Spring 2024 CS 412: Algorithms and Design Project
Analyzing Design Techniques of Cryptographic Algorithms

Iqra Azfar
Habib University
ia07614@st.habib.edu.pk

Rabia Shahab
Habib University
rs07528@st.habib.edu.pk

Zainab Raza
Habib University
zr07532@st.habib.edu.pk

***Abstract*—This report analyzes various design techniques for three cryptographic algorithms: RSA, DES, and AES. This paper also explores the modified approaches for RSA such as Dynamic Approach using Subset Sum Cryptography and MRSA based on Random Prime Numbers; block cipher techniques for DES; and block cipher and further optimized techniques for AES. In this report, we will be examining the strengths, weaknesses, time complexities, and potential attack strategies with their complexities for each algorithm. Moreover, we aim to provide a detailed explanation of these cryptographic algorithms along with their design techniques, so better decisions can be made to select a secure cryptographic application.**

I. Introduction

The cryptography field focuses on concealing information in the form of secret writing by methods of encryption and decryption. It plays a crucial role in building message confidentiality and developing integrity between the senders and the recipients of the message. In today's technological advancements, computer systems, and communication networks are prone to cyber attacks that risk the data security of a message. These cyberattacks are a growing concern for users for risking their data confidentiality. Hence, cryptography is a vital tool to address these issues and provide a solution to maintain confidentiality, authenticity, integrity, availability, and user identification. This can be achieved by employing cryptographic algorithms like RSA, DES, and AES.

This paper will go over the above-mentioned cryptography algorithms to explore their theoretical foundations, design techniques, and their computational efficiency in terms of their memory utilization and their required empirical time. DES and AES are classified as symmetric key cryptographic systems, whereas RSA functions are classified as an asymmetric key cryptographic algorithms. Through our analysis, we seek to provide insights into how these algorithms contribute to data security and privacy.

II. Types of Cryptographic Algorithms

There are two main types of algorithms in cryptography: symmetric and asymmetric.

Symmetric algorithms are known as symmetric-key cryptography since they use a single shared key for the process of encryption and decryption. This means that both parties that are involved in the communication vessel should have access to the same secret key being used to encrypt and decrypt the data.

Symmetric algorithms are helpful in cases where there is a large volume of data since the encryption and decryption methods are computationally inexpensive. However, the task of managing and distributing the shared key between the involved parties can be challenging and risky.

Oh the other hand, asymmetric algorithms are known as public-key cryptographic algorithms. These algorithms use two different keys, a public key and a private key. The public key is used for encryption and is widely available, whereas the private key is supposed to be kept secret and is used for decryption.

Asymmetric algorithms are often used for secure key exchange, digital signatures, and other applications where both parties involved in the communication channel do not share a common and known secret key initially. However, as compared to symmetric algorithms, they are generally computationally more expensive than symmetric algorithms. This makes them less suitable for encrypting large volumes of data as they become computationally expensive. [1]

III. Theoretical Exploration

3.1 RSA (Rivest-Shamir-Adleman)

RSA is a public-key encryption algorithm that was developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. It is one of the most popular asymmetric key cryptographic algorithms and is widely used for providing both secrecy and digital signatures.

3.1.1 Principle and Theoretical Foundations

The RSA cryptosystem is based on the principles of number theory, particularly involving the generation of prime numbers and modular arithmetic. As mentioned, the algorithm utilizes two keys - a public key for encryption and a private key for decryption. These keys are generated from large prime numbers and satisfy certain mathematical properties.

3.1.2 The Algorithm

- **Key Generation**
  1) Choose two distinct large random prime numbers, p and q, such that $p \neq q$.
  2) Compute $n = p \times q$, where n is the modulus.
  3) Calculate $\phi(n) = (p-1) \times (q-1)$, where ϕ is Euler's totient function.

4) Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(\phi(n), e) = 1$.
5) Compute d to satisfy the congruence relation $d \times e \equiv 1 \pmod{\phi(n)}$; d becomes the private key exponent.
6) The public key is represented as (n, e), and the private key is represented as (n, d). Keep all values of d, p, q, and ϕ secret.

- **Encryption**
  1) Plaintext P is represented as an integer such that $0 \leq P < n$.
  2) Ciphertext C is calculated as $C \equiv P^e \pmod{n}$.
- **Decryption**
  1) Ciphertext C is given.
  2) Plaintext P is calculated as $P \equiv C^d \pmod{n}$.

RSA's security relies on the computational difficulty of factoring the modulus n into its prime factors p and q, known as the RSA problem. It is worth noting that RSA encryption and decryption operations can be computationally intensive, especially for large values of n. [19]

3.2 DES: Data Encryption Standard

DES (Data Encryption Standard) is one of the most widely accepted cryptographic systems, initially developed by IBM in the 1970s and later adopted by the National Institute of Standards and Technology (NIST) as Federal Information Processing Standard 46 (FIPS PUB 46). It operates as a symmetric block cipher, designed to encrypt and decrypt blocks of data consisting of 64 bits using a 64-bit key.

3.2.1 Principle and Theoretical Foundations

DES is based on the Feistel block cipher framework, dividing input data into blocks and iteratively applying transformations for encryption and decryption. Its security mechanisms rely on the principles of diffusion and confusion, realized through 16 rounds of operations. Diffusion substitutes and rearranges bits to spread their influence across the ciphertext, obscuring patterns. Confusion involves non-linear S-box substitutions and permutations to obscure the relationship between plaintext and ciphertext. Each round includes bit-shuffling, S-box substitutions, and XOR operations, contributing to the algorithm's robustness against attacks. [19]

3.2.2 The Algorithm

- **Initial Permutation (IP):** Divides plaintext into left and right halves (LPT and RPT).
- **Round Operations:**
  1) Key is split into two 28-bit halves, rotated, and compressed to 48 bits.
  2) Data block split, with one half undergoing expansion permutation.
  3) XOR operation with compressed key, followed by S-box substitution, permutation, and XOR with the other data half.
  4) Final step involves swapping the two data halves.
- **Iterative Process:** Repeated for 16 rounds, with encryption and decryption using the same steps and key.
- **Final Permutation:** Reverses the initial permutation to generate the final ciphertext.

DES, despite known vulnerabilities, remains widely used due to its adoption as a standard in various industries. Its cryptographic mechanisms balance security and efficiency, making it suitable for protecting sensitive online applications.

3.3 AES: Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a cryptographic algorithm recommended by NIST to replace DES in 2001. It supports various combinations of data and key lengths, denoted as AES-128, AES-192, and AES-256, depending on the key size. AES operates through a fixed number of rounds, with 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys, ensuring robust encryption and decryption.

3.3.1 Principle and Theoretical Foundations

AES is designed based on the principles of diffusion and confusion, integral to modern cryptography. Diffusion involves spreading the influence of individual bits across the entire ciphertext, while confusion obscures the relationship between the plaintext and the ciphertext. [19] These principles are realized through a series of transformations in each round of AES.

3.3.2 The Algorithm

The AES algorithm operates on a symmetric block cipher that encrypts data in a fixed block length of 128 bits that can have cipher keys of varying lengths (128, 192, 256 bits). For our analysis, we will consider a key size of 128 bits so we are focusing on AES-128. The data block (with a fixed length of 128 bits) is divided into a 4x4 matrix known as the state. Each round of AES involves four main transformations: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

- **SubBytes:** Each byte of the data block undergoes a byte substitution using an 8-bit substitution box known as Rijndael Sbox.
- **ShiftRows:** Bytes in the last three rows of the state are cyclically shifted, with varying shift amounts depending on the row.
- **MixColumns:** Transformation occurs on each column of four bytes using a specific mathematical function, producing new bytes and enhancing diffusion.
- **AddRoundKey:** Bitwise XOR operation between the state and the round key, contributing to confusion and serving as its own inverse transformation.

The AES decryption process involves applying inverse functions of the transformations in reverse order. AES, with its robust encryption mechanisms and efficient implementation, has become the standard choice for securing sensitive data and communications. It offers advantages such as speed and enhanced security.

IV. Design Techniques Analysis

A. Analysis of RSA

1) Overall Complexity

RSA encryption and decryption processes are dependent on the size of the input that is provided to them which is denoted by n, the number of bits. Encryption processes in RSA are generally simpler than decryption processes since decryption is a lengthy process in RSA involving more steps and would hence need more time. Therefore, the asymptotic time complexity to encrypt RSA is $O(n^2)$ which is lesser than decryption, and the time complexity to decrypt in RSA is $O(n^3)$. [18]

2) Brute Force

Brute force is a method of trying all possible solutions to a problem until the right one is found. It's useful for simple problems like password cracking with weak passwords, but can be slow and inefficient for complex problems. In cryptography, stronger encryption algorithms and longer keys make brute force attacks much harder. While brute force is a basic approach, there are often more efficient algorithms available. Therefore, the strength of encryption relies heavily on both the complexity of the algorithm and the length of the key. Robust encryption algorithms, such as RSA, employ longer keys to significantly raise the difficulty level for successfully executing a brute force attack, rendering them more resilient against unauthorized decryption attempts. [14]

a) A. Generating Prime Numbers:

- The first step in the RSA algorithm involves generating prime numbers. Due to limitations in the hardware environment, the prime numbers are generated within certain bit sizes, typically ranging from 2 to 22 bits. Generating prime numbers beyond this range becomes impractical due to the time complexity of the prime number generation algorithm, which is $O(2^n)$. However, for the purposes of this research focusing on RSA key cracking, this limitation is disregarded.

b) B. Generating and Predicting the RSA Key:

- In this algorithm, the RSA key generation begins with the selection of two random prime numbers, p and q, from a list of pre-generated primes. These generated prime numbers are multiplied together to obtain the modulus, denoted by n. The Euler's totient function of n ($\varphi(n)$) is calculated as $(p-1)(q-1)$. Afterwards, the algorithm will choose a number, e, as the public key, that would be coprime with $\varphi(n)$. Next, the algorithm will find a number, d, as the private key. This leads us to the equation such that $(d \times e) \mod \varphi(n) = 1$.
- To predict the private key generated using only n (the modulus) and e (the public key), the algorithm needs to iterate over the list of pre-generated prime numbers. All the pairs of primes are multiplied together until the right combination is found which results in the product of the modulus, n. The time complexity for this step is $O(n^2)$, where n denotes the number of prime numbers for the given bit size. As soon as the correct prime pair is found, the algorithm will calculate $\varphi(n)$ and will search for a coprime of $\varphi(n)$. This adds another $O(n^2)$ to the complexity. Finally, the algorithm will compute the modular inverse of the coprime using the extended Euclidean algorithm. This requires a time complexity of $O(\log n)$.

$$T(n) = n^2 \times n^2 + n \log n$$

$$T(n) = n^4 + n \log n = O(n^4)$$

Once we combine the complexities deduced from the steps above, the total time complexity to crack an RSA private key using brute force can be determined. Considering the relationship between the number of primes (n) and the bit size (b), where $n(b) \approx 2 \times 3^b$,

$$O(n^4) = O\left((2 \times 3^b)^4\right) = O(1296^b)$$

the overall time complexity is calculated as $O(1296^b)$. Therefore, the **time complexity** of cracking an RSA private key is represented as $O(k^n)$, where k is a constant and n is the maximum bit size of the number. [14]

Hence, we've determined the exact time complexity of guessing an RSA private key with brute force to be a remarkable $O(k^n)$, where n is the maximum bit size of the prime numbers involved. However, it's important to note that the brute force method employed in this research is pure brute force without any optimization. Thus, $O(k^n)$ represents the highest possible time complexity, and it could potentially decrease if better performance enhancements are applied.

3) Dynamic Approach using Subset Sum Cryptography:

- The Subset-Sum cryptosystem, also known as the Knapsack Cryptosystem, represents an asymmetric cryptographic technique. It operates on the subset sum problem, a variant of the knapsack problem. In this problem, given a set of positive integers S = x1, x2, ..., xn and a target integer t, the objective is to determine whether there exists a subset of S whose sum equals t. When the set of numbers, often referred to as the knapsack, follows a particular pattern known as super increasing, solving this problem becomes relatively straightforward using a greedy algorithm. A super increasing knapsack means each element in the set surpasses the cumulative sum of all preceding elements.
- The proposal introduces a novel RSA algorithm inspired by the Subset-Sum cryptosystem Knapsack. This algorithm comprises three main phases: Key Generation, Encryption, and Decryption. Unlike traditional RSA, this approach, termed Modified Subset Sum (MSS), is an asymmetric-key cryptosystem requiring both public and private keys. However, unlike RSA, MSS serves solely as a one-way function; the public key facilitates encryption, while the private key exclusively handles decryption. Consequently, MSS is unsuitable for cryptographic signing or authentication purposes [16]

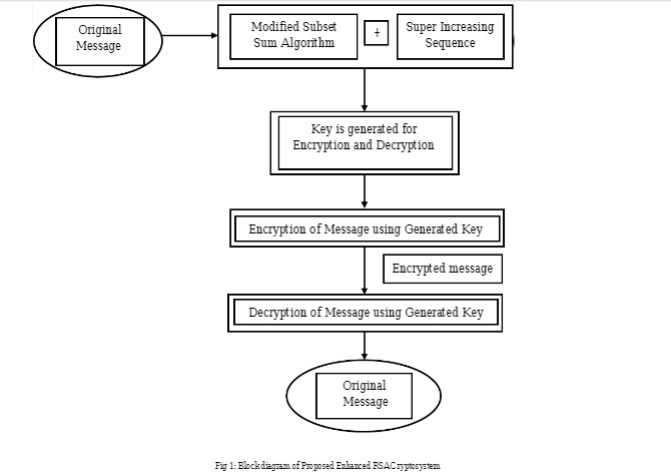

Fig 1: Block diagram of Proposed Enhanced RSA Cryptosystem

Fig. 1. Proposed Enhancement in RSA

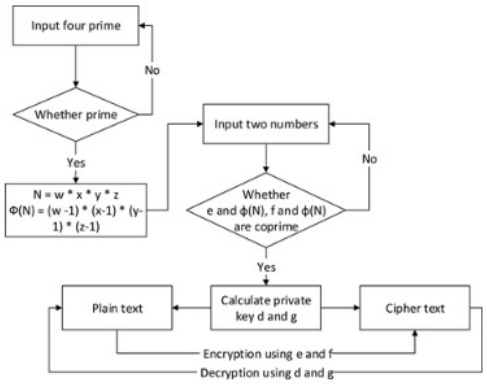

Figure 1. Flow chart of Modified RSA (MRSA) Algorithm.

Fig. 2. Flowchart of MRSA

a) Key Generation Process:

1) Generate two large random primes, p and q, of approximately equal size such that their product $M = p \times q$ is of the required bit length (e.g., 1024 bits).
2) Calculate $M = p \times q$ and $\phi(n) = (p-1) \times (q-1)$.
3) Randomly choose an integer e, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
4) Calculate the secret exponent d, such that $1 < d < \phi(n)$, and $e \times d \equiv 1 \mod \phi(n)$.
5) Choose a super increasing set $A = (a_1, ..., a_n)$.
6) Choose an integer M with $M > \sum_{i=1}^{n} a_i$. M is called the modulus.
7) Choose a multiplier W such that $\gcd(M, W) = 1$ and $1 \leq W < M$. This choice of W guarantees an inverse element U: $UW \equiv 1 \mod M$.
8) Compute the components b_i of the public key B as $b_i = a_i \times W \mod M$, $i = 1, ..., n$. The super increasing property of A is concealed by modular multiplication.

The public key is (B, n, e), and the private key is (A, M, W, n, d). The private key values d, p, q, and $\phi(n)$ must be kept secret, while the public key B is published for everyone. [16]

b) Encryption Process:

- The message p to be encrypted is divided into n-bit groups.
- Compute the ciphertext c as $c = b_1p_1 + b_2p_2 + ... + b_np_n$.
- Compute the ciphertext $c_1 = c^e \mod n$.
- Send the ciphertext c_1 to the recipient.

c) Decryption Process:

- Use the private key to compute $m_1 = c_1^d \mod n$.
- Compute $c' = Um_1 \mod M = W^{-1}c \mod M$.
- Solve the subset sum problem (A, c'). Because A is super increasing, (A, c') is easily solvable.
- Let $X = (x_1, ..., x_n)$ be the resulting vector, and $p_i = x_i$ for $i = 1, ..., n$. Thus, $p = (p_1, ..., p_n)$ is the plaintext.

The RSA cryptosystem is a well-known asymmetric encryption method. Its computational complexity for both encryption and decryption operations is approximately proportional to the cube of the number of bits in the modulus, denoted as k, where k represents the number of bits of the modulus n.

On the other hand, the computational complexity of the subset sum problem is typically linear, denoted as $O(k)$, if a specific approach, such as a greedy algorithm with a super increasing set, is utilized. Here, k refers to the number of items in the set.

The computational complexity of the RSA cryptosystem for both encryption and decryption operations is proportional to k^3, where k represents the number of bits in the modulus n. Similarly, the computational complexity of the subset sum problem is $O(k)$, where k denotes the number of items in the set.

Therefore, in our proposed cryptosystem, the overall computational complexity can be expressed as:

$$O(k^3 + k) = O(k^3)$$

This indicates that both encryption and decryption operations through this proposed method have a **computational complexity of** $O(k^3)$. [17]

4) A Modified and Secured RSA Public Key Cryptosystem Based on "n" Random Prime Numbers:

A modified RSA algorithm is proposed utilizing "n" distinct prime numbers. To bolster security, a pair of a random number and its modular multiplicative inverse is incorporated into the RSA scheme.

The proposed Modified RSA (MRSA) scheme aims to address the primary vulnerabilities of the traditional RSA system. The primary vulnerability lies in the predictability of "N", the product of two prime numbers. Once "N" is obtained, an attacker can compute the keys and compromise the system [18]. The proposed modifications enhance efficiency and security, as discussed below.

A. MRSA Key Generation:

The proposed model employs "n" distinct prime numbers, demonstrating calculations and analysis using four large

prime numbers. The public and private key exponents consist of three components. ”N” is derived from the product of four prime numbers: ”w”, ”x”, ”y”, and ”z”. The public key exponent includes three components (e, f, N), with e and f selected randomly. This adds complexity, as only the value of ”N” is made public, preventing attackers from deducing the four prime numbers underlying ”N”, and subsequently ”e” and ”f”. The private key exponent also comprises three components (d, g, N). To maintain security, the bit length of all four chosen primes matches that of traditional RSA. [18]

B. MRSA Encryption and Decryption:

Encryption utilizes the public key exponent, while decryption relies on the private key exponent. The process involves not only ”N”, which consists of four large prime numbers, but also four random components (”e”, ”f”, ”d”, ”g”), increasing the complexity of breaking the system.

MRSA_Encryption()

Input:

Plain text message, M (

Public key exponent: {e, f, N}

Output:

Cipher text, X

Procedure:

$$X \leftarrow (M^e \bmod N)^f \bmod N$$

MRSA_Decryption()

Input:

Ciphertext message, X

Private key exponent: {d, g, N}

Output:

Decrypted plain text, Y

Procedure:

$$Y \leftarrow (X^g \bmod N)^d \bmod N$$

Fig. 3. Encryption/Decryption in MRSA

C. Performance Analysis:

The MRSA algorithm's performance is evaluated across varying input bit sizes. A comparison with the original RSA algorithm by Rivest, Shamir, and Adleman [3] is presented. Table 2 illustrates the performance of MRSA in terms of key generation, encryption, and decryption times.

Length of w, x (in bits)	Analyzing time for RSA algorithm		
	Key generation time (in ms)	Encryption time (in ms)	Decryption time (in ms)
100	76.63	0.16	0.25
128	90.46	0.17	0.28
256	94.96	0.35	0.96
512	177.47	0.56	5.2
1024	570.90	1.69	26.18
2048	4201.47	3.32	130.83
4096	54,368	11.17	1116.24

Table 1. Performance of RSA.

Length of w, x, and z (in bits)	Analyzing time for Modified RSA (MRSA) algorithm		
	Key generation time (in ms)	Encryption time (in ms)	Decryption time (in ms)
100	244	0.28	1.59
128	252.33	0.66	2.89
256	257.8	1.46	14.26
512	386.8	3.00	87.94
1024	1268.6	7.79	446.32
2048	7098.6	21.90	2472.70
4096	161,913	56.87	19,983.37

Fig. 4. Performance

Key generation time for MRSA is higher than that of RSA, offering an advantage in security due to increased complexity. Figure 2 demonstrates the encryption time comparison between RSA and MRSA, showing that as the bit length increases, the difference in performance becomes more pronounced.

D. Complexity Analysis

The complexity of the MRSA algorithm depends on the number of prime numbers utilized in the process. [18]

Complexity for Randomly Selected Prime Numbers:

- The complexity of finding the first prime number is $O(s \cdot (\log_2 w)^4 \cdot \ln w)$.
- Similarly, the complexity of finding the second prime number is $O(s \cdot (\log_2 x)^4 \cdot \ln x)$.
- The complexity of finding the third prime number is $O(s \cdot (\log_2 y)^4 \cdot \ln y)$.
- Similarly, the complexity of finding the fourth prime number is $O(s \cdot (\log_2 z)^4 \cdot \ln z)$.

Complexity for Calculation of N:

- The complexity of computing N is $O(\log_2 w \cdot \log_2 x \cdot \log_2 y \cdot \log_2 z)$.

Complexity of Computing Euler Phi Value of N:

- The complexity of computing the Euler phi value of N is $O((\log_2 (w-1) \cdot \log_2 (x-1) \cdot \log_2 (y-1) \cdot \log_2 (z-1))^4 \cdot ((w-1)\cdot(x-1)\cdot(y-1)\cdot(z-1)-1))$.

Complexity for Random Variables e and f:

- The complexity of finding the random variable e is $O(\log_2 (w-1)\cdot\log_2 (x-1)\cdot\log_2 (y-1)\cdot\log_2 (z-1)+ \gcd(e,(w-1)\cdot(x-1)\cdot(y-1)\cdot(z-1)))$. Since e and N are coprime, $\gcd(e,(w-1)\cdot(x-1)\cdot(y-1)\cdot(z-1)) = 1$, and the complexity becomes $O((\log_2 (\log_2 (w-1)) \cdot \log_2 (\log_2 (x-1)) \cdot \log_2 (\log_2 (y-1)) \cdot \log_2 (\log_2 (z-1)))^4 + 1)$.
- Similarly, the complexity of finding the random variable f is

$O((\log_2(\log_2(w-1)) \cdot \log_2(\log_2(x-1)) \cdot \log_2(\log_2(y-1)) \cdot \log_2(\log_2(z-1)))^4 + 1)$.

Comparing the complexities above, it's evident that the Modified RSA (MRSA) algorithm is more intricate than the RSA algorithm. The complexity increases depending on the number of primes considered for the algorithm. [18]

B. Analysis of The Data Encryption Standard

The Data Encryption Standard (DES) is a symmetric-key block cipher designed for efficient encryption and decryption of 64-bit blocks of plaintext data using a 56-bit key to encrypt the data into 64-bit ciphertext blocks. Although not explicitly employing common algorithm design techniques, DES incorporates substitutions, permutations, and key mixing operations tailored for cryptographic algorithm design. This combination of techniques to ensure security and efficiency. [6]

1) Design and Operations

a) Initial and Final Permutations: The DES algorithm begins with an Initial Permutation (IP) step, which rearranges the bits of the 64-bit plaintext block according to a predefined permutation table. [3] This initial permutation is followed by dividing the permuted block into two equal halves, referred to as the Left Plaintext (LPT) and Right Plaintext (RPT). After the encryption process, a Final Permutation (FP) is applied, which is the inverse of the Initial Permutation, to produce the final 64-bit ciphertext block.

b) Round Structure and Feistel Network: After the permuted plaintext is divided into two 32-bit halves, the DES uses a round structure consisting of 16 iterative rounds, each involving a sequence of complex processes. At the core of these processes is the Feistel network structure, which is a crucial component shared by many block cipher algorithms.

During each round, operations alternate between the left plaintext (LPT) and right plaintext (RPT) halves. The right half (RPT) is expanded from 32 bits to 48 bits using the Expansion Permutation (E). This expanded RPT is XOR'd with the round key. This result is then divided into eight groups of 6 bits each. Each 6-bit group is passed through a substitution box (S-box) to produce a 4-bit output. The eight 4-bit outputs are combined and permuted using the Permutation Box (P-box). [12] While the left half (LPT) remains unchanged, the permuted right half is then merged with the left half using an XOR operation, and the roles of LPT and RPT are swapped for the next round. [6]

c) Key Schedule: The key schedule algorithm is responsible for generating the subkeys used in each round of the encryption and decryption processes. The algorithm starts with a 64-bit key, from which 8 bits are discarded for parity checking, leaving a 56-bit key. This 56-bit key is then divided into two 28-bit halves, which undergo a series of left shifts and permutations to generate a unique 48-bit subkey for each round. [12]

d) Substitution-Permutation Network: DES employs a substitution-permutation network (SPN) design, which combines substitution and permutation operations.

The substitution operations are performed by eight S-boxes (substitution boxes), each of which replaces a 6-bit input with a 4-bit output based on a non-linear transformation defined by a lookup table.

The permutation operations in DES are carried out by the Expansion Permutation (E) and the Permutation Box (P-box). The Expansion Permutation expands the 32-bit RPT to 48 bits, allowing it to be combined with the 48-bit subkey using an XOR operation. The P-box permutes the 32-bit output of the S-boxes, contributing to the diffusion of the cipher. [12]

2) Algorithmic Efficiency and Time Complexity

a) Algorithmic Efficiency: The DES algorithm was designed with efficiency in mind, making it suitable for implementation on the hardware and software platforms of its time. The algorithm's efficiency can be attributed to several factors such as:

- The use of the iterative design technique of the Feistel network structure which simplifies the algorithms implementation by using the same processing steps for both encryption and decryption, with the only difference being the order in which sub-keys are applied. This design technique also enhances the algorithms security by repeatedly applying a series of transformations to the data, making it more resistant to crypt-analytic attacks.
- The key schedule algorithm generates the sub-keys in an efficient manner, ensuring that each bit of the main key contributes to approximately 14 out of the 16 sub-keys. This key generation enhances the diffusion of the cipher and further increases its resistance to crypt-analytic attacks.
- The substitution boxes (S-boxes) which is the core of DES's algorithms security, as they introduce non-linearity into the cipher, making it resistant to linear cryptanalysis. [12] The design of the S-boxes was carefully crafted to ensure a balance between security and efficiency while avoiding potential weaknesses or biases.
- Lastly, we have the use of bitwise operations, such as permutations, substitutions, and XOR operations, which can be efficiently implemented at the hardware level.

b) Time Complexity: The time complexity of the DES algorithm can be analyzed by examining the individual components and their respective time complexities. The initial permutation (IP), final permutation (FP), and key schedule operations involve a constant number of operations, resulting in a constant time complexity denoted as O(1). Similarly, the expansion permutation (E), substitution box (S-box) operations, and permutation box (P-box) operations can be performed in constant time, contributing an O(1) time complexity. The round function, or Feistel function, combines the expansion permutation, XOR operations with the subkey, S-box substitutions, and the P-box permutation. As each of these operations has a constant time complexity, the overall

time complexity of the round function is O(1). The iterative round structure is the core of the DES algorithm, consisting of 16 rounds. Each round involves the execution of the round function, which has a constant time complexity. Therefore, the time complexity of the iterative round structure can be expressed as O(r), where r is the number of rounds (in this case, r = 16). Consequently, the overall time complexity of the DES algorithm is determined by the iterative round structure, resulting in a time complexity of O(r) = O(16). Since the number of rounds is a constant value, the overall time complexity of DES can be considered to be O(1), or constant time complexity. [11]

3) Cryptanalysis and Known Attacks

DES was designed to be secure against many types of attacks, but over time, experts discovered several advanced methods that can crack DES encryption more efficiently than just trying every possible key (brute-force) [9]. Some of these methods include:
- Differential Cryptanalysis: This approach looks for patterns in the way differences in data inputs affect differences in outputs to figure out the encryption key. [8]
- Linear Cryptanalysis: This method uses simple mathematical approximations to find links between the unencrypted data, the encrypted data, and the key. [8]
- Davies' Attack: This less common attack targets specific structural weaknesses in DES. [7]

These techniques, while powerful in theory, often require a lot of data and computing power, making them impractical for everyday use.

The main weakness of DES comes from its small key size of only 56 bits, which makes it vulnerable to brute-force attacks where attackers use modern computers to try all possible keys quickly. Real-life demonstrations, like the EFF DES cracker, showed that DES keys could be broken in just a few days.

Despite its efficiency in the past, the shortcomings of DES due to its small key size led to the development of stronger encryption methods. The most notable successor is the Advanced Encryption Standard (AES), which uses longer key sizes to provide much greater security and is now widely adopted for protecting sensitive data. [10]

C. Analysis of The AES

1) Block Cipher

AES is a symmetric block cipher and encrypts data in a fixed block size of 128 bits, making it AES 128. This lets each block to be processed individually. In AES encryption or decryption, the operation is done at a constant time depending on the fixed block size of 128 bits [14].

a) Time Complexities of AES Operations :

- **PlainText Processing**: Before we can begin the following 10 rounds of AES for each operation described below, the plaintext first needs to be transformed into a matrix. This matrix is of size 4x4 bytes and the size of the plaintext is 128 bits (hence, one letter takes 8 bits). The order of the characters in the plaintext corresponds to the position on the matrix in order such that the matrix positions P0 to P15 are proceeding from top to bottom and from left to right. This step is linear to the size of the plaintext, which is 128 bits, hence the complexity of this step becomes $O(1)$.

- **SubBytes**: Once the plaintext is converted to a matrix, this matrix can enter the rounds of the following operations starting from SubBytes. This step constructs an S-Box which is a 16x16 matrix, composed of an 8-bit to 8-bit mapping where each byte represented, eg. x0x1x2x3x4x5x6x7, is transformed into S[x0x1x2x3][x4x5x6x7] so S[0][0].
  This operation is done independently on each bye in the 128-bit block, so it executes in constant time for any round. Hence, the time complexity is $O(1)$.

- **ShiftRows:** This step takes the bytes in our 4x4 matrix from the data matrix and shifts their rows. The first row stays the same, the second row is shifted towards the left side b 8 bits (1 byte), the third row is shifted by 2 bytes and so on.
  In this operation, we are permutating the bytes within the block. Since the permutation pattern is fixed, this operation is also done in constant time $O(1)$ and is not affected by the input size.

- **MixColumns:** This operation is done on each column independently; we multiply fixed polynomials in a finite field. A left multiplication is applied by multiplying two two matrices: data matrix with a fixed mix matrix. Like the two operations above, it does not depend on the key length but on the fixed block size, resulting in a constant time complexity per round of $O(1)$.

- **AddRoundKey:** In AES-128, the key is 128 bits long. A key generating function produces in 44 elements (4 bytes each) from W[0] to W[43] such that W[0] tp W[3] make up the original key, and the rest of the elements form the 10 groups corresponding to the 10 encryption rounds. At each encryption round, the data is XORed with their respective key of the round. This step is reversible and we can achieve decryption by applying XOR to the round keys in the reverse order of our rounds.
  This operation does bitwise XOR of the block with the round key and operates directly on the 128-bit block, hence executes in a constant time and the complexity is $O(1)$.

b) Overall Complexity: As explained above, encryption and decryption have a complexity of $O(1)$ as they work on

a block of fixed size. Following the steps from Plaintext processing till AddRoundKeys for 10 rounds would give encrypt to give the ciphertext. Following these steps in a reverse order would not change their complexity and would eventually decrypt the text back into the plaintext. Hence, the complexity for encryption and decryption is $O(1)$ for each round and becomes $O(r)$ where r denotes the number of rounds involved.

However, when the size of the message is larger than 128 bits, there are $n/128$ blocks to be processed, and the complexity becomes $o(n)$ where n is the message size. Each block processes a constant amount of work as mentioned, and increasing the data size causes the number of blocks to scale linearly (for n message bits, we need n/128 blocks).

Hence, the overall time complexity is O(n) corresponding to the increase in workload for each additional data bit added in the message [13].

2) Cryptanalysis and Brute Force Attacks

A brute force attack on the AES algorithm would mean that the attack would apply exhaustive methods over all the possible encryption keys until the right one is found. The time complexity of this attack depends on the block length, and the key length in our algorithm. For AES, the algorithm shows a "perfect time security" resisting a brute force attack. This means that the time needed for the exhaustive search is considerably long, making the brute force attack unfeasible in a reasonable and executable timeframe. Hence, AES has a high resistance level against brute force attacks.

a) Linear Search Through Key Space: : As seen above, regardless of the key size, all the above operations of AES have a time complexity of $O(1)$ for each round. Once those operations are performed, the AES would involve an attempt to find out every possible key until the right combination is found.

This is a linear search through the key space, and the complexity of checking each key is $O(1)$. If the key length is k, then the algorithm has to do a linear search in a key space of 2^k to test all possible keys. Hence, the time complexity is $O(2^k)$ [11]. This reflects the worst-case scenario where every key must be tested.

3) Optimized AES approaches

AES is a widely used cryptographic algorithm and is constantly facing attacks that can leave users and platforms insecure. Hence, research has been made and modifications to the algorithm have been suggested that can make the algorithm perform better and optimized in terms of security and empirical time.

Optimization in ZigBee Networking [14] The S-box is optimized such that the elements are mapped from $GF(2^8)$ in the compound field of $GF((2^4)^2)$ using a linear transformation. This enhances the cryptographic strengths of the S-box as compared to the standard AES algorithm, improving the system security.

Moreover, rather than using a single coefficient matrix for MixColumns, we introduce 3 different coefficient matrices. During the encryption process, one of these matrices will be randomly picked and this would add diversity to the algorithm and improve its strength against attacks.

The results for this optimized version show a higher success rate in power-consumption tests as compared to standard AES, and showed a 100% recovery success rate faster than the standard AES method.

V. Comparative Study

When securing data, choosing the right encryption algorithm is crucial. One key factor to consider is execution time, which refers to how long it takes for the algorithm to transform plain text into cipher text. [19] Analyzing execution time helps us understand the inherent complexity of each algorithm. Additionally, memory utilization and security strength are equally important factors that influence the choice of an encryption algorithm. Memory utilization determines the amount of resources required by the algorithm, while security strength measures the algorithm's resistance against various crypt-analytic attacks.

In this section, we present a comprehensive comparative study of three widely used cryptographic algorithms: AES, DES, and RSA. The analysis is based on the following parameters: computational time, memory utilization, and security strength. The aims is to provide detailed analysis into the effectiveness and efficiency of each algorithm, enabling informed decision-making when selecting an appropriate encryption scheme for various applications.

A. Computational Time

The computational time is the time taken by an encryption algorithm to produce a ciphertext from a plaintext. This parameter provides insights into the complexity of each algorithm. Table I presents the computational time values for AES, DES, and RSA algorithms on different file sizes. [19]

TABLE I
Computation Time Values (in milliseconds)

File Size (bytes)	AES	DES	RSA
4.72	800	1098	367
	1606	1703	4370
5.18	634	1174	483
	1576	1677	5201
7.97	710	1524	523
	1883	2146	4098
10.1	746	1825	376
	2133	2547	5796
16.3	903	2637	510
	2797	3560	5934

The first row for each file size represents the encryption time, while the second row represents the decryption time. As evident from the table, the AES algorithm generally exhibits faster computational times compared to DES and RSA for both

encryption and decryption operations across various file sizes. [19]

This observation is further illustrated in Figures 5 and 6, which provide a visual comparison of the encryption and decryption times, respectively, among the three algorithms.

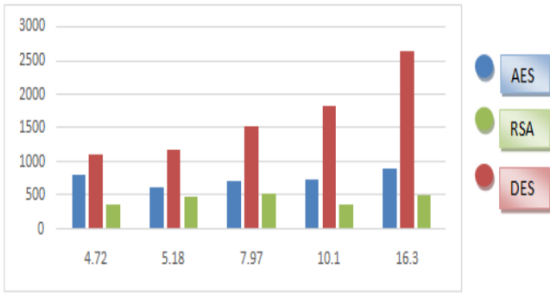

Fig. 5. Comparative Status of Encryption Time among AES, DES, and RSA

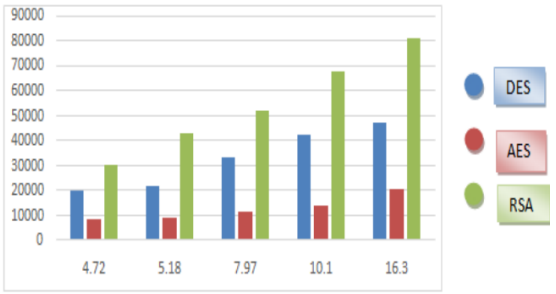

Fig. 6. Comparative Status of Decryption Time among AES, DES, and RSA

From Figure 5, it is evident that the DES algorithm takes the longest time for encryption compared to AES and RSA, while AES and RSA exhibit similar encryption times for smaller file sizes. Figure 6 shows that the RSA algorithm takes the longest time for decryption, followed by DES, with AES being the fastest among the three algorithms.

The computational time analysis suggests that the AES algorithm is the most efficient in terms of both encryption and decryption times, making it a suitable choice for applications requiring fast data encryption and decryption, especially when dealing with larger file sizes.

B. Memory Utilization

Memory utilization is a critical factor in evaluating the efficiency of cryptographic algorithms, as it determines the amount of system resources required during encryption and decryption processes. Table II presents the memory utilization values for AES, DES, and RSA algorithms on different file sizes.

The first row for each file size represents the memory utilization during encryption, while the second row represents the memory utilization during decryption. As evident from the table, the RSA algorithm generally requires more memory compared to AES and DES for both encryption and decryption operations across various file sizes, with AES requiring moderate memory utilization. [19]

TABLE II
MEMORY UTILIZATION VALUES (IN KILOBYTES)

File Size (bytes)	AES	DES	RSA
4.72	20316	14800	668
	19732	8388	30405
5.18	21592	13920	548
	21592	8868	843146
7.97	33020	13856	768
	33020	11724	52187
10.1	42068	21984	845
	42068	13988	67890
16.3	67320	38368	1078
	67320	20316	81241

This observation is further illustrated in Figures 7 and 8, which provide a visual comparison of the memory utilization during encryption and decryption, respectively, among the three algorithms.

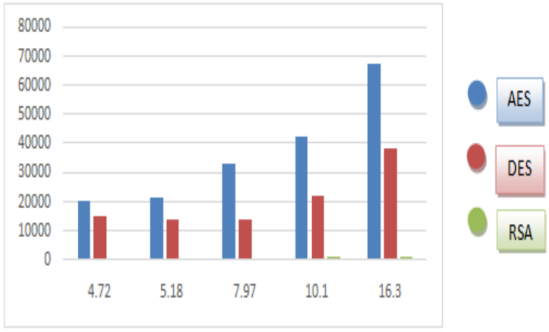

Fig. 7. Comparative Status of Memory Utilization (Encryption) among AES, DES, and RSA

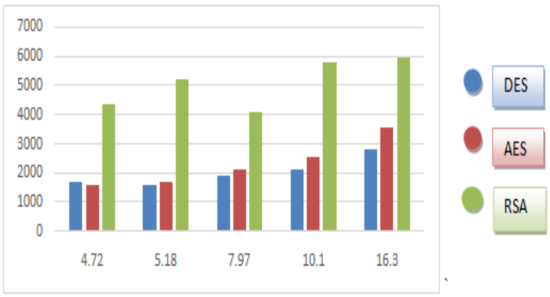

Fig. 8. Comparative Status of Memory Utilization (Decryption) among AES, DES, and RSA

From Figures 7 and 8, it is evident that the RSA algorithm consistently requires more memory compared to AES and DES for both encryption and decryption operations across various file sizes.

The paper also discusses the Big O analysis used to evaluate the efficiency of the algorithms based on memory utilization. It states that AES and DES operate on O(1) complexity because they work on a fixed block size and take approximately the same time regardless of the input size. [19] In contrast, RSA operates on O(n) complexity because its encryption and decryption times depend on the size of the file.

The memory utilization analysis suggests that the DES algorithm is the most efficient in terms of memory usage, followed by AES, while RSA requires significantly more memory resources, especially for larger file sizes.

C. Security Strength of Encryption Algorithms

Encryption strength is a crucial factor when evaluating an algorithm's performance. A strong algorithm offers a high level of security for the encrypted data. This section analyzes the security capabilities of various cryptographic algorithms.

AES: AES boasts a robust security level due to its variable-length key sizes. It utilizes mathematical operations similar to RSA's modulo arithmetic, but with the key difference of being mathematically reversible. The security of AES encryption hinges on two factors: the time it takes to crack the key and the cost associated with such an effort for an attacker. [19] Various attack methods, including square attacks, key attacks, and differential attacks, have been attempted on AES, but none have successfully compromised the algorithm.

DES: DES's security has become a major concern due to its limited 56-bit key length. This vulnerability makes brute-force attacks feasible with powerful computing resources. A large-scale parallel machine with over 2,000 nodes, each capable of searching 50 million keys per second, could potentially crack DES encryption. Additionally, the algorithm's weak S-boxes offer potential weaknesses for cryptanalysis attacks.

RSA: The security of the RSA cryptosystem relies on the mathematical difficulty of factoring large numbers. In simpler terms, an attacker would need to find the prime factors of a large composite number (n) used in the public key to break the encryption. This process, known as factoring, is computationally expensive and time-consuming(in polynomial time), making RSA a strong algorithm overall. However, as computing power continues to advance, the feasibility of factoring large numbers may change in the future. [19]

VI. Conclusion

In conclusion, this research paper aims to provide a comprehensive understanding of RSA, DES, and AES encryption/decryption algorithms. Through theoretical exploration, design analysis, comparative study, and implementation considerations, we aim to shed light on the intricacies and practical implications of these cryptographic schemes.

From our research and the results explained above, we found out that RSA is the most memory-intensive algorithm compared to AES and DES. Secondly, DES is the slowest algorithm among the 3 as it takes the most computational time to encrypt and decrypt a text. And lastly, AES is the most secure cryptographic algorithm as attacking approaches such as brute force require the most time to decipher the cryptic text, making it the most popular and preferred cryptographic algorithm when compared with RSA and DES.

References

[1] T. H. Cormen, "Algorithms Unlocked," *The MIT Press*, 2013.

[2] K. Goyal, "Understanding DES Encryption: A Step-by-Step Tutorial," *upGrad Tutorials*, Mar. 3, 2024. [Online]. Available: https://www.upgrad.com/tutorials/software-engineering/software-key-tutorial/des-algorithm/.

[3] "What is DES?," *Simplilearn*, Available: https://www.simplilearn.com/what-is-des-article.

[4] "Devis Attack: This less common attack targets specific structural weaknesses in DES," *Springer*, Available: https://link.springer.com/content/pdf/10.1007/BFb0053464.pdf.

[5] "Cryptanalytic Attacks on DES Block Cipher," *ResearchGate*, Available: https://www.researchgate.net/publication/306165342_Cryptanalytic_attacks_on_DES_block_cipher.

[6] "Data Encryption Standard (DES)," *TU Berlin*, Available: https://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm.

[7] "Advanced Encryption Standard (AES)," *TechTarget*, Available: https://rb.gy/8b2vaj.

[8] "Big O notation encryption algorithms," *Crypto Stack Exchange*, Available: https://rb.gy/0g4hmw.

[9] "A Comprehensive Guide to the Data Encryption Standard (DES) Algorithm in Cryptography," *Medium*, Available: https://rb.gy/vqs8rm.

[10] "Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2" *Medium*, Available: https://link.springer.com/article/10.1007/s11128-018-2107-3. https://www.scirp.org/journal/paperinformation.aspx?paperid=83244

[11] "Lecture 8: AES: The Advanced Encryption Standard," *Purdue Engineering*, Available: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf.

[12] "Optimization of AES-128 Encryption Algorithm for Security Layer in ZigBee Networking of Internet of Things" *Research Gate*, Available: https://www.researchgate.net/publication/360344064_Optimization_of_AES-128_Encryption_Algorithm_for_Security_Layer_in_ZigBee_Networking_of_Internet_of_Things. https://www.scirp.org/journal/paperinformation.aspx?paperid=83244

[13] "Empirical and Statistical Complexity Analysis of AES-128" *Research Square*, Available: https://assets.researchsquare.com/files/rs-1418564/v1/cae7ba6d-1213-45d9-9a57-78dc7568fd37.pdf?c=1683880141.

[14] "Analyzing the Time Complexity of RSA Encryption Cracking Using the Brute Force Method" *informatika*, Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/Makalah2023/Makalah-Matdis-2023%20(147).pdf

[15] "RSA algorithm using modified subset sum cryptosystem " *researchgate*, Available: https://www.researchgate.net/publication/261036955_RSA_algorithm_using_modified_subset_sum_cryptosystem

[16] "Enhancement in the security of RSA algorithm Using Subset Sum Cryptography" *researchgate*, Available: https://www.researchgate.net/publication/354494677_Enhancement_in_the_security_of_RSA_algorithm_Using_Subset_Sum_Cryptography

[17] "A Modified and Secured RSA Public Key Cryptosystem Based on "n" Prime Numbers" *scirp*, Available: https://www.scirp.org/journal/paperinformation?paperid=83244

[18] "RSA Time Complexity on Best Case, Average Case and Worst Case" *crypto.stackexchange*, Available: https://crypto.stackexchange.com/questions/62429/rsa-time-complexity-on-best-case-average-case-and-worst-case

[19] "Comparative Analysis of Cryptographic Algorithms in Securing Data" *ijettjournal*, Available: https://ijettjournal.org/assets/year/2018/volume-58/number-3/IJETT-V58P223.pdf