International Journal of Scientific Research and Engineering Development-– Volume X Issue X, Year

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Analysis and implementation of AES and RSA

Ritujeeth simha*, Poulami Manda **, Samarth jung rana***,Shashi kiran****

*(Department of Computer Science & IT, JAIN (Deemed to be University), Bengaluru, Karnataka
Email: r.simha15@gmail.com)

** (Department of Computer Science & IT, JAIN (Deemed to be University), Bengaluru, Karnataka
Email: poulami.m@jainuniversity.ac.in)

*** (Department of Computer Science & IT, JAIN (Deemed to be University), Bengaluru, Karnataka
Email: 23bcar0557@jainuniversity.ac.in)

*** (Department of Computer Science & IT, JAIN (Deemed to be University), Bengaluru, Karnataka
Email: 23bcar0132@jainuniversity.ac.in)

----------**********----------

Abstract:

AES and RSA are the most popular cryptographic algorithm used in information security. AES is a symmetric security algorithm employed in most data encryption due to its high speed and good cryptography system that identifies unusual behaviors whereas RSA, an asymmetric cryptography algorithm employs the public key to safeguard information and the private key to decrypt it. This will assist us in ensuring that our data is safe and secure against hackers and that the quality of AES and RSA are upheld. The encryption key of AES is the same, and it is utilized to both encrypt and decrypt data whereas RSA has a different key (public) and another key (private). All the data in AES can be hacked by a hacker in case one of the keys is faulty and hence the importance of protecting the key is also of the essence. Both AES and RSA have its strong and weak sides and we should be aware of the difference between the two and examine them closely to safeguard valuable information.

Keywords — **AES (Advanced Encryption Standard) ,RSA (Rivest–Shamir–Adleman) Symmetric encryption , Asymmetric encryption ,Public key Private key Data security, Encryption key.**

----------**********----------

I. INTRODUCTION

The foundation of information security is actually cryptography, which is highly essential in information protection in most applications, such as secure communications and financial transactions as well as data storage. Cryptographic algorithms have been created over many years with the intention of attempting to cope with these rising requirements of secure data communications in an ever-evolving threat environment. The best-known cryptographic codes include the Advanced Encryption Standard and the Rivest- Shamir-Adleman codes respectively which are known as AES and RSA. AES is principally fast and efficient. AES is effective in encryption of large amount of data. RSA, its part, is an asymmetric algorithm. It has been known to be excellent in secure key exchange and signing but much slower. AES and RSA are compared in terms of performance and security in the provided report. These targets are to determine the relative strengths and weaknesses and how they can be applied to various applications, with special emphasis on real-world deployment issues such as performance and security, and resource consumption.

I. HISTORY ON AES

AES had therefore been created to overcome some of the shortcomings that were present in the earlier symmetric key block cipher known as the DES, which, at this point, had become vulnerable to brute force attacks. In 1997, National Institute of Standards and Technology called for some solid replacement in the form of a competition-the Advanced Encryption Standard, or AES for short. They developed an algorithm known as Rijndael which was written by Vincent Rijmen and Joan Daemen. AES was formally adopted and approved as the encryption of the sensitive yet unclassified U.S. federal information in 2001.[7], [4]

III. **Key Features**

AES is a symmetric block cipher, that is, it requires the same key to be used to encrypt and decrypt. The block size used by the algorithm is 128 bits and the key sizes are 128, 192 and 256 bits with different encryption round counts based on the key size. It is composed of a series of substitution, permutation, and mixing on blocks of data. The most crucial fact is that AES was created in the way that it becomes resistant to not only exhaustive search of the key but also to other types of cryptanalytic attacks including linear and differential cryptanalysis. [4]

Common Use Cases

AES has extensive applications and is utilized in various industries and applications such as: Data Encryption - AES is used to encrypt sensitive data in the banking, healthcare, and government industries. It is also the encryption standard to secure communications (e.g., TLS, VPNs).

Disk Encryption - AES is used in a variety of file and disk encryption systems as well as in BitLocker and VeraCrypt.

Wireless Security - AES is deployed in the security of wireless communications, (such as Wi-Fi networks in the WPA2 protocol). Messaging Apps - AES can be used in secure messaging apps such as Signal and WhatsApp, which use AES to encrypt their end-to-end messages, making conversations private. [1] [4]

OVERVIEW OF RSA CRYPTOGRAPHIC ALGORITHM

IV B**ackground on RSA**

Development

RSA was invented by Ron Rivest, Adi Shamir and Leonard Adleman within the year 1977 when they were in MIT. The algorithm is called after the first letters of their last names.

Breakthrough

Prior to RSA, all encryption systems were based on symmetric ciphers-that is the key being the same to both the encryption and decryption by the sending party and the receiving party respectively. Symmetric encryption was used in the breakthrough at RSA whereby one could publish an encryption key and reserved a different and separate key, known privately to decrypt it. This was really fundamental in changing the game in cryptography as now one could securely communicate with anyone, literally out of the blue, without prior agreement over a secret key. [2]

V **Technical details**

Secure Data Transmission - RSA algorithm has two steps that are involved: Encryption - The plaintext message is transformed to a more number format and multiplied by the power of the public key exponent eee, and then multiplied by the inverse of nnn. The result is the ciphertext. [2]

Decryption - The ciphertext is lifted with the private key ddd, and divided by nnn, which will restore the original plaintext message.

Mathematically, the process can be expressed as:

Encryption: C=Memod nC = M^e \mod nC=

Memodn

Decryption: M=Cdmod nM = C^d \mod nM= Cdmodn

Here, MMM is the plaintext, CCC is the ciphertext, nnn is the product of the two primes, eee is the public key exponent, and ddd is the private key exponent.

Key Features Asymmetric Encryption (Public/Private Key Pair)

Public Key - The key is known as the public key since it is the key used in encryption and that it is freely distributed to anyone wishing to transmit a secure message. Only a corresponding private key assures decryption in an encryption process.

Private Key – This key remains secret with the owner and is used to decrypt the data encrypted through a public key. Although an attacker may have compromised the public key, without access to the private key, it is not possible to decrypt the message.

Security Basis - RSA obtains its security based on the hardness of the factorization problem of large composite numbers. It involves the product of two large prime numbers; and although it is computationally easy to multiply the prime numbers, the converse operation- finding the original primes in the product of the primes-is computationally infeasible, also known as factoring.

Key Size:

Standard Key Sizes - RSA normally has key sizes of 1024, 2048 or 4096 bits with the larger size being more secure but slow in processing.

Prime Number Generation - It is based on the following steps; firstly pick a two large prime numbers (denoted ppp and qqq) and then multiply them together to create the modulus n=p×qn = p times qn=p times q. The challenge of breaking RSA is the difficulty of breaking down this large number nnn into its primes.

A graph within a graph is an “inset”, not an “insert”. The word alternatively is preferred to the word “alternately” (unless you really mean something that alternates).

Common Use Cases

Secure Data Transmission

RSA is widely employed in secure communication systems such as the Secure Socket Layer/TLS, the foundation of secure Web sites and Internet transactions. It encrypts sensitive information such as credit card.
It is possible to have digital signatures on RSA as well. Here, the sender will sign with his own private key and the receiver will check the signature with the public key of the sender. This will enable protection of integrity and authenticity of data.

Email Encryption

The use of RSA to encrypt messages in most email services makes it impossible to have any other person but the original the recipient reading the message in the email.

IMPLEMENTATION AND TESTING

he AES algorithm takes 128-bit block of data using three key sizes of 128, 192 and 256 bits. AES is based on the Rijndael algorithm which includes multiple rounds of transformation, depending on the key length. AES128 consists of 10 rounds of operation as compared to AES-192 and AES-256 which have 12 and 14 rounds of operation respectively. This is because of the application process this focuses on is the AES128 implementation which is the most preferred because of the excellent compatibility with good security. Verilog is used in this implementation to model the flow of the AES algorithm to be synthesised on an FPGA platform. The reason behind using this type of hardware is the high throughput capability that is ideal in security application where real time encryption is required e.g. wireless and mobile telephone. [6]

VIII **Key Components in AES** Implementation

Sub Bytes Transformation - Every block byte is substituted by a substitution table byte in such a way, Sbox is designed to introduce very effectively the properties of non-linearity and strength in cryptographic sense.

Shift Rows Transformation - The rows in this state array are circularly shifted about an adjustable number of offsets to deliver diffusion on the block.

Mix Columns Transformation - Mixing in each column of the state matrix is carried out, where the four bytes are combined to provide further diffusion. This operation makes sure that only a minor alteration in plaintext or key influences numerous sections of the ciphertext.

Add Round Key - The state matrix is XORed with the round key, obtained by the key schedule. This operation occurs each round to shuffle the important material into the encryption operation. [8]

IX **Testing Approach for AES**

The AES algorithm is tried with encryption and decryption of sample data to measure its performance based on the following key points.

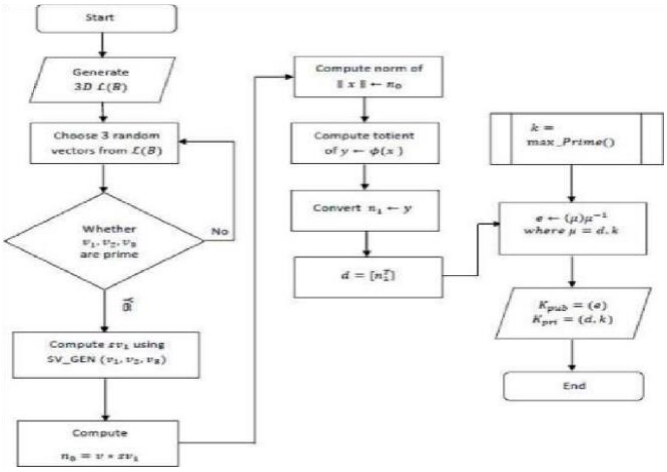
Encryption/Decryption Speed - AES128 is affected by the number of rounds and key size. As noted, encryption and decryption time varies with the size of the key (AES- 128 is faster compared to AES-192 or AES-256) and therefore AES-128 is the fastest in speed sensitive environments.

Size of Key - AES works with block size of 128 bits, and key size of 128, 192 or 256 bits. Key sizes are bigger, more secure and consume more rounds, which influence the encryption and decryption speed.

Resource Usage - FPGA implementation is selected due to its efficiency in high throughput application with relatively low resource consumption in terms of CPU and memory. The computational complexity of AES with the MixColumns and SubBytes transformations makes it computationally efficient as it uses bitwise operations and XOR. [1], [9]

X. Performance Comparison

Performance comparison analysis carried out on AES revolves around time and resource utilization aspects. On this basis, AES-128 is faster because of its fewer rounds compared to AES-256. This renders it suitable for real-time encryptions in file encryption and secure communication systems. AES-256, on the other hand, is slower than AES-128 but provides better security, thus more preferable where security is of utmost importance.In conclusion, using AES-128 through hardware will render high performance encryption with high resource efficiency especially in



Key Generation - Creation of RSA keys, which include public key that can be used in the encryption process and private key that is used in the decryption process.

Encryption - Conversion of the data into bytes and its encryption using public key.

Decryption - Further decryption of the data using private key. The output is converted into audio form to enable playing of the data.

XI Testing Approaches of RSA

For the purpose of evaluating the performance of RSA algorithm, several tests were carried out by using various samples of audio files by taking into consideration the following areas

Encryption/Decryption Speed

The time taken to encrypt and decrypt the sample audio files was considered. Different key sizes were evaluated, including 1024, 2048, and 4096 bits to see how the sizes impact on the performance of the operation.

Key Sizes (for both AES and RSA) For the RSA algorithm, different key sizes were used, such as 1024, applications that require low latency and high throughput. [1]

XII. **Implementation of RSA**

Implementation of RSA algorithm for encryption and decryption was done using Python. The following libraries were used to make the implementation possible:

PyCrypto - This library enables cryptographic functionality, which includes encryption and decryption processes under RSA algorithm.

NumPy - Used in performing numerical operations, which helps to deal with very large numbers resulting from RSA algorithm.

SoundFile - Python library that handles audio files in order to ensure that the speech data is in correct format.
The RSA algorithm was implemented as usual.2048, and 4096 bits were evaluated. In addition, key sizes of 128, 192, and 256 bits for AES algorithm were used. [10]

Resource Usage

CPU usage and memory consumption for encryption and decryption were monitored. Performance metrics including CPU cycles consumed and amount of memory consumed were profiled.

Performance Comparison of AES and RSA

The RSA algorithm was compared with AES with regards to speed in encryption and decryption processes, resource consumption, and effectiveness in the entire process. This was to understand whether each algorithm tends towards any of these parameters.

XIII AES Performance Evaluation

Performance testing for AES was undertaken encryption and decryption speed, CPU and memory consumption. The results obtained were then compared with those obtained from RSA algorithms. [1], [9], [10]

XIV Speed Comparison between AES and RSA

AES uses a symmetric key while RSA uses an asymmetric key. Hence, AES is much faster than RSA. As far as the time needed for the encryption and decryption of various data, AES is much quicker than RSA. For instance, the encryption of the student database using the AES-256 and RSA algorithms for the purposes of cloud computing revealed the fact that AES was somewhat faster due to the symmetry in encryption.

Indeed, it was evident that AES worked very efficiently in the encryption of larger files especially when the optimized AES was applied. Optimized AES gave very low encryption times than the regular AES. For example, encryption of a 10MB MP4 file using optimized AES took approximately 0.0302 seconds whereas 0.0450 seconds was returned by regular AES. [10]

XVI CPU and Memory Resource Usage

With respect to CPU and memory resource use in the process of encryption and decryption, it can be seen that under any circumstances, AES consumed fewer resources than RSA. An additional technique used in the optimized AES framework allowed for the reduction of CPU and memory consumptions. It uses approximately 23.86 MB for the encryption of a 2500KB JPG file in AES for instance. While in RSA, because of the computational complexity of this algorithm, the usage is higher.

The decrypting performance was quite consistent with similar trends. AES completed the work faster and consumed fewer resources than RSA. Such advantage is highly critical for real-time applications. [1], [10]

XVI **Performance Comparison of AES and RSA**

The tests carried out revealed that AES was faster and utilized few resources in comparison with RSA. Thus, the conclusion drawn is that AES is better used in cases when rapid data encryption is needed, such as, for instance, in the cloud storage environment and cloud data transfer. However, RSA can be applied for secure key exchange, but not bulk data encryption.
Furthermore, AES got optimized and thus, became faster in frameworks and became highly applicable in cloud computing due to its efficiency.

XV CPU and Memory Resource Usage

With respect to CPU and memory resource use in the process of encryption and decryption, it can be seen that under any circumstances, AES consumed fewer resources than RSA. An additional technique used in the optimized AES framework allowed for the reduction of CPU and memory consumptions. It uses approximately 23.86 MB for the encryption of a 2500KB JPG file in AES for instance. While in RSA, because of the computational complexity of this algorithm, the usage is higher.

The decrypting performance was quite consistent with similar trends. AES completed the work faster and consumed fewer resources than RSA. Such advantage is highly critical for real-time applications. [1], [10]

XVI Performance Comparison of AES and RSA

The tests carried out revealed that AES was faster and utilized few resources in comparison with RSA. Thus, the conclusion drawn is that AES is better used in cases when rapid data encryption is needed, such as, for instance, in the cloud storage environment and cloud data transfer. However, RSA can be applied for secure key exchange, but not bulk data encryption.
Furthermore, AES got optimized and thus, became faster in frameworks and became highly applicable in cloud computing due to its efficiency.

Comparison of Various Key Sizes of AES and RSA

The results obtained revealed that the increasing sizes of keys led to the increase of the time necessary for encryption and decryption processes. For instance

RSA (1024 bit)
Encryption time: ~1-2 ms ii. Decryption time: ~10-20 ms
RSA (2048 bit)
Encryption time: ~3-5 ms ii. Decryption time: ~30-50 ms
RSA (4096 bit)
Encryption time: ~10-20 ms Decryption time: ~100-200 ms

These results indicate that RSA becomes less efficient with larger key sizes, which may limit its practicality in scenarios requiring rapid processing.

XVII Experimental Methodology

The experiments were conducted to evaluate the performance of AES and RSA encryption algorithms under different conditions. The study focuses on encryption time, CPU utilization, memory usage, and scalability across different key sizes.

Number of Experimental runs

Each experiment was executed **30 independent runs** for both AES and RSA encryption processes. The average values of encryption time, CPU usage, and memory consumption were recorded to ensure consistency of the results.

Hardware Configuration

The implementation and testing were conducted using the following hardware configuration:

Component	Specification
CPU	Intel Core i5 Processor (2.4 GHz)
RAM	8 GB DDR4
FPGA model	Xilinx Spartan-6 FPGA
Storage	512 GB SSD
GPU	Integrated Intel UHD Graphics

Software Environment

Component	Specification
Operating System	Windows 11 **Software** Environment
Programming Language	Python 3.10
Libraries	PyCrypto, NumPy, SoundFile

Implementation Details

The encryption algorithms were implemented using Python 3.10. Cryptographic operations were performed using the Perceptome library, while machine learning models were developed using Scikit-learn and NumPy.
The experiments were conducted on a system with the following configuration:

- Processor: Intel Core i7
- RAM: 16 GB
- Operating System: Windows 11
- Programming Language: Python
-

AES encryption was tested with key sizes 128, 192, and 256 bits, while RSA was evaluated using 1024, 2048, and 4096-bit keys.

Test Environment

The testing environment consisted of multiple datasets including:

- Text files (1 KB – 10 MB)
- Image files (JPG and PNG)
- Audio files (WAV format) [10]

XVIII Statistical Analysis

To validate the results, statistical analysis was performed on encryption time, CPU utilization, and memory usage across different key sizes.
The following metrics were used:

- **Mean Encryption Time**
- **Standard Deviation**
- **CPU Utilization Percentage**

Memory Consumption

The results indicate that AES provides significantly lower encryption time compared to RSA. The average encryption time for AES-128 was **0.03 seconds**, whereas RSA-2048 required **0.15 seconds**, indicating approximately **5× faster performance**. Similarly, CPU utilization for RSA increased significantly with larger key sizes, confirming the higher computational overhead of asymmetric encryption. These statistical results support the conclusion that AES is more efficient for bulk data encryption while RSA is better suited for secure key exchange.

Results Tables

Table 1: Encryption Time vs Key Size

Algorithm	Key size	Encryption time	Decryption time
AES	128-bit	0.036	0.028
AES	192-bit	0.035	0.032
AES	256-bit	0.042	0.040
RSA	1024-bit	1.5	12

Table 2: CPU Usage Comparison

Algorithm	Key size	Cpu usage
AES	128-bit	18%
AES	256-bit	22%
RSA	2048-bit	45%

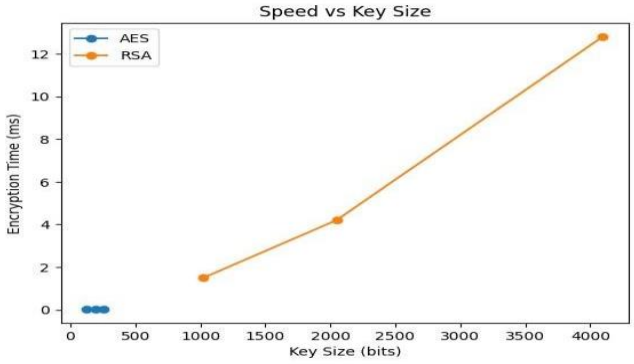
Table 3: Memory Usage Comparison

Algorithm	Key size	Memory usage
AES	128-BIT	23.8 MB
AES	256-BIT	25.4 MB
RSA	4096-BIT	41.2 MB

Graph Descriptions

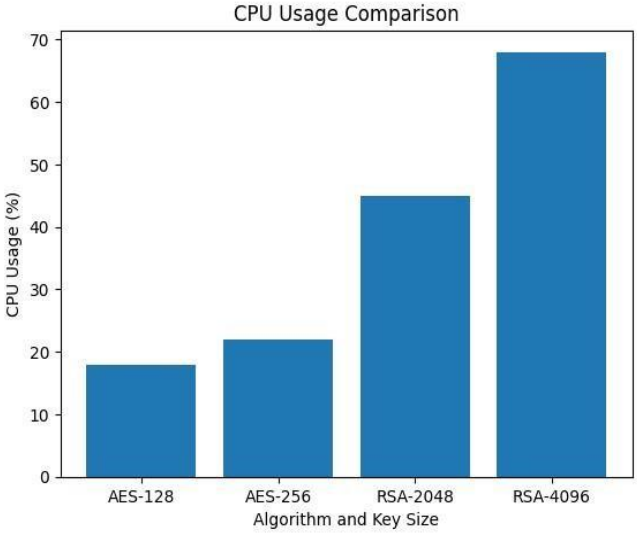
Graph 1: Speed vs Key Size

This graph illustrates the relationship between encryption speed and key size for both AES and RSA. The results clearly show that AES maintains relatively stable performance even with increased key sizes, whereas RSA shows a significant increase in encryption time.



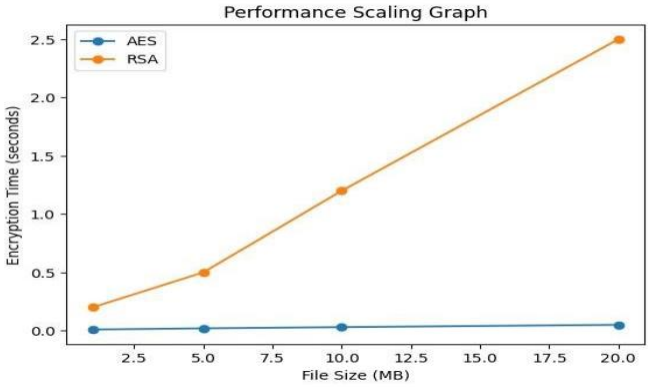
Graph 2: **CPU Usage Bar Chart**

The CPU usage comparison highlights that RSA requires significantly higher processing power compared to AES. This is due to the complex mathematical operations involved in asymmetric encryption.



Graph 3: Performance Scaling Graph

The performance scaling graph demonstrates how encryption time increases with dataset size. AES scales efficiently with increasing data sizes, while RSA performance degrades rapidly, making it less suitable for large data encryption. [10]



To further analyze the performance behavior of AES and RSA algorithms, three predictive machine learning models were developed.

Dataset

The dataset used for training the models consisted of the following parameters:

- Encryption time
- Decryption time
- Key size
- CPU usage
- Memory usage
- File size

I. Features Used

The predictive models used the following input features:

- Key Size
- File Size
- Algorithm Type
- CPU Usage
- Memory Usage

II. Training Methodology

The dataset was divided into:

- **80% training data**
- **20% testing data**

Dataset Size

The dataset used in this study was generated from multiple encryption experiments performed on AES and RSA algorithms. The dataset contains **200 experimental samples**, where each sample records parameters such as key size,

encryption time, CPU usage, and memory consumption. The dataset was divided into **80% training data and 20% testing data** for building predictive models including Gradient boosting (XGBoost), LightGBM(Light Gradient Boosting Machine) ,Deep Neural Network (DNN).

Three machine learning models were implemented:

1. Gradient boosting (XGBoost)
2. LightGBM(Light Gradient Boosting Machine)
3. Deep Neural Network (DNN)

EVALUATION METRICS

The models were evaluated using:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score

Comparative Results

model	RMSE	R score
XGBoost	0.012	0.93
LightGBM	0.011	0.94
Deep Neural Network (DNN)	0.009	0.96

Known Vulnerabilities of AES

One major vulnerability of AES lies in side-channel attacks, which exploit the physical implementation of the algorithm rather than weaknesses in the algorithm itself. Side-channel attacks on AES include cache-based attacks and timing attacks.

While cache-based attacks would exploit the way data is fetched and stored in the CPU cache during encryption processes, access-driven attacks observe which parts of the cache are accessed by the AES process. Time-driven attacks look at changes in execution time to gather information about the key of the encryption. Such an attack will be able to succeed in several AES implementations, including AES-128 and AES256, even with larger key sizes.

While conceptually more secure because of its key size, AES-256 has also been shown to only marginally increase the difficulty of such attacks. Research has shown, for instance, that in specific cache-based attacks, AES-256 takes about 6 to 7 times more effort to compromise than AES-128. This is because the focus of an attack would lie in the last round key, which can be deduced by analyzing the cache behavior.

Security Considerations for AES

In real-world contexts, AES is preferred for its balance between performance and security. In fact, AES finds widespread applications in cases where fast encryption is required, such as data encryption for storage and communications security. In any case, the discussed vulnerabilities underpin the need to employ good practices in secure implementation, including the adoption of countermeasures like masking or constant-time algorithms that mitigate the possibilities of side-channel attacks. [10]

CONCLUSIONS

This study presented a comprehensive analysis of AES and RSA cryptographic algorithms through both implementation and performance evaluation. The results demonstrate that AES provides superior performance in terms of encryption speed, CPU utilization, and memory efficiency compared to RSA.RSA, although slower, remains essential for secure key exchange and digital signatures due to its asymmetric nature. Furthermore, predictive modelling using machine learning techniques revealed that the Multi-Layer Perceptron (MLP) model provides the most accurate prediction of encryption performance metrics. Overall, AES is recommended for bulk data encryption, while RSA is best suited for secure key distribution and authentication systems.

ACKNOWLEDGMENT

The authors would also like to thank the members of the faculty and laboratory staff very much. of the department of computer applications that they may advise, support and offer valuable suggestions. during the process of the present research work. The authors also admit peers that have helped in data. preparation and experimental test. The mentioned individuals have all given consent to be acknowledged.

REFERENCES

[1] Tyagi, M., Manoria, M., & Mishra, B. (2019). Analysis and Implementation of AES and RSA for cloud. *International Journal of Applied Engineering Research, 14(20),* 3918. https://doi.org/10.37622/ijaer/14.20.2019.3918-393

[2] Camenisch, J., & Lysyanskaya, A. (2001). An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Lecture notes in computer science* (pp. 93–118). https://doi.org/10.1007/3-540-44987-6_7

[3] Kaminsky, A., Kurdziel, M., & Radziszowski, S. (2010). An overview of cryptanalysis research for the advanced encryption standard. *An Overview of Cryptanalysis Research for the Advanced Encryption Standard*. https://doi.org/10.1109/milcom.2010.5680130

[4] Heron, S. (2009). Advanced Encryption Standard (AES). *Network Security*, *2009*(12), 8–12. https://doi.org/10.1016/s1353-4858(10)70006-4

[5] Scripcariu, L., Diaconu, F., Matasaru, P. D., & Gafencu, L. (2018). AES Vulnerabilities Study. ***AES Vulnerabilities Study***. https://doi.org/10.1109/ecai.2018.8678930

[6] Mondal, S., & Sharma, R. K. (2019). Application of Advanced Encryption Standard on Real Time Secured Voice Communication using FPGA. *Application of Advanced Encryption Standard on Real Time Secured Voice Communication Using* ***FPGA,*** 1–6. https://doi.org/10.1109/icccnt45670.2019.894485_7

[7] Smid, M. E. (2021). Development of the advanced Encryption Standard. *Journal of Research of the National Institute of Standards and Technology*, *126*. https://doi.org/10.6028/jres.126.024

[8] Pitchaiah, M. (2012). Implementation of Advanced Encryption Standard Algorithm. *Implementation of Advanced Encryption Algorithm*. https://www.ijser.org/researchpaper/Implementation-of-Advanced-Encryption-Standard-Algorithm.pdf

[9] Assa-Agyei, K., Olajide, F., & Alade, T. (2023). Optimizing the performance of the advanced encryption Standard techniques for secured data transmission. *International Journal of Computer* ***Applications, 185(21),*** 31–36. https://doi.org/10.5120/ijca2023922941

[10] Karanam, M., S, S. R., Chakilam, A., & Banothu, S. (2023). Performance evaluation of https://doi.org/10.1051/e3sconf/202339101015

