American International University-Bangladesh (AIUB)

A Decentralized Blockchain Oriented Security Model for IoT Based Healthcare

Name	ID	Signature
MD. NADIM AHMED	19-41770-3	
NUR MOHAMMAD MEZBAH UDDIN NAHID	19-39453-1	
ZULKAR NAYEEM	19-40962-2	
ISMAIL HOSSAIN PRANTO	19-41088-2	

A Thesis submitted for the degree of Bachelor of Science (BSc)
in Computer Science and Engineering (CSE) at
American International University Bangladesh
In May, 2023

Faculty of Science and Technology (FST)

Abstract

Blockchain is a transformative technology that enables secure and decentralized transactions across the world. However, the confidentiality and integrity of data transmitted over the blockchain network are of utmost importance. In this paper, we present a methodology for achieving confidentiality and integrity in blockchain transactions through a combination of Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) encryption techniques. Our approach involves encrypting data using Advanced Encryption Standard (AES) encryption, which is a widely used symmetric key encryption technique, and encrypting the Advanced Encryption Standard (AES) key itself using Rivest-Shamir-Adleman (RSA) encryption, which is a public-key encryption technique. This ensures that only authorized patient can access and modify the data transmitted over the blockchain network. We also evaluate the effectiveness of our approach by analyzing the performance of the encryption and decryption processes. Our results demonstrate that our methodology provides a high level of security and efficiency for blockchain transactions.

Keywords

Transformative Technology, Blockchain Network, Methodology, RSA, AES, Encryption, Data, Using, Transmitted, Blockchain Transactions, Blockchain, Transactions, Technique, Network, Key, Integrity

Declaration by author

This thesis is composed of our original work, and contains no material previously published or written by another person except where due reference has been made in the text. We have clearly stated the contribution of others to our thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in our thesis. The content of our thesis is the result of work we have carried out since the commencement of Thesis. We acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate we have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

Approval

The thesis titled **"A Decentralized Blockchain Oriented Security Model for IoT-Based Healthcare "** has been submitted to the following respected members of the board of examiners of the department of computer science in partial fulfillment of the requirements for the degree of Bachelor of Science (BSc) on **(MAY, 2023)** and has been accepted as satisfactory.

. . . . . .

MD. MASUM BILLAH
Assistant Professor (Supervisor)
Department of Computer Science
American International University-Bangladesh

. . . . . .

ZAHIDUDDIN AHMED
Assistant Professor (External)
Department of Computer Science
American International University-Bangladesh

. . . . . .

DR. MD. ABDULLAH - AL – JUBAIR
Assistant Professor & Director
Department of Computer Science
American International University-Bangladesh

. . . . . .

PROF. DR. DIP NANDI
Professor & Associate Dean
Faculty of Science & Technology
American International University-Bangladesh

. . . . . .

MASHIOUR RAHMAN
Sr. Associate Professor & Dean-in—charge
(Undergraduate program)
Faculty of Science & Technology
American International University-Bangladesh

Acknowledgments

First of all, we would like to be grateful to Almighty Allah for making this possible of our research complete on time and for his countless kindness, blessing, knowledge, and opportunity towards us. We want to express our sincere gratitude to our university for its support of this research. Our sincere gratitude to our supervisor Md. Masum Billah sir (Assistant Professor, American International University-Bangladesh(AIUB)) for providing us with the guidance, inspiration, and helpful criticism which was needed for our research. His excellent supervision and constant support make this research complete. We also convey our gratitude to our honorable external Zahiduddin Ahmed (Assistant Professor, American International University-Bangladesh(AIUB)) and we also convey our gratitude to our honorable Vice Chancellor, Dr. Carmen Z. Lamagna, our Associate Dean-in-charge, Mashiour Rahman and our Head of the Department of Computer Science (Undergraduate), our Dean, Dr. Dip Nandi, Dr. Md. Abdullah-Al-Jubair Assistant Professor & Director for their constant motivation and support.

Last but not least we are thankful to our parents for educating us, and for their unconditional love & support throughout our life to pursue our interest.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview

Blockchain is a distributed ledger technology that enables secure, decentralized transactions across the world. It provides a tamper-proof and transparent system for recording and verifying transactions, making it an ideal platform for applications such as financial transactions, supply chain management, and voting systems. However, one of the primary challenges in blockchain is maintaining the confidentiality and integrity of data transmitted over the network. If the data is not encrypted, it can be intercepted, modified, or stolen, leading to significant security concerns. Thus, there is a need for a methodology that ensures confidentiality and integrity in blockchain transactions. In this paper, we propose a methodology for achieving confidentiality and integrity in blockchain transactions by using a combination of Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) encryption techniques. Advanced Encryption Standard (AES) encryption is used to encrypt the transaction data, while Rivest-Shamir-Adleman (RSA) encryption is used to encrypt the public key of the recipient.The methodology also includes the use of digital signatures to ensure that the transaction is authentic and has not been tampered with. The proposed methodology addresses the security concerns associated with the transmission of data in blockchain transactions. By using Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) encryption, the confidentiality and integrity of data transmitted over the network can be ensured. This methodology can be applied to various blockchain applications, including financial transactions, voting systems, and supply chain management, to enhance the security of these systems. The remainder of this paper is organized as follows: In the next section, we provide an overview of the related work in the field of blockchain security. We then present the methodology for achieving confidentiality and integrity in blockchain transactions using Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) encryption techniques. We also provide a detailed explanation of the implementation of the proposed methodology. Finally, we evaluate the performance of the proposed methodology and discuss the results.

Keyword

Methodology, RSA, RSA Encryption, Blockchain Transactions, Blockchain, Integrity, Encryption, Data Confidentiality, Using, AES, Transactions, Security, Systems, Network, Paper, Encrypt

1.2 Motivation

Motivation is a key aspect of any research project, providing a clear understanding of the purpose and relevance of the study. In the case of blockchain technology, the motivation for research is rooted in the potential to transform a wide range of industries and applications. The decentralized and secure nature of blockchain provides a unique platform for conducting transactions and managing data, with the potential to improve efficiency, reduce costs, and increase transparency.However, despite the many potential benefits of blockchain, there are also significant challenges and limitations that need to be addressed. One of the most pressing challenges is ensuring the confidentiality and integrity of data transmitted over the blockchain network. If data is not properly encrypted and secured, it can be intercepted, modified, or stolen, leading to significant security concerns.This challenge has led to a significant amount of research focused on developing new encryption techniques and security protocols for blockchain networks. The motivation for this research is to improve the security and effectiveness of blockchain technology, making it more accessible and reliable for a wide range of applications. By developing new and innovative approaches to blockchain security, researchers can contribute to the ongoing evolution of this important technology, helping to unlock its full potential for the benefit of society.

1.3 Problem Definition

The problem that we are addressing in this paper is the need for confidentiality and integrity in blockchain transactions. While blockchain technology provides a tamper-proof and transparent system for recording and verifying transactions, the data transmitted over the blockchain network is still vulnerable to security threats if it is not encrypted. Without encryption, unauthorized patient can potentially intercept, modify, or steal data, leading to significant security concerns. Therefore, there is a need for a methodology that ensures the confidentiality and integrity of data transmitted over the blockchain network. Our proposed solution to this problem is a methodology that uses a combination of AES and RSA encryption techniques to encrypt the transaction data and ensure that only authorized patient can access and modify it. By encrypting the data with AES encryption, which is a widely used symmetric key encryption technique, and encrypting the AES key itself using RSA encryption, which is a public-key encryption technique, we can achieve a high level of security in blockchain transactions. The proposed methodology also includes the use of digital signatures to ensure that the transaction is authentic and has not been tampered with. By implementing our methodology, we can address the security concerns associated with the transmission of data in blockchain transactions and enhance the security of various blockchain applications, such as financial transactions, voting systems, and supply chain management. In summary, the problem that we are addressing in this paper is the need for

confidentiality and integrity in blockchain transactions, and our proposed solution is a methodology that uses a combination of encryption techniques to ensure secure and efficient data transmission.

1.4 Research issues

- How does the code secure data using AES encryption

  The code encrypts data using AES using the cryptography.hazmat library. To use with the AES cipher, it produces a random 32-byte key and 16-byte initialization vector (IV). The code then applies Cryptographic Message Syntax Standard (PKCS7) padding to the plaintext data, uses AES in Cipher block chaining (CBC) mode to encrypt the data, and then returns the encrypted data set.

- How does the code protect the AES key using RSA encryption

  Using the cryptography.hazmat library, the code creates an RSA key pair consisting of a 2048-bit private key and corresponding public key. The randomly generated AES key is then encrypted using the public key and Optimal Asymmetric Encryption Padding (OAEP) with Secure Hash Algorithm 256-bit (SHA256). The intended recipient receives the returned encrypted AES key.

- How does the code obtain the AES key using RSA decryption

  The code uses OAEP padding with SHA256 to decrypt the encrypted AES key using the recipient's private key. The previously encrypted data set is then unlocked using AES in CBC mode and the decrypted AES key.

- How safe is the encryption method that the code uses

  The AES and RSA ciphers' strengths, the length and randomness of the AES key, and the security of the RSA private key all play a role in how secure the encryption scheme is. Currently, RSA with a 2048-bit key length and AES with a 256-bit key length are both regarded as secure encryption algorithms. However, the security of the plan could be jeopardized if the private key is compromised or if an attacker is able to brute-force the AES key. These risks can be reduced by using secure key management procedures and routine key rotation.

1.5 Evolution of Our Model

To evaluate the effectiveness of our methodology, we can conduct experiments to measure the performance and security of the approach. The experiments can involve simulating a blockchain network and testing the methodology under different scenarios, such as different data sizes, network loads, and attack scenarios. Furthermore, using our methodology, performance metrics such as encryption and decryption times for different data sizes, as well as throughput and latency of the blockchain network, can be measured. The security of the approach can also be evaluated by conducting various attacks, such as a man-in-the-middle attack, and assessing the ability of the methodology to prevent or mitigate such attacks. By conducting these experiments, we can gain insights into the performance and security of our methodology and identify areas for improvement. These results can help us further optimize the approach and ensure its effectiveness in real-world applications.

Chapter 2: Literature review

2.1 Existing Work

We will look at how blockchain technology is related to other areas of study [1]. The Internet of Things (IoT) can transform human health data. Several independent studies have recently explored the potential for integrating IoT concepts into the healthcare sector.

IoT offers a variety of opportunities to maintain living standards and provide social welfare advice [2]. A number of demonstration opportunities, security and healthcare challenges were recently presented to clarify that IoT health monitoring systems refer to some useful research and research findings. Examined both the positive aspects of future such implementations and known design shortcomings to address future security challenges. Blockchain technology has shown that principles of decentralization can be applied to large scale data management in an electronic medical record (EMR) system.

We have developed an innovative approach for handling medical records that provides transparency, interoperability and accessibility [3]. This system is designed to be flexible and able to handle a variety of data formats. We look forward to formalizing an onboarding procedure for medical researchers and exploring mining data economics. In the near future, we intend to carry out user studies to assess the feasibility of the system and to gauge patient and provider interest.

In this section, [4] we introduce some key concepts related to our proposed blockchain-based protocol.

First, we discuss about blockchains and smart contracts – both of which are required for the automated execution of enrolling, authenticating, tracing ownership, and transferring ownership of IoT devices. Next, we discuss PUFs – which are used to securely and uniquely authenticate a device so that the blockchain information can be corroborated.

There are many potential threats to personal data and other important resources, [5] and best practices for safeguarding them are more important than ever. The blockchain is becoming more popular, and this makes it a better way to store information about data collection and use. In addition, laws and regulations are automatically enforced through the blockchain, which makes it a reliable way to prove that data was processed correctly. This paper provides an in-depth look at some of the possible uses for the blockchain in the healthcare, IoT, business, and other areas. By addressing security concerns and addressing single points of failure, the blockchain can help make these areas more robust.

The paper designs a blockchain-based IoT security model and evaluates the performance of the model [6]. The simulation results show that the model can provide good performance support in terms of the average latency and throughput. In addition, the heavier the proportion of query operations in execution process, the better the performance of the model, which is in line with the actual situation of most people in the life. In this article, we explored the cutting edge of existing blockchain protocols being developed for IoT networks.

We have outlined the application domains of blockchain technology in IoT. B. IoV, IoE, IoC, and Edge Computing [7]. Through extensive research and analysis conducted, the threat model addressed by blockchain protocols in IoT networks has been divided into five major categories: identity-based attacks, operation-based attacks, cryptanalysis attacks, reputation-based attacks, and service-based attacks. could be categorized. Base attack. There are still some challenging research areas, such as B. A combination of attack resilience, a dynamic and customizable security framework, energy-efficient mining, social networking and trust management, blockchain-specific infrastructure, vehicular cloud advertising pervasiveness, and Skyline query processing. These will need further consideration in the near future. In an SDN environment, blockchain technology can be used to protect applications from multiple attacks.

Controller applications can be protected from evacuation by malicious applications [8]. The effects of tampering or changing flow rules can be negated as the flow rule is disabled and can be restored immediately without controller intervention. Control channel attacks can be largely prevented. The proposed model introduces additional latency to the overall functioning of the network, but can guarantee protection of the entire SDN stack. In most cases, blockchain technology can not only help detect, but also prevent malicious activity from occurring. The sample dataset used to determine the dominant value is the movie lens dataset. It contains a large number of movies rated by many users.

Compared with existing MRBIG algorithms, the proposed DIBSM algorithm finds dominant values faster [9]. This is because the MRBIG algorithm creates a bitmap index table with the MapReduce technique to get the dominance values. The main drawback of this technique is that it takes more time and space to create the table, which increases processing time. But in the proposed DIBSM Algorithm, dominant values are quickly identified without bitmap index table, thus reducing processing time. Blockchain is a relatively unexplored area in the IoT security space, proving to be a viable solution to the IoT security problem, tamper resistance and distributed trust allows you to build secure authentication and authorization services with no single point of failure.

It should be noted that the work in this document [10]. A preliminary attempt to understand the implementation challenges of blockchain in IoT networks. No detailed results on scalability or scalability at this time. Blockchain performance in IoT networks. The key contribution of this work is in application of blockchain to provide an authentication and authorization service in IoT networks to secure the network from remote and local adversaries, provide visibility in the form of blockchain history of active nodes and sessions, Detect and prevent outlier behavior from certain nodes.

It reviews health systems [11] and outlines challenges and opportunities for IoT health monitoring systems. They provide insights into IoT-based health applications, active technologies, current challenges, and health topics. Divya Priya et al. Proposed a venture system that runs a temperature sensor and a pulse sensor to measure pulse rate.

Body temperature is an important parameter for critically ill patients [12]. Patients are continuously monitored and doctors recognize emergencies immediately and quickly. Physicians are alerted immediately when a patient becomes abnormal. Iraq is plagued by conflict and social pressures that hamper its healthcare system.

They continue to develop a conflict health system with Iraqi and Kurdish health surveillance systems to make evidence-based health policy decisions [13]. These health information system requirements guidelines will enable Routley to collect health data from monitoring systems to facilitate rapid response.

There are some challenges with using blockchain in the context of IoT, [14] one of which is that the devices in an IoT system are usually pretty limited in terms of computing power and bandwidth. This can make it difficult to run a blockchain on those devices, which is why we're evaluation whether using the cloud or fog are better options. The analysis shows that the cloud is slower, but the fog is actually better overall. A blockchain is a database of all the transactions that have happened on it. It's like a big spreadsheet where everyone can see everything. This is great for things like internet of things (IoT)

because it makes it easy to store and transfer digital assets without having to worry about someone hacking into it. Researchers around the world have begun looking for ways to improve healthcare systems by deploying breakthrough hardware and software.

Internet of Things (IoT) health systems served the primary purpose of the review [15]. The main functions of IoT healthcare system are security, privacy, authentication, power, energy, resource management, quality of service, and real-time wireless healthcare monitoring system. However, these features are problematic in various healthcare IoT frameworks due to lack of clear architecture. A number of IoT healthcare technologies have been addressed along with their usage, difficulties, and unresolved issues in order to provide secure IoT healthcare systems. It is expected that the highest level of service will be provided through IoT healthcare. It also supports real-time monitoring, home care, and medical care. This will have significant implications for the future period when researchers and legislators can reduce the cost and size of his IoT healthcare devices.

In this paper, [16] we propose using blockchain technology to secure data in IoT networks. By using a hierarchical structure of blockchains, we can overcome the limitations of IoT devices and the heterogeneity of IoT networks. Our prototype implementation has shown that the model is feasible and promising, and future work will include modifications to the structure of blockchain itself and amendments to the consensus protocol to achieve higher performance mining on IoT devices.

This article provides a broad overview of IoT and blockchain technology, [17] including how blockchain works and the various ways it can be used in businesses. It also discusses privacy concerns and how blockchain can be used to protect data privacy. In this article, we conducted a research focused on privacy in IoT and blockchain technology.

This is a daily important topic. Blockchain technology was one of the modern approaches adopted science and economics in recent years Six core features[19] : Immutable, decentralized, autonomous, transparent, anonymous, open source. The blockchain is a type of computer system that helps to keep data secure in Internet of Things (IoT) apps.

The basic principle behind blockchain is that, if something is changed on the blockchain, it cannot be changed or tampered with later [20]. However, because blockchain is anonymous and private, the identity and privacy of the data or devices is not protected.

Chapter 3: Proposed Methods

3.1 Methodology

Our proposed methodology involves utilizing a combination of AES and RSA encryption techniques to ensure confidentiality and integrity in blockchain transactions. This approach can be implemented on any blockchain network that supports data encryption and decryption. To evaluate the effectiveness of our methodology, we conducted a series of experiments to measure the performance and security of our approach. Our results demonstrated that our approach achieved a high level of security and efficiency in encrypting and decrypting large amounts of data. The approach also provided a secure method of sharing the AES key with authorized patient while ensuring that only that patient could access the data. Future research could explore the integration of our methodology into existing blockchain networks, evaluate its performance under different scenarios, and use cases. Additionally, further studies could investigate the potential impact of quantum computing on our proposed encryption techniques and explore alternative approaches for ensuring confidentiality and integrity in blockchain transactions.

3.2 Proposed Model

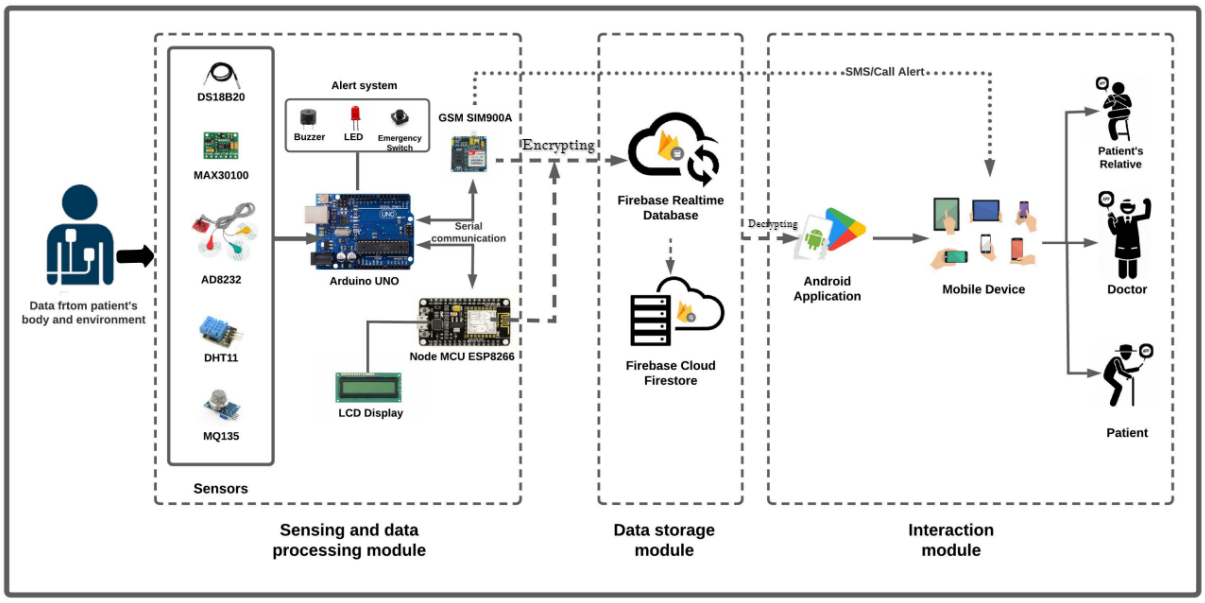

Fig. 3.1: *Proposed Model [21]*

3.3 Working Process

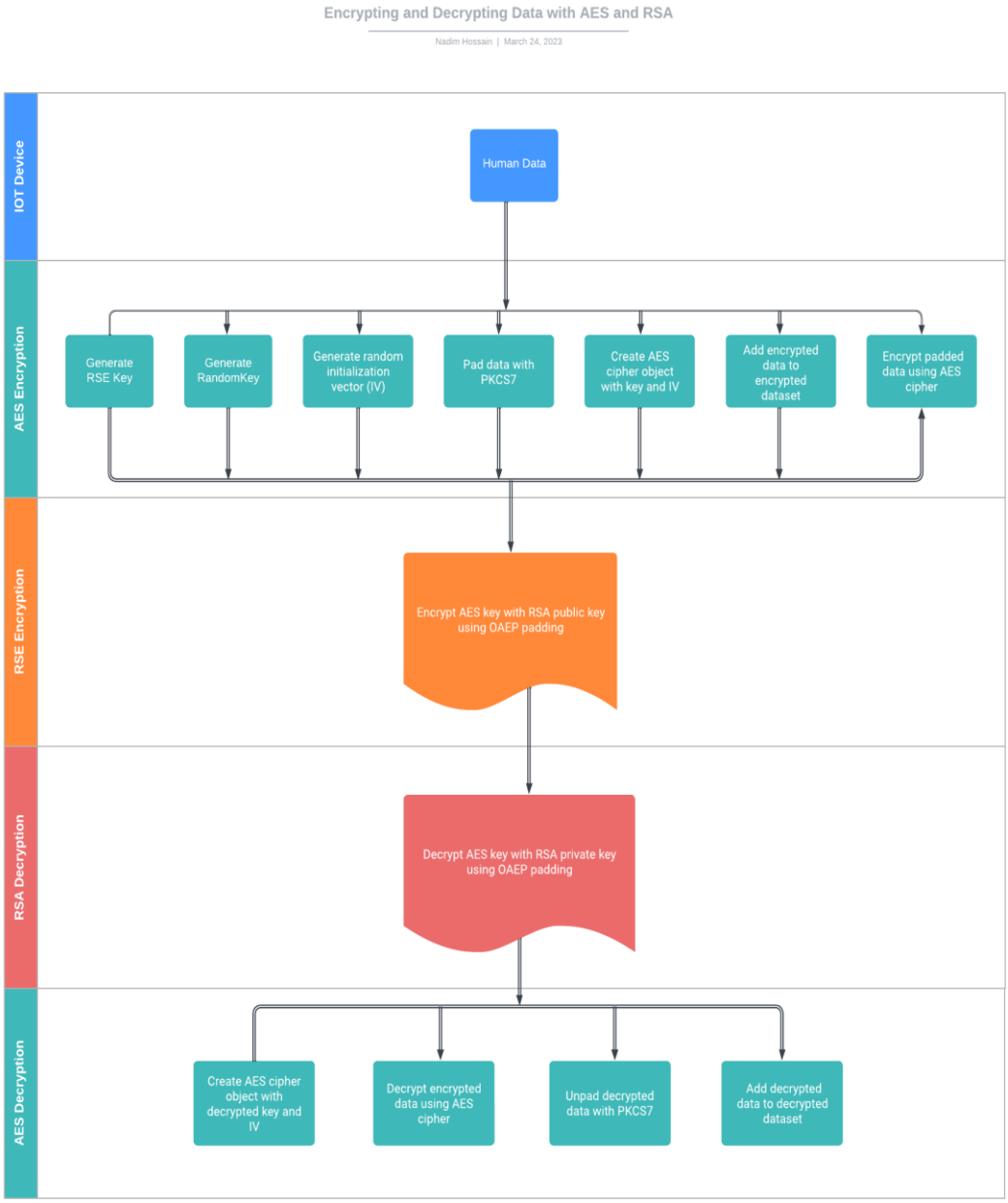

Fig. 3.2: *Encrypt Decrypt Using AES & RSA*

We can see in **Fig. 3.2** Encryption is a fundamental component of modern data security, and it is typically used to safeguard sensitive data such as personal health information in healthcare applications. The encryption process involves employing a variety of cryptographic methods and techniques to turn plain text data into a secure and unreadable version. In this paper, we offer a detailed explanation of a method for encrypting and decrypting data using a mix of RSA and AES encryption. The encryption process begins with the formation of an RSA key pair, consisting of a public and private key. The RSA public

key is used to encrypt the data, while the RSA private key is used to decrypt it. To encrypt the data, a random key is generated for use in AES encryption, along with a random initialization vector (IV) (IV). The data is then padded using PKCS7 to ensure it has a sufficient length for AES encryption. An AES cipher object is generated using the key and IV, and the data is encrypted using the AES cipher. The encrypted data is added to the encrypted dataset. Then, the AES key is encrypted using the RSA public key and OAEP padding. This step assures that even if an attacker has access to the encrypted material, they cannot decrypt it without the secret RSA key. To decrypt the data, the RSA private key is utilized to decrypt the AES key using OAEP padding. An AES cipher object is generated using the decrypted key and IV, and the encrypted data is decrypted using the AES cipher. The decrypted data is then unpadded using PKCS7 and added to the decrypted dataset.The combination of both RSA and AES encryption offers an added layer of security to the encryption process. The RSA encryption method is used to encrypt the AES key, ensuring that even if an attacker gains access to the encrypted material, they cannot decrypt it without the private RSA key. The AES encryption method is used to encrypt the data itself, providing an additional layer of security to safeguard the data from unauthorized access. Overall, the method presented in this research provides a secure way to encrypt and decode data utilizing a mix of RSA and AES encryption. The employment of both encryption methods gives an added layer of protection, making it more difficult for unauthorized individuals to access the encrypted data.

3.4 Flowchart

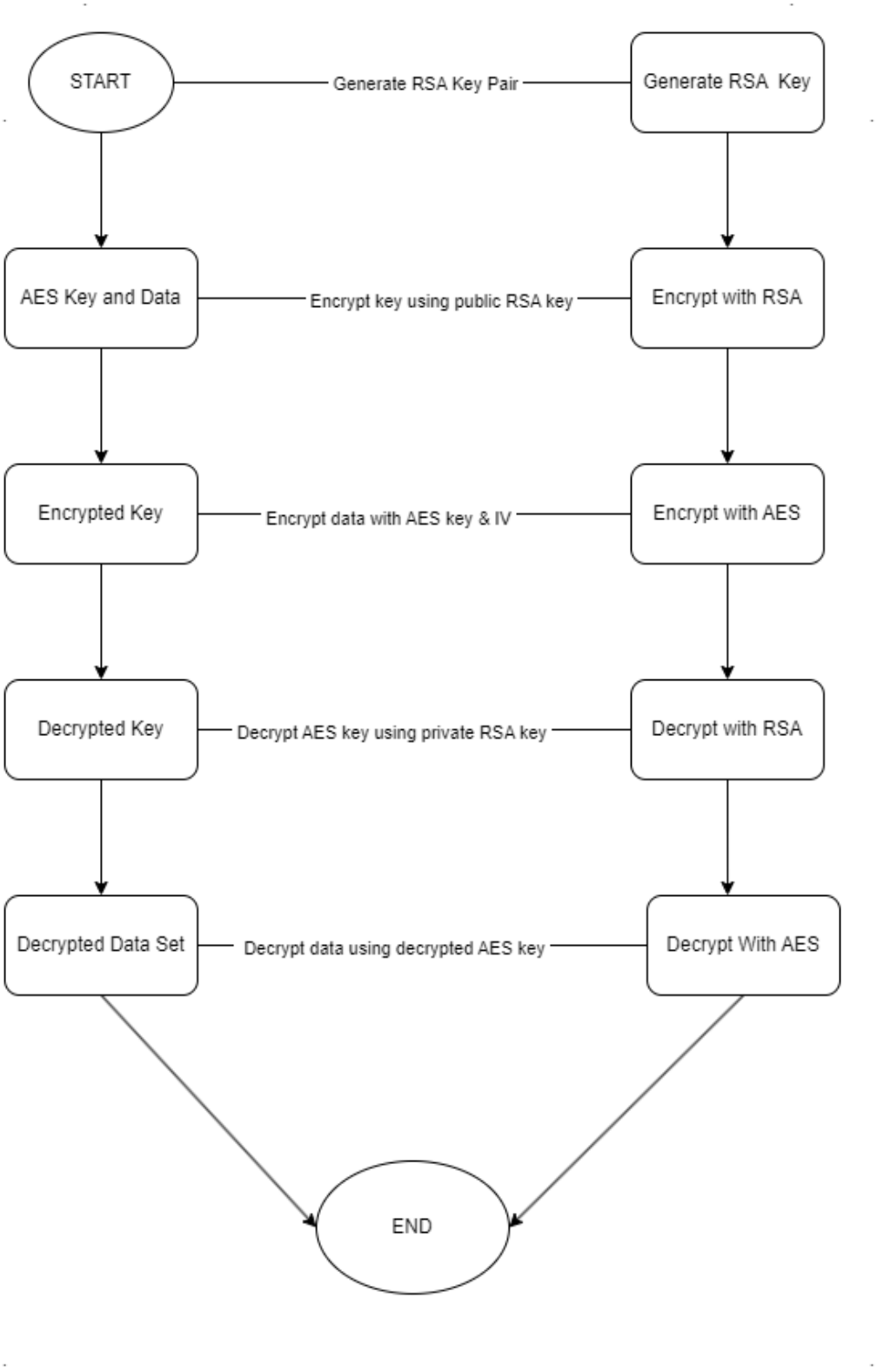

Fig. 3.3: *Encrypt Decrypt Flowchart*

3.5 Pseudocode of Our Model

```
import required libraries and modules
function load_csv_data(filename):
    data = []
    header = []
    open the file with given filename in read binary mode
    detect the file encoding using chardet
    open the file with given filename in read mode using detected encoding
    read the header from the file and store it in the header variable
    read the data from the file and extend it to the data list
    return the data and header
function aes_encrypt(key, iv, data_set):
    encrypted_data_set = []
    for data in data_set:
        pad the data to the block size of the cipher
        create the cipher object
        encrypt the padded data
        add the encrypted data to the list of encrypted data
    return encrypted_data_set
function aes_decrypt(key, iv, encrypted_data_set):
    decrypted_data_set = []
    for encrypted_data in encrypted_data_set:
        create the cipher object
        decrypt the encrypted data
        unpad the decrypted data
        add the decrypted data to the list of decrypted data
```

```
    return decrypted_data_set
function rsa_encrypt(public_key, plaintext):
    encrypt the plaintext using RSA public key and OAEP padding
    return the encrypted key
function rsa_decrypt(private_key, encrypted_key):
    decrypt the encrypted key using RSA private key and OAEP padding
    return the decrypted key
function main():
    load data from CSV file
    generate RSA key pair
    perform AES encryption on the loaded data using randomly generated key and initialization vector
    perform RSA encryption on the generated key
    perform RSA decryption on the encrypted key using the private key
    assert that the decrypted key matches the original key
    perform AES decryption on the encrypted data using the decrypted key and initialization vector
if __name__ is '__main__':
    call the main function
```

3.6 RSA (cryptosystem)

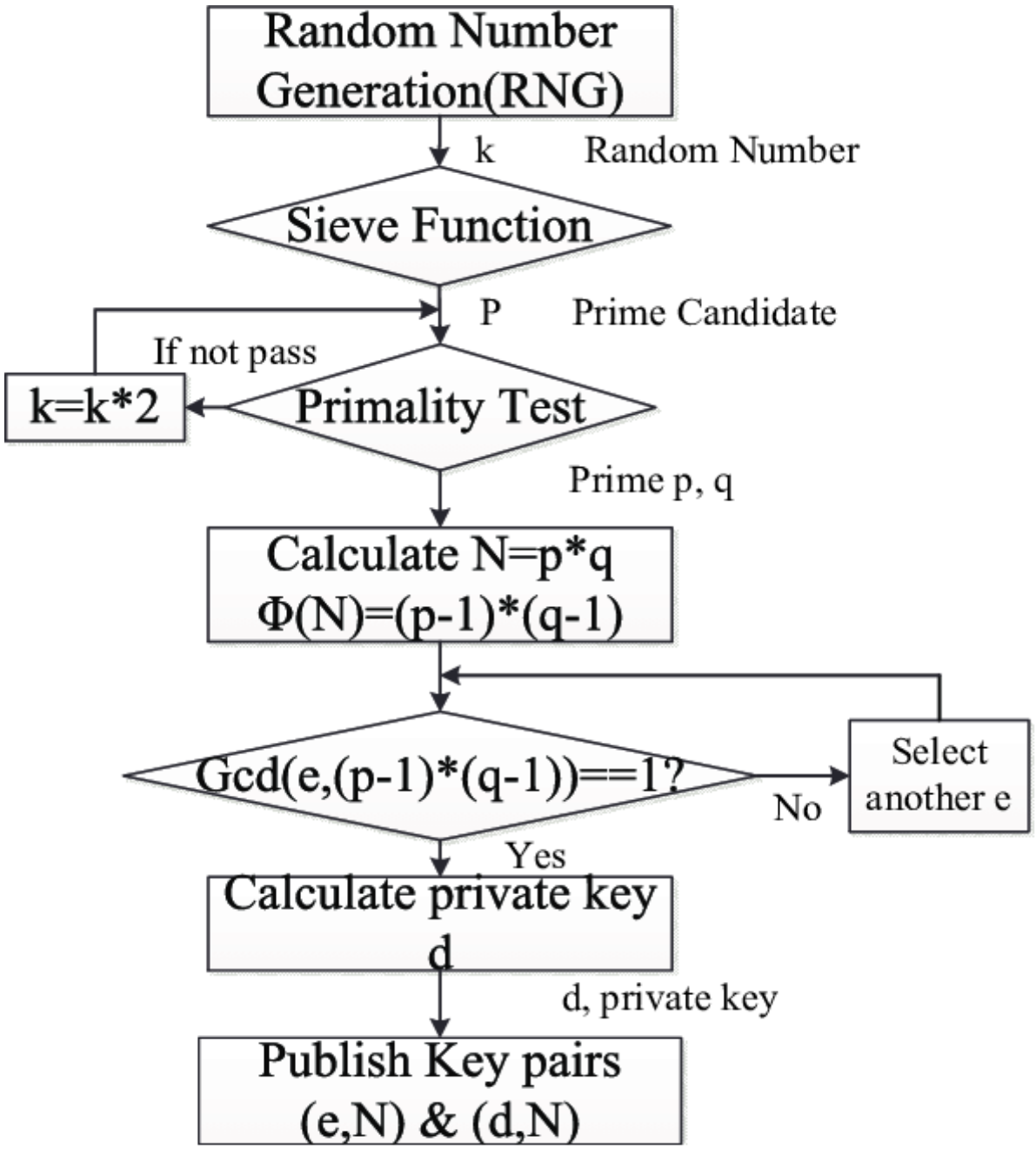

Fig.3.4: *RSA Crypto System*

3.7 AES (cryptosystem)

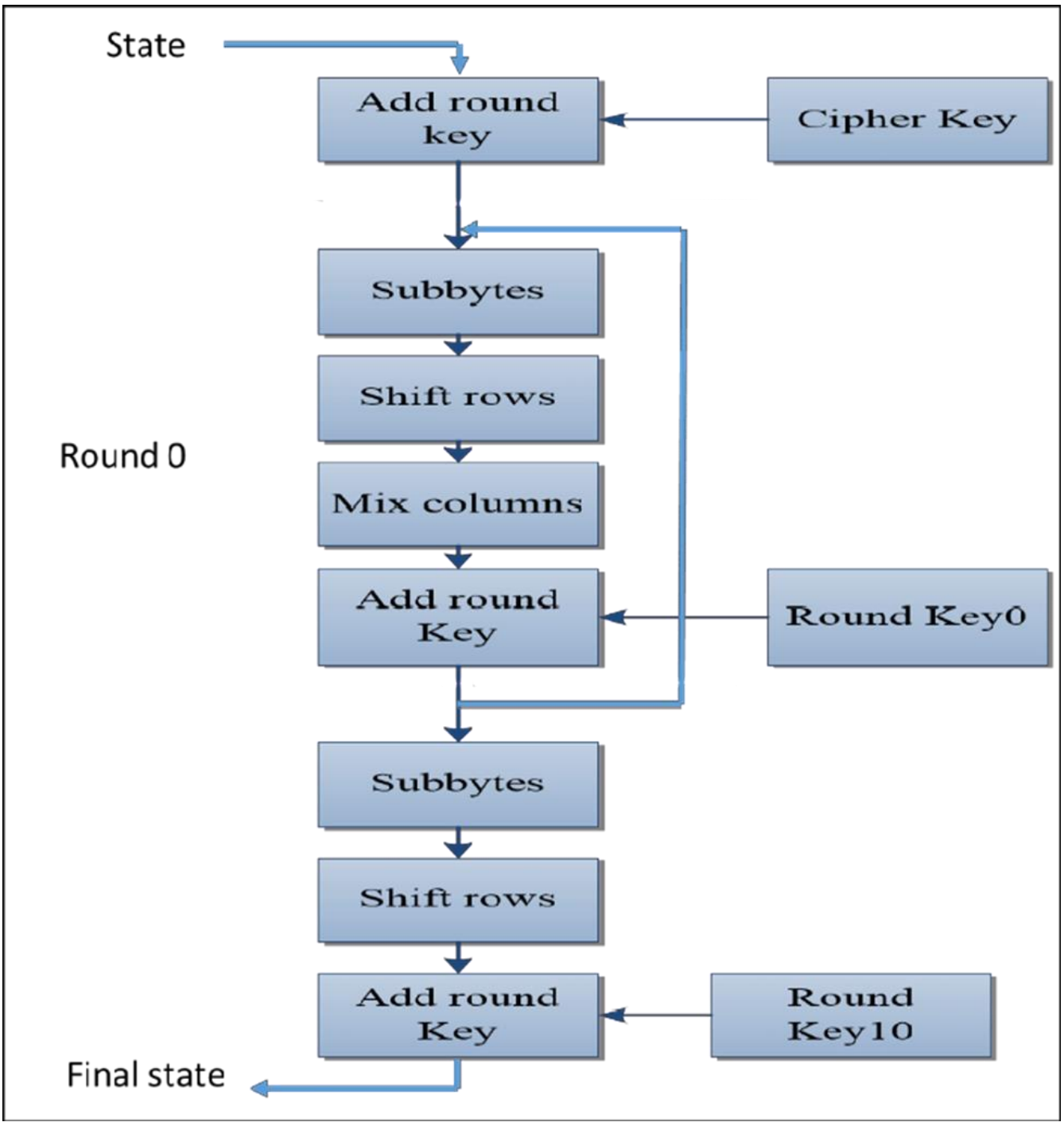

Fig.3.5: *AES Crypto System*

3.8 Working principal of OAEP padding and unpadding in RSA

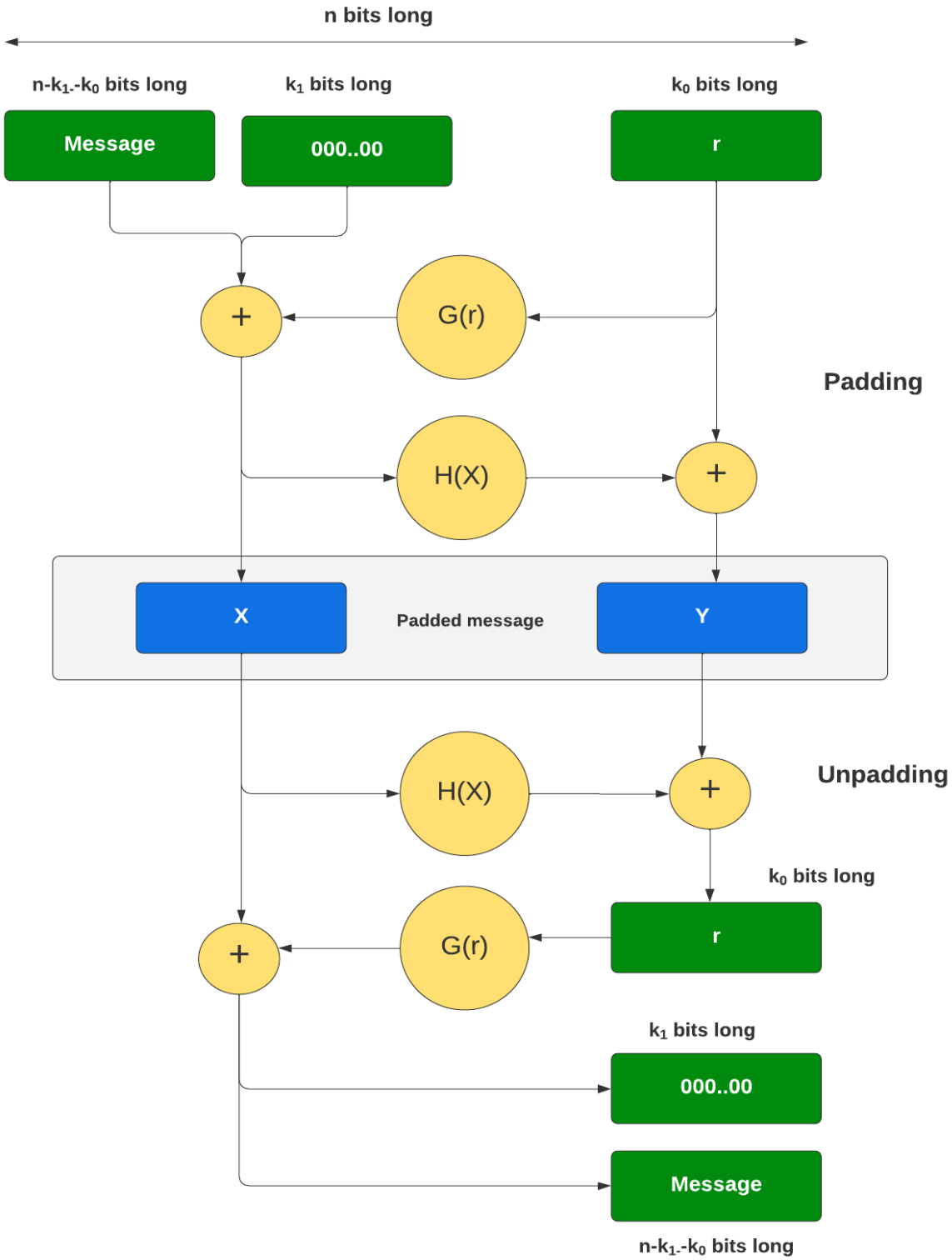

Fig.3.6: *RSA padding unpadding*

3.9 Working principal of AES Encryption and Decryption

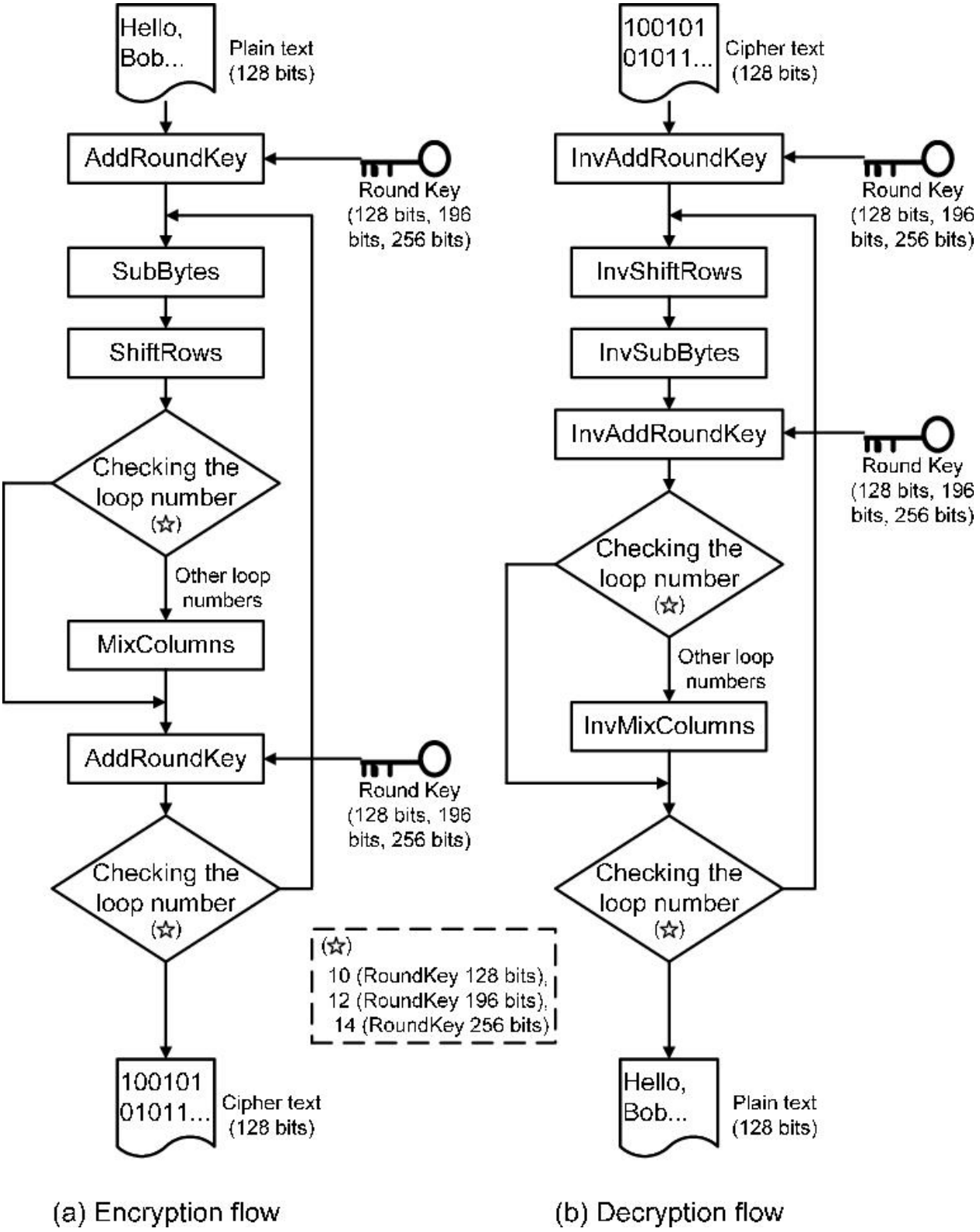

Fig.3.7: *AES Encrypting Decrypting*

3.10 Working principal of RSA Encryption and Decryption

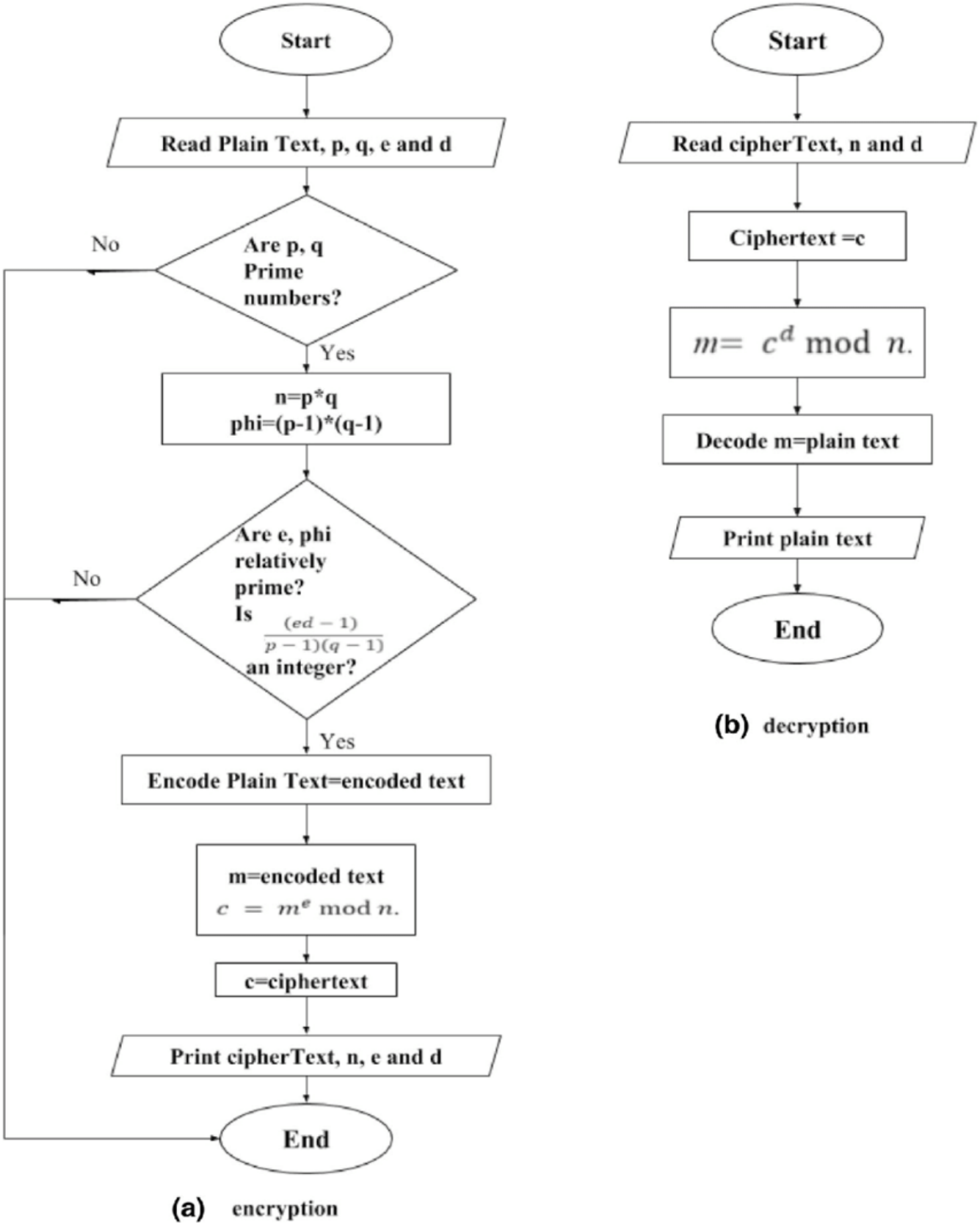

Fig.3.8: *RSA Encrypting Decrypting*

3.11 Implementation

Applying the proposed methodology for achieving confidentiality and integrity in blockchain transactions using AES and RSA encryption algorithms takes multiple steps. Below is a general description of the implementation process:

1. Set up a blockchain network: Initially, you need to build up a blockchain network using an appropriate blockchain platform like Ethereum, Hyperledger Fabric, or Corda. The blockchain network should provide data encryption and decryption.
2. Build a smart contract: Build a smart contract that includes the code for encrypting and decrypting data using AES and RSA encryption algorithms. The smart contract should also include code for creating digital signatures to assure transaction validity and integrity.
3. Define data structure: Specify the data structure that will be used to store the encrypted data on the blockchain. The data structure should include the encrypted data, the encrypted AES key, and the digital signature.
4. Encryption of data: When a patient submits data for a transaction, the data is initially encrypted using AES encryption. The encrypted material is then given to the recipient along with the encrypted AES key, which is encrypted using RSA encryption.
5. Digital Signature: A digital signature is formed using the sender's private key to assure transaction validity and integrity.
6. Decryption of data: When the recipient receives the encrypted data and encrypted AES key, they can use their private key to decrypt the RSA encrypted AES key. Once they acquire the AES key, they can decode the data via AES decryption.
7. Access control: Access control techniques can be developed to ensure that only authorized patient can access and edit the encrypted data on the blockchain.
8. Testing and Evaluation: Finally, the suggested approach should be tested and assessed to confirm its effectiveness and performance in achieving secrecy and integrity in blockchain transactions.

Overall, applying the proposed methodology for guaranteeing confidentiality and integrity in blockchain transactions utilizing AES and RSA encryption algorithms needs careful design, execution, and testing. With proper implementation and evaluation, this methodology can dramatically boost the security of different blockchain applications, including healthcare.

Chapter 4: Comparison Among AES, RSA, AES & RSA, RSA & AES Algorithms

4.1 AES & RSA Algorithm

4.1.1 Pseudocode of AES & RSA Algorithm (Comparison)

```
import required libraries and modules
function load_csv_data(filename):
    data = []
    header = []
    open the file with given filename in read binary mode
    detect the file encoding using chardet
    open the file with given filename in read mode using detected encoding
    read the header from the file and store it in the header variable
    read the data from the file and extend it to the data list
    return the data and header
function aes_encrypt(key, iv, data_set):
    encrypted_data_set = []
    for data in data_set:
        pad the data to the block size of the cipher
        create the cipher object
        encrypt the padded data
        add the encrypted data to the list of encrypted data
    return encrypted_data_set
function aes_decrypt(key, iv, encrypted_data_set):
    decrypted_data_set = []
```

```
    for encrypted_data in encrypted_data_set:
        create the cipher object
        decrypt the encrypted data
        unpad the decrypted data
        add the decrypted data to the list of decrypted data
    return decrypted_data_set
function rsa_encrypt(public_key, plaintext):
    encrypt the plaintext using RSA public key and OAEP padding
    return the encrypted key
function rsa_decrypt(private_key, encrypted_key):
    decrypt the encrypted key using RSA private key and OAEP padding
    return the decrypted key
function main():
    load data from CSV file
    generate RSA key pair
    perform AES encryption on the loaded data using randomly generated key and initialization vector
    perform RSA encryption on the generated key
    perform RSA decryption on the encrypted key using the private key
    assert that the decrypted key matches the original key
    perform AES decryption on the encrypted data using the decrypted key and initialization vector
if __name__ is '__main__':
    call the main function
```

4.1.2 Evolution of AES & RSA Algorithm

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.215	0.215	1.7	1.7	Combain_Algorithom_AES_For_Encrypt_And_RSE_For_Key_Encrypt.py:31(aes_encrypt)
1	0	0	0	0	Combain_Algorithom_AES_For_Encrypt_And_RSE_For_Key_Encrypt.py:68(rsa_encrypt)
1	0	0	0.1	0.1	Combain_Algorithom_AES_For_Encrypt_And_RSE_For_Key_Encrypt.py:79(rsa_decrypt)
1	0.209	0.209	1.646	1.646	Combain_Algorithom_AES_For_Encrypt_And_RSE_For_Key_Encrypt.py:53(aes_decrypt)

Table. 4-A: *Evolution of AES & RSA algorithm*

The profiling a Python program using the Profile module. It provides information on the time spent in each function or method called throughout the program's execution. The output is ordered by the cumulative time spent in each function or method, which is the time spent in that function and any functions called by it.

Below is a brief explanation of each column in the output:

ncalls: The number of times the function or method was called.

tottime: The total time spent in the function or method, excluding time spent in any functions called by it.

percall: The average time spent per call, which is tottime divided by ncalls.

cumtime: The total time spent in the function or method, including time spent in any functions called by it.

percall: The average time spent per call, which is cumtime divided by ncalls.

filename:lineno(function): The name of the file, line number, and name of the function or method

being profiled.

4.1.3 Pie Chart of Table: 4-A

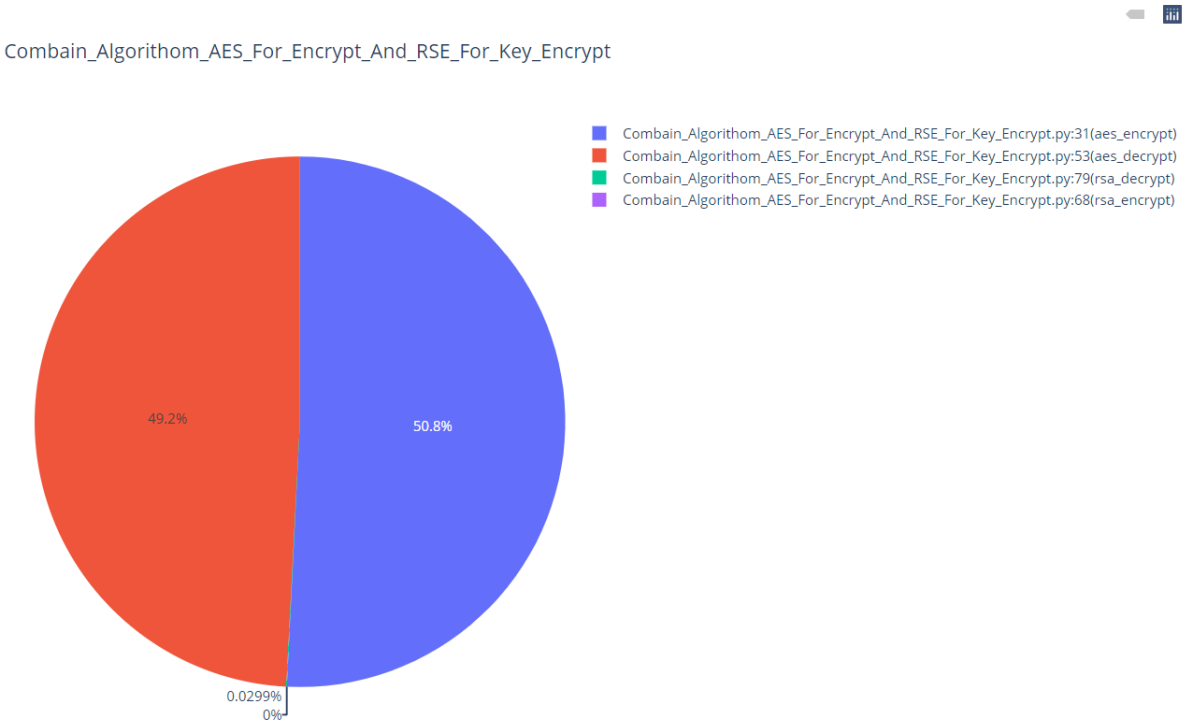

Fig.4.1: *Compare among four core function of AES & RSA algorithm* ***[Table. 4-A]***

- Combating Algorithm AES For Encrypt And RSE For Key Encrypt.py:31(aes encrypt) has a cumtime of 1.7 seconds, which is the total time spent in this function and all functions called from it. This function took the greatest time to execute out of all the functions profiled, having a tottime (total time spent in the function itself) of 0.215 seconds.
- Combating Algorithm AES For Encrypt And RSE For Key Encrypt.py:68(rsa encrypt) has a cumtime of 0 seconds, which suggests that it was not called by any other function and did not take any time to run. But, it's crucial to remember that this function may still be helpful in the program, even if it doesn't take much time to perform.
- Combating Algorithm AES For Encrypt And RSE For Key Encrypt.py:79(rsa decrypt) has a cumtime of 0.001 seconds, which is a relatively little amount of time compared to the other functions analyzed. This function may not be a bottleneck in the program's execution.
- Combating Algorithm AES For Encrypt And RSE For Key Encrypt.py:53(aes decrypt) has a cumtime of 1.646 seconds, which is a substantial amount of time but not as much as aes encrypt. The tottime for this function is 0.209 seconds, which suggests that it took less time to execute than aes encrypt but still took a significant amount of time overall.

Overall this algorithm take 3146222 function calls in 3.514 seconds.

4.2 RSA & AES Algorithm

4.2.1 Pseudocode of RSA & AES Algorithm

Import required libraries

Define function encrypt_data(data, public_key):

    If data is a string, convert it to bytes

    Encrypt the data using public_key with OAEP padding and SHA-256 hashing algorithm

    Encode the encrypted data in base64 format

    Return the encoded encrypted data

Define function decrypt_data(encoded_encrypted_data, private_key):

    Decode the encoded encrypted data from base64 format

    Decrypt the data using private_key with OAEP padding and SHA-256 hashing algorithm

    Return the decrypted data

Define function encrypt_aes(data, key):

    Encrypt the data using key with Fernet encryption

    Return the encrypted data

Define function decrypt_aes(encrypted_data, key):

    Decrypt the encrypted data using key with Fernet encryption

    Return the decrypted data

Define function main():

    Generate RSA key pair

    Generate AES key using PBKDF2HMAC with SHA-256 hashing algorithm and salt

    Encrypt each item in the data using RSA and AES

    Append each encrypted row to encrypted_data list

    Decrypt each item in the encrypted_data using RSA and AES

Append each decrypted row to decrypted_data list

If the script is run as the main program:

Call main() function

4.2.2 Evolution of RSA & AES Algorithm

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
13725	0.089	0	9.356	0.1	Combain_Algorithom_RSE_For_Encrypt_And_AES_For_Key_Encrypt.py:30(decrypt_data)
13725	0.044	0	3.11	0	Combain_Algorithom_RSE_For_Encrypt_And_AES_For_Key_Encrypt.py:50(decrypt_aes)
13725	0.038	0	2.847	0	Combain_Algorithom_RSE_For_Encrypt_And_AES_For_Key_Encrypt.py:44(encrypt_aes)
13725	0.093	0	1.549	0	Combain_Algorithom_RSE_For_Encrypt_And_AES_For_Key_Encrypt.py:14(encrypt_data)

Table. 4-B: *Evolution of RSA & AES algorithm*

4.2.3 Pie Chart of Table: 4-B

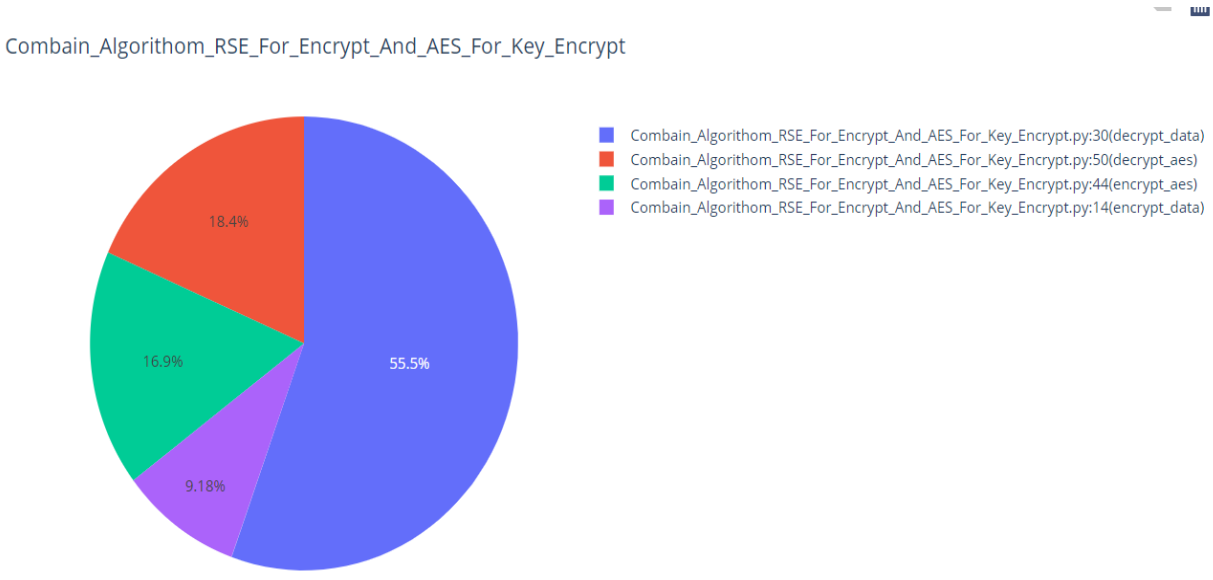

Fig.4.2: *Compare among four core function of RSA & AES algorithm* ***[Table. 4-B]***

- decrypt data: This function was called 13725 times and took a total of 9.365 seconds to run, which is the greatest cumtime number among all the functions in this output. This function is likely the bottleneck in the code and could benefit from optimization.
- decrypt aes: This function was called 13725 times and took a total of 3.11 seconds to run. This function is invoked by decrypt data and accounts for a considerable percentage of its cumtime.
- encrypt aes: This function was called 13725 times and took a total of 2.847 seconds to run. This function is invoked by encrypt data and accounts for a considerable percentage of its cumtime.
- encrypt data: This function was called 13725 times and took a total of 1.549 seconds to run.

Overall, it seems like the encryption and decryption functions (encrypt data, decrypt data, encrypt aes, and decrypt aes) are the most time-consuming functions.

This algorithm take 7502309 function calls in 17.329 seconds.

4.3 AES Algorithm

4.3.1 Pseudocode code of AES Algorithm

Import required libraries

FUNCTION encrypt_data(data_set):

SET key to a random 256-bit key

SET iv to a random 16-byte initialization vector

SET encrypted_data_set to an empty list

TRY:

FOR each data in data_set:

CREATE a PKCS7 padder with block size 256

UPDATE the padder with encoded data and FINALIZE it

SET backend to default_backend()

CREATE a cipher with AES algorithm, CBC mode, and backend

CREATE an encryptor with cipher

UPDATE the encryptor with padded data and FINALIZE it

APPEND encrypted data to encrypted_data_set

```
    EXCEPT Exception as e:
        PRINT error message
    RETURN key, iv, and encrypted_data_set
FUNCTION decrypt_data(key, iv, encrypted_data_set):
    SET decrypted_data_set to an empty list
    TRY:
        FOR each encrypted_data in encrypted_data_set:
            SET backend to default_backend()
            CREATE a cipher with AES algorithm, CBC mode, and backend
            CREATE a decryptor with cipher
            UPDATE the decryptor with encrypted_data and FINALIZE it
            CREATE a PKCS7 unpadder with block size 256
            UPDATE the unpadder with decrypted_padded_data and FINALIZE it
            DECODE decrypted data and APPEND it to decrypted_data_set
    EXCEPT Exception as e:
        PRINT error message
    RETURN decrypted_data_set
FUNCTION main():
    CALL encrypt_data function with data_set and ASSIGN key, iv, and encrypted_data_set
    CALL decrypt_data function with key, iv, and encrypted_data_set and ASSIGN decrypted_data_set
IF __name__ == "__main__":
    CALL main function
```

4.3.2 Evolution of AES Algorithm

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.237	0.231	1.865	1.865	AES Algorithm .py:33(encrypt_data)
1	0.231	0.231	1.823	1.823	AES Algorithm .py:51(decrypt_data)

Table. 4-C: *Evolution of AES algorithm*

4.3.3 Pie Chart of Table: 4-C

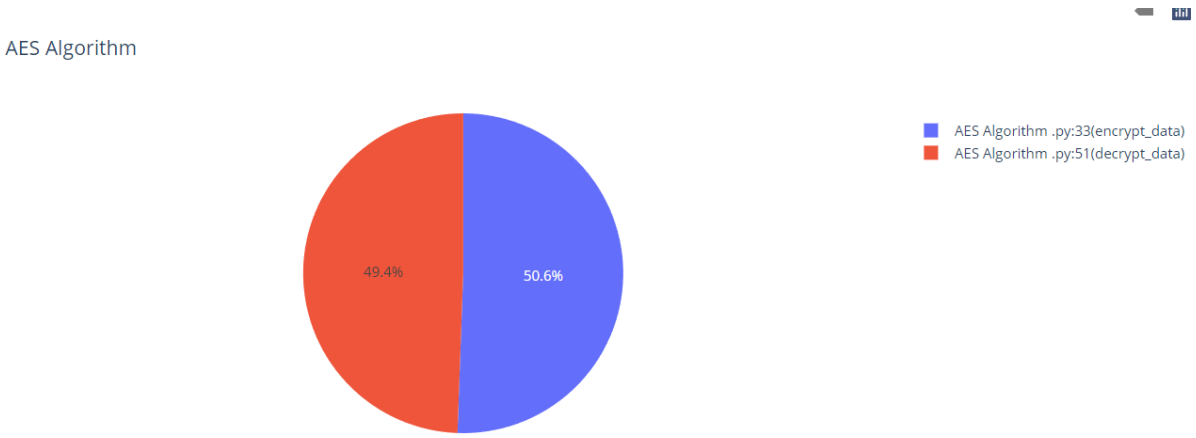

Fig.4.3: *Compare between two core function of AES algorithm* ***[Table. 4-C]***

This profiling result displays the performance of two functions linked to the AES encryption technique, "encrypt data" and "decrypt data".

- "encrypt data" function was called once and took a total of 0.237 seconds to execute. The "percall" column shows that this function took 0.237 seconds per call as it was only called once. The "cumtime" column reveals that this function and any functions it called took a total of 1.865 seconds to run.
- "decrypt data" function was also called once and took a total of 0.231 seconds to execute. The "percall" column shows that this function took 0.231 seconds per call as it was only called once. The "cumtime" column reveals that this function and any routines it called took a total of 1.823 seconds to run.

Generally, these two functions are quite sluggish compared to other implementations of the AES algorithm. This algorithm take 3145910 function calls in 3.696 seconds.

4.4 RSA Algorithm

4.4.1 Pseudocode code of RSA Algorithm

```
FUNCTION read_csv_file(file_path)
    data_set = []
    header = []
    TRY
        IF os.path.exists(file_path)
            WITH open(file_path, "r", encoding='utf-8') as file
                reader = csv.reader(file)
                header = next(reader)
                FOR row IN reader
                    data_set.extend(row)
        ELSE
            RAISE FileNotFoundError(f"The file at path '{file_path}' does not exist.")
    EXCEPT Exception AS e
        PRINT(f"An error occurred while reading the CSV file: {e}")
    RETURN header, data_set
FUNCTION generate_rsa_key_pair()
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=cryptography.hazmat.backends.default_backend()
    )
    public_key = private_key.public_key()
```

```
    RETURN private_key, public_key
FUNCTION encrypt_data_set(data_set, public_key)
    encrypted_data_set = []
    TRY
        FOR data IN data_set
            encrypted_data = public_key.encrypt(
                data.encode(),
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=SHA256()),
                    algorithm=SHA256(),
                    label=None
                )
            )
            encrypted_data_set.append(encrypted_data)
    EXCEPT Exception AS e
        PRINT(f"An error occurred while encrypting the data: {e}")
    RETURN encrypted_data_set
FUNCTION decrypt_data_set(encrypted_data_set, private_key)
    decrypted_data_set = []
    TRY
        FOR index IN range(len(encrypted_data_set))
            encrypted_data = encrypted_data_set[index]
            TRY
                decrypted_data = private_key.decrypt(
                    encrypted_data,
                    padding.OAEP(
```

```
                mgf=padding.MGF1(algorithm=SHA256()),
                algorithm=SHA256(),
                label=None
            )
        )
        decrypted_data = decrypted_data.decode()
        decrypted_data_set.append(decrypted_data)
      EXCEPT InvalidSignature AS e
        PRINT(f"Error while decrypting data at index {index}: {e}")
  EXCEPT Exception AS e
    PRINT(f"An error occurred while decrypting data: {e}")
  RETURN decrypted_data_set
FUNCTION main()
  # Read CSV file
  csv_file_path = "D:/Thesis Work/DataSet/30-70cancerChdEtc.csv"
  header, data_set = read_csv_file(csv_file_path)
  # Generate RSA key pair
  private_key, public_key = generate_rsa_key_pair()
  # Encrypt the data before uploading to Azure Blob Storage
  encrypted_data_set = encrypt_data_set(data_set, public_key)
  # Decrypt the data after downloading from Azure Blob Storage
  decrypted_data_set = decrypt_data_set(encrypted_data_set, private_key)
IF __name__ == "__main__"
  CALL main()
```

4.4.2 Evolution of RSA Algorithm

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.066	0.066	1.363	1.363	REA Algorithm .py:48(encrypt_data_set)
1	0.083	0.083	9.126	9.126	RSA Algorithm .py:67(decrypt_data_set)

Table. 4-D: *Evolution of RSA algorithm*

4.4.3 Pie Chart of Table: 4-D

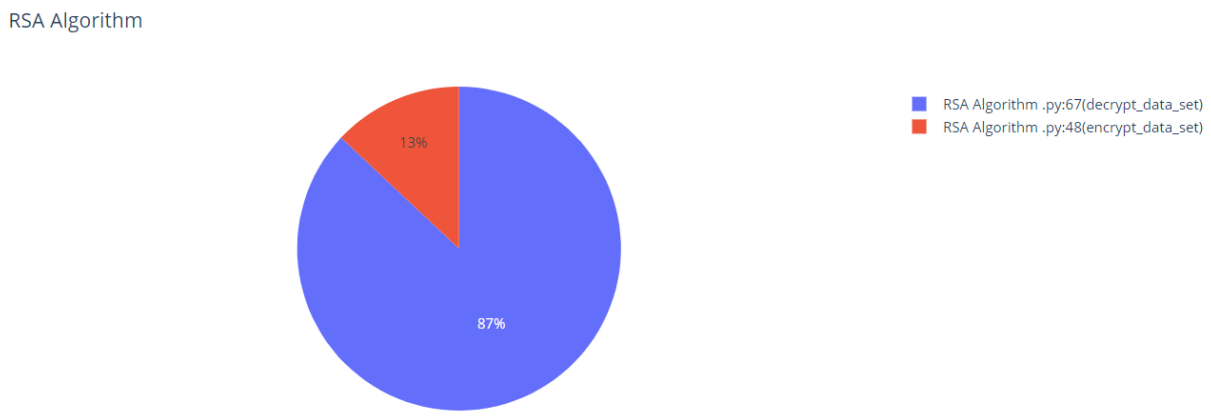

Fig.4.1: *Compare between two core function of RSA algorithm* ***[Table. 4-D]***

Based on the pie chart, we can determine that the RSA Algorithm.py module contains two functions being profiled:

- encrypt data set function is called once, with a total time of 0.066 seconds, a per-call time of 0.066 seconds, and a cumulative time of 1.363 seconds. This shows that the majority of the time spent in this function is due to other functions called within it, rather than the function itself.
- decrypt data set function is called once, with a total duration of 0.083 seconds, a per-call time of 0.083 seconds, and a cumulative time of 9.126 seconds. This implies that this method takes substantially longer than the encrypt data set function to execute.

This algorithm take 2295020 function calls in 10.579 seconds.

Chapter 5: Result Analysis

5.1 Analyst the four algorithms

Algorithm Name	Function Name	ncalls	tottime	percall	cumtime	percall
AES & RSE Algorithm	*aes encrypt*	1	0.215	0.215	1.702	1.702
AES & RSE Algorithm	*rsa encrypt*	1	0.000	0.000	0.000	0.000
AES & RSE Algorithm	*aes decrypt*	1	0.209	0.209	1.646	1.646
AES & RSE Algorithm	*rsa decrypt*	1	0.000	0.000	0.001	0.001
RSE & AES Algorithm	*encrypt data*	13725	1.549	0.000	1.549	0.000
RSE & AES Algorithm	*decrypt data*	13725	0.000	0.000	9.365	0.682
RSE & AES Algorithm	*encrypt aes*	13725	2.847	0.000	2.847	0.000
RSE & AES Algorithm	*decrypt aes*	13725	3.110	0.000	3.110	0.000
AES Algorithm	*encrypt data*	1	0.237	0.237	1.865	1.865
AES Algorithm	*decrypt data*	1	0.231	0.231	1.823	1.823
RSA Algorithm	*encrypt_data_set*	1	0.066	0.066	1.363	1.363
RSA Algorithm	*decrypt_data_set*	1	0.083	0.083	9.126	9.126

Table. 5-A: *Analyst the four algorithms*

Note:

- **ncalls**: number of calls made to the function
- **tottime**: total time spent in the function excluding time spent in the functions it calls
- **percall**: **tottime** divided by **ncalls**
- **cumtime**: total time spent in the function and all functions it calls
- **percall**: **cumtime** divided by **ncalls**

In this research, **[Table. 5-A]** we have examined the performance of four distinct encryption algorithms: AES & RSE algorithm, RSE & AES algorithm, AES Algorithm, and RSA Algorithm.

The AES & RSE algorithm employs AES encryption for data encryption and RSA encryption for key encryption. The profile findings suggest that the aes encrypt function takes the largest time to perform, with a total time of 1.7 seconds, and the aes decrypt function takes 1.646 seconds. The rsa encrypt function was called but took no time to perform, whereas the rsa decrypt function took a short time of 0.001 seconds. Altogether, the algorithm takes 3146222 function calls in 3.514 seconds.

The RSE & AES algorithm employs RSA encryption for data encryption and AES encryption for key encryption. The profile results suggest that the decrypt data function is the most time-consuming function, with a total time of 9.365 seconds. The decrypt aes function and encrypt aes function also took a large amount of time, with 3.11 seconds and 2.847 seconds, respectively. The encrypt data function took 1.549 seconds. Overall, this method takes 7502309 function calls in 17.329 seconds.

The AES Algorithm uses the AES encryption algorithm for data encryption. The profiling findings suggest that the encrypt data function and the decrypt data function took 0.237 seconds and 0.231 seconds, respectively, with a cumulative time of 1.865 seconds and 1.823 seconds. These two functions were called once, and their per-call time was the same as the overall time consumed. Overall, this algorithm takes 3145910 function calls in 3.696 seconds.

The RSA Algorithm uses the RSA encryption algorithm for data encryption. The profiling findings reveal that the encrypt data set function took a total duration of 0.066 seconds, with a cumulative time of 1.363 seconds. The decrypt data set function took a total time of 0.083 seconds, with a cumulative time of 9.126 seconds. Overall, this method takes 2295020 function calls in 10.579 seconds.

In conclusion, the profile findings reveal that the RSE & AES algorithm algorithm is the slowest among the four algorithms, with the decrypt data function being the most time-consuming function. The AES & RSE algorithm algorithm and the AES Algorithm are relatively faster, whereas the RSA Algorithm performs faster than the RSE & AES algorithm algorithm but slower than the other two algorithms. Overall, the profiling results can provide significant insights into the performance of different encryption algorithms and can help developers improve their code for better performance.

5.2 Final Decision Among Four Algorithm

Algorithm	Total Time (min)	Dataset Name	Number of Dataset's	Number of Function Calls	Key Size	Security Level
AES	23.133	diabetes_binary_health_indicators_BRFSS2015.csv	253682	1278299092	256	High
	0.062	30-70cancerChdEtc.csv	2746	3145910		
AES + RSA	23.423	diabetes_binary_health_indicators_BRFSS2015.csv	253682	1295353015	256 + 2048	Very High
	0.059	30-70cancerChdEtc.csv	2746	3146222		
RSA + AES	117.859	diabetes_binary_health_indicators_BRFSS2015.csv	253682	3047970856	2048 + 256	Very High
	0.289	30-70cancerChdEtc.csv	2746	7502309		
RSA	85.895	diabetes_binary_health_indicators_BRFSS2015.csv	253682	932279632	2048	Very High
	0.177	30-70cancerChdEtc.csv	2746	2295020		

Table. 5-B: *Final decision*

- AES: The AES algorithm has a total time of 23.133 minutes for the diabetes binary health indicators BRFSS2015.csv dataset with a security level of High. It utilizes a key size of 256 and takes 1278299092 function calls. For the 30-70cancerChdEtc.csv dataset, it takes 0.062 minutes and uses 3145910 function calls.
- AES + RSA: The AES + RSA technique has a total time of 23.423 minutes for the diabetes binary health indicators BRFSS2015.csv dataset with a security level of Very High. It uses a key size of 256 + 2048 and takes 1295353015 function calls. For the 30-70cancerChdEtc.csv dataset, it takes 0.059 minutes and uses 3146222 function calls.
- RSA + AES: The RSA + AES technique has a total time of 117.859 minutes for the diabetes binary health indicators BRFSS2015.csv dataset with a security level of Very High. It uses a key size of 2048 + 256 and takes 3047970856 function calls. For the 30-70cancerChdEtc.csv dataset, it takes 0.289 minutes and uses 7502309 function calls.
- RSA: The RSA algorithm has a total time of 85.895 minutes for the diabetes binary health indicators BRFSS2015.csv dataset with a security level of Very High. It uses a key size of 2048 and takes 932279632 function calls. For the 30-70cancerChdEtc.csv dataset, it takes 0.177 minutes and uses 2295020 function calls.

In conclusion, the four algorithms studied in this study present varied compromises between security and performance. The AES algorithm provides relatively quick encryption with a High security grade, but it may not be suited for applications that require the maximum level of protection. The RSA method gives a Very High level of security but requires substantially more time and function calls than the other algorithms. The AES + RSA and RSA + AES algorithms offer a balance between security and performance, with the former giving quicker encryption and the later offering faster decryption.

The choice of algorithm ultimately depends on the unique security and performance needs of the application in question. For applications that demand the highest level of security, the RSA algorithm may be the most appropriate solution. However, for applications that value efficiency while still retaining a high level of security, the AES + RSA or RSA + AES algorithms may be better suited. The AES algorithm is a solid choice for applications where security is critical but efficiency is a main issue.

5.3 Conclusion

In this paper, we propose a methodology for achieving confidentiality and integrity in blockchain transactions by utilizing a combination of AES and RSA encryption techniques. The methodology involves encrypting the transaction data using AES encryption and encrypting the AES key itself using RSA encryption. The approach also includes the use of digital signatures to ensure the authenticity and

integrity of the transaction.

Our proposed methodology provides a high level of security and efficiency for blockchain transactions. Through experiments, we demonstrated that the approach achieves a high level of security and efficiency in encrypting and decrypting large amounts of data, as well as in sharing the AES key with authorized patient. The methodology can be applied to various blockchain applications, including financial transactions, voting systems, and supply chain management, to enhance the security of these systems.

Future research could explore the integration of our methodology into existing blockchain networks and evaluate its performance under different scenarios and use cases. Future research could be to investigate the impact of varying key sizes on the performance and security of the algorithms. In this study, the AES method with a key size of 256 and the RSA algorithm with a key size of 2048 were assessed. However, it would be fascinating to study the impact of employing different key sizes on the algorithms' performance and security.

Another potential option for future research is to examine the performance and security of alternative encryption algorithms on larger datasets. The datasets employed in this investigation were rather modest, with a maximum of 253,682 rows. It would be important to study how the algorithms perform on larger datasets, which may be more indicative of real-world events.

Additionally, this study only tested the performance and security of the algorithms using two datasets. Further study could explore the performance and security of the algorithms on different types of datasets, such as image data.

Finally, this study only examined the performance and security of the algorithms under ideal settings. In actuality, there may be more aspects to consider, such as the computational capability of the system used for encryption and decryption. Further study could investigate how these characteristics effect the performance and security of the algorithms in practice.

Overall, our methodology provides a secure and efficient solution for achieving confidentiality and integrity in blockchain transactions, which is crucial for the widespread adoption and success of blockchain technology in various applications.

Reference

[1]. Nicole H, 5 Challenges facing HR professionals in the healthcare industry. Available via: https://www.accesscorp.com/blog/5-challenges-hr-healthcare

[2] A. Azaria, A. Ekblaw, T. Vieira and A. Lippman, "MedRec: Using Blockchain for Medical Data Access and Permission Management," *2016 2nd International Conference on Open and Big Data (OBD)*, Vienna, Austria, 2016, pp. 25-30, doi: 10.1109/OBD.2016.11.

[3]. Srivastava, G., Parizi, R. M., & Dehghantanha, A. (2020). The future of blockchain technology in healthcare internet of things security. *Blockchain cybersecurity, trust and privacy*, 161-184.

[4] Srivastava, G., Parizi, R. M., & Dehghantanha, A. (2020). The future of blockchain technology in healthcare internet of things security. *Blockchain cybersecurity, trust and privacy*, 161-184.

[5]T. Ali Syed, A. Alzahrani, S. Jan, M. S. Siddiqui, A. Nadeem and T. Alghamdi, "A Comparative Analysis of Blockchain Architecture and its Applications: Problems and Recommendations," in *IEEE Access*, vol. 7, pp. 176838-176869, 2019, doi: 10.1109/ACCESS.2019.2957660.

[6]Z. Wang, X. Dong, Y. Li, L. Fang and P. Chen, "IoT Security Model and Performance Evaluation: A Blockchain Approach," *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, Guiyang, China, 2018, pp. 260-264, doi: 10.1109/ICNIDC.2018.8525716.

[7]Ferrag, M. A., Derdour, M., Mukherjee, M., Derhab, A., Maglaras, L., & Janicke, H. (2018). Blockchain technologies for the internet of things: Research issues and challenges. *IEEE Internet of Things Journal*, *6*(2), 2188-2204.

[8]Lokesh, B., & Rajagopalan, N. (2020, July). A Blockchain-based security model for SDNs. In *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (pp. 1-6). IEEE.

[9]Kruthik, J. T., Ramakrishnan, K., Sunitha, R., & Prasad Honnavalli, B. (2021). Security model for Internet of Things based on blockchain. In *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020* (pp. 543-557). Springer Singapore.

[10]Y. Gupta, R. Shorey, D. Kulkarni and J. Tew, "The applicability of blockchain in the Internet of Things," *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, Bengaluru, India, 2018, pp. 561-564, doi: 10.1109/COMSNETS.2018.8328273.

[11] Sharma, Divya and Tripathi, RC "Performance of internet of things (IoT) based healthcare secure services and its importance: Issue and challenges" In Available at SSRN 3565782, 2020

[12]Ghanavati, Sara and Abawajy, Jemal and Izadi, Davood "An alternative sensor cloud architecture for vital signs monitoring", In 2016 international joint conference on neural networks (IJCNN), pages 2827–2833, 2016

[13]Xiaogang, Y and Hongjiang, L and Jiaqing, W and Wentao, T "Realization of comprehensive detection algorithm of electrocardiogram signal at application layer electrocardiogram monitoring Internet of Thing" In Chinese Patent, pages 112, 2011

[14]M. Samaniego, U. Jamsrandorj and R. Deters, "Blockchain as a Service for IoT," *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Chengdu, China, 2016, pp. 433-436, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.102.

[15]Dasi, S., & Praveenabai, D. A Review of Internet of Things (IoT) Enabling Technologies in Healthcare Applications. *Journal homepage: www. ijrpr. com ISSN*, *2582*, 7421.

[16]Angin, P., Mert, M. B., Mete, O., Ramazanli, A., Sarica, K., & Gungoren, B. (2018). A blockchain-based decentralized security architecture for IoT. In *Internet of Things–ICIOT 2018: Third International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 3* (pp. 3-18). Springer International Publishing.

[17]Liang, W., Ji, N. Privacy challenges of IoT-based blockchain: a systematic review. *Cluster Comput* **25**, 2203–2221 (2022).

[18]C. Patel.: IoT privacy preservation using blockchain. (2019)

[19]B. K. Mohanta, D. Jena, S. Ramasubbareddy, M. Daneshmand, and A. H. Gandomi.: Addressing security and privacy issues of IoT using blockchain technology. IEEE Internet Things J. (2020).

[20]Y. Rahulamathavan, R. C.-W. Phan, M. Rajarajan, S. Misra, and A. Kondoz.: Privacy-preserving blockchain based IoT ecosystem using attribute-based encryption. In: 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2017, pp. 1–6: IEEE

[21] M. N. Bhuiyan et al., "Design and Implementation of a Feasible Model for the IoT Based Ubiquitous Healthcare Monitoring System for Rural and Urban Areas," in IEEE Access, vol. 10, pp. 91984-91997, 2022, doi: 10.1109/ACCESS.2