CS149	SAKET R VAISHNAW

For the academic year 2022-23

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“Jnana Sangama”, Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Computer Network Security(18CS52) Mini Project** entitled **“Hybrid Technique for Data Encryption”** has been successfully completed by **Nishit Khamesra (1BI20CS117), Priyam (1BI20CS132), Gaurav Saraiwala (1BI20CS065), Saket R Vaisnaw (1BI20CS149)** of V semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2022-2023**.

In charge: Prof. Pooja

Prof. Pooja P
Assistant Professor
Dept. of CS&E, BIT

Dr. J.Girija.
Professor and Head
Department of CS&E
Bangalore Institute of Technology

ABSTRACT

Data encryption has been widely applied in many data processing areas. Various encryption algorithms have been developed for processing text documents, images, video, etc. If we are able to collaborate on the advantages of the different existing encryption methods, then a new hybrid encryption method can be developed which offers better security and protection. So, in order to accomplish the Hybrid encryption technique, data encryption techniques using Fibonacci series, XOR logic, PN sequence are studied, analyzed and their performance is compared in this paper. The message is divided into three parts and these three different techniques are applied to these parts and the performance is again analyzed. The application of these three different methods to different parts of the same message along with two keys, namely, segmenting key and encrypting key to provide further authentication and validation is the basis of our paper.

Data security is an important aspect of communication systems that has always been a focus for exchanging information among parties at location physically apart. In today's competitive market, different techniques have been developed to send data securely. We present a hybrid technique which combines the speed of Advanced Encryption Standard (AES) for encryption of data and Rivest Shamir Adleman (RSA) algorithms to encrypt the AES secret key for proper key distribution and management. The proposed technique was implemented in Visual Studio Code using Python and the performance of the hybrid technique was assessed based on encryption and decryption time and throughput for different input text and image data samples of varying sizes. The results obtained show that the developed hybrid technique has better performance in terms of speed of encryption(encryption time), throughput and the central processing unit (CPU) power consumption, when compared to other hybrid techniques in literature.

Hence, the developed hybrid technique is recommended for enhancing data security in modern applications, and systems where a high speed of encryption is required, without compromising the CPU power consumption.

CONTENTS

Chapter 1

INTRODUCTION

1.1 Overview

Hybrid encryption is an approach to encoding and decoding data that blends the speed and convenience of a public asymmetric encryption scheme with the effectiveness of a private symmetric encryption scheme.

In this approach to cryptography, the sender generates a private key, encrypts the key by using a public key algorithm and then encrypts the entire message (including the already-encrypted private key) with the original symmetric key. The encoded cipher can only be decoded if the recipient knows the private key the sender originally generated.

1.1.1 How hybrid encryption works?

Hybrid encryption is achieved through data transfer using unique session keys along with symmetrical encryption. Public key encryption is implemented for random symmetric key encryption.The recipient then uses the public key encryption method to decrypt the symmetric key. Once the symmetric key is recovered, it is then used to decrypt the message.

1.2 Problem statement

“To implement Hybrid Technique for data encryption”

Input: Data to be transferred securely.

At Sender side data is encrypted by using an AES algorithm .

Output: Data reached at destination correctly.

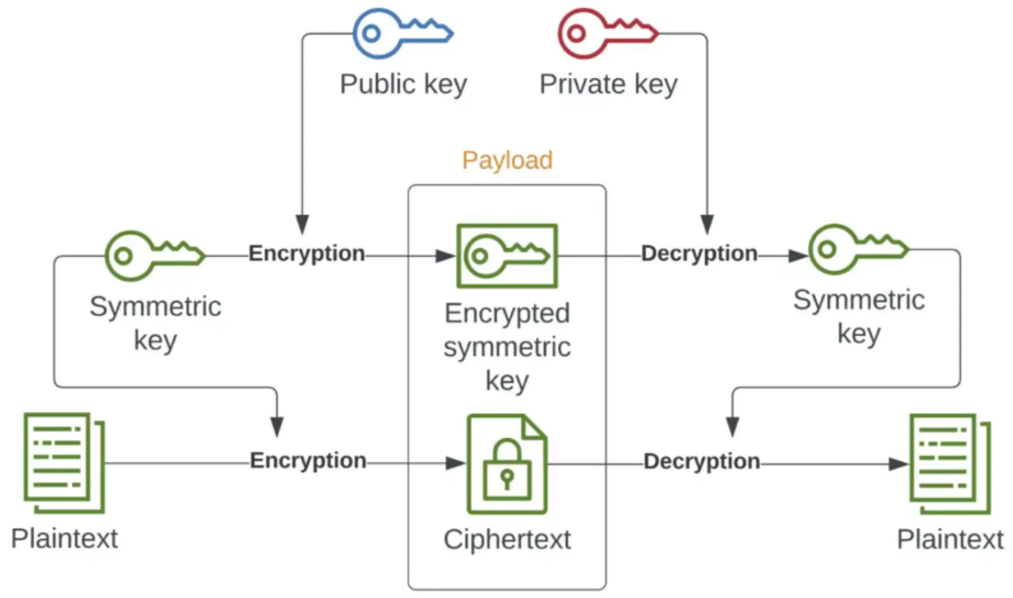
At receivers side decrypted and used

1.3 Objectives

- To collaborate the advantages of different existing encryption methods which offer better security and protection.
- The primary purpose of encryption is to protect confidentiality of digital data stored on computer systems or transmitted over the internet or any other computer network.
- The main aim of cryptography is to maintain the confidentiality, availability and integrity of the information.

Chapter 2

SYSTEM ARCHITECTURE/BLOCK DIAGRAM



Chapter 3

SYSTEM REQUIREMENT SPECIFICATIONS

3.1 Software Requirements

- Windows XP, 7, 8, 8.1, 10, 11
- Python Version 3 and Above
  In Python - AES Module
  get_random_bytes
  PBKDF2
  pad, unpad
  In all Python modules are imported from crypto.* as it is the basic requirement for the hybrid encryption .
- MacOS venture 13 and Above

3.2 Hardware Requirements

- RAM - 1 GB
- Disk Space - 30GB 20% free disk space
- Network Connectivity - Communication with PolicyServer required for managed agents

Chapter 4 DESIGN

4.1 Algorithms/Flowchart

Algorithm:

- Generate public + private keypair and share the private key with the receiver (a one-time thing)
- Generate a symmetric key
- Encrypt our data with the symmetric key
- Encrypt the symmetric key with the public key
- Encode the data and the symmetric key with *base64*
- Send the data along with the encrypted symmetric key
- Receive
- Decode the data and the symmetric key from *base64*
- Decrypt the symmetric key with the private key
- Decrypt data with the symmetric key

Chapter 5

IMPLEMENTATION

5.1 MODULE DESCRIPTION

Python Packages and modules identified and implemented :

- **Pycrypto Package:**

  This is a collection of both secure hash functions (such as SHA256 and RIPEMD 160), and various encryption algorithms (AES, DES, RSA, ElGamal, etc.).
  General:
  Function : open(*file, mode*)
  Use :Opens a file stored internally and returns it as a file object.
  Function : write(*byte*)
  Use : On file object, to insert specific text or byte object.
  Function : read(*size*)
  Use : On file object, to return a specified number of bytes.

- **Cipher Module**

  Secret-key (AES, DES, ARC4) and public-key encryption (RSA PKCS#1) algorithms

- **AES sub-module**

  AES (Advanced Encryption Standard) is a symmetric block cipher standardized by NIST. It has a fixed data block size of 16 bytes. Its keys can be 128, 192, or 256 bits long.
  Function : new(key, *args, **kwargs)
  Use : Create a new AES cipher
  Functions performed on AES cipher object :
  Function : encrypt(self, plaintext)
  Use : Encrypt data with the key and the parameters set at initialization.
  Function : denrypt(self, plaintext)
  Use : Decrypt data with the key and the parameters set at initialization.

- **AES sub-module Keyword arguments :**
  Arguments : iv (*bytes*, *bytearray*, *memoryview*)
  Use : The initialization vector to use for encryption or decryption.
  Arguments : *MODE_CBC*
  Use : Here *Cipher-Block Chaining (CBC)*. Each of the ciphertext blocks depends on the current // and all previous plaintext blocks. An Initialization Vector (*IV*) is required.
  AES sub-module Keyword variables :
  Variable : block_size
  Use : Size of a data block (in bytes)

- **Protocol module**
  Implements various cryptographic protocols (Chafing, all-or-nothing transform, key derivation functions). This package does not contain any network protocols

- **KDF submodule**
  It contains key derivation functions that derive one or more secondary secret keys from one primary secret (a master key or a passphrase).
  Function : PBKDF2(password, salt, dkLen=16, count=1000, prf=None)
  Use : Derive one or more keys from a password (or passphrase) according to PKCS#5 standard (v2.0) by means of PBKDF2 algo

- **Util module**
  Various useful modules and functions (long-to-string conversion, random number generation, number theoretic functions)
  Function : pad(*data_to_pad*, *block_size*, *style='pkcs7'*)
  Use : Apply standard padding.
  Function : unpad(*padded_data*, *block_size*, *style='pkcs7'*)
  Use : Remove standard padding.

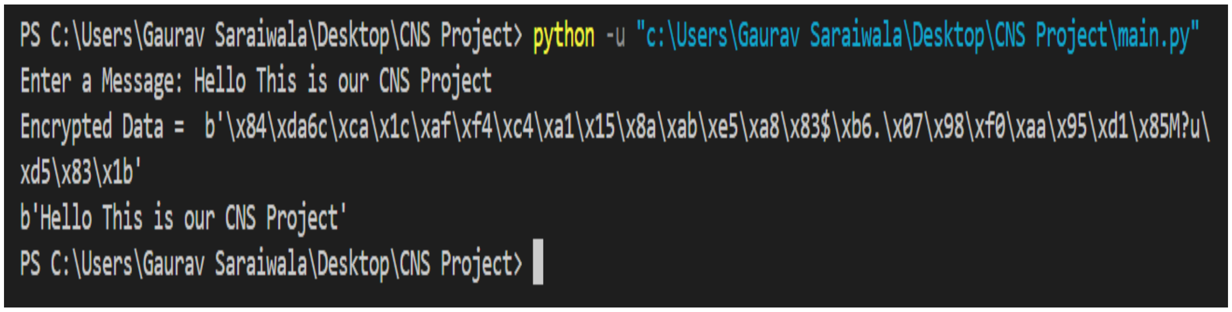
- **Random sub-module**
  used to generate random numbers
  Function : get_random_bytes(*N*)
  Use : Return a random byte string of length *N*.

5.2 Source Code:

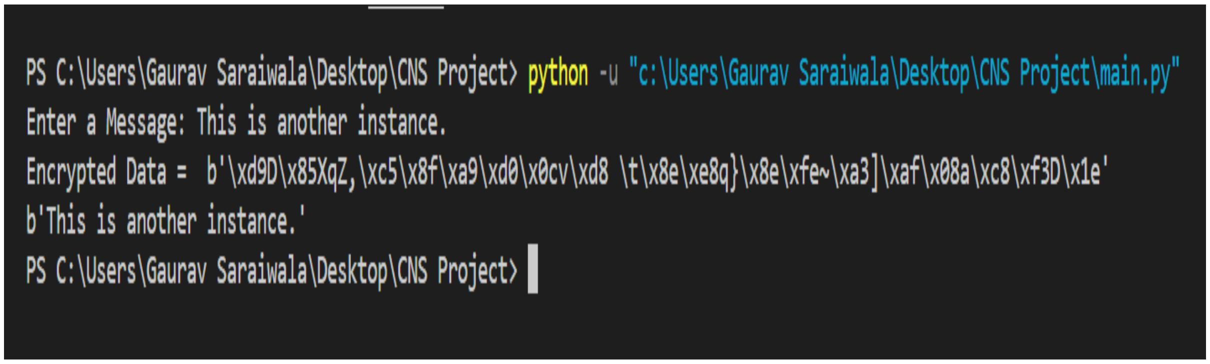
```
from Crypto.Random import get_random_bytes
from Crypto.Protocol.KDF import PBKDF2
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
salt=b'\xe2\x16]\xe3\x08\x91\xe8\x15\xe84\x05\x11\xbf\xea\x17?\x1f]#G\xb8\x00\x1e*\
xa7DGJX\xe414'
password = 'mypassword'
key = PBKDF2(password, salt, dkLen=32)
msg = input("Enter a Message: ")
message = msg.encode()
cipher = AES.new(key, AES.MODE_CBC)
cipher_data = cipher.encrypt(pad(message, AES.block_size))
print("Encrypted Data = ", cipher_data)
with open('encrypted.bin', 'wb') as f:
   f.write(cipher.iv)
   f.write(cipher_data)
with open('encrypted.bin', 'rb') as f:
   iv = f.read(16)
   decrypt_data = f.read()
cipher = AES.new(key, AES.MODE_CBC, iv=iv)
original = unpad(cipher.decrypt(decrypt_data), AES.block_size)
print(original)
```

Chapter 6

RESULTS/SNAPSHOTS



```
PS C:\Users\Gaurav Saraiwala\Desktop\CNS Project> python -u "c:\Users\Gaurav Saraiwala\Desktop\CNS Project\main.py"
Enter a Message: Hello This is our CNS Project
Encrypted Data =  b'\x84\xda6c\xca\x1c\xaf\xf4\xc4\xa1\x15\x8a\xab\xe5\xa8\x83$\xb6.\x07\x98\xf0\xaa\x95\xd1\x85M?u\
xd5\x83\x1b'
b'Hello This is our CNS Project'
PS C:\Users\Gaurav Saraiwala\Desktop\CNS Project>
```



```
PS C:\Users\Gaurav Saraiwala\Desktop\CNS Project> python -u "c:\Users\Gaurav Saraiwala\Desktop\CNS Project\main.py"
Enter a Message: This is another instance.
Encrypted Data =  b'\xd9D\x85XqZ,\xc5\x8f\xa9\xd0\x0cv\xd8 \t\x8e\xe8q}\x8e\xfe~\xa3]\xaf\x08a\xc8\xf3D\x1e'
b'This is another instance.'
PS C:\Users\Gaurav Saraiwala\Desktop\CNS Project>
```

Chapter 7

APPLICATIONS

- All practical implementations of public key cryptography today employ the use of a hybrid system.

1. TLS protocol and the SSH protocol, that use a public-key mechanism for key exchange (such as Diffie-Hellman) and a symmetric-key mechanism for data encapsulation (such as AES).

2. Can be implemented for Bluetooth (IJSR research paper).

3. To protect high-valued sensitive health records against malicious users and ensure integrity of patients' databases. (IJCSE research paper).

Chapter 8

CONCLUSION

Hybrid Encryption combines the efficiency of symmetric encryption with the convenience of public-key (asymmetric) encryption. Only users with the private key can decrypt the data. To encrypt a message, a fresh symmetric key is generated and used to encrypt the plaintext data. This provides a safe mechanism for data transmission over the network.

Chapter 9

REFERENCES

[1] IJSR(International Journal of Science and Research),"Cryptography Algorithm based on DES and RSA in Bluetooth Communication", Veeresh K. , Arunkumar Madupu, E-ISSN : 2319-7064,
https://www.ijsr.net/archive/v3i12/U1VCMTQ3ODI=.pdf

[2]IJCSE (International Journal of Computer Sciences and Engineering),"Hybrid Encryption for Radio Frequency Identification in Healthcare System: Object-Oriented Analysis and Design Approach ", Amanze, B.C ., Ononiwu, C. C., Eleberi, E.L & Chilaka , U.L. , E-ISSN: 2347-2693, Vol. 8, Issue.4, Apr 2020
https://www.ijcseonline.org/pub_paper/12-IJCSE-07890-22.pdf

[3] Medium.com,"Hybrid encryption in python", Jun 14
https://medium.com/@igorfilatov/hybrid-encryption-in-python-3e408c73970c

[4] diltz.net , "Package Crypto"
https://www.dlitz.net/software/pycrypto/api/2.6/Crypto-module.html

[5] pycryptodone.readthedocs.io , "AES"
https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

[6] w3schools.com , " Python File Read Method"
https://www.w3schools.com/python/ref_file_read.asp

[7] pypi.org,"pycrypto 2.6.1"
https://pypi.org/project/pycrypto/

[8] wikipedia , "PKBDF2"
https://en.wikipedia.org/wiki/PBKDF2