An

Industry Oriented Mini Project Report

on

IMPLEMENTATION OF SECURITY MECHANISMS USING AES AND RSA HYBRID ALGORITHM WITH DIGITAL SIGNATURES

A Report submitted in partial fulfilment of the requirements for the award of degree of Bachelor of Technology

by

Shaista Firdous
(20EG105442)

M. Sathvika Reddy

(20EG105428)

K. Shiva Sai

(19H61A05L6)

Under the guidance of

Dr. K. Madhuri

Associate Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG UNIVERSITY
VENKATAPUR-500088
GHATKESAR
TELANGANA
2023-2024

DECLARATION

We hereby declare that the report entitled **Implementation of security mechanisms using AES and RSA with Digital Signatures** submitted for the award of Bachelor of Technology Degree is our original work and the report has not formed the basis for the award of any Degree, associateship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any Degree.

Shaista Firdous
(20EG105442)

M. Sathvika Reddy
(20EG105428)

K. Shiva Sai
(19H61A05L6)

ANURAG UNIVERSITY | SCHOOL OF ENGINEERING

CERTIFICATE

This is to certify that the Report / Dissertation entitled **Implementation of security mechanisms using AES and RSA with Digital Signatures** that is being submitted by **Ms. Shaista Firdous** bearing Hall ticket number **20EG105442, Ms. Sathvika Reddy** bearing Hall ticket number **20EG105428** and **Mr. K. Shiva Sai** bearing Hall ticket number **19H61A05L6** in partial fulfilment for the award of B. Tech Computer Science and Engineering to Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this report have not been submitted to any other University or Institution for the award of any Degree.

Signature of The Supervisor

Dr. K. Madhuri
Associate Professor

Dean, CSE

External Examiner 1

External Examiner 2

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. K. Madhuri, Associate Professor, Department of CSE** for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Department. of CSE, Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram**, Academic co-ordinator, **Dr. Pallam Ravi**, Project in-Charge and Class in-Charge, Project co-ordinator and Project review committee members whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean, School of Engineering, Anurag University, for his encouragement and timely support in our B. Tech program.

Shaista Firdous
(20EG105442)

M. Sathvika Reddy
(20EG105428)

K. Shiva Sai
(19H61A05L6)

ABSTRACT

In our modern interconnected digital environment, safeguarding sensitive data from unauthorized access is paramount due to the prevalence of cyber threats like hacking, malware, phishing, and data breaches. Maintaining confidentiality, integrity, and authentication of transmitted information, whether in business or personal domains, requires robust security measures.

To address this issue, we have implemented a security solution utilizing encryption algorithms such as AES (Advanced Encryption Standard) and RSA (Rivest, Shamir, Adleman). AES plays a vital role in rendering stolen data useless to unauthorized parties by encrypting it. However, as AES is a symmetric algorithm, there is a potential risk of key leakage. To mitigate this risk and ensure secure transmission, we utilize the RSA algorithm to securely transfer the encryption key. By combining these two algorithms into a hybrid approach, we create a formidable defense against unauthorized access, making it challenging to decode or manipulate sensitive data. This hybrid encryption methodology significantly contributes to secure data transfer between two parties.

The hybrid encryption approach, combining AES and RSA algorithms, has demonstrated its efficacy in enhancing data security during transmission. The encryption ensures the confidentiality and integrity of sensitive information, making it highly resistant to unauthorized access and manipulation. The secure transfer of encryption keys using RSA further strengthens the overall security, reducing the risks associated with key leakage. The results showcase a significant advancement in the domain of secure data transmission.

Our project has successfully addressed the critical concern of securing sensitive data during transmission in our interconnected digital era. By employing a hybrid encryption approach that utilizes AES and RSA algorithms, we have provided a robust defense mechanism against cyber threats. This approach enhances data confidentiality, integrity, and authentication, establishing a foundation for secure communication in both personal and business domains. The successful implementation and demonstrated results underscore the importance of encryption in ensuring secure data transfer and pave the way for future advancements and studies in the field of data security.

Key Words: AES algorithm, RSA algorithm, Hybrid algorithm

LIST OF FIGURES

LIST OF TABLES

LIST OF CONTENTS

1.INTRODUCTION

As we know in today's world large amount of data is created because of wide use of internet and computers. The security of data in the process of data transmission over the network attracted much attention. Many intruders attack to access the confidential data which might poses some serious threat to political, military, economy of the country. It is important to protect such data from the intruders who may use this data against. This created an attention in the Information technology and network technology field that how to encrypt import important information reliably [1]. The standard encryption algorithms are divided into symmetric, asymmetric and hash algorithms [2]. Symmetric encryption algorithm means encrypting and decrypting the message with the same key. Some of the algorithms are DES, AES, Double DES, Triple DES. Asymmetric algorithm encrypting and decrypting of the algorithm happens with different keys that is public and private keys. Some of the algorithms are RSA, and ECC. AES and RSA algorithms are widely used symmetric and asymmetric algorithms for data confidentiality. The AES and RSA hybrid algorithm are applied to the encrypted transmission of messages for security.

The proliferation of information communication brings with it an increasing spectrum of associated risks. Unauthorized access, eavesdropping, and data tampering constantly loom as threats in the digital domain. Conventional services are susceptible to these vulnerabilities, and the imperative for robust security measures has never been more pronounced. This project derives its impetus from the urgency to offer individuals and organizations a comprehensive solution for secure information transfer. It aims to secure both the content of messages and the authenticity of the sender.

Information can be susceptible to interception, alteration, or false attribution to unwitting senders, casting shadows on the privacy and trustworthiness of digital communication. The challenge at hand is the development of a system that guarantees the confidentiality of information content and validates the sender's identity, thereby mitigating potential risks and vulnerabilities.

Implement AES encryption to fortify the confidentiality of information content, rendering intercepted information incomprehensible to unauthorized parties. Apply RSA encryption and digital signatures to verify the legitimacy of the sender's identity, providing recipients with the assurance that the information originates from a trustworthy source and has remained unaltered during transit. Forge a user-friendly interface, facilitating seamless and secure information communication, making it accessible to both individuals and organizations. Assess the system's efficacy concerning security, performance, and user-friendliness. This project aspires to elevate the security of information communication, reinforcing the trust we place in our digital interactions while advocating for the confidentiality and integrity of information in the online realm.

1.1. AES ALGORITHM

AES is Advanced Encryption Standard algorithm, which implements block ciphering technique. It is a symmetric cryptographic algorithm with fixed message and key length. It is an improved algorithm based on DES algorithm. The key length of AES algorithm are divided into- 128 bits, 192 bits, 256 bits. The message length is of 128bit grouped data blocks [3]. The number of rounds are depended on the key length. That relation is showed in table 1. AES algorithm performs 4 steps at each round, those are- Sub Bytes, Shift Rows, Mix Columns and Adding Round Key.

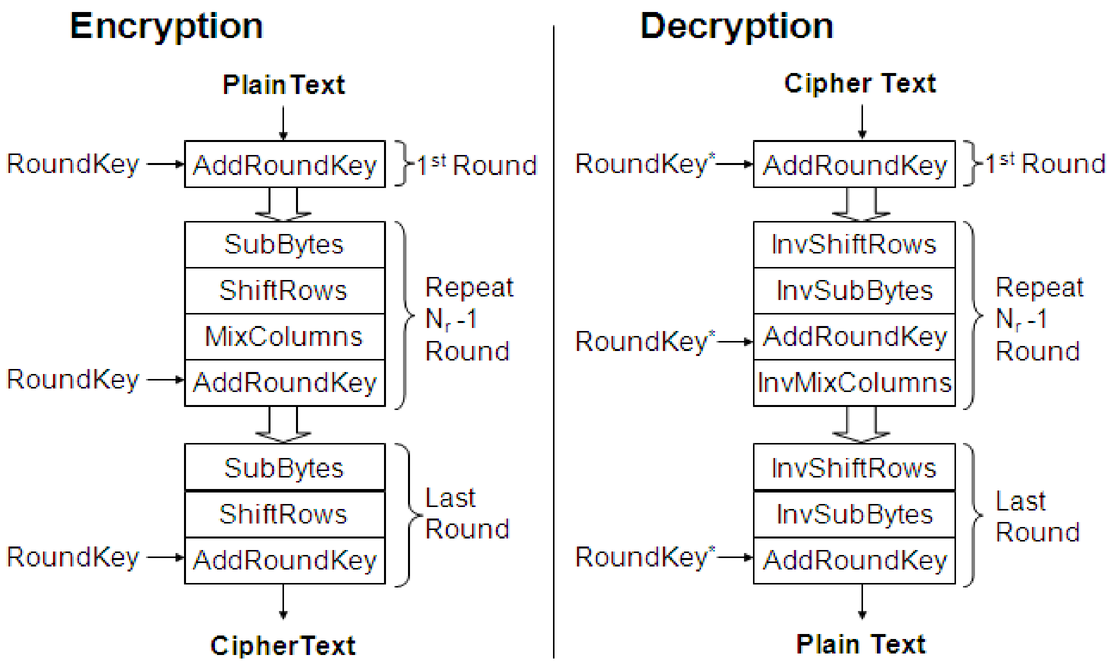


Figure 1.1 AES symmetric encryption

KEY SIZE	ROUNDS
128 bits	10 rounds
192 bits	12 rounds
256 bits	16 rounds

Table-1.1: Algorithm Round Table

Let us take an example to understand AES algorithm

Plaintext-This is a algorithm.

Key- Encryption is a key.

TEXT	HEXADECIMAL
This is algorithm	74 68 69 73 69 73 61 61 6c 67 6f 72 69 74 68 6d
Encryption is a key	65 6e 63 72 79 70 74 69 6f 6e 69 73 61 6b 65 79

74	68	69	73
69	73	61	61
6c	67	6f	72
69	74	68	6d

XOR

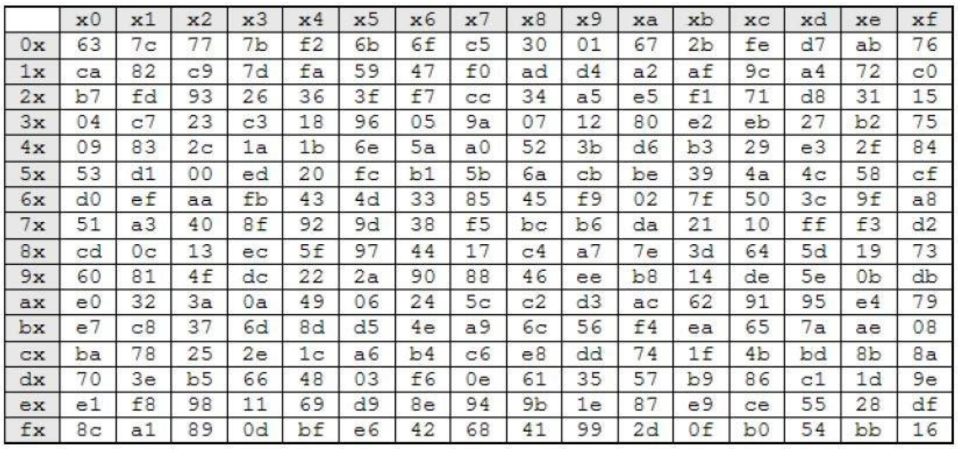
$\oplus$

65	6e	63	72
79	70	74	69
6f	6e	69	73
61	6b	65	79

11	06	0a	01
10	03	15	08
03	09	06	01
08	1f	0d	14

Table 1.2. XOR operation

Step 1: **Sub Bytes-**Each byte is replaced with another byte in this phase. The S-box, is employed. A byte is never replaced by itself or by a byte that is a complement of the current byte because of the manner this substitution is carried out. This process yields the same 16-byte (4 x 4) matrix as previously.



	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 1.3. S-box

From above table

11	06	0a	01
10	03	15	08
03	09	06	01
08	1f	0d	14

Substituting Bytes $\longrightarrow$

82	6f	67	7c
ca	7b	59	30
7b	01	6f	7c
20	C0	d7	fa

Table 1.4 Substitution Bytes

Step 2: **Shift Rows-** Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left

```
[ b0 | b1 | b2 | b3 ]          [ b0 | b1 | b2 |b3  ]
| b4 | b5 | b6 | b7 |   --->   | b5 | b6 | b7 | b4 |
| b8 | b9 | b10| b11|          | b10| b11| b8 | b9 |
[ b12| b13| b14| b15]          [ b15| b12| b13| b14]
```

Example:

82	6f	67	7c
ca	7b	59	30
7b	01	6f	7c
20	C0	d7	fa

$\longrightarrow$

82	6f	67	7c
7b	59	30	ca
6f	7c	7b	01
fa	20	C0	d7

Table 1.5. Shift Rows

***Step 3*-Mix Columns-**This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

```
[ c0 ]  =   [ 2  3  1  1 ]  [ b0 ]
| c1 |  =   | 1  2  3  1 |  | b1 |
| c2 |  =   | 1  1  2  3 |  | b2 |
[ c3 ]  =   [ 3  1  1  2 ]  [ b3 ]
```

Example:

82	6f	67	7c
7b	59	30	ca
6f	7c	7b	01
fa	20	C0	d7

X

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

=

c0	8f	4e	3c
66	f9	c6	a5
47	17	4c	81
31	ec	48	87

Table 1.6 Mix Columns

Step 4: **Add round keys-** Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.

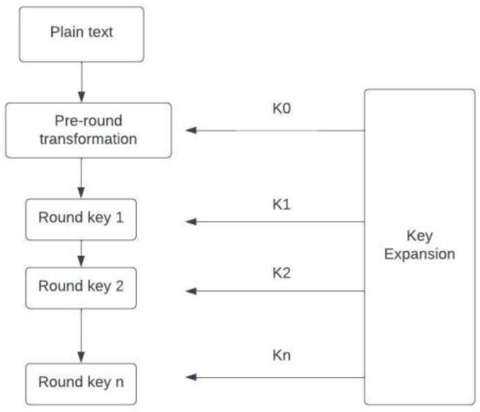


Figure 1.2 Adding Round keys

Round	0	Key:	65	6e	63	72	79	70	74	69	6f	6e	69	73	61	6b	65	79
Round	1	Key:	42	69	5e	35	21	f0	8d	88	5d	20	ac	07	b4	10	ff	48
Round	2	Key:	c2	9b	9d	61	b1	d1	c5	ef	45	6f	39	d7	95	8a	3a	6b
Round	3	Key:	ef	79	c0	4e	30	f6	a0	ad	3f	a6	07	f6	5c	68	84	5f
Round	4	Key:	e2	a4	97	64	db	c6	3f	53	14	e6	43	77	89	f3	5f	75
Round	5	Key:	b1	f3	e4	45	a4	c0	c9	4c	5d	be	f2	0e	08	10	84	74
Round	6	Key:	c5	cb	83	3e	d6	14	14	28	a4	57	e4	55	36	86	4f	97
Round	7	Key:	75	1b	f6	a0	b2	15	b2	c2	ea	0d	8f	69	b1	19	16	63
Round	8	Key:	f8	16	98	fb	ec	16	41	7a	43	b7	d7	67	48	c4	15	15
Round	9	Key:	c9	1b	5e	91	e0	27	93	8f	49	0c	85	77	81	21	6c	a6
Round	10	Key:	fe	87	e3	0e	4c	2a	5b	10	98	9b	75	88	77	34	ab	44
Round	11	Key:	6b	f9	81	c4	56	d5	54	3b	7f	12	6b	e9	94	f0	6d	08

Step 1: Adding Round 0 key

c0	8f	4e	3c
66	f9	c6	a5
47	17	4c	81
31	ec	48	87

XOR

$\oplus$

65	6e	63	72
79	70	74	69
6f	6e	69	73
61	6b	65	79

a5	e1	0d	4f
1f	89	b2	dc
a8	77	3f	e4
50	85	2d	fe

Table 1.7 Adding Round keys

Similarly, we have to add round keys for all 14 rounds
The output for the last round is

b0	95	aa	00
9b	5e	fc	cd
6a	07	0f	91
d1	08	4e	c4

Plaintext	This is a algorithm
Key	Encryption is a key
Encrypted Text	ZmUxbzfpW8V3t3sousx635nUVu4/F/IcNDIj8QeppkHS i7icVMmzm/ah9tvEx+jC

- If we reverse the encryption algorithm we can get the decrypted message.

1.2. RSA ALGORITHM

RSA is an asymmetric encryption algorithm proposed by Ron Rivest, Shamir, Adleman in 1978 and it is named with the initials of three scholar’s surnames [5]. The security of RSA is dependent on the difficulty of decomposing of largest prime numbers [6]. Here is the algorithm for RSA.

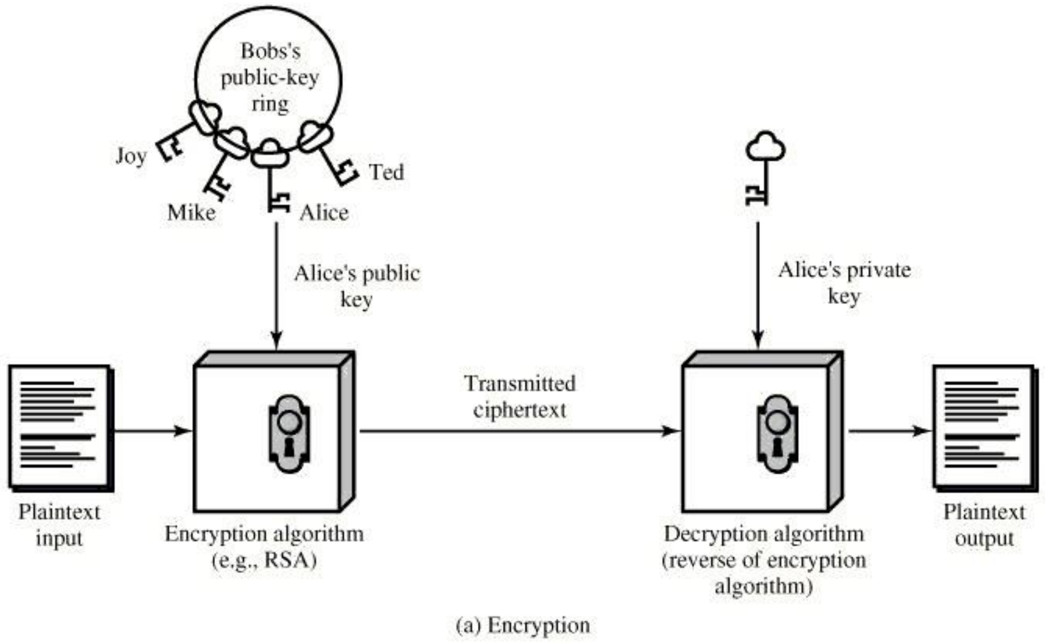


Figure 1.3 Asymmetric Encryption

Step 1: Firstly, system randomly generates two largest prime numbers p and q.

Step 2: Then calculate public and private keys using mathematical calculations.

$n=p*q,\ \phi(n)=(p-1)\ (q-1)$

$GCD\ (e,\ \phi(n))\ =1$

$d=e^{-1}\ mod(\phi(n))$

Step 3: Identifying Public key and Private key

Public key: (e, n)   Private key: (d, n)

Step 4: Formula to encrypt

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

Step 5: Formula to decrypt

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

For Example,

Step 1: The large prime numbers p=61 and q=53

Step 2: Compute n=p x q, which is n= 61 x 53= 3233.

Step 3: Compute $\phi(n)=(p-1)(q-1) = 60 \times 52 = 3120$

Step 4: Choosing the public exponent $e=17$, where $GCD(e, \phi(n)) = 1$

Step 5: Compute the private exponent $d=e^{-1} \bmod(\phi(n))$ where $d=2753$

Step 6: Public key (e, n) = (17,3233)
Private key (d, n) = (2753, 3233)

Encryption

Plaintext	Encryption is a key
Plaintext converted to integer	2067
Public Key	(17, 3233)
Encrypted text in integer	689

Decryption

Encrypted text in integer	689
Private key	(2753, 3233)
Decrypted plaintext in integer	2067
Decrypted Original text	Encryption is a key

2. LITERATURE SURVEY

Seth et al. [2] has done the comparative analysis of three algorithms; RSA, DES and AES while considering certain parameters such as computation time, memory usage and output byte. These parameters are the major issues of concern in any encryption algorithms. Experimental results show that AES algorithm has least memory usage while encryption time difference is minor in case of AES algorithm. RSA consumes longest encryption time and memory usage is also very high but output byte is least in case of RSA algorithm.

Amrita Sahu et.al [3] proposed a new key generation algorithm based on palm print which is used for encryption and decryption of a message. Our scheme allows one party to send a secret message to another party over the open network, even if many eavesdroppers listen. This scheme gives reliable security.

Kakkar et al. [4] studied various techniques and algorithms used for data security in MN (Multimode Network). It has been observed that strength of system depends upon the key management, type of cryptography (public or private keys), number of keys, number of bits used in a key. Longer key length and data length consumes more power and results in more heat dissipation. Larger the number of bits used in a key, the more secure the transmission. All the keys are based upon the mathematical properties and their strength decreases with respect to time. The keys having more number of bits requires more computation time which simply indicates that system takes more time to encrypt the data**.**

Alanazi et al. [5] has done the comparative analysis of three encryption algorithms (DES, 3DES and AES) within nine factors such as key length, cipher type, block size, security, possible keys at 50 billion keys per second etc. study shows that AES is better than DES and 3DES**.**

Pavithra et al. [6] compares the performance evaluation of various cryptographic algorithms. On the basis of parameter taken as time, various cryptographic algorithms are evaluated on different video files. Different video files are having different processing speed on which various sizes of files are processed. Calculation of time for encryption and decryption in different video file format such as .vob and .dat, having size from 1MB to 1100 MB. Results shows that AES algorithm was executed in lesser processing time and more throughput level as compared to DES and Blowfish.

S. Koko and A. Babiker et al. [10] compared the measured encryption speed to several methods and then provide a summary of the algorithms' other characteristics. AES is one of the encryption methods that is taken into consideration here (with 128 and 256-bit keys).

Mandal et al. [9] compared two most commonly used symmetric techniques i.e. Data Encryption Standard (DES) and Advanced Encryption Standard (AES) on the basis of avalanche effect due to one bit variation in plaintext constant, memory required for implementation and simulation time required for encryption.

Arora et al. [8] studied about the performance of different security algorithms on a cloud network and on a single processor for different input sizes. This paper aims to find in quantitative terms like speed up ratio that benefits of using cloud resources for implementing security algorithms (RSA, MD5 and AES) which is used by businesses to encrypt large volume of data.

2.1 Comparison of Existing Methods

Author(s)	**Method**	**Advantages**	**Disadvantages**
Ye Liu et al.	**File and Disk encryption:** AES is used to encrypt files and folders on computers.	**1)Confidentiality** **2)Efficiency** **3)Security:** AES provides security and resistant to brute-force attack.	As AES is symmetric algorithm, there are high chances of leakage of data.
Wei Gong et al.	**RSA Algorithm:** Asymmetric encryption algorithm	**1)Security:** RSA algorithm is a very secure method of encrypting and decrypting sensitive information. **2)Key Exchange:** RSA algorithm can be used to safely transmit the key.	1)Computational Intensity 2)Key Management 3)Performance impact.
Wenqing Fan et al.	**AES algorithm:** Symmetric encryption algorithm using Java language.	1)Platform Independent 2)Security 3)Readability and Maintainability 4)Community Support	1)Hard to implement with software. 2)Every block is always encrypted in the same way.

3. AES AND RSA HYBRID ALGORITHM

We are working on a method which is already proposed that is using AES and RSA hybrid algorithm to secure the data. Incorporating AES and RSA hybrid algorithm within email security enhances the security.

3.1 DIGITAL SIGNATURES

RSA is also used for digital signatures, which is a technique to verify the authenticity and integrity of messages. In this approach, RSA is used for both encryption (for generating the digital signature) and decryption (for verifying the digital signature).

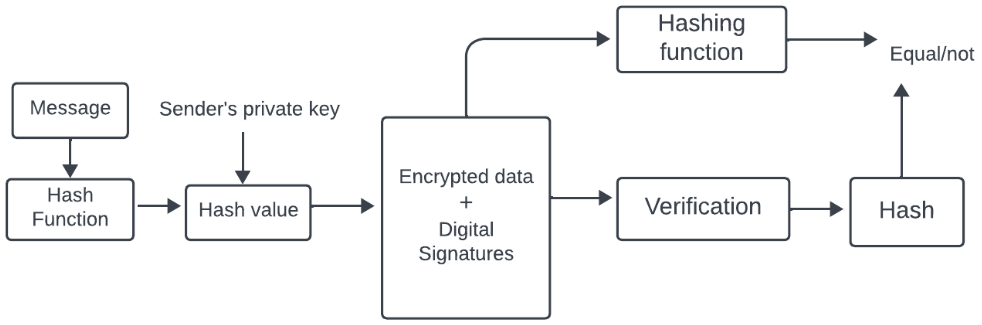


Figure 3.1 Digital Signatures

3.1.1. Generating Digital Signature (Encryption)

Step 1: Generate an RSA asymmetric key pair (public and private key) for digital signatures.

Step 2: Generate a hash value by hashing the message using an algorithm; SHA-256.

Step 3: Encrypt the hash value using the sender's RSA private key to create the digital signature.

Step 4: The digital signature is the encrypted hash, which is sent along with the original message.

3.1.2. Digital Signature Verification (Decryption)

Step 1: Decrypt the digital signature using the sender's public key (RSA public key) to obtain the decrypted hash value.

Step 2: Hash the original message again using the same hash function used during signature generation to generate a new hash value.

Step 3: Compare the decrypted hash value obtained from the signature with the newly generated hash value from the received message.

Step 4: If the decrypted hash matches the newly generated hash, the digital signature is valid, and the message is considered authentic and unaltered. Otherwise, the verification fails, indicating that the message may be altered or not authentic.

3.2. HYBRID AES AND RSA ALGORITHM

The proposed hybrid algorithms are AES and RSA. According to algorithm analysis AES algorithm is much faster than RSA and it is suitable to encrypt large data without any complex issues, whereas RSA algorithm is suitable to encrypt small amount of data and can also provide services like authentication by performing digital signatures [7]. RSA algorithm has small size of keys which has an advantage of key management. In this we are using AES algorithm to encrypt the message and RSA algorithm used to encrypt the AES that is it uses public key in encryption and private key in decryption. And RSA algorithm is also used in Digital Signature for sender authentication.

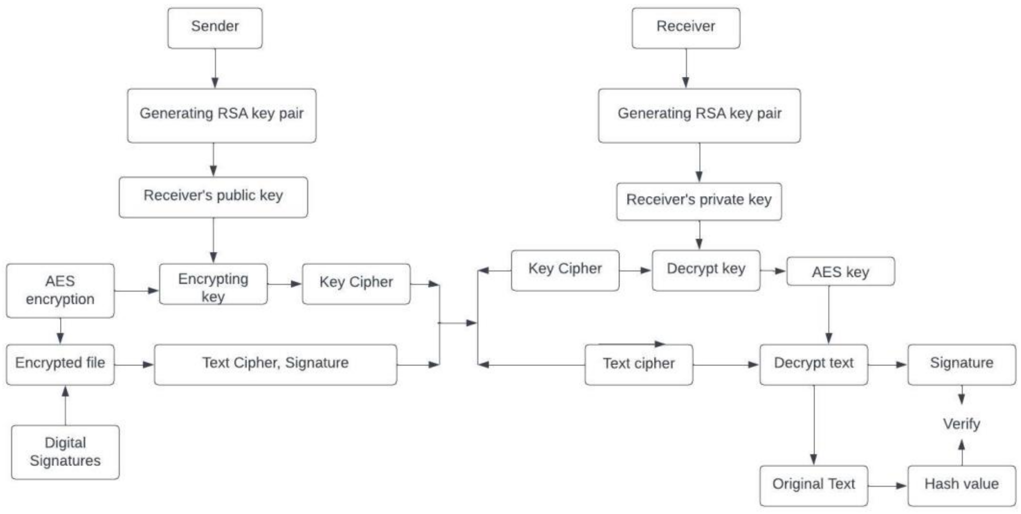


Figure 3.2 AES and RSA Hybrid Algorithm

3.2.1. Application of AES and RSA Hybrid Algorithm

Combining AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) in a hybrid encryption scheme is a common practice in secure communication and data storage. AES is a symmetric encryption algorithm, while RSA is an asymmetric encryption algorithm. The hybrid approach utilizes the efficiency of symmetric encryption for data encryption and the security of asymmetric encryption for key exchange and securing the symmetric key.

ENCRYPTION ALGORITHM

Step 1: Input the AES Symmetric key.

Step 2: Input the Plaintext message.

Step 3: Input the prime numbers for RSA algorithm, that is p and q values.

Step 4: Convert the plaintext into 128 bits block size by padding the extra bits.

Step 5: Convert the key until it matches the length of the padded plaintext.

Step 6: Now, encrypt the plaintext using AES symmetric key using AES algorithm.

Step 7: Generate public and private keys from above prime numbers taken.

Step 8: Now encrypt the AES symmetric key with RSA public key using RSA algorithm.

Step 9: Generate a hash value from the plaintext.

Step 10: Encrypt the hash value with sender's private key using RSA approach.

Step 11: Now we send both cipher key, cipher text along with digital signature to the receiver.

S.NO	**LEVEL OF PROCESSING**	**RESULTS**
1.	Input plaintext	This is algo
2.	Input AES key	Encryption is a key
3.	Prime Numbers	p=61 and q=53
4.	Padded plaintext	This is algoxxxxxx
5.	Generated keys	Public key-(17, 3233) Private key-(2753, 3233)
6.	Encrypted plaintext	ZmUxbzfpW8V3t3sousx635nUVu4/F/ IcNDIj8QeppkHSi7icVMmzm/ah9tvEx+jC
7.	Encrypted key in integer	689
8.	Generated hash value	b'j0\xa2\xa4\x99n\xf8\xb1\xbb\xa0\xa522~5\x01zM\ xaa%\xfbz\x8f\xdc/\xc5\xc4)[\x02d\r'
9.	Encrypted hash value in integer	1173

Table 3.1 Hybrid Encryption algorithm

DECRYPTION ALGORITHM

Step 1: Receive the cipher text along with encrypted signature and cipher key.

Step 2: Decrypt the cipher key with RSA private key using RSA algorithm.

Step 3: Divide the padded plaintext into 128 bits block size.

Step 4: Now, decrypt the padded plaintext using the decrypted key using AES algorithm.

Step 5: Now un-pad the extra bits from the plaintext.

Step 6: Generate a hash value from the plaintext
Step 7: Now, decrypt the encrypted signature using sender’s public key.

Step 8: Now, verify whether the decrypted signature and generated hash value are

1.	Received cipher text	ZmUxbzfpW8V3t3sousx635nUVu4/F/ IcNDIj8QeppkHSi7icVMmzm/ah9tvEx+jC
2.	Received cipher key	689
3.	Received encrypted signature in integer	1173
4.	Decrypted cipher key	Encryption is a key
5.	Decrypted cipher text	This is algoxxxxxx
6.	Unpadded cipher text	This is algo
7.	Decrypted signature	b'j0\xa2\xa4\x99n\xf8\xb1\xbb\xa0\xa522~5\x01zM\ xaa%\xfbz\x8f\xdc/\xc5\xc4)[\x02d\r'
8.	Generated hash value	b'j0\xa2\xa4\x99n\xf8\xb1\xbb\xa0\xa522~5\x01zM\ xaa%\xfbz\x8f\xdc/\xc5\xc4)[\x02d\r'
9.	Verifying digital signature	True

same or not.

Table 3.2 Hybrid Decryption algorithm

4. IMPLEMENTATION

Program file is **hybrid_cryptography.py**

Used **python IDLE (3.11)** to develop hybrid cryptography system.

4.1 PACKAGES

Pycryptodome is a Python library providing cryptographic functions, including symmetric and asymmetric encryption, hash functions, and more. It's a popular choice for secure data handling and cryptographic operations due to its robustness, efficiency, and ease of use in implementing cryptographic algorithms.

1) **hashlib:** This Python standard library module is used for hashing functions, often used for data integrity and security.

2) **Crypto.PublicKey:** A module within Pycryptodome that provides functionalities related to public key cryptography, including RSA encryption and decryption.

3) **Crypto.Cipher:** Another module within Pycryptodome, offering various encryption and decryption algorithms, such as AES (Advanced Encryption Standard).

4) **Crypto.Random:** Part of Pycryptodome, this module provides functions for generating random bytes, essential for cryptographic operations.

5) **Crypto.Protocol.KDF:** A module within Pycryptodome used for key derivation functions (KDFs), like PBKDF2 (Password-Based Key Derivation Function 2).

6) **base64:** A Python standard library module for encoding and decoding binary data into base64 representation.

7) **math:** A Python standard library module providing mathematical functions and operations.

8) **time:** A Python standard library module that provides various time-related functions, which might be used for timing or delays.

9) **matplotlib.pyplot:** A popular third-party Python package used for creating data visualizations like plots and graphs.

4.2. FUNCTIONALITIES:

1)**AES_Encrypt:** Encrypts the 128bit block plaintext using AES encryption using the AES symmetric key.

2) **AES_Decrypt:** Decrypts the 128bit block plaintext using AES decryption using the AES symmetric key.

3) **RSA_Encrypt:** Encrypts the key with receiver's public key using RSA encryption algorithm.

4) **RSA_Decrypt:** Decrypts the key with receiver's private key using RSA decryption algorithm.

5) **PBKDF2_key:** PBKDF2 is used to derive a cryptographic key from a user entered password.

6) **get_random_bytes:** Generation of random bytes for IV (Initialization Vector) in AES encryption.

7) **Pad:** Padding the plaintext to be a multiple of the AES block size.

8) **Un-pad:** Removing the bits that are added to the plaintext.

9) **encode and decode:** Base64 encoding is used to represent binary data in an ASCII string format using 64 different printable characters. Decoding for converting binary data to a human-readable ASCII string.

10) **Time Measurement**: Measuring the time taken for encryption and decryption operations.

11) **Plotting**: Plotting line charts to visualize encryption and decryption times.

12) **convert_to_int:** Used to convert the AES key into integer, if it is in its ASCII value then also it converts into integer.

13) **sign:** It is used to generate the hash value using SHA-256 hash function and also to encrypt the hash value.

14) **Verify:** Used to decrypt the hash value and to verify that the user is authentic or not.

15) **is_prime:** To check whether the p and q values taken are prime or not.

16) **generate_keypair:** To generate the public key and private key for RSA

encryption and decryption algorithms.

4.3. ATTRIBUTES:

1)**Message:** Copies the plaintext and add the padded bits to it.

2) **Key:** Takes the key input from user and matches with the length of the plaintext.

3) **public_key:** The generated public key using p and q values are stored, where (e, n) are stored.

4) **private_key:** The generated private key using p and q values are stored, where (d, n) are stored.

5)**message1:** The encrypted AES key is stored, which is in integer format.

6)**encrypted_message:** Stores the encrypted ciphertext.

7)**encrypted_password:** Stores the encrypted AES key.

8) **decrypted_message:** Stores the decrypted ciphertext.

9) **decrypted_password:** Stores the decrypted AES key.

10) **encryption_time_aes:** Stores the encryption time of AES algorithm.

11)**encryption_time_rsa:** Stores the encryption time of AES algorithm.

12) **decryption_time_aes:** Stores the decryption time of AES algorithm.

13)**decryption_time_rsa:** Stores the decryption time of RSA algorithm.

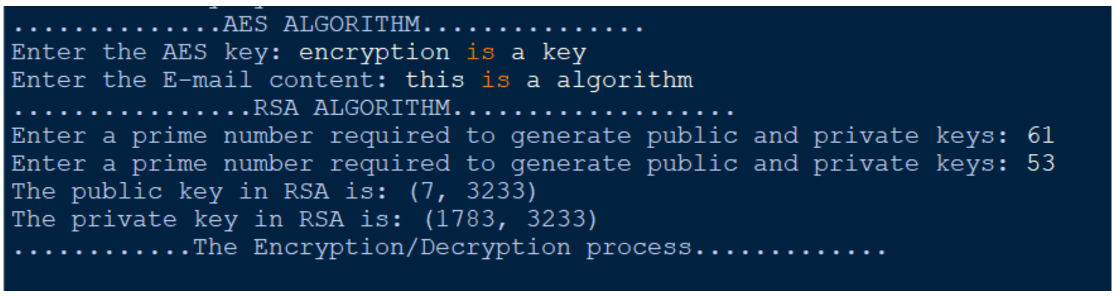
4.4 SAMPLE CODE

```
import hashlib
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Protocol.KDF import PBKDF2
import base64
import math
import time
import matplotlib.pyplot as plt
print("Email Security System")
print("..............AES ALGORITHM..............")
password = input("Enter the AES key: ")
message = input("Enter the E-mail content: ")
```

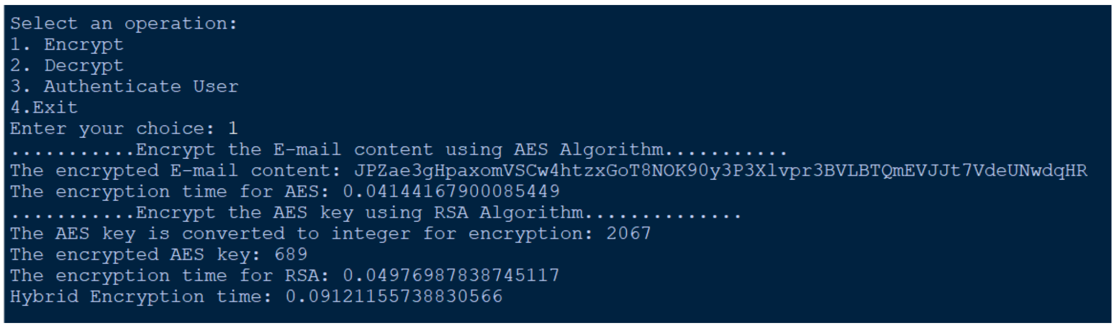
```
# To add the extra bits to become a multiple of 128
def pad(text):
    return text + (16 - len(text) % 16) * chr(16 - len(text) % 16)
# To remove extra bits added to the plaintext
def unpad(text):
    return text[:-ord(text[-1])]
#To encrypt plaintext
def encrypt(key, plaintext):
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    padded_plaintext = pad(plaintext)
    padded_plaintext_bytes = padded_plaintext.encode('utf-8')
    ciphertext = cipher.encrypt(padded_plaintext_bytes)
    return base64.b64encode(iv + ciphertext).decode('utf-8')
#To decrypt plaintext
def decrypt(key, ciphertext):
    ciphertext = base64.b64decode(ciphertext)
    iv = ciphertext[:16]
    ciphertext = ciphertext[16:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_text = cipher.decrypt(ciphertext)
    decrypted_text = unpad(decrypted_text.decode('utf-8'))

    return decrypted_text
print("................RSA ALGORITHM..................")
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

def get_user_input_prime():
    while True:
        try:
            prime = int(input("Enter a prime number required to generate public and
private keys: "))
            if is_prime(prime):
                return prime
            else:
                print("Please enter a valid prime number.")
        except ValueError:
            print("Please enter a valid integer.")

# Get user input for p and q
p = get_user_input_prime()
q = get_user_input_prime()
def find_coprime(num):
    value = 2
    while True:
        if math.gcd(num, value) == 1:
            return value
        value += 1
def mod_inverse(a, m):
    m0, x0, x1 = m, 0, 1
    while a > 1:
        q = a // m
        m, a = a % m, m
        x0, x1 = x1 - q * x0, x0
    return x1 + m0 if x1 < 0 else x1
#To generate public and private keys
def generate_keypair(p, q):
    n = p * q
    phi_n = (p - 1) * (q - 1)
    e = find_coprime(phi_n)
    d = mod_inverse(e, phi_n)
    return (e, n),(d, n)
public_key, private_key = generate_keypair(p, q)
#To encrypt AES key
def encrypt1(public_key, plaintext):
    e,n=public_key
    return pow(plaintext, e, n)
#To decrypt AES key
def decrypt1(private_key, ciphertext):
    d,n=private_key
    return pow(ciphertext, d, n)
#To generate digital signature
def sign(message, private_key):
    hash_value = hashlib.sha256(message.encode()).digest()
    print("The generated hash value from the message for digital
signature:",hash_value)
    integer_value=int.from_bytes(hash_value, 'big')
    integer_value1=integer_value%private_key[1]
    print("The hash value which should be encrypted is:",integer_value1)
    signature1=encrypt1(private_key,integer_value1)
    print("The encrypted signature is:",signature1)
    return signature1
#To verify digital signature
def verify(message, signature, public_key):
    hash_value = hashlib.sha256(message.encode()).digest()
    integer_value=int.from_bytes(hash_value, 'big')
    integer_value1=integer_value%private_key[1]
    computed_hash=decrypt1(public_key,signature)
    print("The decrypted signature is:",computed_hash)
    return integer_value1 == computed_hash
#To identify encryption and decryption times
def encryption_time(encryption_time_aes,encryption_time_rsa):
    time=encryption_time_aes+encryption_time_rsa
    return time
def decryption_time(decryption_time_aes,decryption_time_rsa):
    time=decryption_time_aes+decryption_time_rsa
    return time
#Visual representation of time
def plot_line_chart(labels,rsa_times,hybrid_times,title):
    plt.plot(labels, rsa_times, marker="o",label='RSA')
    plt.plot(labels, hybrid_times,marker="o",label='Hybrid (AES+RSA)')
    plt.xlabel('Message Length (bytes)')
    plt.ylabel('Time(s)')
    plt.title(title)
    plt.legend()
    plt.grid()
    plt.show()

def main():
    def convert_to_int(password):
        if password.isdigit():
            return int(password)
        else:
            concatenated_ascii = ''.join(str(ord(char)) for char in password)
            return concatenated_ascii
    password1=str(convert_to_int(password))
    key = PBKDF2(password1, b"salt", dkLen=32, count=1000000)
    public_key, private_key = generate_keypair(p,q)
    print("The public key in RSA is:",public_key)
    print("The private key in RSA is:",private_key)
    n=public_key[1]
    message1=int(password1)%n
    print("............The Encryption/Decryption process............")
    while True:
        print("\nSelect an operation:")
        print("1. Encrypt")
        print("2. Decrypt")
        print("3. Authenticate User")
        print("4.Exit")
        choice = int(input("Enter your choice: "))
        if choice == 1:
             start_time=time.time()
             encrypted_message = encrypt(key, message)
             print("...........Encrypt the E-mail content using AES Algorithm...........")
             print("The encrypted E-mail content:", encrypted_message)
             end_time=time.time()
             encryption_time_aes=end_time-start_time
             print("The encryption time for AES:",encryption_time_aes)
             start_time1=time.time()
             encrypted_password=encrypt1(public_key,message1)
             print("...........Encrypt the AES key using RSA Algorithm.............")
             print("The AES key is converted to integer for encryption:",message1)
            print("The encrypted AES key:",encrypted_password)
            end_time1=time.time()
            encryption_time_rsa=end_time1-start_time1
            print("The encryption time for RSA:",encryption_time_rsa)
            encrypt_time=encryption_time(encryption_time_aes,encryption_time_rsa)
            print("Hybrid Encryption time:",encrypt_time)
        elif choice == 2:
            start_time=time.time()
            decrypted_message = decrypt(key, encrypted_message)
            print("............Decrypt the E-mail content using AES Algorithm............")
            print("The decrypted E-mail content:", decrypted_message)
            end_time=time.time()
            decryption_time_aes=end_time-start_time
            print("The decryption time for AES:",decryption_time_aes)
            print("............Decrypt the AES key using RSA Algorithm............")
            start_time=time.time()
            decrypted_password=decrypt1(private_key,encrypted_password)
            print("The decrypted AES key:",decrypted_password)
            end_time=time.time()
            decryption_time_rsa=end_time-start_time
            print("The decryption time for RSA:",decryption_time_rsa)
            decrypt_time=decryption_time(decryption_time_aes,decryption_time_rsa)
            print("Hybrid Decryption time:",decrypt_time)
        elif choice == 3:
            print("............Authenticating the user using Digital Signature...........")
            signature=sign(message, private_key)
            is_valid = verify(message, signature, public_key)
            print("Is the signature valid?", is_valid)
        elif choice == 4:
            break
        else:
            print("Invalid choice. Please try again.")
    message_lengths = [128,256,512,1024,2048,5096]  # Message lengths in bytes
    rsa_times = [0.04,0.11,0.21,0.31,0.49,0.59]  # Placeholder RSA encryption times
    hybrid_times = [0.025,0.07,0.15,0.19,0.32,0.38]

    # Plot line chart for encryption times
    plot_line_chart(message_lengths,rsa_times, hybrid_times, 'Encryption')

    # Placeholder decryption times (modify these with actual times)
    rsa_dec_times = [0.60,0.86,1.31,1.76,2.68,3.2]
    hybrid_dec_times = [0.51,0.55,0.65,0.69,0.81,0.88]

    # Plot line chart for decryption times
    plot_line_chart(message_lengths,rsa_dec_times,hybrid_dec_times,'Decryption')

if __name__ == "__main__":
    main()
```

5. EXPERIMENT RESULTS

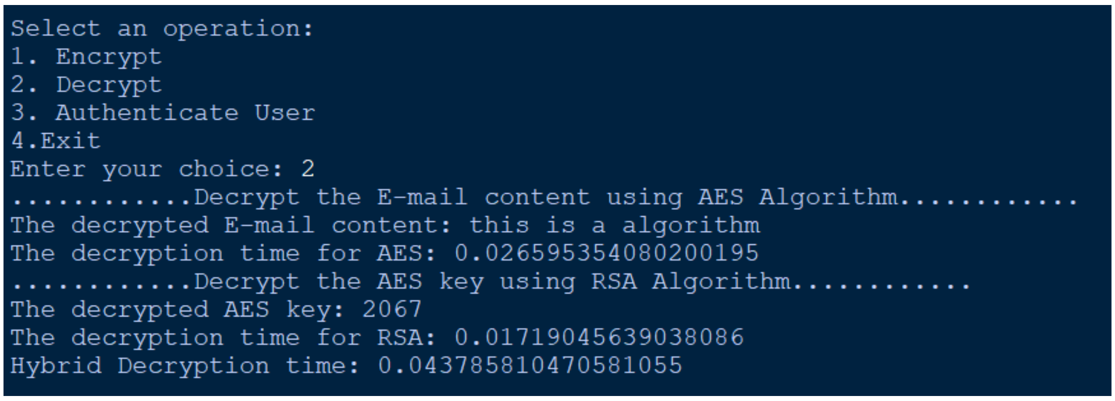
5.1. EXPERIMENT SCREENSHOTS



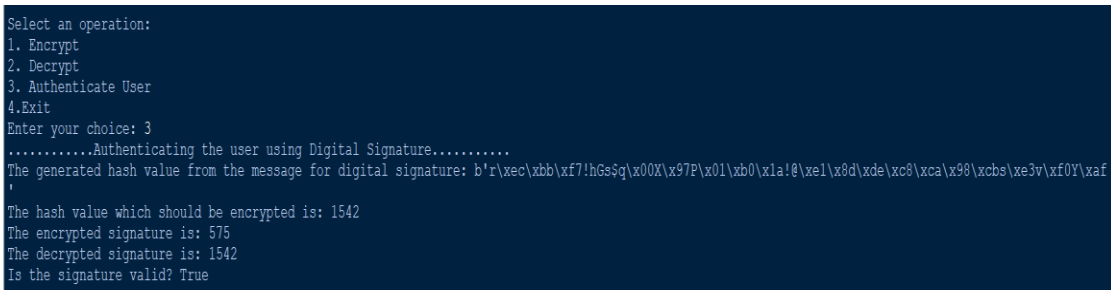
```
..............AES ALGORITHM...............
Enter the AES key: encryption is a key
Enter the E-mail content: this is a algorithm
................RSA ALGORITHM..................
Enter a prime number required to generate public and private keys: 61
Enter a prime number required to generate public and private keys: 53
The public key in RSA is: (7, 3233)
The private key in RSA is: (1783, 3233)
............The Encryption/Decryption process............
```



```
Select an operation:
1. Encrypt
2. Decrypt
3. Authenticate User
4.Exit
Enter your choice: 1
..........Encrypt the E-mail content using AES Algorithm..........
The encrypted E-mail content: JPZae3gHpaxomVSCw4htzxGoT8NOK90y3P3Xlvpr3BVLBTQmEVJJt7VdeUNwdqHR
The encryption time for AES: 0.04144167900085449
..........Encrypt the AES key using RSA Algorithm.............
The AES key is converted to integer for encryption: 2067
The encrypted AES key: 689
The encryption time for RSA: 0.04976987838745117
Hybrid Encryption time: 0.09121155738830566
```



```
Select an operation:
1. Encrypt
2. Decrypt
3. Authenticate User
4.Exit
Enter your choice: 2
............Decrypt the E-mail content using AES Algorithm............
The decrypted E-mail content: this is a algorithm
The decryption time for AES: 0.026595354080200195
............Decrypt the AES key using RSA Algorithm............
The decrypted AES key: 2067
The decryption time for RSA: 0.017190456390380 86
Hybrid Decryption time: 0.0437858104705810 55
```



```
Select an operation:
1. Encrypt
2. Decrypt
3. Authenticate User
4.Exit
Enter your choice: 3
...........Authenticating the user using Digital Signature..........
The generated hash value from the message for digital signature: b'r\xec\xbb\xf7!hGs$q\x00X\x97P\x01\xb0\x1a!@\xe1\x8d\xde\xc8\xca\x98\xcbs\xe3v\xf0Y\xaf
'

The hash value which should be encrypted is: 1542
The encrypted signature is: 575
The decrypted signature is: 1542
Is the signature valid? True
```

5.2. PARAMETERS

5.2.1. Advanced Encryption Standard (AES)

1) **Key Size:** AES supports various key sizes, including 128-bit, 192-bit, and 256-bit keys. Longer keys provide better security, but they also require more computational power.

2) **Security:** AES is a symmetric encryption algorithm, making it highly secure against known attacks. However, the security depends on the key size, and brute force attacks can become feasible with longer key sizes.

3) **Performance:** AES is known for its fast encryption and decryption, making it suitable for bulk data encryption. It is highly efficient in hardware and software implementations.

4) **Bandwidth:** AES encryption and decryption processes do not significantly impact the bandwidth, making it suitable for applications where bandwidth conservation is essential.

5.2.2. Rivest-Shamir-Adleman (RSA)

1) **Key Size:** AES supports various key sizes, including 128-bit, 192-bit, and 256-bit keys. Longer keys provide better security, but they also require more computational power.

2) **Security:** RSA is an asymmetric encryption algorithm that relies on the mathematical difficulty of factoring large composite numbers. It is considered secure if sufficiently large key sizes are used, making it resistant to brute force attacks. However, RSA can be vulnerable to quantum computing attacks.

3) **Performance:** RSA encryption and decryption are slower than AES, especially for larger key sizes. This can make RSA less suitable for encrypting large amounts of data.

4) **Bandwidth:** RSA key exchange and digital signatures can introduce some overhead, especially when using large key sizes, which may affect the available bandwidth.

5.2.3. AES-RSA Hybrid

1) In a hybrid system, AES is typically used to encrypt the data, and RSA is used to encrypt the AES key. The AES key is relatively short (128-256 bits), while the RSA key is larger (2048-4096 bits).

2) The hybrid system combines the security of both AES and RSA. AES provides confidentiality, and RSA provides key exchange and digital signatures. The overall security depends on the key sizes of both algorithms and the specific implementation.

3) The hybrid system typically uses RSA for key exchange and AES for data encryption, combining the strengths of both algorithms. This approach offers a balance between security and performance.

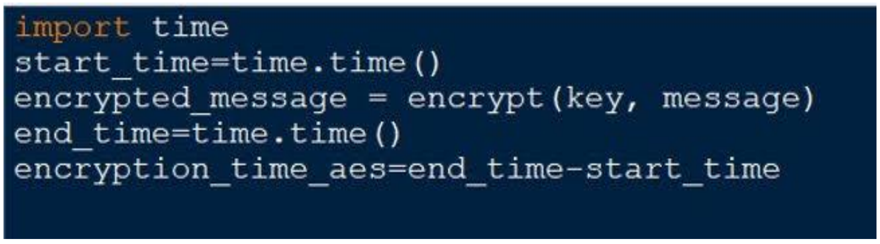
4) The hybrid system's impact on bandwidth will depend on the specific use case. RSA key exchange and AES data encryption can work well together, but the overhead from RSA operations should be considered.

The formulas that are used to compare the AES algorithm, RSA algorithm and Hybrid AES and RSA algorithm.

Encryption Time and Decryption Time

Encryption Time = End Time – Start Time

Decryption Time= End Time – Start Time



```
import time
start_time=time.time()
encrypted_message = encrypt(key, message)
end_time=time.time()
encryption_time_aes=end_time-start_time
```

5.2.4. Security Comparison between AES and Hybrid

To perform security one such parameter is to know upto how many bits security can be provided by an algorithm.

i.e 2^k, where k is key size

ALGORITHM	KEY SIZE	SECURITY PROVIDED
AES	256 bits	2^{256} bits
RSA	2048 bits	2^{2048} bits
HYBRID	2034 bits	2^{2034} bits

Table 5.1 Security Comparison based on key sizes

6. DISCUSSION OF RESULTS

Analysis and comparisons were conducted between AES algorithm, RSA algorithm and the Hybrid AES and RSA algorithm, accessing various parameters such as encryption time, decryption time and key sizes. The experimentation utilized an intel i5 processor with 16 GB main memory and another setup with an i5 processor featuring 8GB of memory. The implementation of these algorithms was carried out using Python IDLE 3.11.

6.1 Encryption time of RSA v/s Hybrid Algorithm

The table below shows the data collected for different message sizes input.

Message size(bits)	RSA Encryption time(in seconds)	Hybrid Encryption time(in seconds)
128	0.04	0.025
256	0.11	0.07
512	0.21	0.15
1024	0.31	0.19
2048	0.49	0.32
5096	0.59	0.38

Table 6.1 Encryption Time for different message sizes

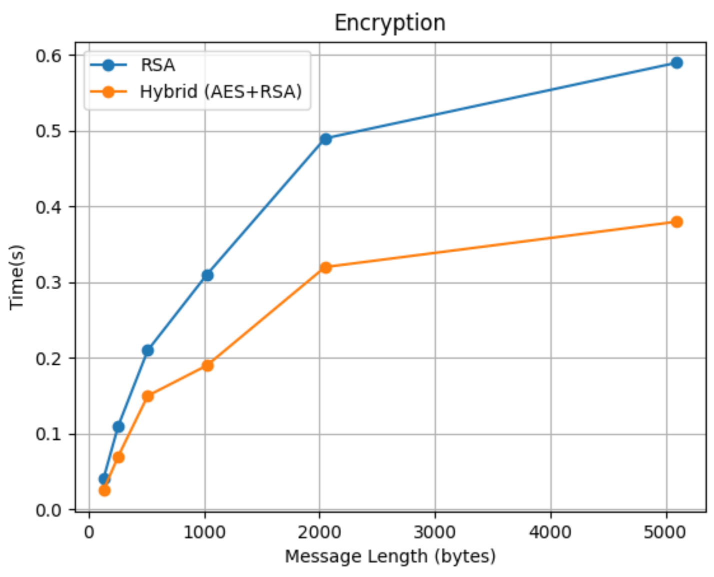


Figure 6.1 Encryption time comparison

6.2. Decryption time of RSA v/s Hybrid Algorithm

Message size (bits)	RSA Encryption time (in seconds)	Hybrid Encryption time (in seconds)
128	0.60	0.51
256	0.86	0.55
512	1.31	0.65
1024	1.76	0.69
2048	2.68	0.81
5096	3.2	0.88

Table 6.2 Decryption time for different message sizes

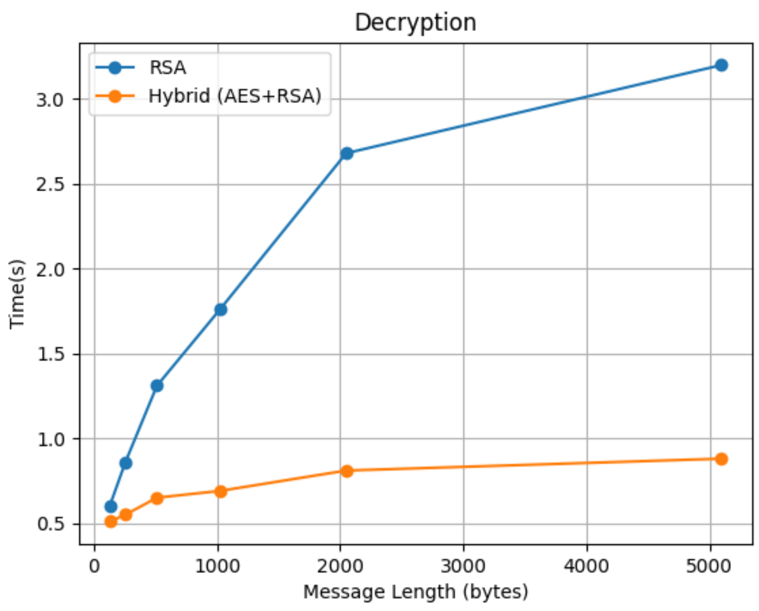


Figure 6.2 Decryption time comparison

6.3. Key length comparison of AES and Hybrid Algorithm

AES algorithm uses key size of 256 bits, where it can provide security upto 2^{256} bits and hybrid algorithm key size is, Key size of AES is 256 and RSA is 2048 bits.

Key Size of Hybrid is 256+2048=2304

Where it can provide security upto 2^{2304} bits.

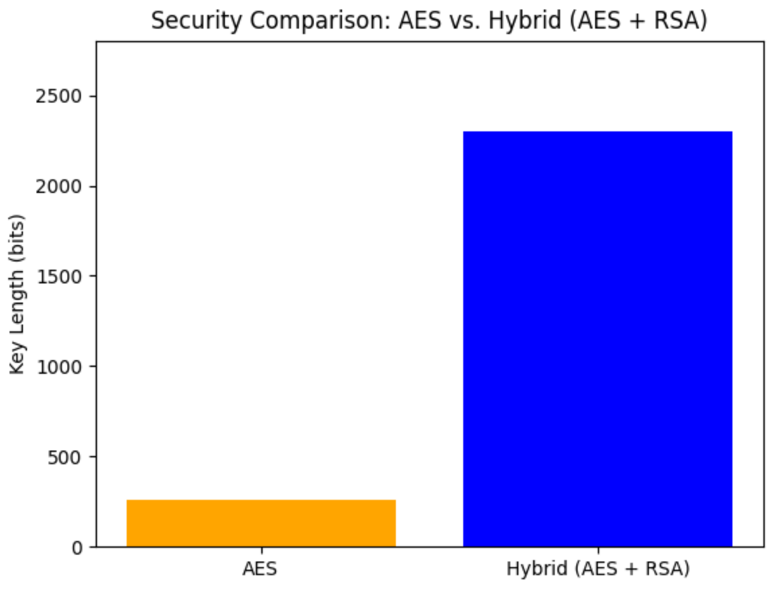


Figure 6.3 Security comparison of AES and Hybrid algorithm

7. CONCLUSION

The implementation of AES encryption within this project enhances security significantly. This encryption method effectively safeguards email contents, rendering it challenging for unauthorized entities to gain access to sensitive information. Beyond security, AES encryption also upholds the principle of confidentiality, guaranteeing that only the intended recipient possesses the means to decrypt and read the email, preserving the privacy of communication.

Furthermore, the project incorporates digital signatures, typically based on the RSA or similar cryptographic methods. This practice not only assures data integrity, ensuring that emails remain unaltered during transmission but also serves as a robust authentication mechanism. RSA-based digital signatures authenticate the sender, unequivocally verifying the message's origin, which instills trust in the communication process. In addition to these security measures, the utilization of AES encryption thwarts eavesdropping attempts by preventing malicious actors from intercepting and comprehending email content. Secure key exchange, facilitated by the RSA algorithm, further bolsters the project's security posture. This method guarantees that both the sender and recipient securely share a secret key, immune to interception. Moreover, the combination of RSA with digital signatures goes a step further in safeguarding against man-in-the-middle attacks. This form of intrusion, where an attacker intercepts and manipulates email content, is effectively mitigated through this dual-layer security approach. In summary, this project embodies a comprehensive strategy for secure email communication. It prioritizes security, privacy, and trustworthiness by employing AES encryption, digital signatures, secure key exchange, and countermeasures against eavesdropping and man-in-the-middle attacks. Overall, it ensures that email communication remains secure, private, and dependable.

8. REFERENCES

[1] Ye Liu, Wei Gong, Wenqing Fan School of Computer Science Community University of China Beijing, China implemented Application AES and RSA Hybrid algorithm in E-mail in IEEE ICIS 2018.06.

[2] Seth, S. M., & Mishra, R. (2011). Comparative analysis of encryption algorithms for data communication 1.

[3] Amrita Sahu, Yogesh Bahendwar, Swati Verma, Prateek Verma, "Proposed Method of Cryptographic Key Generation for Securing Digital Image", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012.

[4] Kakkar, A., Singh, M. L., & Bansal, P. K. (2012). Comparison of Various Encryption Algorithms and Techniques for Secured Data Communication. In in Multinode Network", International Journal of Engineering and Technology Volume.

[5] Alanazi, H., Zaidan, B. B., Zaidan, A. A., Jalab, H. A., Shabbir, M., & Al-Nabhani, Y. (2010). New comparative study between DES, 3DES and AES within nine factors. arXiv preprint arXiv:1003.4085.

[6] S. Pavithra, E. Ramadevi, (2012), Performance Evaluation of Symmetric Algorithms.

[7] Luo, Z., Shen, K., Hu, R., Yang, Y., & Deng, R. (2022). Optimization of AES-128 Encryption Algorithm for Security Layer in ZigBee Networking of Internet of Things. Computational Intelligence and Neuroscience.

[8] Arora, P., Singh, A., & Tyagi, H. (2012). Evaluation and comparison of security issues on cloud computing environment. World of Computer Science and Information Technology Journal (WCSIT), 2(5), 179-183.

[9] Mandal, A. K., Parakash, C., & Tiwari, A. (2012, March). Performance evaluation of cryptographic algorithms: DES and AES. In 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science (pp. 1-5). IEEE.

[10] M. Koko, A. Babiker, and N. Mustafa, "Comparison of Various Encryption Algorithms and Techniques for improving secured data Communication," IOSR J. Comput. Eng. Ver. III, vol. 17, no. 1, pp. 2278–661, doi: 10.9790/0661-17136269.