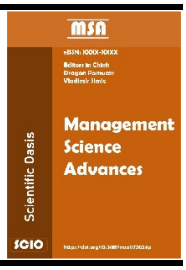
SCIENTIFIC OASIS

Management Science Advances

Journal homepage: www.msa-journal.org
eISSN: xxxx-xxxx



A Comparative Analysis of AES, RSA, and 3DES Encryption Standards based on Speed and Performance

Mirza Hamza Munir Baig[1], Hafiz Burhan Ul Haq[1*], Waleed Habib[1]

[1] Department of Information Technology, Faculty of, Computer Sciences, Lahore Garrison University, Lahore, Pakistan

ARTICLE INFO

Article history:
Received 10 November 2024
Received in revised form 14 November 2024
Accepted 16 November 2024
Available online 17 November 2024

Keywords:
Encryption; Symmetric Cryptography; Asymmetric Cryptography; Hashing; Information Security

ABSTRACT

Information security is important in the digital era, with encryption emerging as a fixed solution to safeguard sensitive data from unauthorized access. This paper provides an overview of encryption by categorizing cryptographic algorithms into symmetric and asymmetric, along with hashing. Symmetric cryptography uses a single key for encryption and decryption, and asymmetric cryptography utilizes distinct public and private keys. Hashing maps input messages into compact bit strings. The literature survey explores the performance analysis of encryption algorithms, focusing on Rivest Shamir Adleman (RSA), Data Encryption Standard (DES), and Advanced Encryption Standard (AES), considering parameters like computation time and memory usage. Furthermore, the comparative analysis of encryption algorithms, including AES, RSA, and 3DES, is discussed. The paper delves into the motivation for cryptography in secure document transfer systems, emphasizing the need for confidentiality, integrity, and availability. Finally, the study presents detailed insights into AES, RSA, and 3DES encryption standards, highlighting their principles and applications.

1. Introduction

In the current world where information is considered to be secure, only a few persons can access such information, and this is made possible by encryption. Since its inception, the crave for tough encryption algorithms as the technology advanced has remained high. It is necessary to point out that among the methods for protecting data, the best information security solution is encryption, which is capable of protecting different types of data.

Security can be defined as the mechanism that makes it impossible for any unauthorized or undesirable access to information and services [1]. As developers continue to advance new applications, there are more of them enlisted at Play Store and many can be pirated with similar looks and features hence encryption does play a great deal at this juncture [2].

Encryption is the technique of converting normal readable text into a non-readable form by means of a special technique called algorithm and a key. Such information is scrambled and can only

* *Corresponding author.*
E-mail address: burhanhashmi64@lgu.edu.pk

https://doi.org/10.31181/msa1120244

be deciphered back when using the proper key by the receiver who intended to use the scrambled information in the first place [3]. Generally speaking, this can be divided into two major categories: based on the usage of the key during encryption and decryption in cryptographic algorithms, the types are symmetric, asymmetric, and hashing [4]. The cryptography types are mentioned below in Figure 1.

The major difference between asymmetric and symmetric cryptography systems is that symmetric cryptography system uses the same key to encrypt as well as to decrypt the information and therefore makes the algorithm less complex than asymmetric cryptography systems [5].

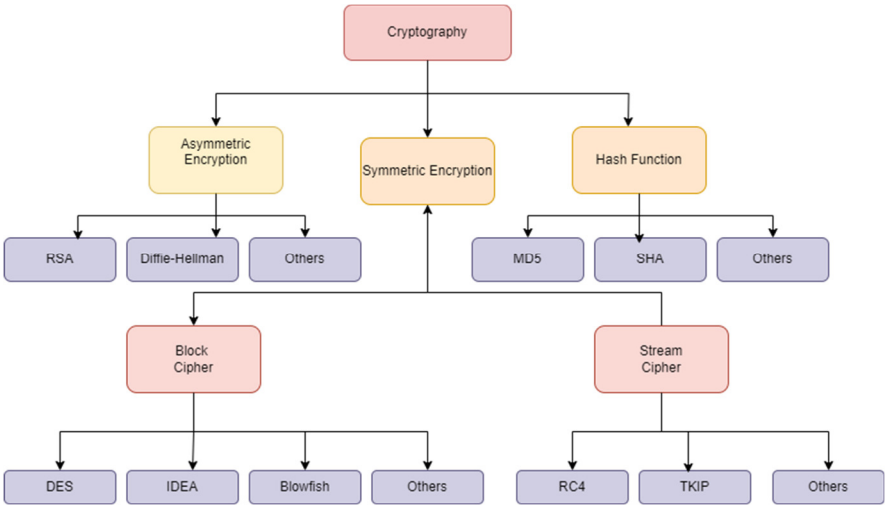


Fig. 1. General idea of encryption and decryption

On the basis of input data, symmetric key cryptography algorithms are divided into two categories; i.e. block ciphers and stream ciphers. A block cipher-based system processes or encrypts data on a fixed-size group of bits called a block. Stream cipher-based systems process data on a stream of bits [6]. Figure 2 shows symmetric encryption.

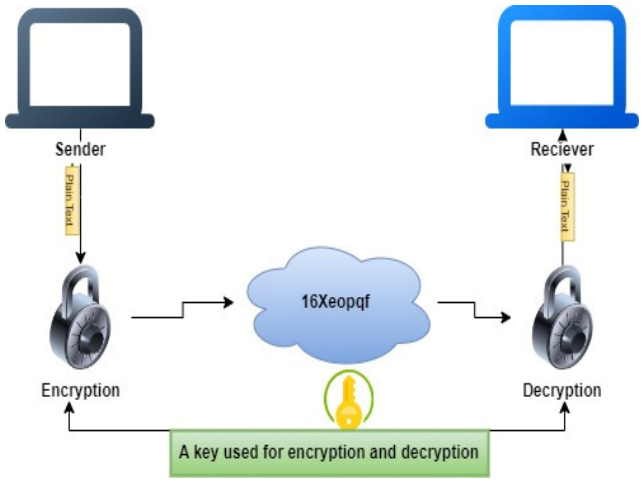


Fig. 2. Symmetric encryption

In asymmetric cryptography, there is the use of two keys. One is for encryption and the other is for decryption although both are private keys [7]. Within any asymmetric key cryptographic system, there are two keys with one of those keys known only to the key-holder and the other commonly known as a public key. Used for encryption is the public key, whereas, for decryption of the text that

has been encrypted, a secret key is used. Mathematically, these keys are in some way connected. While the asymmetric systems are showing more security than symmetric ones they may not be as suitable for the large size documents. This is because the speed is relatively slow as compared to the systems based on symmetric keys. Also, they log a higher rate of CPU usage [6]. Figure 3 illustrates the process of asymmetric encryption.

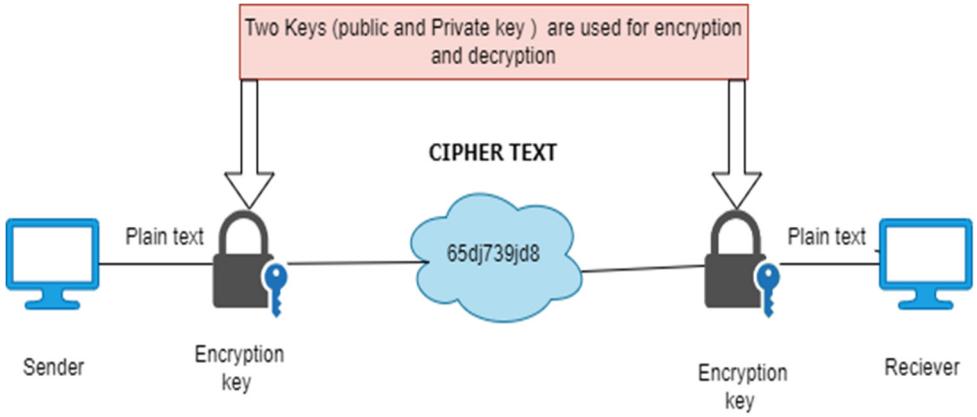


Fig. 3. Asymmetric encryption

The third kind of cryptographic algorithm is hashing. The process involves mapping an input message into a compact, fixed-size bit string called a hash [8].

2. Literature Survey

Keeping our data more secure has become very important in today's generation. Some of the related research papers are already working on another topic which is the performance analysis of encryption algorithms on different platforms. This work entails a literature review on Rivest Shamir Adleman (RSA), Data Encryption Standard (DES), and Advanced Encryption Standard (AES) encryption algorithms. We aim to elaborate on the various encryption algorithms that were used in information security. As the intelligence of humans arose, cryptography also became more complicated to provide information security [9].

In this survey, the performances of various types of security algorithms at cloud networks and single processors with varying data sizes are compared. As the paper attempted to discover, there would be certain advantages that can be attributed to the utilization of cloud resources to implement alarm security such as speed-up ratio [10]. Comparative analysis of the three algorithms with respect to RSA, DES, and AES under parameters such as computation time, memory usage, and output byte was observed [11,12].

In our case we do, for sure, require an exceptionally secure means of document transfer; i.e. the use of cryptography is occasioned by the need to shield the communication. The applications where data has to be transferred in encrypted form have been used enormously in the financial sector and other businesses where privacy and security concerns are vital; so based on the security in AES, DES, DSS, and RSA the systems have been used for protecting information during communication and for fast and efficient identification of files [13]. Cryptography explanation in the context of the CIA triad and each of the foregoing standards has been accomplished in a review paper on DES, AES, and RSA [14]. Figure 4 summarizes the literature.

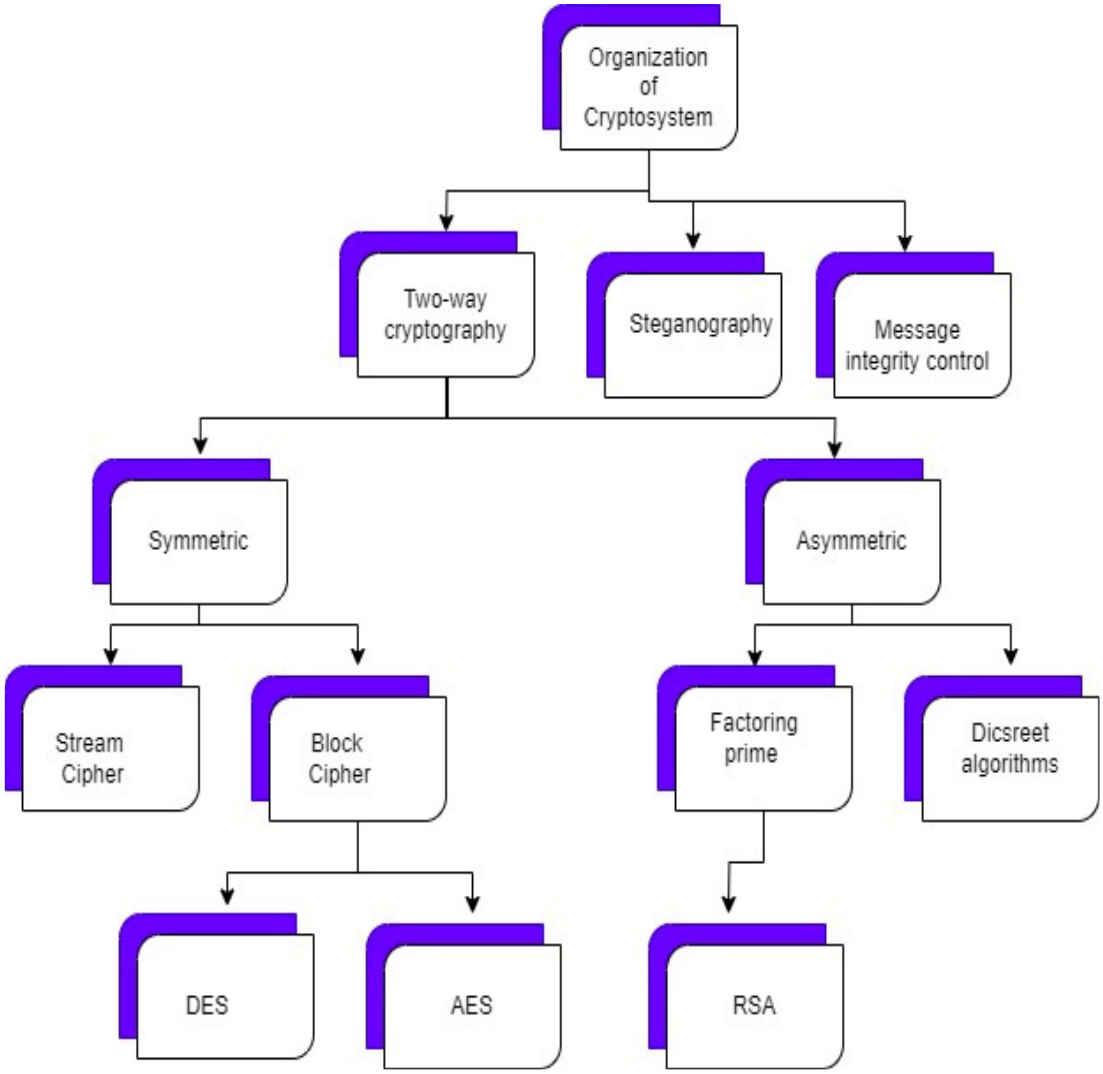


Fig. 4. Hierarchy of DES, AES, and RSA encryption standards

3. AES, RSA and 3DES

3.1 Advanced Encryption Standard

AES is a type of cryptographical algorithm in which the same key is used for encoding and decoding the data [15]. Currently, it is a symmetric key encryption algorithm that has replaced DES as the new-generation encryption standard. AES was built to become a standard for encryption by the U.S. National Institute of Standards and Technology (NIST). It is a block cipher operating on fixed-size blocks of data and supporting three different key sizes. A possible size is 128, 192, or 256 bits. AES is secure and fast hence making it adopted in the encryption of sensitive information thus making it regarded as the standard encryption algorithm [5]. Among them, all published by NIST are the submissions and all the analyses that did not attain the classification level. The AES algorithm was the new generation of block ciphers, and block size enhanced significantly from the preceding 64-bit to the latest 128; and, in the same fashion, from 128 to 256 keys. This was partly due to DES and RC-5 exhaustive key search demonstrations carried out at only 64 bits [17,18] as mentioned in Figure 5.

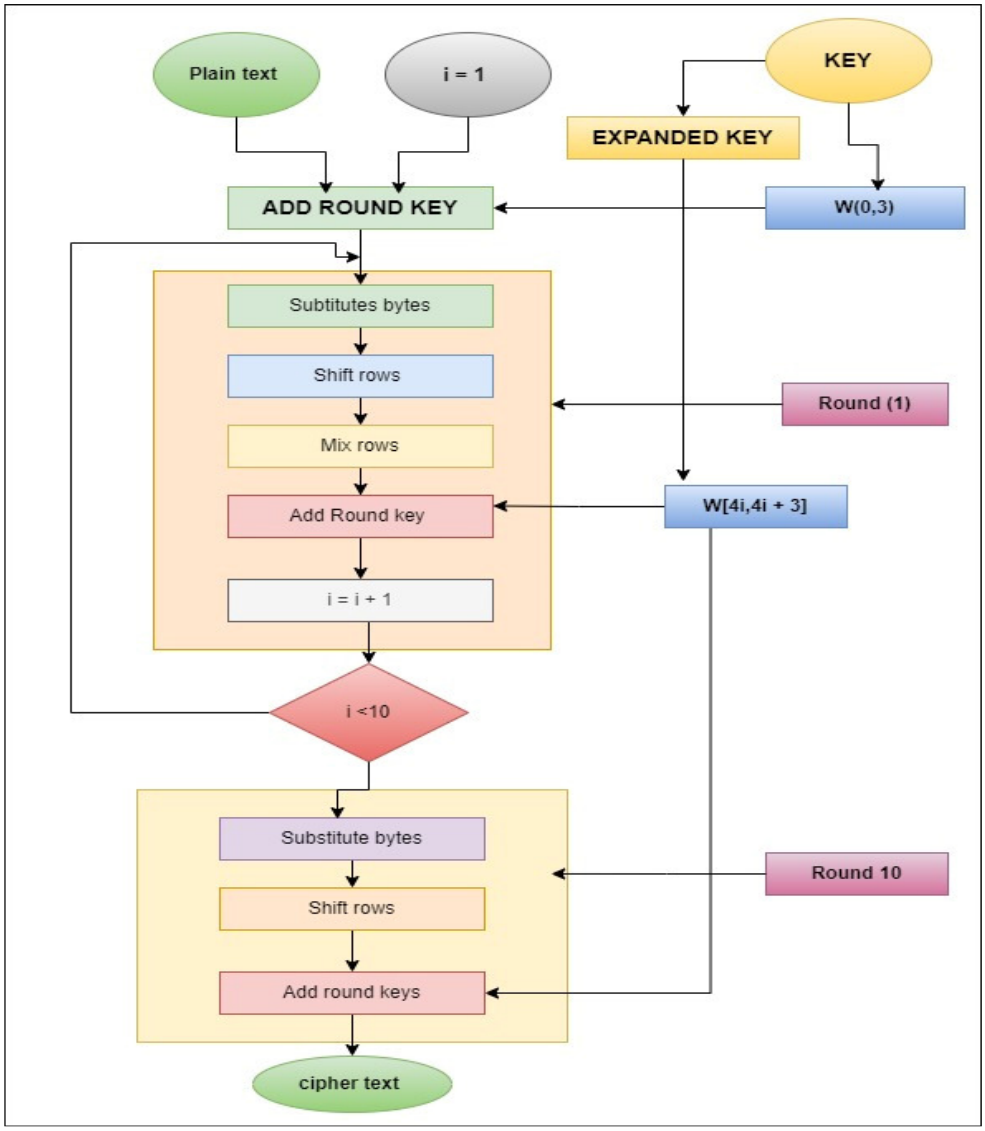


Fig. 5. Processing of AES

3.2 Rivest Shamir Adleman

RSA is an algorithm for asymmetric key encryption named after its inventors; i.e. all three of them namely, Ron Rivest, Adi Shamir, and Leonard Adleman. It was published in 1977. RSA is very central in protecting information, especially in secured communication, digital signatures, and key exchange. Its security lies in the computation of the product of two large prime numbers; i.e. it offers an effective means of encryption and decryption over the incompetent communication channels [19,20].

RSA employs a block of a given size and a key of a given size, too. A mobile communication scheme that is based on the number theory and classified under the asymmetric, public key cryptosystems. It creates the public and private keys with the help of two prime numbers only. These two specifically distinct keys are used for encryption and decryption. In the case where a particular message is to be passed to the receiver, it will be encrypted using the public key and in parallel, the same will be decrypted using a sender's private key [21]. The multiple blocks process of RSA can be depicted in Figure 6.

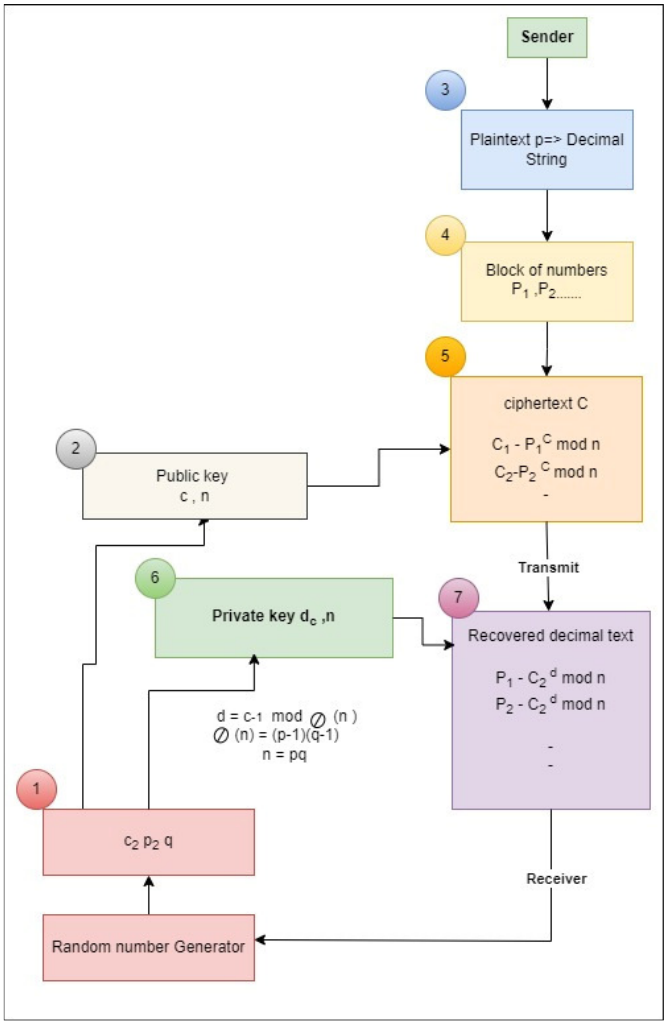


Fig. 6. RSA multiple blocks processing

3.3 Triple Data Encryption Standard

3DES stands for the triple data encryption algorithm and is a symmetric block cipher applying the DES algorithm three times to every block. Hence, it becomes more secure than the original DES [22]. 3DES is an encryption process where plaintext is encrypted three times in succession using two or three different keys by applying the DES. This multiple encryption process enhances security, addressing the key length limitations of the original DES. Despite its enhanced security, 3DES is gradually being replaced by more modern encryption algorithms due to advances in computing power [23]. Figure 7 shows the processing of DES.

Algorithm of DES	**Algorithm of RSA**	**Algorithm of AES**
Inputs**: 64-bit plaintext, 56-bit key* ***Output**: 64-bit ciphertext* ***Start: ***Initial Setup*** *plaintext = get_64_bit_plaintext()* *key = get_56_bit_key()* ***Step 1: Initial Permutation (IP)*** *initial_permuted_text= initial_permutation(plaintext)*	***Inputs**: plaintext (string), key_size (integer)* ***Outputs**: ciphertext (list of integers), decrypted_text (string)* *e, n: Public key.* *d, n: Private key.* *P: Block of plaintext numbers.* ***Start:***	***Inputs:*** *plaintext: 128-bit block (string or byte array)* *key: encryption key (128-bit, 192-bit, or 256-bit)* ***Outputs:*** *ciphertext: 128-bit encrypted message* ***Start:***

```
    Divide permuted text into two 32-bit halves
L = initial_permuted_text[:32]  # Left half
R = initial_permuted_text[32:]  # Right half
# Generate 16 subkeys from the 56-bit key
subkeys = generate_subkeys(key)  # K1, K2, ..., K16 (48-bit each)
# Step 2: 16 Rounds of Encryption

 for i in range(1, 17):
        # Expand R from 32 to 48 bits
        expanded_R = expansion_permutation(R)
        # XOR with subkey Ki
        xor_result = xor(expanded_R, subkeys[i - 1])
        # S-box substitution (48 -> 32 bits)
        substituted = s_box_substitution(xor_result)
        # Straight permutation
        Permuted = straight_permutation (substituted)
        temp = R  # Save the value of R
        # XOR L with permuted
        R = xor(L, permuted)
        result
        L = temp  # Swap L and R
  end for
After 16 rounds, swap the halves
pre_output = R + L
# Step 3: Inverse Initial Permutation (IP^-1)
ciphertext = inverse_initial_permutation(pre_output)
# Output the ciphertext (64-bit)
return ciphertext
```

```
def start():
        plaintext = get_input_text()
        key_size = get_key_size()
        # Generate public and private keys
        public_key,private_key= generate_keys(key_size)
        # Encrypt the plaintext
        ciphertext= encrypt(plaintext, public_key)
        # Decrypt the ciphertext
        decrypted_text= decrypt(ciphertext, private_key)
        print("Decryptedtext:", decrypted_text)
 return decrypted_text
def generate_keys(key_size):
        p,q=generate_prime (key_size // 2),
        generate_prime (key_size // 2)
        n = p * q
        phi_n = (p - 1) * (q - 1)
        e = 65537
        # Common choice for e
        d = mod_inverse(e, phi_n)
 return (e, n), (d, n)
def encrypt(plaintext,
    public_key):
        e, n = public_key
        blocks = text_to_blocks(plaintext, n)
return [(block ** e) % n for block in blocks]
 def decrypt(ciphertext,
    private_key):
        d, n = private_key
        blocks = [(block ** d) % n for
                block in
         ciphertext]
return blocks_to_text(blocks)

 def mod_inverse(e, phi_n):
 return extended_gcd(e, phi_n)[0] % phi_n
# Additional helper functions like `text_to_blocks`,
`blocks_to_text`,
#  `generate_prime`, and `extended_gcd` should also be defined.
# Main function
start()
```

```
def AES_encrypt(plaintext, key):
    state = add_round_key(plaintext,
     key_expansion(key, 0))
      # Initial round key
     for i in range(1, 10):  # Main rounds
            state = substitute_bytes(state)
            state = shift_rows(state)
            state = mix_columns(state)
            state = add_round_key(state,
             key_expansion(key, i))
      end for
    # Final round (no MixColumns)
     state = substitute_bytes(state)
     state = shift_rows(state)
      ciphertext= dd_round_key(state,
      key_expansion(key, 10))
return ciphertext
# Helper functions
def key_expansion(key, round_num):
    # Expand key to get the round key
     for the given round number
 return  expand_key(key)[round_num *
        4:(round_num + 1) * 4]

Def add_round_key(state,
     round_key):
return xor(state, round_key)

def substitute_bytes(state):
return s_box_substitution(state)

def shift_rows(state):
 return shift_rows_operation(state)

def mix_columns(state):
return mix_columns_operation(state)

# Main execution
plaintext = get_plaintext_input()
key = get_key_input()
ciphertext  =  AES_encrypt(plaintext, key)
print("Ciphertext:", ciphertext)
```

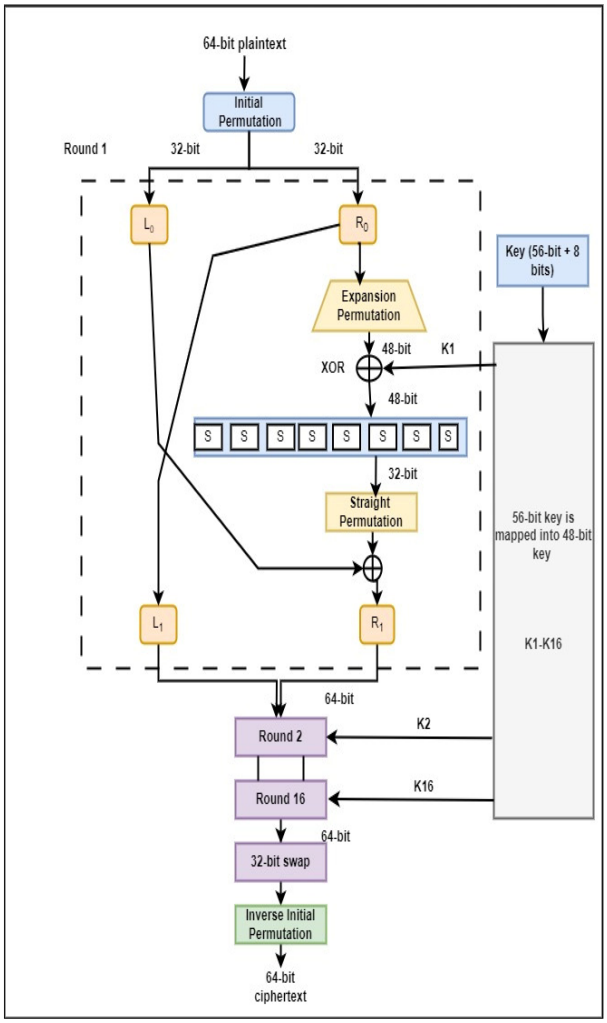


Fig. 7. General depiction of DES

4. Results

4.1. Results of RSA

The given output consists of the OpenSSL command to benchmark the performance of RSA encryption/decryption messages with various key sizes. Some of the data output encompasses the number of operations carried out per second for distinct key sizes and the time taken for the operations. RSA performance benchmark is mentioned in Table 1.

Table 1
Performance of RSA

Key size (bits)	Private sign ops	Public verify ops	Private encrypt ops	Private decrypt ops	Time (s)
512	**171,410**	**1,361,549**	**1,052,593**	**72,116**	**9.70**
1024	**45,216**	**636,894**	**562,237**	**38,129**	**9.86**
2048	**7,140**	**229,613**	**216,295**	**6,716**	**7.72**
3072	**2,401**	**115,135**	**111,789**	**2,371**	**8.27**
4096	**1,099**	**68,179**	**65,328**	**1,077**	**8.03**
7680	**119**	**20,379**	**20,438**	**116**	**8.13**
15360	**22**	**5,326**	**5,269**	**22**	**8.09**

The reference data obtained shows the characteristics of RSA operations depending on the key length. Indeed, greater key sizes are associated with slower operations due to the increase in the computational load of the operations. These results may be helpful for researchers and practitioners in the area of cryptography when determining the necessary key sizes together with their security class and the available computational power.

4.2 Results of AES

The output shown here summarizes the OpenSSL performance in different operations related to AES with different key sizes. These are given in kilobytes per second (KB/s) and show the throughput that was obtained for each setting. AES performance benchmark results as given in Table 2.

Table 2
Performance of AES

Data size (Bytes)	AES-128-CBC (KB/s)	AES-192-CBC (KB/s)	AES-256-CBC (KB/s)
16	**655,700.04**	**282,367.28**	**249,350.52**
64	**1,228,004.46**	**563,391.15**	**429,989.71**
256	**1,036,851.16**	**645,022.34**	**433,893.55**
1024	**957,730.33**	**619,488.65**	**423,229.39**
8192	**508,868.57**	**508,868.57**	**423,987.45**
16384	**500,841.30**	**495,412.70**	**423,987.45**

Analyzing the obtained benchmark, it is possible to see the throughput of AES encryption (CBC mode) depending on key size and block size. In general, bigger blocks will lower the achieved throughput because each encryption function operates with a larger amount of data. In addition to this, as hypothesized above, encryption with large key sizes especially 192 and 256-bit key sizes bear the general tendency of having higher levels of performance cost than 128-bit key size. It is possible to consider these results as indicators of AES encryption operations' efficiency and speed in different conditions. Therefore, researchers and practitioners can apply this information directly about the choice of an appropriate key size and block size for the desired security and performance levels.

4.3 Results of DES

The following output depicts the OpenSSL performance evaluation for DES and DES-ede3 by performing operations at different block sizes (Table 3).

Table 3
Performance of DES

Data size (Bytes)	DES-ede3 (3DES) (ops/s)	DES-CBC (throughput (KB/s)
16	**1,750,916**	**0.00**
64	**460,434.7**	**27,973.59**
256	**95,713.89**	**0.00**
1024	**12,782.49**	**0.00**
8192	**1,606.73**	**0.00**
16384	**883.67**	**0.00**

For DES-ede3, the benchmark data provides the throughput in which kilobytes per second for a given block size. Regarding this classification, the reader must understand that 3DES is a symmetric-key block cipher, and the benchmark offers insights into the effectiveness of its operations depending on the blocks' size. In turn, it seems as if DES-CBC is tagged as being only

supported when the block size is 64 bytes. There are several modes of DES among them is DES-CBC, which is used as cipher block chaining mode. It was observed from the output that the OpenSSL version used does not support DES-CBC mathematical block size in all 64 bytes. These results can be useful in evaluating the results of the external testing of DES and 3DES security in various situations. From this information, both the researchers and the practitioners can be informed about the suitability of such encryption algorithms depending on the security needs and throughput.

5. Conclusion

In conclusion, this paper highlighted how encryption is crucial for information security, focusing on a comparison of RSA, DES, and AES algorithms. The literature survey stressed the importance of cryptography in secure document transfers. The benchmark results gave useful insights into how these algorithms perform, showing differences in efficiency based on key and block sizes. While RSA is slower with larger keys, AES varies in throughput, and 3DES addresses key limitations but faces gradual obsolescence. These findings help practitioners make informed decisions, finding the right balance between speed and security in their encryption choices.

References

[1] Forouzan, B. A. (2007). *Data Communication and Networking*. Huga Media.
[2] Ammiudin, A. (2021). Android assets protection using RSA and AES cryptography to prevent app piracy. In *Proceedings of the 2021 IEEE International Conference on Information Technology and Electrical Engineering (ICOIACT)* (pp. 1–6). https://doi.org/10.1109/ICOIACT50329.2020.9331988.
[3] Singh, S. L. (2011). *The Code Book: The Secret History of Codes and Code-Breaking*. HarperCollins Publishers.
[4] Asaad, R. R., Abdulrahman, S. M., & Hani, A. A. (2017). Advanced Encryption Standard Enhancement with Output Feedback Block Mode Operation. *Academic Journal of Nawroz University, 6*(3), 1-10. https://doi.org/10.25007/ajnu.v6n3a70.
[5] Aumasson, J.-P., & Green, M. D. (2017). *Serious Cryptography: A Practical Introduction to Modern Encryption*. No Starch Press.
[6] Alenezi, M. N., Alabdulrazzaq, H., & Mohammad, N. Q. (2020). Symmetric encryption algorithms: Review and evaluation study. International Journal of Communication Networks and Information Security, 12(2), 256-272.
[7] Delfs, H., Knebl, H., & Knebl, H. (2002). *Introduction to Cryptography: Principles and Applications*. Springer. https://doi.org/10.1007/978-3-662-47974-2.
[8] Katz, J., & Lindell, Y. (2014). *Introduction to Cryptography*. Chapman and Hall/CRC.
[9] Sood, R. H. K. (2023). A literature review on RSA, DES, and AES encryption algorithms. *Journal of Information Security, 12*, 37–50. https://doi.org/10.56155/978-81-955020-3-5-07.
[10] Arora, A., & Kumar, T. (2012). Evaluation and comparison of security issues on cloud computing environment. In *Proceedings of the World Conference on Computer Science and Information Technology* (pp. 1–5). https://doi.org/10.1109/WCSIT.2012.XX.
[11] Wang, Y., Chen, C., & Jiang, Q. (2019). Security algorithm of Internet of Things based on ZigBee protocol. *Cluster Computing, 22*, 14759-14766. https://doi.org/10.1007/s10586-018-2388-4.
[12] Elminaam, D. S. A., Kader, H. M. A., & Hadhoud, M. M. (2008). Performance evaluation of symmetric encryption algorithms. *IJCSNS International Journal of Computer Science and Network Security, 8*(12), 280-286. https://doi.org/10.22937/IJCSNS.2008.8.8.2.
[13] Singh, G. P. (2021). A DES, AES, DSS, and RSA-based security system for protecting sensitive information during communication and providing fast, reliable file identification. *Webology, 18*. https://doi.org/10.14704/web/v18n2/a52.
[14] Hamza, A., & Kumar, B. (2020). A review paper on DES, AES, RSA encryption standards. In *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)* (pp. 333-338). IEEE. https://doi.org/10.1109/SMART50582.2020.9336800.
[15] Tayde, S., & Siledar, S. (2015). File encryption, decryption using AES algorithm in android phone. International Journel of Advanced Research in computer science and software engineering, 5(5), 53-56.
[16] National Institute of Standards and Technology (NIST). (2001). *FIPS PUB 197: Advanced Encryption Standard (AES)*.

[17] Singh, G. (2015). A study of encryption algorithms (RSA, AES, 3DES, DES) for information security. *International Journal of Computer Applications, 115*(7), 1-5. https://doi.org/10.5120/11507-7224.
[18] Commey, D., Griffith, S., & Dzisi, J. (2020). Performance comparison of 3DES, AES, Blowfish and RSA for Dataset Classification and Encryption in Cloud Data Storage. International Journal of Computer Applications, 177(40), 17-22. https://doi.org/10.5120/ijca2020919897.
[19] Mandal, A. K., Parakash, C., & Tiwari, A. (2012, March). Performance evaluation of cryptographic algorithms: DES and AES. In 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science (pp. 1-5). IEEE. https://doi.org/10.1109/SCEECS.2012.6184991.
[20] Juniawan, F. P. (2016, August). RSA implementation for data transmission security in BEM chairman E-voting Android based application. In 2016 1st International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE) (pp. 93-98). IEEE. https://doi.org/10.1109/ICITISEE.2016.7803054.
[21] Stallings, W. (2017). *Cryptography and Network Security: Principles and Practices*. Pearson.
[22] Barker, E., & Roginsky, A. (2018). Transitioning the use of cryptographic algorithms and key lengths (No. NIST Special Publication (SP) 800-131A Rev. 2 (Draft)). National Institute of Standards and Technology.
[23] National Institute of Standards and Technology (NIST). (1999). Data encryption standard. Federal Information Processing Standards Publication, 112, 3.