Indian Journal of Computer Science and Technology
Volume 3, Issue2 (May-August 2024), PP: 01-04
www.indjcst.com
ISSN No: 2583-5300

Parallel Processing in Hybrid Encryption Using AES and RSA

Denny Santhosh[1], Kripa Shibu[2], Mirin Mathew[3], R Gayathry Krishna[4], Jinu Mathai[5]

[1,2,3,4] B. Tech, Department of Computer Science, St. Thomas College of Engineering and Technology (STCET), Chengannur, Kerala, India.

[5]Assistant Professor, Department of Computer Science(AI&ML), St. Thomas College of Engineering and Technology(STCET), Chengannur, Kerala, India.

To Cite this Article: *Denny Santhosh[1], Kripa Shibu[2], Mirin Mathew[3], R Gayathry Krishna[4], Jinu Mathai[5], "Parallel Processing in Hybrid Encryption Using AES and RSA", Indian Journal of Computer Science and Technology, Volume 3, Issue2 (May-August 2024), PP: 01-04.*

Abstract: *In the realm of information security, the demand for robust encryption techniques has never been greater. This project delves into the integration of parallel processing techniques to enhance the efficiency and security of hybrid encryption, combining the strengths of Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) algorithms. The primary objective of this research is to develop a novel approach that leverages parallel processing to accelerate the encryption and decryption processes while maintaining a high level of security. AES is known for its speed and symmetric key encryption strength, while RSA excels in asymmetric key management. By synergizing these two cryptographic methods and harnessing the power of parallelization, this project aims to address the computational demands of modern encryption. Throughout this study, we will explore various parallelization strategies, including multi-threading, and distributed computing, to optimize the encryption and decryption phases. The performance gains achieved through parallel processing will be evaluated against traditional sequential encryption methods, demonstrating the practicality and advantages of this hybrid approach. Furthermore, the security aspects of the proposed hybrid encryption will be thoroughly examined, ensuring that parallel processing does not compromise the confidentiality and integrity of encrypted data. The project will also investigate key management, key distribution, and the scalability of parallelized hybrid encryption to accommodate diverse use cases and applications.*

Keywords*:* *Parallel processing, hybrid encryption, AES parallel processing.*

I.INTRODUCTION

Data security events such as information leaking and tampering are becoming increasingly common. The large transfer volume gives the attacker more possibilities to intercept data during transmission. However, if valuable data is captured, it can have major consequences for individuals, businesses, and governments. Thus, information security has always been an important concern. Data security requires effective strategies.

Combining both AES and RSA in a hybrid encryption scheme allows for the benefits of both symmetric and asymmetric encryption. Any individual encryption algorithm is not strong enough to reduce the chance of being successfully cracked, since any of them has been studied thoroughly and attackers are already familiar with the vulnerabilities of these single algorithms. Leveraging parallel processing in this context is more applicable to the symmetric encryption phase (AES) due to its suitability for parallel operations. Asymmetric encryption (RSA) operations remain sequential in nature and don't readily benefit from parallel processing.

Parallel processing in hybrid encryption using AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) involves leveraging multiple computational units simultaneously to enhance the efficiency of encryption and decryption operations. In this context, AES and RSA are frequently combined to take use of their individual advantages: RSA is known for its robustness in safely transferring encryption keys for asymmetric key encryption, while AES is known for its speed in processing huge volumes of data for symmetric key encryption. By employing parallel processing techniques, such as dividing the encryption tasks into smaller chunks and distributing them across multiple processing units, the overall encryption process can be significantly accelerated. For example, during encryption, multiple blocks of data can be encrypted concurrently using AES, while RSA can be utilized to securely encrypt and exchange the AES session key. Similarly, during decryption, parallel processing can be employed to simultaneously decrypt multiple data blocks using AES, while RSA is used to securely decrypt the session key. Overall, hybrid encryption utilizing parallel processing with AES and RSA offers a potent method to accomplish security and efficiency in cryptographic procedures, particularly when working with large amounts of data.

II.EXISTING SYSTEM

Hybrid encryption is a combination of both symmetric and asymmetric encryption schemes. The existing system of hybrid encryption involves using the strengths of both encryption schemes to improve the overall security of the system. In this system, symmetric encryption is used to encrypt the data using a secret key that is shared by both the sender and the receiver. This ensures that the data is secure during transmission and cannot be accessed by unauthorized parties. However, the problem with symmetric encryption is that the secret key needs to be shared between the sender and the receiver. This creates a security risk, as anyone who gains unauthorized access to the key can easily decrypt the data.

To overcome this issue, asymmetric encryption is used. Asymmetric encryption involves using a public key to encrypt the data, and a private key to decrypt it. The public key is freely available to anyone, while the private key is kept secret by the owner. This ensures that the data can only be accessed by authorized parties, and adds an extra layer of security to the system. In the hybrid encryption system, the sender encrypts the data using symmetric encryption and the secret key, and encrypts the secret key using the receiver's public key. The receiver then decrypts the secret key using their private key, and uses it to decrypt the data. This system is highly secure and provides a high level of protection for sensitive data.

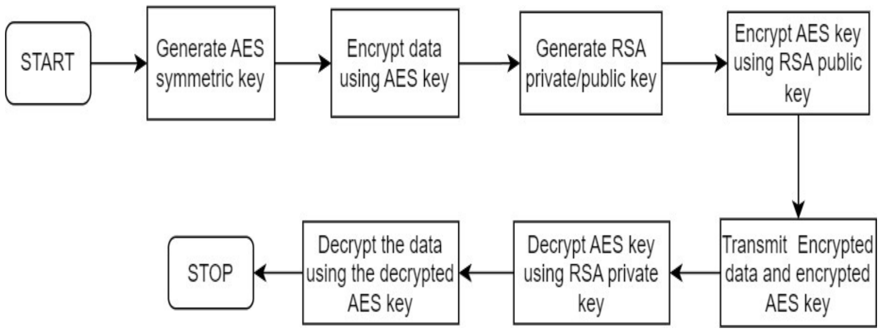

START
Generate AES symmetric key
Encrypt data using AES key
Generate RSA private/public key
Encrypt AES key using RSA public key
Transmit Encrypted data and encrypted AES key
Decrypt AES key using RSA private key
Decrypt the data using the decrypted AES key
STOP

Figure 1: Block diagram of Existing System

Disadvantages of Existing System:

- **Slower Processing:** Sequential encryption often takes longer than parallel processing since it requires performing encryption operations one after another, leading to increased processing time. Encrypting large volumes of data sequentially can be time-consuming and slow due to the linear nature of the encryption process, especially for very large datasets.
- **Resource-Intensive:** Traditional AES encryption does not take full advantage of modern multi-core processors and hardware acceleration. This method is resource-intensive as it may require significant memory and processing power, potentially causing bottlenecks or delays in systems with limited resources.
- **Scalability:** Sequential AES encryption may face scalability issues when encrypting large datasets or when encryption performance needs to be scaled up to meet higher demand.

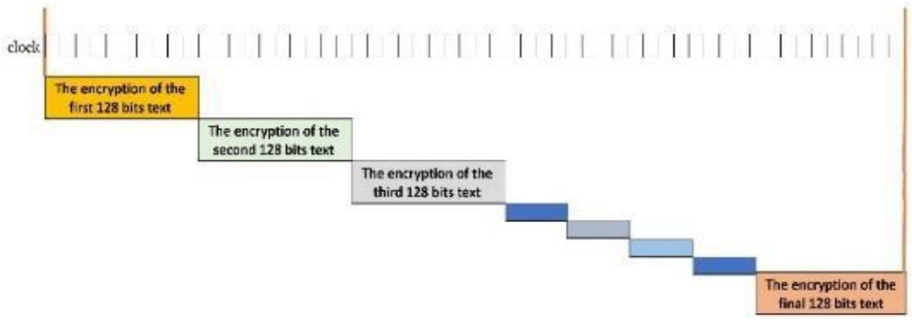

clock
The encryption of the first 128 bits text
The encryption of the second 128 bits text
The encryption of the third 128 bits text
The encryption of the final 128 bits text

Figure 2: The traditional AES processing

III.PROPOSED SYSTEM

Parallel processing is a computing technique where multiple calculations or processes are carried out simultaneously. This approach contrasts with serial processing, where tasks are completed one after the other. In parallel processing, tasks are divided into smaller sub-tasks that can be executed concurrently by multiple processing units. This can significantly improve computational speed and efficiency, especially for tasks that are inherently parallelizable, such as large-scale data analysis, simulations, rendering graphics, and scientific computations.

The proposed system aims to harness the benefits of AES in a parallelized environment, addressing the challenges posed by the encryption of vast datasets. In this system, data is partitioned into manageable chunks, each processed concurrently by multiple processors utilizing the same AES key. Subsequently, the AES key is encrypted using the recipient's public key, forming a robust hybrid encryption process that combines the efficiency of parallel processing with the security features of the RSA algorithm.

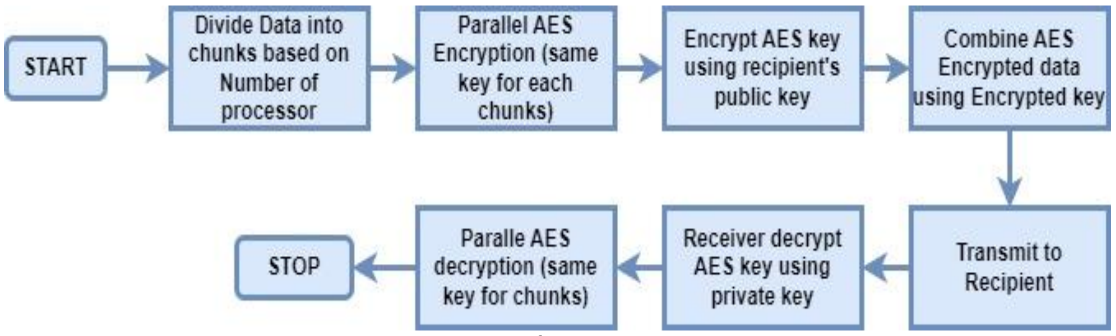

START
Divide Data into chunks based on Number of processor
Parallel AES Encryption (same key for each chunks)
Encrypt AES key using recipient's public key
Combine AES Encrypted data using Encrypted key
Transmit to Recipient
Receiver decrypt AES key using private key
Paralle AES decryption (same key for chunks)
STOP

Figure 3: Flowchart of parallel hybrid encryption

Encryption Process:

The encryption process begins with user interaction, where the user selects a file to encrypt and provides necessary inputs like a password, file ID, password length, and receiver email through a custom input dialog. Upon user input, a random salt is generated using a cryptographic function. This salt, along with the user-provided password, is utilized to derive a secure encryption key using the PBKDF2 key derivation function. The derived key's length is then checked to ensure it meets the required criteria of being either 16, 24, or 32 bytes.

Following key generation, the salt is encrypted using the recipient's public key obtained based on the provided email address. This step ensures the secure transmission of the encryption key to the intended recipient. The input file is then divided into chunks, and multiple parallel processes are spawned, with each process responsible for encrypting a chunk of the file simultaneously using the AES-CTR encryption mode. This parallelization enhances the efficiency of the encryption process, particularly for large files.

Once the encryption of all file chunks is completed, information about the encryption process, including the file ID and chunk sizes, along with the encrypted salt, is stored in a database for future reference. The duration of the encryption process is calculated to provide feedback to the user, indicating the time taken for the encryption operation to complete successfully.

- File Chunking: Before parallelization, the input file is divided into smaller, manageable chunks. Each chunk represents a portion of the file's data, which can be processed independently.
- Parallel Encryption: Multiple processes are spawned, with each process responsible for encrypting a chunk of the file simultaneously. This parallelization allows for the encryption of multiple file chunks to occur concurrently.
- Each process operates independently on its assigned chunk, utilizing the derived encryption key to perform AES-CTR encryption.
- By distributing the encryption workload across multiple processes, the overall encryption time is reduced significantly, especially for large files.
- The use of multiprocessing ensures that multiple CPU cores are utilized concurrently, maximizing computational efficiency.
- Synchronization: To ensure data integrity and avoid race conditions, synchronization mechanisms such as semaphores are employed.
- A semaphore ensures that only one process can access shared resources, such as the output file, at a time, preventing conflicts and ensuring orderly execution.

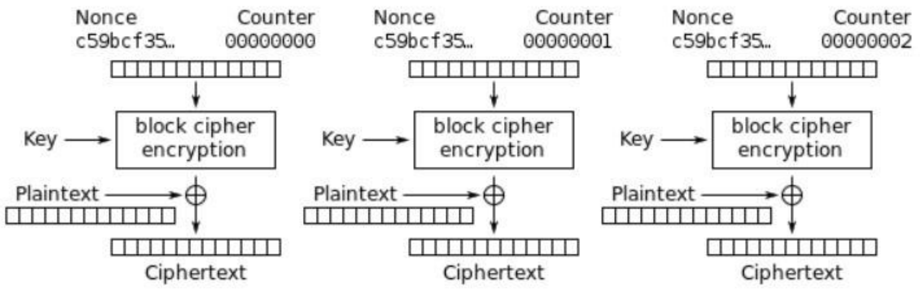


Figure 4: The Counter(CTR) mode encryption

Decryption Process:

On the decryption side, the user initiates the process by selecting the file to decrypt and providing the required inputs, such as the password and file ID, through a user interface. Subsequently, encryption-related information, including the chunk sizes and the encrypted salt, is retrieved from the database based on the provided file ID.

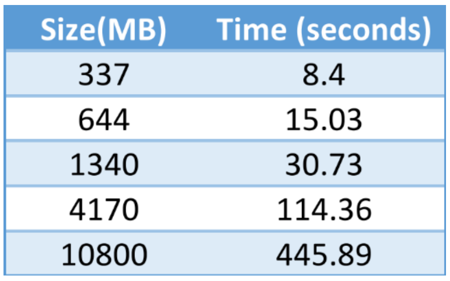
The encrypted salt is then decrypted using the user's private key, which is securely stored locally. This step ensures that only the intended recipient possessing the corresponding private key can access the decryption key derived from the password and salt. With the decryption key derived, the input file is read in chunks, and multiple parallel processes are employed to decrypt each chunk simultaneously using the AES-CTR decryption mode.

Upon successful decryption of all file chunks, the duration of the decryption process is calculated to inform the user about the time taken for the operation to complete. This systematic decryption process ensures the secure retrieval of the original content from the encrypted file while maintaining the confidentiality of the data throughout the decryption operation.

- File Chunking: Similar to the encryption process, the input encrypted file is divided into smaller chunks to enable parallel processing.
- Parallel Decryption: Multiple processes are created, each tasked with decrypting a chunk of the encrypted file concurrently.
- Each process operates independently on its assigned chunk, utilizing the derived decryption key to perform AES - CTR decryption.
- Parallel decryption allows for faster processing of the encrypted file, as multiple chunks can be decrypted simultaneously. Like in the encryption phase, multiprocessing ensures efficient utilization of available CPU cores, leading to improved performance.
- Synchronization: Synchronization mechanisms, such as semaphores, are employed to ensure thread safety and prevent conflicts during file write operations.

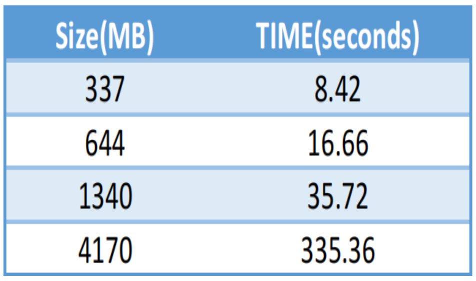
- Semaphores regulate access to shared resources, such as the output file, ensuring that only one process writes to the file at a time to maintain data integrity.

IV.RESULT ANALYSIS



Size(MB)	Time (seconds)
337	8.4
644	15.03
1340	30.73
4170	114.36
10800	445.89

Table 1. Parallel encryption



Size(MB)	TIME(seconds)
337	8.42
644	16.66
1340	35.72
4170	335.36

Table 2. Sequential encryption

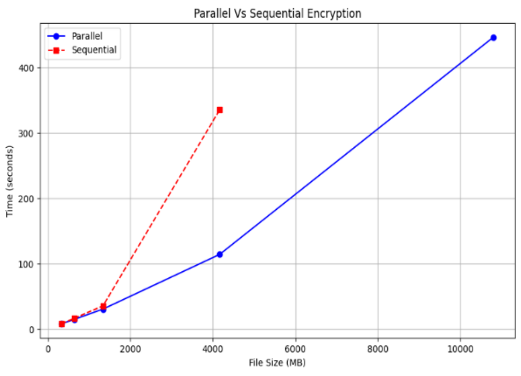


Figure 5. Parallel v/s Sequential computational line chart

Parallel and sequential computation time can differ significantly based on file size. In parallel computing, tasks are divided among multiple processors, enabling simultaneous processing, which can lead to faster execution times for large files. However, the overhead of task distribution and coordination may impact performance for small files, favoring sequential processing. In contrast, sequential processing handles one task at a time, which may be more efficient for small files due to reduced overhead but can become slower as file size increases due to processing bottleneck. Ultimately, the optimal approach depends on factors like file size, hardware capabilities, and the nature of the computation.

V.CONCLUSION

By integrating parallel processing into hybrid encryption offers significant advantages in terms of speed and efficiency. Encryption and decryption tasks can be completed much more quickly, especially with regard to high volumes of data, when using the capabilities of a number of processing units in parallel. This increased speed is critical in today's fast-paced digital environment, where data security is top priority. Parallel processing can help distribute computational load more evenly across hardware resources, leading to better resource utilization and scalability. However, it's important to ensure that parallel processing techniques are implemented securely, as any vulnerabilities could potentially compromise the confidentiality of encrypted data. Overall, the incorporation of parallel processing into hybrid encryption represents a promising avenue for improving the performance and robustness of encryption systems in various applications.

REFERENCES

1. *Satish B Basapur, B.S. Shylaja, Venkatesh. (2021). "A Hybrid Cryptographic Model Using AES and RSA for Sensitive Data Privacy Preserving". Webology, Vol 18, Special Issue On Current Trends in Management and Information Technology.*
2. *Divya , D., & Santhi , G. (2021). "Secured Multi-Party Data Release on Cloud for Big Data Privacy-Preserving Using Fusion Learning". Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(3), 4716-472.*
3. *Y. Hui and L. Zesong, "Research on Real-time Analysis and Hybrid Encryption of Big Data,"* 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD), *Chengdu, China, 2019, pp. 52-55, doi: 10.1109/ICAIBD.2019.8836992.*
4. *L. Zhang, B. Li and X. Zhao, "Reconfigurable Hardware Implementation of AES-RSA Hybrid Encryption and Decryption,"* 2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP), *Nanjing, China, 2020, pp. 970-974, doi: 10.1109/ICSIP49896.2020.9339251.*
5. *W. hong, W. sheng, Z. qing and Z. jian, "Research on Data Encryption of Network Communication Based on Big Data," 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Fuzhou, China, 2020, pp. 129-131, doi: 10.1109/ICBAIE49996.2020.00033.*