Hybrid Lightweight Cryptography for Data Security

Sriraam C
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai, TamilNadu
sriraam.c2021@vitstudent.ac.in

Balaji Ramesh
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai, TamilNadu
balaji.r2021@vitstudent.ac.in

Rojesh S
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai, TamilNadu
rojesh.s2021@vitstudent.ac.in

***Abstract*— Security is becoming important for both individuals and business needs. Many researchers have studied security extensively. Still, security gaps or threats are increasing, and data protection is a primary issue. This paper presents a novel, effective lightweight hybrid cryptographic algorithm with two layers of encryption. The first layer uses a new effective, lightweight cryptographic algorithm, and the second layer uses a Dynamic RSA Technique. This approach offers both symmetric and asymmetric cryptography features. The proposed approach's performance is evaluated using various metrics, including computational time, key sensitivity, statistical analysis, image histogram, and entropy change analysis. The experimental findings showed a high level of security and an apparent improvement in encryption execution time, memory usage, and throughput compared to widely used cryptographic systems.**

Keywords - Security, RSA, Lightweight hybrid cryptography, ASCON-128,

1. INTRODUCTION

Data transfer applications play a crucial role in facilitating communication and collaboration, the security and integrity of transmitted information are of paramount importance. Conventional encryption techniques often struggle to strike a balance between security and efficiency, particularly in real-time data transfer scenarios. To address these challenges, this study proposes a novel hybrid cryptographic algorithm tailored specifically for data transfer applications by incorporating lightweight and traditional methods.

Unlike traditional approaches, which typically rely on either symmetric or asymmetric encryption, The method we propose is a unique hybrid approach that incorporates both symmetric and asymmetric algorithms into the encryption process. This hybridization enhances the security posture of the cryptographic system while optimizing the performance in data transfer applications by improving time complexity.

The key aspect of our approach lies in the strategic integration of symmetric and asymmetric algorithms within the hybrid encryption layers, leveraging the strengths of both types of encryption to mitigate their respective limitations. This innovative fusion of cryptographic techniques aims to deliver enhanced security and efficiency in data-transfer scenarios.

To assess the effectiveness of the proposed algorithm, extensive performance evaluations were conducted, focusing on vital indicators such as computational time and time complexity. Leveraging real-world data transfer scenarios, the experimental results exhibit promising outcomes, showcasing a substantial improvement in time complexity compared to existing methods. These results corroborate the potential of our hybrid cryptographic algorithm to effectively address the evolving security requirements of data-transfer applications.

The introduction of this novel lightweight hybrid cryptographic algorithm aims to significantly advance the security and efficiency of data-transfer processes. The innovative fusion of symmetric and asymmetric encryption techniques offers a versatile solution that can be tailored to a diverse array of data-transfer applications, ranging from file sharing to real-time communication platforms. This study aspires to make a substantial contribution to the progression of encryption technologies, thereby fostering a safer and more secure digital environment for data transmission and communication.

The Proposed hybrid cryptographic algorithm makes a several significant contributions to the fields of data transfer applications in terms of Enhanced Security by combining symmetric and asymmetric encryption techniques, the algorithm leverages the strengths of both methods to enhance the overall security posture. Symmetric encryption provides efficiency in encrypting and decrypting large volumes of data, while asymmetric encryption offers robust key management and authentication mechanisms. The Proposed hybrid algorithm optimizes the performances by improving time complexity, making it well-suited for real-time data transfer scenarios where speed is crucial. The strategic integration of symmetric and asymmetric algorithms ensures that encryption and decryption processes are efficient without compromising security.

The proposed hybrid cryptographic algorithm can be integrated with the cloud services such as storage solutions, databases and communication platforms, creating a comprehensive ecosystem for secure and efficient data transfer. APIs and SDKs provided by cloud providers facilitate seamless integration and interoperability.

Finding new and efficient approaches is a never-ending task in the world of academic research. One such approach that has gained significant attraction in recent years is the dynamic approach. Repeating complex problems are challenges that persistently arise in various domains [9]. Conventional problem-solving methods need help to provide effective solutions due to the dynamic nature of these problems. The

dynamic approach follows two ways: (i) Overlapping subproblem - We store the solution to the overlapping subproblem (ii) Optimal Substructure - If the solution for the subproblem is optimal, then the solution of the original problem is also optimal. By adopting the dynamic approach, researchers make effective strategies to address complex problems. In the current paper, we have proffered a dynamic approach on the RSA algorithm that involves storing each alphabet's encrypted and decrypted values for potential reuse. The encryption process can be accelerated by employing a dynamic approach that stores pre-computed encryption and decryption values.

This approach capitalizes on the repetitive nature of the data, where the same character combination frequently appears in different contexts. By utilizing the repository of encrypted and decrypted values, the encryption process can quickly retrieve and apply the pre-computed values, reducing the computation overhead and increasing the speed of encryption and Decryption. Additionally, this dynamic approach does not compromise data security, as the stored values are securely managed and can only be accessed by authorized parties.

The significant contributions of the proposed work can be described as follows:

1. ASCON 128-bit encryption: The use of ASCON, a lightweight authenticated encryption algorithm, enhances security and efficiency in the encryption process.

2. Dynamic RSA: This novel approach dynamically adjusts the RSA encryption process based on the characteristics of the input data, improving encoding and decoding times while ensuring optimal performance across a range of data transfer scenarios.

3. Balanced security and performance: ASCON encryption provides a balance between security and speed, making the hybrid algorithm well-suited for real-world applications requiring both rapid processing and reliability.

4. Enhanced security: Incorporation of Dynamic RSA improves the overall security posture of the algorithm, supporting various data transfer scenarios with greater assurance.

5. Experimental validation: The proposed algorithm demonstrates superior efficiency and effectiveness compared to existing encryption methods, showcasing its potential to address evolving security challenges in data transfer applications.

The following are the portions of the paper: Section 1 is meant to serve as an overview. Section 2 outlines a study that is linked to the research. Section 3 explains the suggested algorithm. The simulation and implementation environments are described in Section 4. Section 5 summarizes the findings and comments. Section 6 contains a comparative analysis. The CIA's security analysis and accomplishments are described in Section 7. Other conclusions and recommendations were being stated in Section 8.

2. Related Works

Data security has become a paramount concern in contemporary computing environments, prompting extensive research into cryptographic techniques to address evolving threats. Pritilata and Mahmood [1] discuss the utilization of multi-level cryptography algorithms to strengthen data security, offering insights into novel approaches for safeguarding sensitive information.

Hybrid cryptographic algorithms have gained prominence in securing cloud environments, as highlighted by Kumar et al. [2], who propose hybrid cryptography algorithms for enhancing cloud security. Additionally, Yang et al. [6] promote a hybrid cryptosystem combining Fresnel lens and RSA Algorithm to bolster security in data transmission, underscoring the significance of hybrid cryptographic solutions.

In the domain of lightweight cryptography, Hatzivasilis et al. [3] provide a comprehensive review of lightweight block ciphers, shedding light on their efficacy in resource-constrained environments. Moreover, Diogenes, introduced by Chen et al. [4], offers a lightweight scalable RSA modulus generation technique, contributing to efficient cryptographic solutions in constrained settings.

The efficacy of cryptographic algorithms, particularly RSA, has been a subject of extensive research, with studies such as Somani and Mangal [5] presenting an improved RSA cryptographic system, while Mohamad et al. [7] conduct a research trend review on RSA scheme of asymmetric cryptography techniques, providing insights into recent advancements and emerging trends in RSA-based cryptographic solutions.

Efforts to evaluate cryptographic algorithms' performance under runtime scenarios have been undertaken by researchers such as Kumar et al. [8], who perform a performance-based comparison of various symmetric cryptographic algorithms, aiding in identifying suitable algorithms for specific application scenarios.

Moreover, Kamatchi and Kumari [9] propose a hybrid homomorphic model integrating RSA algorithm and modified enhanced homomorphic encryption technique, offering a versatile cryptographic solution for secure data processing in communication, IoT, and security applications. Additionally, Munjal and Bhatia [10] conduct a systematic review of homomorphic encryption and its contributions in the healthcare industry, highlighting the potential of cryptographic techniques in safeguarding sensitive healthcare data.

Network security is a critical concern in modern computing environments, driving research into innovative cryptographic techniques. Kapoor and Yadav [11] propose a hybrid cryptography technique to enhance network security, showcasing the potential of hybrid cryptographic solutions in safeguarding communication channels.

Elliptic curve lightweight cryptography has emerged as a promising area of research, as highlighted by Lara-Nino et al. [12], who provide a comprehensive survey of elliptic curve lightweight cryptography techniques. This survey contributes to the understanding of lightweight cryptographic solutions for wireless sensor networks, the Internet of Things, and other resource-constrained environments.

AbdElminaam [13] addresses cloud security concerns by introducing new hybrid cryptography algorithms, emphasizing the importance of robust cryptographic solutions in securing cloud computing environments.

Efforts to improve cryptographic algorithms' efficiency have led to the development of efficient hybrid cryptography algorithms, as demonstrated by Hoobi [15], who introduces an efficient hybrid cryptography algorithm, highlighting the significance of optimizing cryptographic techniques for performance-critical applications.

In the context of cloud computing, researchers such as Kalpana, Parsi, and Singaraju [16] explore data security using the RSA algorithm, shedding light on cryptographic solutions tailored for cloud computing environments.

Moreover, advancements in cryptographic techniques, including fast implementations of RSA cryptography [17] and fast variants of RSA [18], contribute to enhancing the performance and efficiency of RSA-based cryptographic solutions, addressing the growing demand for secure and efficient cryptographic algorithms.

Data security challenges have also been addressed through innovative approaches such as data security using compression and cryptography techniques [19] and data security with DNA cryptography [20], illustrating the diverse range of cryptographic solutions available for safeguarding sensitive information in various application domains.

Sharma and Kakkar [21] provide a comprehensive overview of cryptography algorithms and approaches used for data security, offering insights into the diverse range of cryptographic techniques employed to safeguard sensitive information.

Cryptanalysis plays a crucial role in evaluating cryptographic algorithms' security, as demonstrated by Shoeb and Gupta [22], who conduct a cryptanalysis of the tiny encryption algorithm in key generation, contributing to the understanding of cryptographic algorithm vulnerabilities.

Advancements in lightweight cryptography have garnered significant attention, particularly in the context of IoT security. Gunathilake et al. [23] discuss next-generation lightweight cryptography for smart IoT devices, addressing implementation challenges and applications in securing IoT environments.

Furthermore, researchers have proposed innovative lightweight cryptographic schemes, such as Xoodyak [24] and Hummingbird [25], tailored for resource-constrained devices, contributing to the development of efficient cryptographic solutions for IoT and other resource-limited environments.

Al-Rahman et al. [26] introduce NVLC, a new variant lightweight cryptography algorithm for the Internet of Things, emphasizing its applicability in securing IoT devices and networks.

In the domain of RSA algorithm implementation, Rahman et al. [27] explore the implementation of the RSA algorithm for speech data encryption and decryption, highlighting its potential applications in securing voice-based communication systems.

Additionally, researchers have proposed modifications to the RSA algorithm to enhance its security, as demonstrated by Al_Barazanchi et al. [28], who present a modified RSA-based algorithm with a double-secure approach, contributing to strengthening RSA-based cryptographic solutions.

In the context of cloud computing, researchers such as Kalpana, Parsi, and Singaraju [29] explore data security using the RSA algorithm, addressing data protection challenges in cloud environments.

Moreover, efforts to enhance the security and performance of the RSA algorithm have been undertaken by researchers like Patel and Shah [30], who focus on security enhancement and speed monitoring of the RSA algorithm, contributing to improving the efficiency and effectiveness of RSA-based cryptographic solutions.

Security challenges in cloud computing have been extensively explored by Thabit et al. [31], who investigate issues, threats, and attacks in cloud environments, along with their alleviating techniques. This study provides valuable insights into mitigating security risks in cloud computing.

In the realm of mobile wireless sensor networks (WSNs), Huang [32] proposes SEA (Secure Encrypted Data Aggregation), a secure encrypted data aggregation technique, addressing security concerns in mobile WSNs and contributing to enhancing data security in these networks.

Advancements in lightweight encryption design have been pivotal in addressing security concerns in embedded systems, as demonstrated by Bansod et al. [33], who implement a new lightweight encryption design for embedded security, offering efficient cryptographic solutions for resource-constrained devices.

Furthermore, researchers have proposed various cryptographic algorithms and block ciphers to address security challenges. Rivest et al. [34] introduce the RC6 block cipher, while Hong et al. [35] present HIGHT, a new block cipher suitable for low-resource devices. Additionally, the Blowfish algorithm, introduced by Valmik and Kshirsagar [37], and the Advanced Encryption Standard (AES), discussed by Wright [36], contribute to strengthening data security through robust encryption techniques.

In the context of the Internet of Things (IoT), researchers have developed lightweight encryption algorithms tailored for IoT device security. Usman et al. [38] propose SIT, a lightweight encryption algorithm for secure IoT communication, addressing security concerns in IoT deployments.

Among these efforts, the utilization of fully homomorphic encryption (FHE) stands out as a promising approach. Jabbar's research on "Using fully homomorphic encryption to secure cloud computing" sheds light on the potential of FHE to bolster the security posture of cloud-based systems [40]. By enabling computations on encrypted data without the need for decryption, FHE holds the key to preserving data confidentiality while facilitating secure data processing and analysis in cloud environments.

These insightful studies collectively contribute to the ever-growing body of knowledge surrounding cryptographic algorithms and their indispensable role in fortifying data security across a myriad of domains. With their thorough analyses and innovative approaches, they enrich our understanding of cryptographic mechanisms, shedding light on their efficacy in safeguarding sensitive information in today's interconnected digital landscape.

3. Proposed Methodology

This section discusses the proposed cryptographic algorithm, an improved variation of cryptography algorithm that combines two levels of encryption to enhance data security. The first layer is ASCON, an effective light-weight cryptographic algorithm and the second layer consists of a Dynamic RSA algorithm, which has an improved run-time compared to the traditional RSA. This Hybrid Cryptographic Algorithm offers both symmetric and asymmetric cryptography features, which improves the data security and maintains confidentiality. The following subsections describe this algorithm.

3.1. Description

In the proposed technique, a sophisticated amalgamation of two encryption layers fortifies data security. The primary layer harnesses the formidable ASCON Encryption, renowned for its lightweight attributes and robust symmetric-key foundation shown in Figure 1. This layer employs a structural framework blending elements of substitution/permutation, drawing inspiration from the renowned Feistel Structure, and enriched by the theoretical underpinnings of Shannon's cryptography. The infusion of logical operations such as XOR, AND, OR, shifting, and swapping further enhances the encryption's resilience against cryptographic attacks.

The second layer, anchored by Dynamic RSA, embraces an asymmetric cryptographic paradigm. This facet empowers the encryption process with dynamic computational capabilities, allowing specific operations on encrypted data to yield encoded results akin to the product of equivalent operations on plaintext. This bi-layered approach bestows a dual-tiered security mantle upon the plaintext, ensuring robust protection. To unveil the original content, decryption follows a reverse sequence, systematically reversing the encryption layers applied to the ciphertext.

1. Key Generation Process
    - ASCON
    - RSA
2. Encryption
    - Level – 1 (ASCON)
    - Level – 2 (RSA)
3. Decryption
    - Level – 2 (RSA)
    - Level – 1 (ASCON)

3.2. Key Generation Process

The key is the critical component of the algorithm in the process of encryption and decryption. The diffusion and confusion techniques used to generate the proposed algorithm strengthen the key. The strength of key generation results in increased security, better encryption complexity, and decreased knowledge of the key by attackers.

The key for ASCON is generated by using the help of random library in python. Firstly, A random number is generated using this library and a 128-bit key is generated by using a random function. In Addition, We generate a 128-bit nonce before the encryption process takes place.

In RSA key generation, two large prime numbers are chosen randomly. Their product forms the modulus of the RSA key pair. The Euler's totient function is computed, and a public exponent is selected, typically a small prime number. The public key is formed with the modulus and the public exponent. Finally, the private exponent is calculated as the modular multiplicative inverse of the public exponent. The public key is shared openly, while the private key is kept secret, ensuring secure communication and data integrity.

RSA Key Generation

Select two large prime numbers p and q.

Calculate $n = p \times q$

Calculate $\phi(n) = (p-1) \times (q-1)$

Select $e \rightarrow 1 < e < \phi(n)$

Public Key (e, n)

Select $d \rightarrow d = e^{-1} mod(\phi(n))$

Private key => (d, n)

3.3. Encryption

In the hybrid cryptography model where ASCON-128 bit is used for the first level encryption and Dynamic RSA for the second level encryption, the encryption process unfolds in two stages:

3.3.1. First Level Encryption (ASCON-128 bit Encryption):

The plaintext message is first encoded using ASCON-128 bit encryption algorithm, providing confidentiality. ASCON encrypts the plaintext message, generating ciphertext. The resulting ciphertext serves as the input for the next encryption stage.

ASCON-128 Bit Encryption

Authentication Encryption

$$\varepsilon_{k,r,a,b}(K, N, A, P)$$

Input

$$key\ K \in 0{,}1^k, k \leq 160, nonce\ N \epsilon \{0,1\}^{128},$$
$$associated\ data\ A \in \{0,1\}^*, plaintext\ P \epsilon \{0,1\}^*$$

Output:

$$ciphertext\ C \in \{0,1\}^{|P|}, \quad tag\ T \in \{0,1\}^{128}$$

Initialization

$$\boldsymbol{S \leftarrow IV_{k,r,a,b} \,||\, K \,||\, N}$$
$$\boldsymbol{S \leftarrow p^a(S) \oplus (0^{320-k} || K)}$$

Processing Associated Data

if $|A| > 0$ **then**
  $A_1 \ldots A_s \leftarrow r - bit\ blocks\ of\ A||1||0^*$
  for $i = 1, \ldots, s$ **do**
    $S \leftarrow p^b((S_r \oplus A_i) \,||\, S_c)$
$S \leftarrow S \oplus (0^{319} || 1)$

Processing Plaintext

$P_1 \ldots P_t \leftarrow r - bit\ blocks\ of\ A||1||0^*$
for $i = 1, \ldots, t-1$ **do**
  $S_r \leftarrow S_r \oplus P_i$
  $C_i \leftarrow S_r$
  $S \leftarrow p^b(S)$
$S_r \leftarrow S_r \oplus P_t$
$C_t \leftarrow \lfloor S_r \rfloor_{|P| \bmod r}$

Finalisation

$S \leftarrow p^a(S \oplus (0^r \,||\, K \,||\, 0^{320-r-k}))$
$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
return $C_1 \,||\, \ldots \,||\, C_{t-1} \,||\, C_t, T$

3.3.2. Second Level Encryption (Dynamic RSA Encryption):

The ciphertext from the first level encryption is encrypted using Dynamic RSA encryption. RSA's public key, which was dynamically generated for this session, is used for encryption. The RSA encryption process transforms the ASCON encrypted ciphertext into a 2-level encrypted ciphertext, ensuring additional security to the data.

The hybrid encryption model combines the efficiency and simplicity of ASCON-128 bit encryption at the first level with the flexibility and strength of Dynamic RSA encryption at the second level. This approach ensures robust protection of sensitive data against cryptographic attacks.

Dynamic RSA Encryption

$$C = M^e \bmod (n)$$

3.4 Decryption

In the hybrid cryptography model where ASCON-128 bit is used for the first level encryption and Dynamic RSA for the second level encryption, the decryption process mirrors the encryption process in reverse, consisting of two stages:

3.4.1. Second Level Decryption (Dynamic RSA Decryption):

The encrypted ciphertext from the second level encryption (RSA encryption) is decrypted using Dynamic RSA decryption. RSA's private key, which corresponds to the public key used for encryption, is used for decryption. The RSA decryption process transforms the encrypted ciphertext back into the derived plaintext, recovering the data encrypted at the second level.

Dynamic RSA Decryption

$$M = C^d \bmod (n)$$

3.4.2. First Level Decryption (ASCON-128 bit Decryption):

The derived plaintext from the second level decryption (resulting from RSA decryption) is decrypted using ASCON-128 bit decryption algorithm. ASCON decrypts the derived plaintext, recovering the original plaintext message encrypted at the first level.

The decryption process of the hybrid cryptography model involves reversing the encryption process, first decrypting the data encrypted with Dynamic RSA at the second level and then decrypting the resulting plaintext with ASCON-128 bit at the first level. This approach ensures that the original plaintext message is recovered securely and efficiently, providing end-to-end confidentiality and data integrity.

ASCON-128 Bit Decryption

Verified Decryption

$$D_{k,r,a,b}(K, N, A, C, T)$$

Input:

$$key\ K \in 0{,}1^k, k \leq 160, nonce\ N \epsilon \{0,1\}^{128},$$
$$associated\ data\ A \in \{0,1\}^*, ciphertext\ C \epsilon \{0,1\}^*, tag\ T \epsilon \{0,1\}^{128}$$

Output:

$$plaintext\ P \in \{0,1\}^{|C|}\ or\ \perp$$

Initialization

$$\boldsymbol{S \leftarrow IV_{k,r,a,b} \,||\, K \,||\, N}$$
$$\boldsymbol{S \leftarrow p^a(S) \oplus (0^{320-k} || K)}$$

Processing Associated Data

if $|A| > 0$ **then**
  $A_1 \ldots A_s \leftarrow r - bit\ blocks\ of\ A||1||0^*$
  for $i = 1, \ldots, s$ **do**
    $S \leftarrow p^b((S_r \oplus A_i) \,||\, S_c)$
$S \leftarrow S \oplus (0^{319} || 1)$

Processing Ciphertext

$C_1 \ldots C_{t-1} C_t \leftarrow r - bit\ blocks\ of\ C, 0 \leq |C_t| < r$
for $i = 1, \ldots, t-1$ **do**
  $P_i \leftarrow S_r \oplus C_i$
  $S \leftarrow C_i \,||\, S_c$
  $S \leftarrow p^b(S)$
$P_t \leftarrow \lfloor S_r \rfloor_{|C_t|} \oplus C_t$
$S_r \leftarrow S_r \oplus (P_t \,||\, 1 \,||\, 0^*)$

Finalisation

$S \leftarrow p^a(S \oplus (0^r \,||\, K \,||\, 0^{320-r-k}))$
$T^* \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$
if $T = T^*$ **return** $P_1 || \ldots || P_{t-1} || P_t$
else return $\perp$

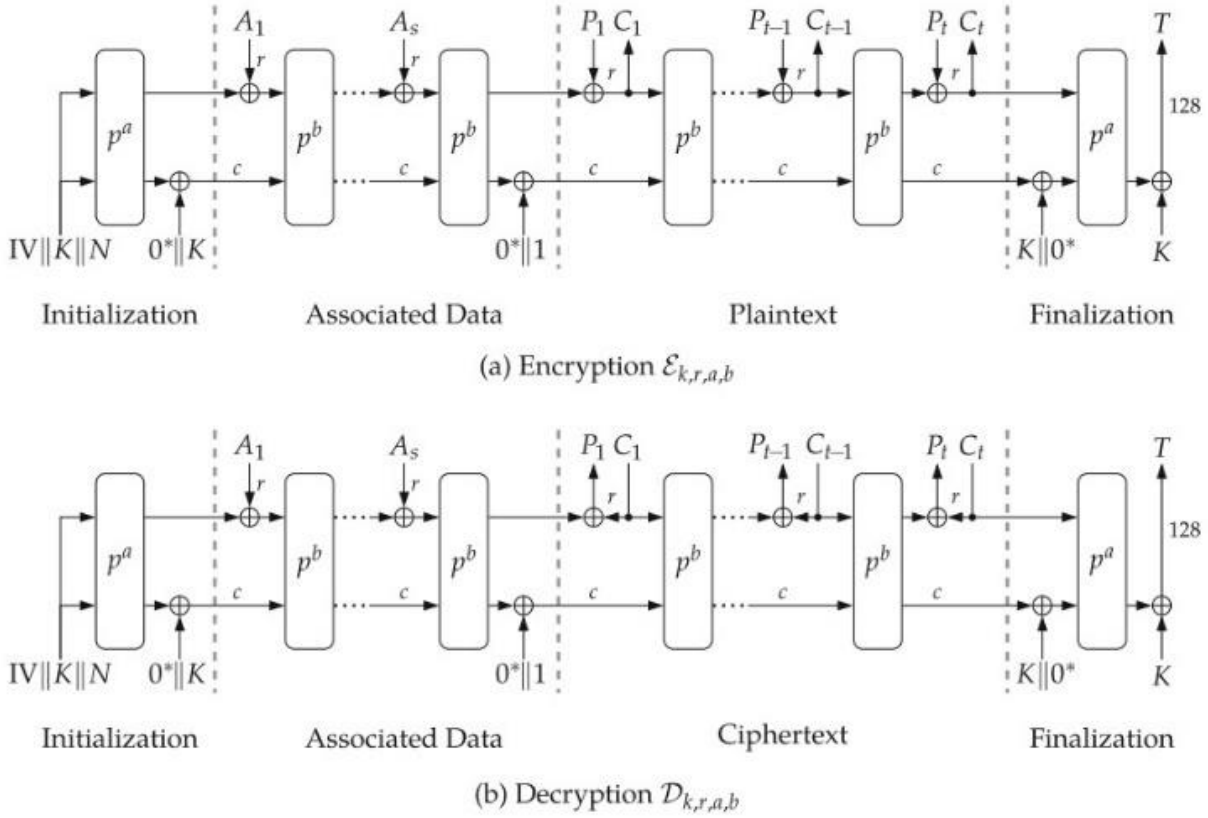

Figure 1 ASCON 128-bit encryption and decryption architecture

4. Implementation and Simulation

In the realm of cybersecurity, fast encryption algorithms hold immense significance. With the increase in data breaches and cyber threats, the timeliness of encryption becomes crucial. Efficient encryption techniques ensure that sensitive information is securely protected, minimizing exposure to potential attacks. Hence, fast encryption is a vital cornerstone of modern cybersecurity, safeguarding sensitive data with efficiency and speed.

This section of the research paper delineates the practical implementation of the proposed algorithm within the Jupyter Notebook environment. Leveraging the versatile capabilities of Jupyter Notebook, the algorithm is translated from theoretical constructs into executable code, facilitating rigorous experimentation and analysis.

Sieve of Eratosthenes algorithm is used to efficiently find all the prime numbers up to a given limit. In our algorithm, primefiller () is responsible for generating set of prime numbers. The function initializes a Boolean array with all values initially set to 'True' except number 0 and 1 as they are not prime. Starting from 2, it iterates through array and mark all multiple of each prime number as 'False''. After iteration the remaining 'True' value in array represent prime numbers. The function adds these prime numbers to a set and returns it.

The function pickrandomprime() is used to randomly select a prime number from the set of generated prime numbers. It takes generated set of prime numbers as input. The function generates a random index within the range of set size and select the prime numbers at that index. Once the prime number is selected, it is removed from the set to ensure that the same prime is not choose again. The function returns the selected prime numbers.

Generating public and private keys for the RSA encryption algorithm is the role of the `setkeys()` function. This function ensures that secure encryption and decryption are possible by selecting appropriate key pairs and setting them for use in the RSA process. The function begins by selecting two different prime numbers, typically referred to as prime1 and prime2. It then calculates the modulus' n' by multiplying the two prime numbers. The Euler's totient function 'fi' is computed as (prime1 - 1) * (prime2 - 1), which represents the count of positive integers less than 'n' that are coprime with 'n'. The function proceeds to find a suitable value for the public key 'e'. It starts with a value of 2 and increments it until it finds a value that is coprime with 'fi'. Once 'e' is determined, the function calculates the corresponding private key d. It starts with a value of 2 and increments it until it finds a value that satisfies the equation (d * e) % fi = 1. The final values of public_key (e), private_key (d), and n are set.

To secure a given message using the RSA algorithm, the `encrypt()` function takes a plaintext message as input and then performs the encryption process. It uses the recipient's public key, specifically the modulus n and the public exponent e, for encryption. The function converts each character of the message into its corresponding ASCII value. It applies modular exponentiation to each ASCII value using the public key components n and e. This process involves raising the ASCII value to the power of e and then taking the modulus n of the result. The resulting encrypted values represent the ciphertext, which is the encrypted form of the

original message. The function returns the ciphertext as the output.

`encoder()` transforms the plaintext message into a format appropriate for encryption, while also using a dynamic approach to verify whether each character has been previously encrypted. The function takes a message as input, typically in the form of a string. It iterates over each character of the message. For each character, the function performs a lookup to check if it has been previously encoded and stored in a dictionary ('enq'). If so, it retrieves the previously assigned encoded value for that character. If the character has not been previously encoded, the function encrypts the ASCII value of the character using the 'encrypt()' function. The function records the encoded value for the character in the dictionary for future reference. The encoded values for all the characters are collected and returned as the output.

The decryption of an encrypted message using the RSA algorithm is performed by the function `decrypt()`. It accepts the encrypted ciphertext as input and leverages the recipient's private key for the process. The function takes the encrypted ciphertext as input. It uses the recipient's private key. The function applies modular exponentiation to the ciphertext using the private key components n and d. This process involves raising the ciphertext to the power of d and then taking the modulus n of the result. The resulting decrypted values represent the original ASCII values of the characters in the plaintext message. The function converts each decrypted ASCII value back into its corresponding character. The decrypted characters are concatenated to reconstruct the original plaintext message. The function returns the plaintext message as the output.

In decryption, `decoder()` adjusts encrypted values for easier decryption and checks dynamically whether a character has been previously decrypted. The function takes the decrypted numerical values as input. For each iterated value, the function performs a lookup to check if it has been previously decoded and stored in a dictionary. If so, it retrieves the previously assigned decoded character for that value. If the value has not been previously decoded, the function converts it into its corresponding ASCII character. The function records the decoded character for the value in the dictionary for future reference. The decoded characters are concatenated to reconstruct the original plaintext message and returned.
`ascon_encrypt()` uses the Ascon authenticated encryption algorithm to transform plaintext into ciphertext and generate a tag for integrity verification. It takes a key, nonce, associated data, plaintext, and variant as inputs and returns a byte's object containing the ciphertext and tag. The key and the nonce must be either 16 bytes. The function begins by initializing the internal state array `S` based on the key size, rate, and number of rounds specified by the chosen variant. It processes the associated data and then encrypts the plaintext using the Ascon encryption algorithm, adjusting the rate and number of rounds depending on the variant. After processing the plaintext, the function finalizes the encryption by computing a tag. The final output is a byte's object containing the ciphertext and the tag, ensuring both confidentiality and integrity of the plaintext.

In the initial phase of Ascon authenticated encryption, `ascon_initialize()` configures the internal state `S` using the specified key, nonce, and other parameters. The function takes the Ascon state `S` (a list of five 64-bit integers), key size `k` in bits, block size `rate` in bytes, the number of initialization and finalization rounds (`a`), the number of intermediate rounds (`b`), and the key and nonce as inputs. It constructs an initial value using `k`, `rate`, `a`, `b`, and zero bytes to match the length of the key, followed by the key and nonce. This initial value is converted into the Ascon state format using bytes_to_state() and assigned to the state `S`. If debug mode is enabled, the initial state is printed for verification. The function then applies the ascon_permutation() function to the state `S` with the specified number of initialization rounds (`a`). Finally, the function XORs the state `S` with a zero-padded key, effectively setting the key in the state. If debug mode is enabled, the updated state is printed.

In the Ascon authenticated encryption process, `ascon_process_associated_data()` modifies the internal state `S` according to the associated data. It takes the Ascon state `S` (a list of five 64-bit integers), the number of intermediate rounds `b` for permutation, the block size `rate` in bytes, and the associated data as inputs. If there is any associated data, the function prepares it by appending a byte with value `0x80` and zero bytes for padding. The padded data is split into blocks of length equal to the block size (`rate`) and processed. For each block, the function XORs the block with the state `S` (using either 8 or 16 bytes), and then applies the ascon_permutation() function with the specified intermediate rounds (`b`). After processing all blocks, the function sets the least significant bit of `S[4]` to 1, marking the end of the associated data processing phase. This phase ensures proper integration of associated data into the internal state for further encryption or decryption steps.

During Ascon encryption, `ascon_process_plaintext()` transforms plaintext into ciphertext while simultaneously updating the internal state `S`. It takes the Ascon state `S`, the number of intermediate rounds `b` for permutation, the block size `rate` in bytes, and the plaintext as inputs. The function pads the plaintext to match the block size using a byte with value `0x80` and zero bytes. It processes plaintext in blocks of size `rate`, XORing each block with the appropriate portions of the state `S` and collecting the resulting ciphertext. After each block, it applies the ascon_permutation() function to the state `S` using the specified rounds `b`. The function handles the final block separately to match the length of the original plaintext. If debug mode is enabled, it prints the updated state. This phase securely transforms plaintext into ciphertext using Ascon encryption.

`ascon_process_ciphertext()` oversees the decryption phase in the Ascon algorithm, converting ciphertext back to plaintext and updating the internal state `S`. It processes the ciphertext in blocks, applying XOR operations and `ascon_permutation()` for transformation, and handles padding and length adjustments for accurate decryption. It takes the Ascon state `S`, the number of intermediate rounds `b` for permutation, the block size `rate` in bytes, and the

ciphertext as inputs. The function pads the ciphertext with zero bytes to match the block size and then processes it in blocks. For each block, the function performs an XOR operation with the appropriate portions of the state `S` and the ciphertext block, producing plaintext. It then updates the state `S` with the current block of ciphertext and applies the ascon_permutation() function. In the final block, the function manages padding and ciphertext length to extract the plaintext and update the state accordingly. If debug mode is enabled, the updated state is printed. This phase ensures accurate decryption of ciphertext back to plaintext.

The closing phase of the Ascon encryption algorithm is managed by the `ascon_finalize()` function, which generates the tag and updates the internal state `S`. It takes the Ascon state `S` (a list of five 64-bit integers), the block size `rate` in bytes, the number of rounds `a` for permutation, and a key of size 16 or 20 bytes as inputs. The function XORs the state `S` with portions of the key and applies the ascon_permutation() function to the state using the specified rounds `a`. It then XORs the state again with the key and returns the tag, which is formed from the last two 64-bit integers in the state `S`. This finalization phase ensures the tag is correctly computed for authentication and verification purposes.

The function ascon_permutation() is an internal helper function that applies the core permutation in the Ascon sponge construction algorithm. It takes the Ascon state `S` (a list of five 64-bit integers) and the number of rounds to perform (defaulting to one round) as inputs. The function uses a series of operations to update the state `S` based on the specified rounds. In each round, it starts by adding round constants to one of the state values. Then, it performs a substitution layer, where each state value undergoes a series of XOR operations with other state values and a temporary variable `T` based on specific bitwise operations. This is followed by a linear diffusion layer, where each state value undergoes bitwise rotations and XOR operations to introduce further randomness. The function's purpose is to enhance the security of the Ascon algorithm by transforming the state `S` through multiple rounds of permutations. Debug information can be printed if debugging flags are enabled.

The function that we have used to print plain texts and cipher texts is demo_hybrid_print() emonstrates hybrid encryption and decryption using Ascon-128 and RSA. It encrypts the plaintext message using a randomly generated key and nonce with Ascon, and prints the resulting encoded bytes and hexadecimal representation. The Ascon ciphertext (in hexadecimal) is then used as input for RSA encryption and decryption, and the function prints the RSA-encoded and decoded values. The RSA-decoded hexadecimal string is converted back to bytes and decrypted using the original Ascon key, nonce, and associated data, resulting in the original plaintext, which is then printed. If decryption fails, an error message is displayed. This function showcases the integration of Ascon and RSA for secure communication.

The proposed algorithm, aimed at improving data security in data transfer applications, and was simulated and implemented on a Lenovo IdeaPad L340 laptop. Equipped with an Intel CoreTM i5-9300H processor, the laptop provided robust computational power for intensive algorithmic processing. Complementing the processor, the inclusion of a GTX1650 GPU accelerated graphics processing tasks, crucial for image rendering and analysis.

The laptop's 8GB of RAM allowed seamless transitions between algorithm development, testing, and evaluation tasks. Utilizing Python 3.9 within Jupyter Notebook, developers harnessed the versatility of Python's scientific computing libraries, such as NumPy for code execution. The interactive environment of Jupyter Notebook provided a user-friendly interface for iterative algorithm refinement and experimentation.

5. Results and Discussions

Several security experiments are performed to verify the algorithm's quality, compared to other lightweight encryption algorithms and current symmetric and asymmetrically key encryption techniques such as A.E.S., D.E.S. and Blowfish R.S.A., E.G.A.M.A.L. The analysis of the algorithms is given below. We experimented with different file sizes in kilobytes (K.B.) as well as time estimates for encryption and decryption (seconds).

5.1. Time Complexity

With a 128-bit key size, the attacker would need to find 2^128 possible keys. As a result, the time complexity of 2^128 for obtaining the proper key is O on average (1). In reality, the proposed algorithm has a time complexity comparable to AES, but it is more efficient since there are no more repetitions than A.E.S. and the rest of the similar algorithms.

5.2. Execution Time

One of the most crucial aspects to consider while building cryptography is execution time. The overall time taken to encrypt/decrypt the unique data is known as the encryption cryptographic execution time. This cryptographic algorithm was implemented in Python and Jupyter Notebook.

Execution time is the algorithm's overall time to complete the execution and the processes. Execution time has two times-encryption and decryption time. Encryption time is the overall time the code takes to convert the original message into the cipher text. Decryption time is the time the algorithm takes to convert the cypher text back into the message, .i.e, in a readable format. We have compared our improved rsa algorithm with the El Gamal encryption algorithm based on execution time. Our improved rsa algorithm is also compared based on encryption and decryption times with the original rsa algorithm. We have also compared a symmetric algorithm AES based on execution time.

The evaluation test has been done in different sizes of text files. The experiment was performed 5 times on each file size and Table 1 shows the average time to execute the encryption and decryption process for text files. The tables indicate the execution time in milliseconds of equivalent algorithms with various file sizes. It is obvious that the proposed algorithm takes less time than the existing N.E.L.C algorithm as depicted in Figure 2.

5.3. Throughput

The throughput rate can be used to evaluate the algorithm's effectiveness. The algorithm's throughput is directly related to

its performance; the higher the performance/the higher the throughput. The formula for calculating the transfer rate of encoder technology is as follows

Throughput = Plain Text Size/Encoding Time.

The results of throughput is shown in Table 1. The throughput comparison of the proposed algorithm with N.E.L.C hybrid algorithm is shown in Figure 3.

Plaintext Size (kB)	Enc/Sec Time (S)	Thooughput (Kb/Sec)	Delay (S)	Latency (s)
255 kb	0.32	767.34	0.35	0.01
500kb	0.68	734.43	0.72	0.03
750kb	1.00	747.01	1.08	0.06
1 MB	1.33	749.06	1.47	0.09
Average	0.83	749.46	0.905	0.04

Table 1 Computation Time and Throughput Analysis

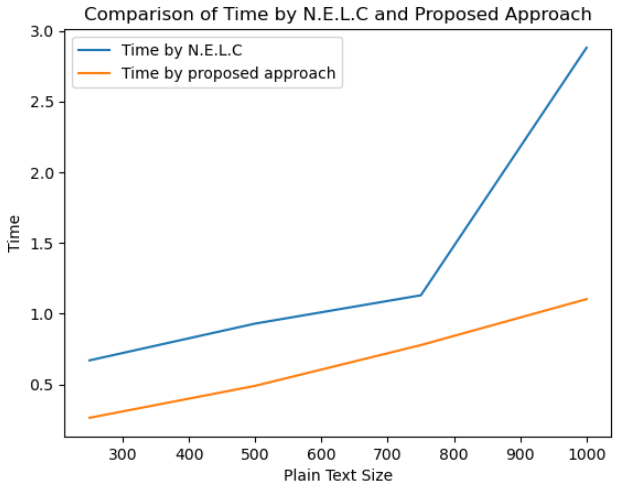

Figure 2 Execution Time Comparison with N.E.L.C algorithm

5.4. Delay and Latency

In cryptography, delay and latency play crucial roles in determining the efficiency of an encryption algorithm. Delay refers to the time taken for a message to travel from the sender to the receiver, while latency represents the time delay incurred during the processing of the message.

The proposed algorithm's delay and latency characteristics were evaluated through comprehensive testing. Multiple experiments were conducted, varying the input sizes and types of data. Each experiment was repeated five times to ensure statistical reliability.

Table 1 presents the average delay and latency values obtained from the experimental tests. It illustrates the delay in message transmission and the latency introduced during the encryption and decryption processes across different file sizes. Figure 4 graphically represents the Delay and Latency of the proposed algorithm

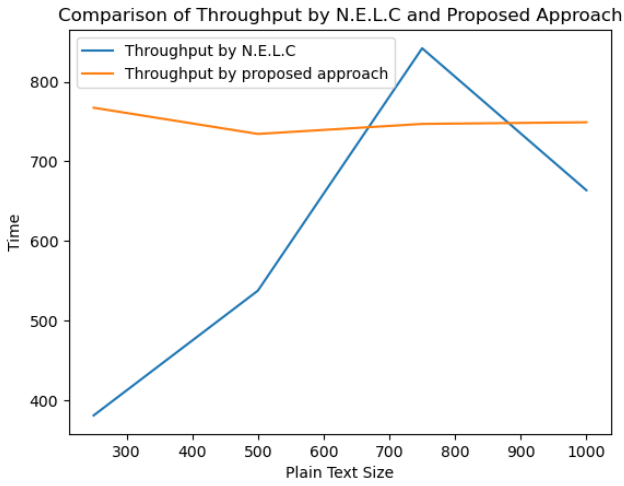

Figure 3 Throughput Comparison with N.E.L.C algorithm

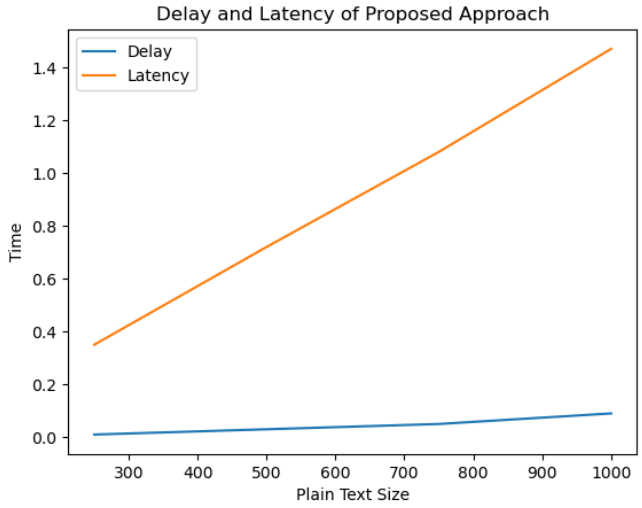

Figure 4 Delay and Latency of the proposed approach

6. Comparative Analysis

In this section, comparative studies are conducted to present and verify the proposed algorithm's feasibility and describe the analysis of the difference between the proposed algorithm, namely with a hybrid, light-weight, symmetric and asymmetric encryption algorithm that is commonly used.

The comparative studies divided into two sections describe the following.

6.1. Comparative study of some symmetric and asymmetric algorithm

The proposed algorithm was compared to several symmetric and asymmetric encryption algorithms frequently used for security information in cloud computing in the second section. The comparative study based on evaluated parameters commonly used for evaluated the Enc/Dec processes, such as Structure, Key size, Block Size, Possible Key, Execution Time, Cipher Type and Security Power parameters, shown in Table 2.

A faster algorithm requires fewer computational resources, such as memory. By reducing the execution time of an

algorithm, the cost required in terms of resources can be reduced. In the modern era, cloud computing platforms such as amazon web service (AWS) employ virtualization techniques to consolidate physical resources into virtual instances. The cost of using cloud resources is calculated in terms of time. Hence by reducing the execution time, usage of cloud resources is minimized, leading to cost saving in terms of hardware requirement.

The algorithm is often used in real-time applications where speed is crucial. Reducing the time required for an algorithm can improve operational efficiency and reduce associated costs. As the size of the input data increases, algorithms with higher time complexity can become prohibitively expensive to execute. By reducing the time complexity, an algorithm can scale more effectively.

Table 2
Comparative study of some of the symmetric and asymmetric algorithm.

	AES [36]	BLOWFISH [37]	*SIT* [38]	HOMOMORPHIC RSA [39]	HOMOMORPHIC ELGAMAL [40]	PROPOSED ALGORITHM
Structure	Su-lbs-Per	Feistel	Feistel/SP	modular exponentiation	modular exponentiation	SP/modular exponentiation
Algorithm	symmetric	symmetric	symmetric	asymmetric	asymmetric	symmetric/asymmetric
Key Size	128 bit	64 bits	64	Key Bit Random	Key Bit Random	128
Block Size	128, 192, 256 bits	32–448 bits	64	512/1024	512/1024	128
Key space analysis	2128, 2192 Or 2256 bits	232–2448 bits	264	Random	Random	2128 bits
Deposit Of Keys	needed	needed	needed	no	no	needed
No. Of Round	10, 12, 14	16	5	Random	Random	18
Encryption Process	Moderate	Moderate	Faster	faster	Moderate	Faster
Decryption Process	Moderate	Moderate	Moderate	faster	Moderate	Faster
Power Consumption	Low	Low	Moderate	high	high	Moderate
Security	Secure	Secure	Secure	Secure	Secure	High Secure

Table 3
Comparative in terms of flexibility, architecture, security and limitations

Algorithm	HIGHT [31]	SEA [32]	LED [34]	RC6 [34]	NLCA [35]	Proposed Algorithm
Structure	Feistel	Feistel	Feistel	Feistel	Feistel + SP	SP
Layer	1	1	1	1	1	2
Block size	64 bits	48, 96, 144 bit	64 or 128	128 bits	128, or 256	128
Key size	128 bits	48, 96, 144 bit	64 or 128	128, 192, 256 bits	1,28,256	128
No. of Round Possible key	32 2^{56} bits	Variable 248, 296 Or 2144 bits	Variable 264,2128 bits	20 2128,2196 bits	4 2128,2256 bits	18 2128,
Average time (S)	2.4	4.2	2.9	2.63	1.89	0.8364
Mathematical Operations Used	Addition, subtraction, XOR, Shifting. (8 bits)	XOR, rotations, 2n mod addition, substitution (8 bits)	XOR, rotations, 2n mod addition, substitution (6 bits)	Addition (2's comp). Variable Rotation, XOR, (16 bits)	Shifting, Substitution (4 bits) XOR, XNOR,	Shifting, Substitution (16 bits), XOR
Security rate	Secure	Secure	Secure	Secure	Secure	Highly Secure

Figure 5 Comparative analysis with other light weight techniques

6.2. Comparative analysis of the proposed algorithm with a light-weight hybrid cryptographic algorithm

The first section was conducted, taking into account some evaluated parameters commonly used for evaluated the encoding and decoding processes, such as Block Size, execution time, Key Length, Possible Key, Mathematical Operations, Cipher Type and Security Power parameters, as shown in Table 3and Figure 5.

7. Security Analysis and C.I.A Achievements

The strength of a cryptographic process hinges on its resilience to various forms of attack, as the attacker's endeavor to compromise the confidentiality of information through multiple avenues. Brute force attacks involve exhaustively trying all possible keys to decrypt encrypted data, aiming to exploit weak keys or vulnerabilities in the encryption scheme. Cryptographic processes must be designed to withstand these and other potential attacks to ensure the sustained security of sensitive information.

7.1. Brute force attack

The proposed algorithm demonstrates strong defense capabilities against brute force attacks owing to the combination of ASCON 128-bit and RSA encryption techniques. ASCON 128-bit encryption, with its 128-bit key size, creates a formidable barrier against brute force attacks by offering an extensive keyspace that renders exhaustive search impractical. Similarly, RSA encryption, leveraging large key sizes ranging from 2048 to 4096 bits, presents a formidable challenge to attackers attempting to factor the modulus.

By employing a hybrid model that integrates both ASCON-128 bit and RSA encryption, the algorithm further enhances its resilience against brute force attacks. Even if one layer of encryption were compromised, the other layer would remain intact, significantly increasing the computational effort required for attackers to succeed. This multi-layered defense strategy ensures that the encrypted data remains secure and protected against brute force attacks, bolstering the overall security posture of the algorithm.

Table 4
RSA Operations and time taken table

Digits	No of operations	Time (years)
50	1.4×10^{10}	0.0004
75	9.0×10^{12}	0.28
100	2.3×10^{15}	74
200	1.2×10^{23}	3.8×10^{9}
300	1.5×10^{29}	4.9×10^{15}
500	1.3×10^{39}	4.2×10^{25}

Cracking RSA encryption involves factoring the RSA modulus n, the product of two large prime numbers, which is mathematically complex and computationally intensive. The Schroeppel algorithm, referenced by the authors of RSA in 1978, estimates factoring time based on the size of n. Assuming one microsecond per operation, factoring 50-digit n would take around 0.0004 hours, while 100-digit n requires 74 hours, and 500-digit n necessitates 4.2 septillion hours. These time estimates illustrate how the duration required to factor n rises rapidly with the size of the RSA modulus shown in Table 4.

7.2. Linear and differential cryptanalysis

Linear and differential attacks are absolutely unsuccessful when completely encrypted. If you use the linear approximation for two rounds, the ratio of input and output is very strong. Circular transformation is often maintained

uniformly, comparably treating any bit and opposing differential attacks.

7.3. C.I.A. Achievement

• Confidentiality

This indicates that unauthorized individuals are involved when information is shared. The suggested solution establishes confidentiality by encrypting all transmitted entities and parameters.

• Data integrity

Data integrity ensures that no modifications are made to the user as a result of insertion, deletion, or alteration. In other words, although the data has been manipulated, the receiver should offer specific processes to ensure that new information is obtained. Data integrity was achieved in this approach via segmentation.

• Availability

This means to ensure availability and access knowledge to users wherever the need arises. The suggested cryptographic algorithms are hypothetical techniques and are therefore useable at all times. It also supports text formats for broadband data encryption and decryption without loss of information, without losing any scheme where not lost any bit during transmission. The scheme also tests the proposed algorithm encryption using the white spaces and special characters sizes up to 10,000 plain text character.

8. CONCLUSION

The conclusion underscores the pressing need for enhanced data security measures amidst the growing popularity of remote connectivity and data storage solutions. The proposed research introduces a novel Lightweight Hybrid Cryptographic Algorithm, leveraging a two-layer encryption approach for heightened data secrecy. The first layer employs a unique 128-bit Lightweight cryptography technique, integrating festial and permutation/substitution processes with Shannon's diffusion/confusion theory to bolster encryption complexity. Meanwhile, the second layer capitalizes on the dynamic property of the RSA algorithm to further enhance data security and data transfer speeds.

In today's digital landscape, computational speed plays a crucial role in ensuring efficient and secure data transmission. With the increasing volume of data being exchanged and processed, the ability to perform encryption and decryption operations effectively is of utmost importance. In this paper, we have used a dynamic approach to enhance the RSA algorithm, focusing on computational speed and optimizing the encryption and decryption processes. Our suggested dynamic approach stores the encryption and decryption value of every character, and once the character gets repeated, the already stored encrypted and decrypted value is being used, which reduces computational complexity.

We have compared our proffered approach with other existing strategies, i.e. Asymmetric algorithm- Classical RSA, EL Gamal and Symmetric algorithm – AES and the outputs received from the experimentation with regards to encryption and decryption time is found to be better than other asymmetric algorithms and comparatively faster in execution as the number of words increases in terms of symmetric algorithm, i.e. AES algorithm. Furthermore, the Proposed RSA minimized the energy consumption and cost reduction for the encryption and decryption process in data transmission. The outcome attained from the experimental study supports the efficacy of the proposed algorithm over other existing algorithms. Moreover, as there is a threat, that attacker can attack the encryption value of the character that is being stored and use that information to encrypt the message sent. To avoid this type of threats, different encryption processes can be used to encrypt the character that is being stored along with the encryption value of that character. Encryption is done using the same public key and needs to be done at last after the completion of encryption of the complete message. The stored information can also be deleted after the completion of the encryption of the complete message. In this way, the security of the algorithm can be maintained along with the faster computation.

Experimental validation demonstrates the algorithm's superiority over existing lightweight and symmetric/asymmetric encryption techniques across various metrics, including ciphertext size, encryption time, throughput, and security level. Rigorous testing against common cryptographic attacks confirms the algorithm's resilience, meeting the principles of confidentiality, integrity, and availability.

The result of the proposed algorithm outperforms another Lightweight Cryptographic Algorithm. (HIGHT, SEA, LED., RC6, and NLCA) and some of the symmetric and asymmetric algorithm (AES, BLOWFISH, SIT, HOMOMORPHIC RSA, HOMOMORPHIC ElGamal) in regard to the size of the ciphertext, encryption time, throughput, and security level.

The suggested technique might be implemented in the future in various applications, resulting in significantly better outcomes. Like optimizing performance through refined encryption processes and parallelization techniques will be explored to enhance the efficiency of the proposed hybrid algorithm. Standardization efforts and integration with cloud services will be pursued while prioritizing usability with user-friendly interfaces will be emphasized to facilitate practical deployment. Real-world testing across diverse scenarios will remain essential for validating scalability, reliability, and overall suitability of the algorithm.

REFERENCES

[1] Pritilata, and Md Ashiq Mahmood. "Strengthening Data Security Using Multi-Level Cryptography Algorithm." Advanced Computational Paradigms and Hybrid Intelligent Computing: Proceedings of ICACCP 2021. Springer Singapore, 2022.

[2] Kumar, Sanjeev, et al. "Cloud security using hybrid cryptography algorithms." 2021 2nd international conference on intelligent engineering and management (ICIEM). IEEE, 2021.

[3] Hatzivasilis, George, et al. "A review of lightweight block ciphers." Journal of cryptographic Engineering 8 (2018): 141-184.

[4] Chen, Megan, et al. "Diogenes: lightweight scalable RSA modulus generation with a dishonest majority." 2021 IEEE Symposium on Security and Privacy (SP). IEEE, 2021.

[5] Somani, Nikita, and Dharmendra Mangal. "An improved RSA cryptographic system." International Journal of Computer Applications 105.16 (2014).

[6] Yang, Peng, et al. "Promoting a Hybrid Cryptosystem System's Security based on Fresnel lens and RSA Algorithm." 2022 8th International Conference on Systems and Informatics (ICSAI). IEEE, 2022.

[7] Mohamad, Mohd Saiful Adli, Roshidi Din, and Jasmin Ilyani Ahmad. "Research trends review on RSA scheme of asymmetric cryptography techniques." Bulletin of Electrical Engineering and Informatics 10.1 (2021): 487-492.

[8] Kumar, Praveen, et al. "A performance-based comparison of various symmetric cryptographic algorithms in run-time scenario." 2016 International Conference System Modeling & Advancement in Research Trends (SMART). IEEE, 2016.

[9] Kamatchi, T. P., and K. Anitha Kumari. "A hybrid homomorphic model with RSA algorithm and modified enhanced homomorphic encryption technique." 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS). IEEE, 2023.

[10] Munjal, Kundan, and Rekha Bhatia. "A systematic review of homomorphic encryption and its contributions in healthcare industry." Complex & Intelligent Systems 9.4 (2023): 3759-3786.

[11] Kapoor, Vivek, and Rahul Yadav. "A hybrid cryptography technique for improving network security." International Journal of Computer Applications 141.11 (2016): 25-30.

[12] C. A. Lara-Nino, A. Diaz-Perez and M. Morales-Sandoval, "Elliptic Curve Lightweight Cryptography: A Survey," in IEEE Access, vol. 6, pp. 72514-72550, 2018, doi: 10.1109/ACCESS.2018.2881444.

[13] AbdElminaam, Diaa Salama. "Improving the security of cloud computing by building new hybrid cryptography algorithms." International Journal of Electronics and Information Engineering 8.1 (2018): 40-48.

[14] Rohit Minni, Kaushal Sultania, Saurabh Mishra, Prof Durai Raj Vincent PM, VIT University

[15] Hoobi, Mayes M. "Efficient hybrid cryptography algorithm." Journal of Southwest Jiaotong University 55.3 (2020).

[16] Kalpana, Parsi, and Sudha Singaraju. "Data security in cloud computing using RSA algorithm." International Journal of research in computer and communication technology, IJRCCT, ISSN (2012): 2278-5841.

[17] M. Shand and J. Vuillemin, "Fast implementations of RSA cryptography," Proceedings of IEEE 11th Symposium on Computer Arithmetic, Windsor, ON, Canada, 1993, pp. 252-259, doi: 10.1109/ARITH.1993.378085. keywords: {Hardware;Public key cryptography;Application specific integrated circuits;Laboratories;Performance gain;Software performance;Performance analysis;Software measurement;Electronics packaging;Coprocessors},

[18] Boneh, Dan, and Hovav Shacham. "Fast variants of RSA." CryptoBytes 5.1 (2002): 1-9.

[19] Sharma, Ruchita, and Swarnalata Bollavarapu. "Data security using compression and cryptography techniques." International Journal of Computer Applications 117.14 (2015).

[20] Das, Anupam, Shikhar Kumar Sarma, and Shrutimala Deka. "Data security with DNA cryptography." Transactions on Engineering Technologies: World Congress on Engineering 2019. Springer Singapore, 2021.

[21] Sharma, Gaurav, and Ajay Kakkar. "Cryptography Algorithms and approaches used for data security." International Journal of Scientific & Engineering Research 3.6 (2012): 1-6.

[22] Shoeb, Mohammad, and Vishal Kumar Gupta. "A crypt analysis of the tiny encryption algorithm in key generation." International Journal of communication and computer Technologies 1.1 (2012): 9-14.

[23] N. A. Gunathilake, W. J. Buchanan and R. Asif, "Next Generation Lightweight Cryptography for Smart IoT Devices: : Implementation, Challenges and Applications," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 707-710, doi: 10.1109/WF-IoT.2019.8767250. keywords: {Internet of Things;Ciphers;Cloud computing;Encryption;Data processing;Lightweight cryptography (LWC);Internet-of-Things (IoT);encryption},

[24] Daemen, Joan, et al. "Xoodyak, a lightweight cryptographic scheme." (2020).

[25] Engels, Daniel, et al. "Hummingbird: ultra-lightweight cryptography for resource-constrained devices." International conference on financial cryptography and data security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[26] S. Abd Al-Rahman, A. Sagheer and O. Dawood, "NVLC: New Variant Lightweight Cryptography Algorithm for Internet of Things," 2018 1st Annual International Conference on Information and Sciences (AiCIS), Fallujah, Iraq, 2018, pp. 176-181, doi: 10.1109/AiCIS.2018.00042. keywords: {Ciphers;Encryption;Germanium;Internet of Things;Hardware;Lightweight, Cryptographic, Internet of Things (IoT), Block cipher, Symmetric cipher, Advanced Encryption Standard (AES)},

[27] Rahman, Md Mijanur, Tushar Kanti Saha, and Md Al-Amin Bhuiyan. "Implementation of RSA algorithm for speech data encryption and decryption." IJCSNS International Journal of Computer Science and Network Security 12.3 (2012): 74-82.

[28] Al_Barazanchi, Israa, et al. "Modified RSA-based algorithm: A double secure approach." TELKOMNIKA (Telecommunication Computing Electronics and Control) 17.6 (2019): 2818-2825.

[29] Kalpana, Parsi, and Sudha Singaraju. "Data security in cloud computing using RSA algorithm." International Journal of research in computer and communication technology, IJRCCT, ISSN (2012): 2278-5841.

[30] Patel, Sarthak R., and Khushbu Shah. "Security enhancement and speed monitoring of RSA algorithm." International Journal of Engineering Development and Research 2.2 (2014): 2057-63.

[31] F. Thabit, A. Prof, S. Alhomdy, A.H.A. Al-ahdal, Exploration of Security Challenges in Cloud Computing : Issues , Threats , and Attacks with Their Alleviating Techniques Exploration of Security Challenges in Cloud Computing : Issues , Threats , and Attacks with Their Alleviating Techniques, vol. 1, 2020. December.

[32] S.S.S.I. Huang, SEA: secure encrypted data aggregation in mobile WSNs, in: Int. Conf. Comput. Intell. Secur, IEEE, 2007, pp. 526–530, 2007.

[33] G. Bansod, N. Raval, N. Pisharoty, Implementation of a new lightweight encryption design for embedded security, IEEE Trans. Inf. Forensics Secur. (2015), https://doi. org/10.1109/TIFS.2014.2365734.

[34] R. Rivest, M.J.B. Robshaw, R. Sidney, Y.L. Yin, "The RC6 block cipher," first adv, Encryption ... (1998).

[35] D. Hong, et al., HIGHT: A New Block Cipher Suitable for Low-Resource Device, 2006, https://doi.org/10.1007/11894063_4.

[36] M.A. Wright, The advanced encryption standard, Netw. Secur. (2001), https://doi. org/10.1016/S1353-4858(01)01018-2.

[37] M.N. Valmik, P.V.K. Kshirsagar, Blowfish algorithm, IOSR J. Comput. Eng. (2014), https://doi.org/10.9790/0661-162108083.

[38] M. Usman, I. Ahmed, M. Imran, S. Khan, U. Ali, SIT: a lightweight encryption algorithm for secure Internet of things, Int. J. Adv. Comput. Sci. Appl. (2017), https://doi.org/10.14569/ijacsa.2017.080151.

[39] I. Jabbar, Using fully homomorphic encryption to secure cloud computing, Internet Things Cloud Comput. 4 (2) (2016) 13, https://doi.org/10.11648/j. iotcc.20160402.12.

[40] I. Jabbar, Using fully homomorphic encryption to secure cloud computing, Internet Things Cloud Comput. 4 (2) (2016) 13, https://doi.org/10.11648/j. iotcc.20160402.12.