A Novel Data Transmission Model Using Hybrid Encryption Scheme for Preserving Data Integrity

Riyaz Fathima Abdul*, Saravanan Arumugam

Department of Computer Science, Sree Saraswathi Thyagaraja College, Tamil Nadu, India

Received 09 August 2024; received in revised form 28 September 2024; accepted 07 October 2024

DOI: https://doi.org/10.46604/aiti.2024.14114

Abstract

The objective of the study is to introduce a novel hybrid encryption scheme, combining both symmetric and asymmetric encryptions with a data shuffling mechanism, to enhance data obfuscation and encryption security. The approach uses RSA for asymmetric encryption and ChaCha20-Poly1305 for symmetric encryption. To increase the complexity, an additional phase involves reorganizing the RSA-encrypted data blocks. Furthermore, symmetric key generation using the key derivation function is employed to generate the key for symmetric encryption through an asymmetric private key. Decryption entails reversing these procedures. This model significantly enhances security through an additional shuffling step, measured by performance metrics like encryption and decryption times, throughput rate, and the avalanche effect. The method, despite increasing execution time compared to symmetric models, yields comparable results for asymmetric models and ensures robustness. The proposed method outperforms traditional methods regarding resistance to cryptanalytic attacks, including chosen-plaintext and pattern analysis attacks.

Keywords: hybrid encryption, cryptographic security, RSA, ChaCha20-Poly1305, data shuffling

1. Introduction

The widespread adoption of cloud computing has radically transformed data processing, storage, and access in the current digital world [1]. Businesses across multifarious sectors and other private and public organizations have been heavily relying on cloud services to manage their infrastructure, applications, and data. The growing reliance on cloud computing offers benefits like data access, storage, and sharing. On the other hand, however, cloud computing can raise privacy and security concerns, incurring potentially increased security threats and data breaches [2]. Thus, securing sensitive information and guaranteeing privacy against unauthorized access, security breaches, and other advanced cyber threats requires ensuring the confidentiality, integrity, and availability of data in a cloud environment [3-4]. Specifically, encryption is crucial for maintaining the confidentiality and integrity of data in the cloud. To ensure the safety of sensitive information while being communicated or stored, conventional data encryption methods, such as symmetric and asymmetric encryption, have been widely used for a considerable amount of time [5].

Several types of symmetric encryption, including advanced encryption standard (AES), data encryption standard (DES), triple DES, Blowfish, Twofish, Rivest Cipher 4 (RC4), and ChaCha20, use the same key for both encryption and decryption. Such a phenomenon ensures data protection expeditiously and securely. Asymmetric encryption methods, on the other hand, use public-private key pairs to effectuate secure communication, and popular methods include Rivest-Shamir-Adleman (RSA),

* Corresponding author. E-mail address: riyazfathimarf@gmail.com

English language proofreader: Chih-Wei Chang

elliptic curve cryptography (ECC), digital signature algorithms (DSA), and ElGamal encryption. Concerning the practices, researchers utilized the aforementioned methods directly or with some modifications to enhance their functionality on the cloud [6-8].

However, a panoply of limitations and challenges to encryption methods emerges, particularly in cloud environments, despite their critical role in data security. The distributed infrastructure, virtualization, multi-tenancy, and dynamic nature that define cloud computing introduce particular difficulties for data transmission security [9]. Well-established cryptographic methods face challenges such as the need for effective key management, defense against complex cyber threats, and maintaining data integrity during transmission. Thus, an urgent need for innovative solutions is presented to address data security concerns without compromising efficiency or scalability to the growing amount and diversity of data transmitted in cloud systems.

Recently, researchers and practitioners in the field of cloud security have begun to focus on hybrid encryption schemes as a potential and feasible solution to cloud security in a distributed environment [10-11]. To overcome the weaknesses of each method, the hybrid encryption methods integrate the key strengths of both symmetric and asymmetric encryption. Hence, hybrid schemes often combine the two types of encryption methods to facilitate the transfer of data to and from the cloud in a secure, efficient, and scalable way. The most commonly suggested hybrid encryption schemes are combinations of AES with the RSA algorithm [12-13] and AES with ECC [14].

Apropos of confidentiality, authentication, and security, researchers proposed a three-phase hybrid cryptographic (TPHC) algorithm combining AES, DES, and RSA [15]. A study utilized AES, proxy re-encryption, and honey encryption to improve data privacy and authentication, necessitating further security and efficiency analysis, as introduced in Dutta et al. [16]. Moreover, a Twofish encryption algorithm with manifold optimization techniques was also suggested, including Bald Eagle Pelican Optimization [17] and Ant Lion Optimization [18]. Recently, ChaCha20 and secure hash algorithm 3 (SHA3)-based hashing were proposed to mitigate threats in cloud computing, requiring further investigation into vulnerabilities [19].

Notwithstanding significant advancements in hybrid encryption systems, a research gap remains in addressing the challenge of maintaining data integrity during transmission [9, 20]. Existing research primarily emphasizes confidentiality concerns, often overlooking the critical role of data integrity. Trust, regulatory compliance, and prevention of unauthorized modifications all depend on data integrity, which ensures the quality and consistency of information. This highlights the need for new hybrid encryption schemes that prioritize data integrity during cloud data transmission. The study addresses this need by developing advanced solutions to enhance data security in cloud environments, thereby eliciting the limitations of traditional encryption methods, and improving both data protection and integrity.

This study aims to develop and evaluate a new hybrid encryption method to protect data in cloud environments, considering the difficulties, challenges, and gaps in the existing literature. The primary objective of the study is to design a hybrid encryption model that combines symmetric and asymmetric techniques and evaluate its performance and effectiveness in preserving data integrity during transmission. The findings of the study indicate that the proposed model significantly enhances cloud data security by integrating advanced encryption techniques. It is ideal for applications demanding stringent data protection, such as sensitive data storage, secure communications, and regulatory compliance, despite its higher execution time.

The paper is structured as follows: Section 2 explains the framework of the proposed hybrid encryption model. Subsequently, Section 3 outlines the experimental setup and the performance metrics used in assessing the model. Furthermore, Section 4 presents the results from data integrity assessments, performance evaluations, and security analysis against multitudinous attacks. Finally, Section 5 concludes the research findings and suggests future research directions in the field of cloud security and encryption.

2. Proposed Methodology

To enhance data security during transmission in cloud computing environments, the proposed hybrid encryption scheme integrates established symmetric and asymmetric algorithms into a novel hybrid model. The scheme integrates established algorithms into a unified hybrid model, rendering comprehensive protection against cryptographic attacks and ensuring data protection even if one layer is compromised. Fig. 1 depicts the two main phases of this model: encryption and decryption, each involving several steps.

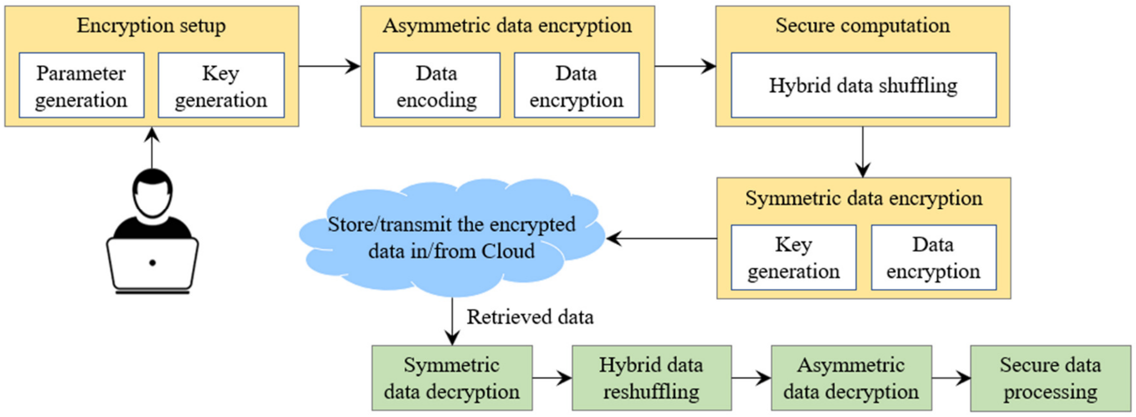

Fig. 1 Overall framework of the proposed data transmission model

Initially, the necessary parameters and keys are generated for the proposed model. The first phase of hybrid encryption commences by encrypting the input data using asymmetric encryption, producing the initial ciphertext. Such a ciphertext is shuffled to improve data confidentiality during transmission thereafter. Finally, symmetric encryption is applied to the shuffled data, and the resulting encrypted text is transmitted or stored in the cloud.

The second phase, on the receiver side, initiates with the symmetric decryption of the data. The symmetrically decrypted data is subsequently subjected to a hybrid data reshuffling step. Finally, asymmetric decryption is performed on the reshuffled data to recover the plaintext. The asymmetric encryption algorithm encrypts the data using a public key and decrypts it using a private key, yielding robust encryption and ensuring confidentiality. Based on this mechanism, the authorized parties are permitted to access the plain text, and the unauthorized ones will be excluded. The hybrid data shuffling randomizes the order of encrypted data, furnishing an additional layer of randomness and security with data unpredictability.

Moreover, symmetric encryption enhances security by ensuring data confidentiality, integrity, and detection of modifications or tampering while being more efficient, faster, and requiring less computational complexity than asymmetric encryption. Therefore, by combining asymmetric and symmetric encryption, the model renders more resilient to various cryptographic attacks. Despite compromising one layer of defense, the other layer continues to proffer additional protection. The sub-sections below visualize a comprehensive explanation of the encryption and decryption phases.

2.1. *Hybrid data encryption phase*

Fig. 2 graphically presents the overall workflow for the encryption phase. The proposed hybrid data transmission model uses Unicode, a character encoding scheme, to accurately represent plain text as numerical values, preserving its original information. This model deploys the Unicode Transformation Format (UTF-8), 8-bit code units to represent characters.

2.1.1 *Asymmetric data encryption*

Once the plain text is encoded (m), asymmetric encryption is carried out. The proposed model employs the RSA algorithm for asymmetric encryption, due to its strong security with large keys, proven reliability, efficient performance, and robust resistance against brute-force attacks. Initially, the parameters and keys are generated for the asymmetric encryption.

Regarding the large primes p and q, n and $\phi(n)$ are calculated as $n = p \times q$ and $\phi(n) = (p - 1) \times (q - 1)$. The encryption component e is chosen such that it is relatively prime to $\phi(n)$, and the decryption component d is computed such that d is the modular multiplicative inverse of e modulo $\phi(n)$, represented as $d \equiv e^{-1} \bmod \phi(n)$. As a result, the public key comprises (n, e), and the private key is the decryption component d. Finally, the output of the asymmetric encryption step is the ciphertext (c_1) computed, as presented in:

$$\text{AsymmetricEncryptedData } c_1 = m^e \bmod n \tag{1}$$

where m represents the plaintext message, e is the encryption component, n is a product of 2 large primes, and c_1 is the encrypted text. In the next stage, this ciphertext is used as input for further processing.

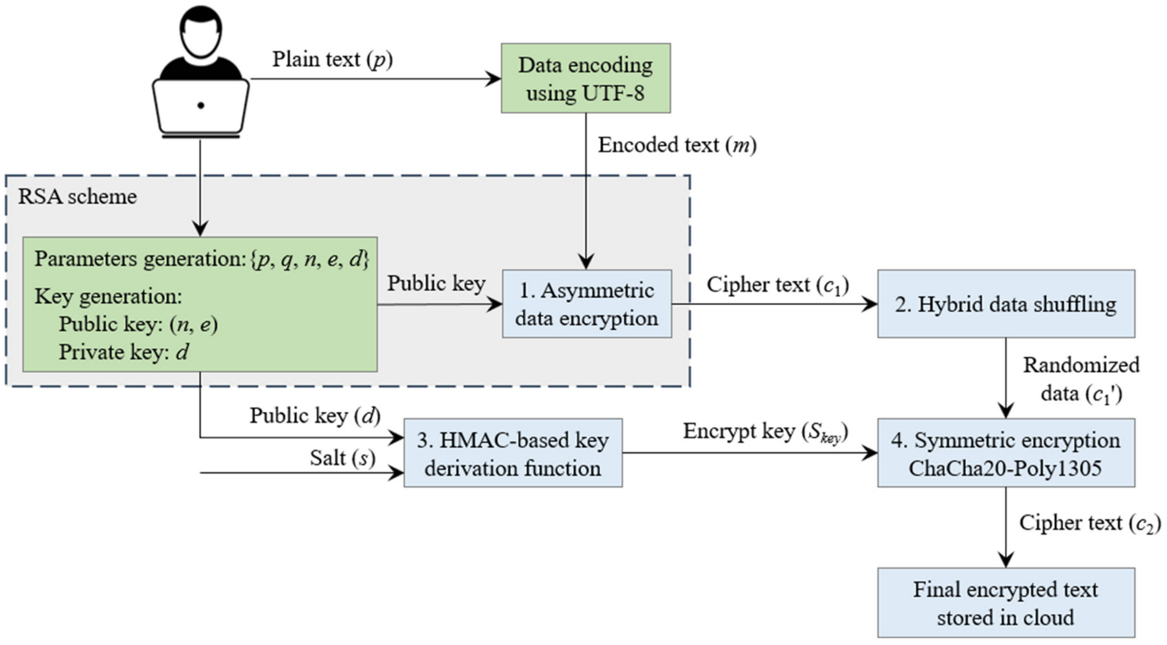

Fig. 2 Workflow for the encryption phase

2.1.2. *Hybrid data shuffling*

The proposed model utilizes hybrid shuffling, enhancing randomness, unpredictability, and security in cryptographic procedures while maintaining integrity and confidentiality through the efficiency and simplicity of the inside-out algorithm. The algorithm encompasses the following steps:

(1) The array is initially partitioned into two sections

(2) The inside-out procedure is applied to shuffle each partition, where each element is iterated over, a random index is selected, and the element is exchanged with the one at that index

(3) The final shuffled array is created by combining the shuffled partitions

Thus, it accepts input from the previous step (c_1) and shuffles the data, producing the permuted data (c_1'), as mentioned in:

$$\text{PermutedData } c_1' = \text{DataShuffle}(c_1) \tag{2}$$

The approach optimizes computational performance by partitioning the array and shuffling each partition independently, ensuring quicker operations and a more uniform distribution of elements. This hybrid technique enhances the balance between computational complexity and randomness, extending a veritable cornucopia of applications where both speed and randomness are critical factors in achieving optimal performance and security. Algorithm 1 explains the pseudocode for the hybrid shuffling algorithm.

```
Algorithm 1: Hybrid shuffling algorithm
Input: $c_1$ – ciphertext to be shuffled
Output: Shuffled array $c_1'$
Procedure HybridShuffling(array $c_1$)
Begin
   //Determine the length of the array and partition them
   $n$ = len($c_1$)
   $L_{partition}$ = $c_1$[0:n/2]; $R_{partition}$ = $c_1$[n/2:n]
   //Shuffle each partition independently
   For each element $a$ in $L_{partition}$
      //Select a random index $j$ such that $0 \leq j <$ index of $a$.
      $j$ = random.randint(0, $c_1$.index($a$) - 1)
      Swap $a$ and $c_1[j]$
   For each element $b$ in $R_{partition}$
      //Select $a$ random index $j$ such that index of $b \leq j <$ len($R_{partition}$)
      $j$ = random.randint($c_1$.index($b$), len($c$) - 1)
      Swap $b$ and $c_1[j]$
      //Combine the shuffled partitions
      $c_1'$ = $L_{partition}$ + $R_{partition}$
   Return ShuffledArray $c_1'$.
End Procedure
```

2.1.3. Symmetric key generation using key derivation function

The algorithm uses a key derivation function (KDF) to securely derive symmetric keys from asymmetric key pairs, simplifying key management, reducing storage and distribution complexity, and increasing efficiency and security. The model utilizes the hash-based message authentication code (HMAC)-based KDF (HKDF), a robust and efficient cryptographic method known for its flexibility. The system uses HMAC and a secure hash function to generate a pseudorandom key (PRK), ensuring the key's integrity and authenticity using SHA-256. The HKDF process can be apportioned in the following steps:

(1) Input parameters: The RSA private key (d), a randomly generated salt (s), and the desired key length (L_{key})
(2) PRK generation: An HMAC function is applied using SHA-256 as the hash function. This function takes the RSA private key and salt as inputs to produce a PRK
(3) Key expansion: The PRK is then expanded to generate a symmetric key of 256 bits using HKDF's key expansion algorithm

Thus, the model applies an HMAC function, which includes private key d and salt s, to generate a PRK, then expands it to the desired length (S_{key}) while maintaining its security properties, as outlined in:

$$\text{SymmetricKey } S_{key} = \text{HMAC\_SHA-256}(d, s) \tag{3}$$

```
Algorithm 2: HMAC-based key derivation function
Input: RSA private key ($d$), a randomly generated salt $s$, key length $L_{key}$
Output: $S_{key}$ symmetric key of length 256 bits
Procedure HKDF (RSA private key $d$, Salt $s$, key length $L_{key}$)
Begin
   //Calculate the HMAC using SHA-256 as the hash function
   hmac_digest = HMAC_SHA-256 (private key $d$, salt $s$)
   //Extract the pseudorandom key (PRK)
   PRK = hash_digest
   //Expand the pseudorandom key to the desired output using the key expansion algorithm
   $S_{key}$ = HKDF_Expand(PRK, $L_{key}$)
   Return $S_{key}$
End Procedure
```

The HKDF algorithm enables the sender and receiver to independently generate symmetric encryption keys, improving communication without direct key transmission, and thereby enhancing security. The technique enhances cryptographic resistance by providing variability and unpredictability in key generation, while adding salt ensures unique, secure hash outputs, preventing precomputed attacks. Algorithm 2 presents the algorithm pseudocode for the HMAC-based KDF using SHA-256.

2.1.4. Symmetric data encryption

After generating a symmetric key, the symmetric key encryption is applied to the shuffled data c_1'. The proposed model applies ChaCha20-Poly1305 for symmetric encryption, which combines ChaCha20, a stream cipher, with the Poly1305 message authentication code (MAC). This combination is known as authenticated encryption with additional data (AEAD). The ChaCha20 symmetric stream cipher is an efficient and secure encryption algorithm in which the data is encrypted using a key, nonce, and counter by XORing it with a stream created by ChaCha20. Poly1305 is a secure MAC that ensures the authenticity of encrypted data by converting the message and secret key into a fixed-size authenticator. Concerning the function, this algorithm works as follows:

(1) Creating the symmetric key (S_{key}) for encryption
(2) Initialing the ChaCha20 cipher state using the symmetric key (S_{key}) and a nonce (n)
(3) For each block of the input data (c_1'), generating a ChaCha20 keystream and exclusive or (XOR) it with the data to produce ciphertext (c_2)
(4) Setting up the Poly1305 authentication state using S_{key}
(5) Processing the encrypted data blocks to generate the authentication tag (A_{tag}) for verifying data integrity
(6) Output the ciphertext (c_2) and the authentication tag (A_{tag})

Hence, the symmetric key encryption takes c_1' and provides c_2 as the cipher text and authentication tag A_{tag}, as shown in:

$$\text{SymmetricEncrytedData } c_2 = \text{ChaCha20\_Poly1305\_Encrypt}(c_1', S_{key}) \tag{4}$$

ChaCha20-Poly1305 is a method that offers confidentiality, authenticity, and integrity verification through cipher text and a cryptographic hash-based tag, offering efficiency and robust message authentication. Upon decryption, the receiver computes the tag using the received cipher text and compares it with the received tags. Therefore, the authenticity of the received ciphertext is confirmed if the computed and received tags match, while a non-match indicates potential tampering or modification during transmission. Algorithm 3 presents the pseudocode for symmetric encryption using ChaCha20-Poly1305.

```
Algorithm 3: ChaCha20-Poly1305 encryption and authentication
Input: input text $c_1$', Symmetric encryption key $S_{key}$, Nonce $n$
Output: Ciphertext $c_2$, Authentication tag $A_{tag}$
Procedure ChaCha20_Poly1305_Encrypt($c_1$', $S_{key}$, $n$):
Begin
  State = InitializeState($S_{key}$, $n$) //ChaCha20 key setup
  For each block in $c_1$' //Generate ChaCha20 keystream blocks
    KeystreamBlock = GenerateChaCha20Keystream(State)
    IncrementNonce($n$)
  $c_2$ = $c_1$' XOR KeystreamBlock //Encrypt input text with ChaCha20 keystream
  PolyState = InitializePoly1305State($S_{key}$) //Initialize Poly1305 state with key
  For each block in $c_1$'  //Process message blocks
    PolyState = Poly1305Update(PolyState, block)
  $A_{tag}$ = Poly1305Finalize(PolyState) //Finalize and obtain the authentication tag
  Return ($c_2$, $A_{tag}$)
End Procedure
```

2.2. *Hybrid data decryption phase*

The second phase of the proposed model is the hybrid data decryption phase. Fig. 3 depicts the overall workflow for the decryption phase of the proposed data transmission model using a hybrid scheme. Initially, for asymmetric encryption, the parameters and keys are generated. The n and $\phi(n)$ are determined for the huge prime integers p and q as follows: $n = p \times q$ and $\phi(n) = (p-1) \times (q-1)$. The decryption component d is selected so that, $d \equiv e^{-1} \bmod \phi(n)$ and the encryption component e is determined by choosing a relative prime to $\phi(n)$. At this point, (n, e) and d will be the public and private keys, respectively. The private key is utilized to create the symmetric key, and the cipher text is converted into plain text sequentially. The steps are explained in the below subsections.

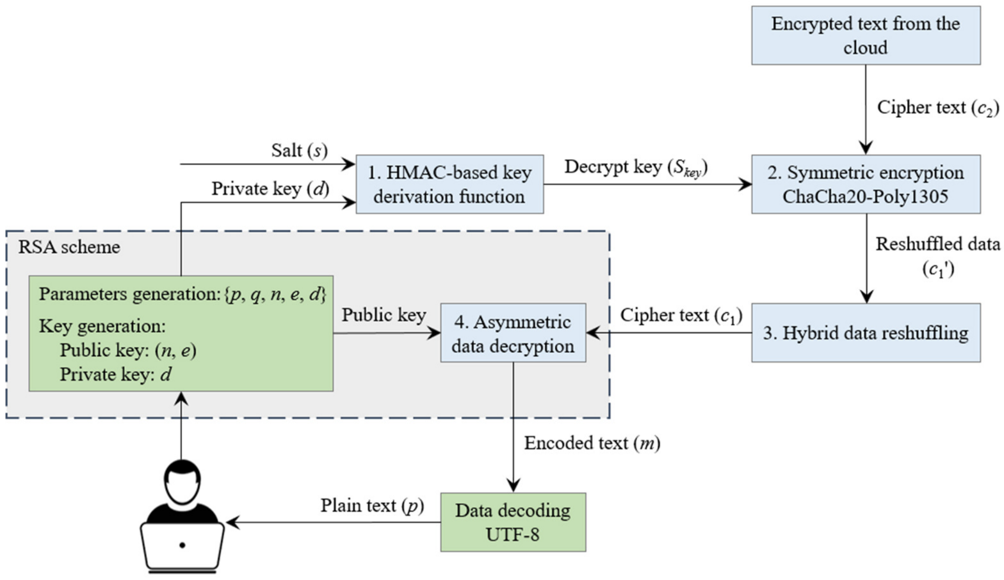

Fig. 3 Workflow for the decryption phase

2.2.1. *Symmetric data decryption with KDF*

As a first step, the proposed algorithm deploys symmetric decryption, which uses the KDF to securely derive a symmetric key from asymmetric key pairs. The proposed model employs HKDF for deriving symmetric keys from shared secret keys. The method generates symmetric keys using the HMAC function, which is explained in Section 2.1.3. Algorithm 2 explains the pseudocode, using the private key d and salt s to create PRK, which expands to produce the symmetric key with a length of 256 bits thereafter.

Upon generating a symmetric key, the receiver applies the symmetric decryption to the inputs, such as the cipher text c_2 and authentication tag A_{tag}. The algorithm works as follows:

(1) The decryption process commences by setting up the ChaCha20 cipher state using the symmetric key (S_{key}) and nonce (n)

(2) ChaCha20 keystream block is generated and XORed with the ciphertext to retrieve the decrypted text (c_1')

(3) The Poly1305 authentication state is initialized using the same symmetric key (S_{key})

(4) Regarding each block of the decrypted text (c_1'), the Poly1305 state is updated to authenticate the data

(5) After processing all blocks, the Poly1305 state is finalized to produce the verified authentication tag ($A_{tag\_verified}$)

(6) The computed authentication tag ($A_{tag\_verified}$) is compared with the received tag (A_{tag}). If they match, the decrypted text (c_1') is returned. Otherwise, an error message is returned, indicating that authentication failed.

Thus, as in the formula below, the symmetric key decryption takes c_2 and A_{tag} as inputs and renders c_1' as the decrypted text. Algorithm 4 presents the pseudocode for symmetric decryption using ChaCha20-Poly1305 with a symmetric key.

$$\text{SymmetricDecryptedData } c_1' = \text{ChaCha20\_Poly1305\_Decrypt}(c_2, S_{key}) \tag{5}$$

Algorithm 4: ChaCha20-Poly1305 decryption and authentication

```
Input: Ciphertext $c_2$, Authentication tag $A_{tag}$ input text $c_1'$, Symmetric decryption key $S_{key}$, Nonce $n$
Output: decrypted shuffled text $c_1'$
Procedure ChaCha20_Poly1305_Decrypt($c_2$, $S_{key}$, $n$):
Begin
   State = InitializeState($S_{key}$, $n$) //ChaCha20 key setup
   For each block in $c_2$ //Generate ChaCha20 keystream blocks
      KeystreamBlock = GenerateChaCha20Keystream(State)
      IncrementNonce($n$)
   $c_1'$ = $c_2$ XOR KeystreamBlock //Decrypt input text with ChaCha20 keystream
   PolyState = InitializePoly1305State($S_{key}$) //Initialize Poly1305 state with key
   For each block in $c_1'$ //Process message blocks
      PolyState = Poly1305Update(PolyState, block)
   $A_{tag\_verified}$ = Poly1305Finalize(PolyState) //Finalize and obtain the authentication tag
   If $A_{tag\_verified} = A_{tag}$ then return ($c_1'$) //Return decrypted text if authentication is successful
   Else return $Err\_Msg$
End Procedure
```

2.2.2. *Hybrid data reshuffling*

In the next step of the decryption phase, the proposed model applies hybrid reshuffling, which applies an inside-out algorithm. The algorithm initially partitions the array into two sections. Section 2.1.2 explains the shuffling algorithm. Algorithm 1 presents the pseudocode for the hybrid shuffling algorithm, which accepts input c_1' and outputs c_1. The algorithm initially divides the array into two halves, and then the inside-out procedure is used to shuffle each partition by randomly selecting an index for each element and exchanging it with the one at that index. Finally, the shuffled partitions are combined to generate a final shuffled array. As a result, it takes the data from the previous step (c_1') and shuffles it to produce the reshuffled data (c_1), as specified in the formula below. The next step of the decryption phase then uses the reshuffled data as an input.

$$\text{ReshuffledData } c_1 = \text{DataShuffle}(c_1') \tag{6}$$

2.2.3. *Asymmetric data decryption*

In the next step, the asymmetric decryption is carried out using the RSA algorithm, which uses the generated private key *d*. Thus, the asymmetric key *d* decrypts the reshuffled data (c_1) from the previous step. As a result, the output of the asymmetric decryption step is the decoded text (m), computed as in:

$$\text{AsymmetricDecryptedData } m = c_1^d \bmod n \tag{7}$$

where n is determined for the prime integers p and q as $n = p \times q$. As the decrypted text is obtained as encoded text, it must be decoded using methods used in the encryption phase. Thus, UTF-8 decoding is applied at the end of the hybrid decryption phase to convert the decrypted text m to its plain text form m.

3. Experimental Setup

This section describes the experimental setup and evaluation parameters used to evaluate the proposed model's efficiency. The proposed model is implemented in a Python environment and simulated using CloudSim. The proposed model's performance was evaluated using multifarious parameters such as cryptographic execution time, throughput, and the avalanche effect as discussed below:

Cryptographic execution time: This metric considers the overall time taken to encrypt or decrypt the unique data. The evaluation is carried out on the plain text, which has different sizes. To ensure the reliability of the results, the experiments are conducted 10 times, and the average time for encryption and decryption is used for the analysis.

Throughput rate: This metric is an essential indicator to evaluate the efficiency of the cryptographic algorithm, as there is a direct correlation between the throughput rate and the performance of the algorithm. Thus, increased performance leads to a higher throughput rate. The formula below provides the calculation method for the throughput rate.

$$\text{ThroughputRate} = \frac{\text{PlainText\_Size}}{\text{Execution\_Time}} \tag{8}$$

Avalanche test: It is a metric or property in which a small change in the input results in a significant change in the output. A robust algorithm should have a significant avalanche effect, rendering output unpredictable and unrelated to input, thereby hindering the susceptibility of the detection patterns and breaking encryption to attackers. The optimal assessment of a ciphertext is determined by strict avalanche criteria, which require a single modification to alter 50% of bits.

4. Results and Discussion

This section presents the results obtained from the experimental analysis of the proposed model. The analysis involves evaluating each phase of the model, including its execution time, throughput analysis, and avalanche effect. Furthermore, it presents a comparative analysis of the proposed model with existing standard symmetric and asymmetric algorithms, and further discussion regarding the security analysis against various attacks is established.

4.1. Analysis of the proposed model

Initially, the proposed study conducts a simple analysis to evaluate the performance of the model at each step of the encryption and decryption process. Table 1 presents the execution time (in milliseconds) of each step in the encryption and decryption process for different plain text sizes varying from 32, 64, 96, 128, 160, 192, 224, 256, 288, and 320 bytes. The results indicate that the execution time of the proposed decryption process is higher than that of the encryption process. It can be attributed that decryption often involves complex operations, especially in RSA involving modular exponentiation and the use of a private key. Therefore, the time complexity of the proposed model is highly influenced by the asymmetric decryption process rather than the encryption process.

Table 1 Evaluation of execution time

Size (bytes)	Time in ms									
	Encod.	RSA Enc.	Shuffling	Sym. Enc.	Total Enc.	Decod.	RSA Dec.	Reshuffling	Sym. Dec.	Total Dec.
32	0.0001	0.0091	0.0000	0.0002	0.0094	0.0000	0.3344	0.0000	0.0001	0.3345
64	0.0000	0.0160	0.0001	0.0003	0.0163	0.0000	0.6725	0.0000	0.0001	0.6726
96	0.0000	0.0227	0.0001	0.0002	0.0230	0.0000	1.0221	0.0000	0.0001	1.0222
128	0.0000	0.0364	0.0001	0.0003	0.0367	0.0000	1.6060	0.0000	0.0001	1.6062
160	0.0000	0.0494	0.0001	0.0003	0.0498	0.0000	2.1625	0.0000	0.0001	2.1627
192	0.0000	0.0495	0.0001	0.0003	0.0499	0.0000	1.9945	0.0000	0.0001	1.9946
224	0.0000	0.0606	0.0001	0.0003	0.0610	0.0000	2.3229	0.0001	0.0001	2.3231
256	0.0000	0.0626	0.0002	0.0003	0.0631	0.0000	2.6608	0.0001	0.0002	2.6611
288	0.0000	0.0721	0.0002	0.0003	0.0725	0.0000	3.1720	0.0000	0.0002	3.1722
320	0.0000	0.1027	0.0002	0.0004	0.1033	0.0000	3.9098	0.0001	0.0002	3.9100

Furthermore, the average time to execute the encryption and decryption process for text with varying sizes is presented in Table 2. The results signify that the average total time for encryption and decryption is 69.28 seconds, with 2.8 seconds for encryption and 66.66 seconds for decryption. Additionally, the throughput rate increases with the size of the plaintext.

Table 2 Analysis of the proposed model with performance metrics

Plaintext size (KB)	Enc. Time (s)	Dec. Time (s)	Total time (s)	Throughput (Kb/sec)
250	1.550	37.922	39.472	6.334
500	2.433	58.207	60.640	8.245
750	3.315	78.493	81.080	9.250
1000	3.903	92.017	95.920	10.425
Average	2.800	66.660	69.278	8.564

The avalanche effect for the proposed model is computed using plaintext sizes ranging from 10 KB to 100 KB, and the results are presented in Fig. 4. The avalanche effect, represented as a bit difference, refers to the number of differing bits between the encrypted data of the original plaintext and the modified plaintext. The results indicate that the average avalanche effect for the proposed model is 50%, which is a strong indication of the effectiveness of the cryptographic algorithm.

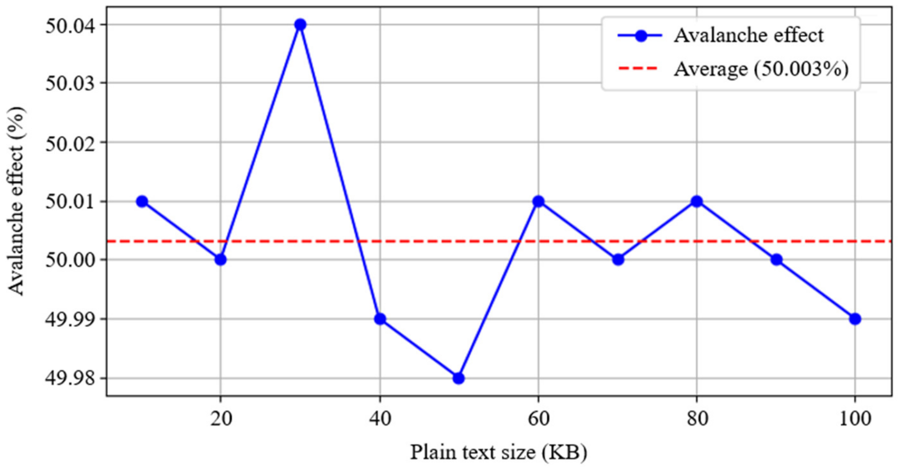

Fig. 4 Avalanche effect of the encryption process

4.2. *Comparative analysis*

This section presents a comparative analysis of the proposed model with the standard symmetric and asymmetric encryption algorithms. The proposed hybrid cryptographic model is compared with other individual symmetric encryption algorithms, including AES, DES, 3DES, Blowfish, Twofish, RC4, and ChaCha20, by varying the plain text size from 10 KB to 100 KB. Fig. 5 displays the average time (in seconds) required for the encryption and decryption processes for various symmetric cryptographic models. The results indicate that the average encryption time for RC4 yields the lowest, followed by Chacha20, Blowfish, AES, and Twofish, while DES, 3DES, and the proposed model take more time than others. Thus, the proposed hybrid model exhibits a high execution time compared to other standard symmetric encryption models.

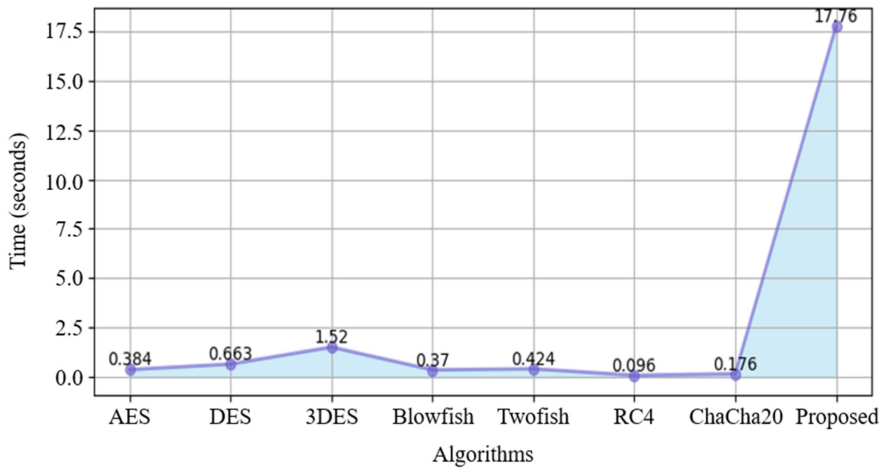

Fig. 5 Average execution time of symmetric encryption models

Furthermore, the proposed hybrid cryptographic model is also compared with other individual asymmetric encryption algorithms such as RSA, elliptic curve digital signature algorithm (ECDSA), DSA, and ElGamal encryption models by varying the plain text size from 10 KB to 100 KB. The average time taken (in seconds) for the encryption and decryption processes for different asymmetric cryptographic models is depicted in Fig. 6, in which dotted lines represent the average values of the respective models. The results indicate that the average encryption for ECDSA (13.15s) and DSA (14.47s) takes the lowest time, followed by ElGamal (16.85s) and RSA (16.86s), while the proposed algorithm takes 18.41 seconds. This symbolizes that the execution time of the proposed model is similar to the other standard asymmetric encryption models.

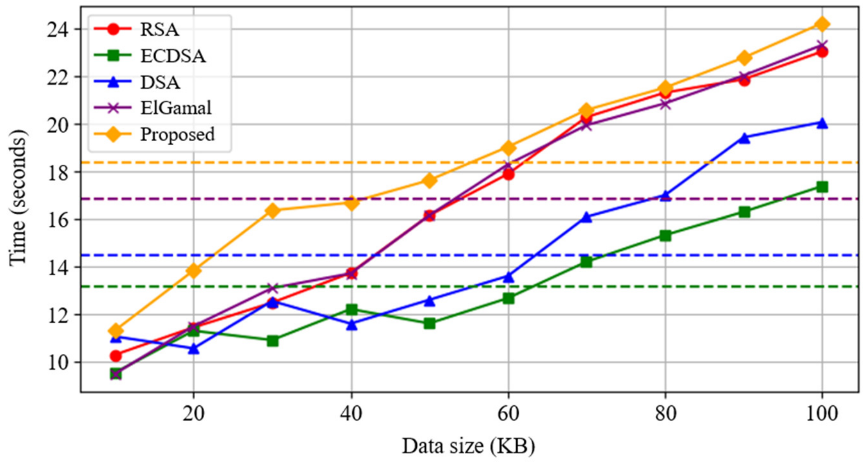

Fig. 6 Performance comparison with asymmetric encryption models

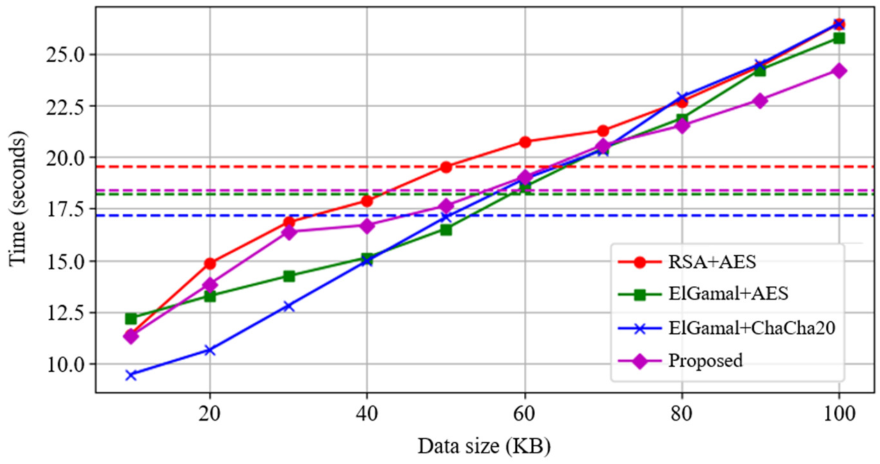

Fig. 7 Performance comparison with hybrid encryption models

Besides, the proposed hybrid cryptographic model is also compared with other hybrid models by replacing the symmetric (RSA) and asymmetric models (ChaCha20). The hybrid algorithms for comparison are created by selecting the most effective combinations from the analysis: RSA+AES, ElGamal+AES, and ElGamal+ChaCha20. RC4 is not selected due to its vulnerabilities and weaknesses in security, such as susceptibility to various attacks. These hybrid models are assessed by varying the plain text size from 10 KB to 100 KB. The time taken (in seconds) for the encryption and decryption processes for different hybrid models is shown in Fig. 7, with dotted lines representing average values. The results indicate that the average encryption for ElGamal+AES and ElGamal+Chacha20 is 17.32 seconds and 17.4 seconds, while that of RSA+AES and the proposed model is 19.5 seconds and 18.4 seconds.

Similarly, the throughput rate and avalanche effects are assessed for significant symmetric, asymmetric, and hybrid models. Fig. 8 displays the average throughput rate for various models, with plaintext sizes ranging from 10 KB to 100 KB. The study reveals that symmetric models like AES and ChaCha20 achieve high throughput rates, while single asymmetric models like RSA and ElGamal have lower rates due to simplicity.

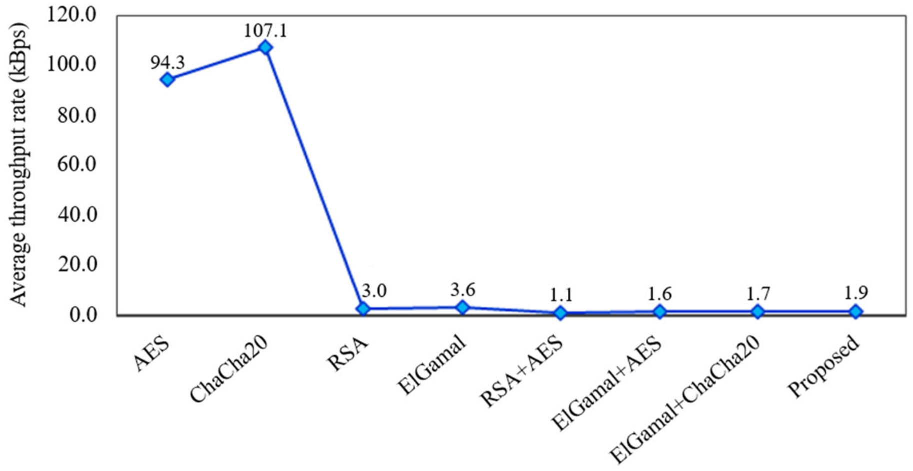

Fig. 8 Throughput rate comparison with hybrid encryption models

Moreover, Fig. 9 shows the average avalanche effect for significant symmetric, asymmetric, and hybrid models, varying the plaintext size from 10 KB to 100 KB. The proposed model has the highest average avalanche effect at 50%, exhibiting the highest sensitivity to single-bit changes among tested encryption methods.

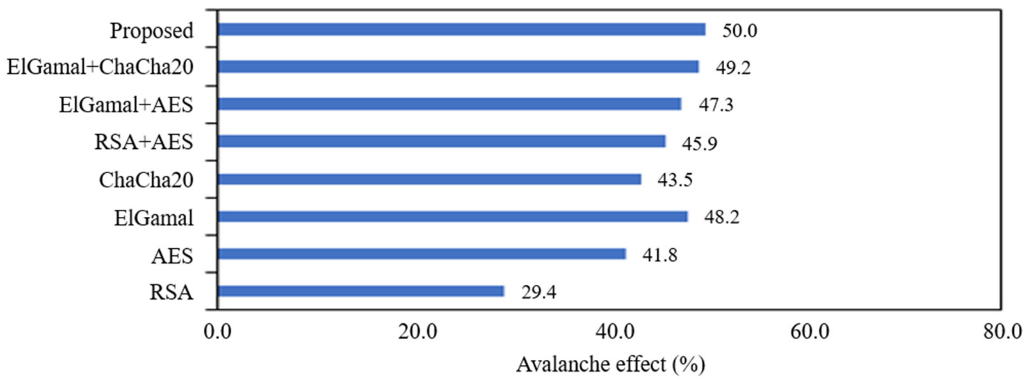

Fig. 9 Average Avalanche effect comparison

4.3. Security analysis against various attacks

Any cryptographic model's sustainability is determined by its reliability and robustness against various attacks performed by attackers to compromise the confidentiality and integrity of the data. The proposed algorithm, which combines the RSA for asymmetric encryption, ChaCha20-Poly1305 for symmetric encryption, and a hybrid data shuffling technique, is secure against various attacks, as discussed below.

- *Brute force attacks*: RSA encryption, with a key size of 2048 bits or more, hampers the effectiveness of brute-force attacks due to the vast number of check possibilities. Moreover, ChaCha20-Poly1305 is also a robust security tool due to its 256-bit key size, making brute force attacks impractical due to its large key space.
- *Cryptanalysis attacks*: ChaCha20 encryption is resistant to a variety of cryptanalysis methods, including differential and linear cryptanalysis. Also, using it with Poly1305 ensures the authenticity and integrity of the original plain text.
- *Chosen ciphertext/plaintext attacks:* By utilizing secure padding strategies, RSA encryption can prevent ciphertext attacks. Moreover, the ChaCha20-Poly1305, an authenticated encryption scheme, ensures ciphertext integrity and authenticity, rendering resistance to specific ciphertext attacks. Additionally, shuffling adds a layer of security to the chosen plaintext attack, increasing its complexity.
- *Replay attacks*: The nonce used in ChaCha20-Poly1305 ensures that each message is unique. However, reusing nonces with the same key can lead to security vulnerabilities.

- *Man-in-the-middle attacks*: The combination of RSA and ChaCha20-Poly1305 enables a secure method of exchanging keys and encrypting data, protecting against attacks that include a man in the middle.
- *Data tampering and integrity attacks*: Poly1305 ensures message integrity and authenticity by verifying the integrity of the ciphertext before decryption and detecting any tampering with the encrypted message.

Thus, the security analysis indicates that the model is strong in terms of confidentiality and integrity, with efficient performance contributing to availability. The security analysis of the proposed model highlights its efficiency apropos confidentiality, integrity, reliability, and robustness to various attacks.

5. Conclusion

The study proposed a novel hybrid encryption scheme that combines RSA for asymmetric encryption and ChaCha20-Poly1305 for symmetric encryption. By adding a hybrid shuffling mechanism, the model significantly improves security and efficiency. Manifold metrics, such as encryption and decryption times, throughput rate, and the avalanche effect, evaluate the performance of the proposed model. The model's average execution time is 69.28 seconds, and its average throughput rate is 8.564 kBps. Furthermore, the proposed model has a 50% avalanche effect, which is a strong indication of a good cryptographic algorithm. Despite the longer execution time than symmetric methods, the model performs similarly to asymmetric models and is resistant to cryptanalytic attacks.

However, the shuffling mechanism and the dual encryption procedure cause a higher computational overhead, which is the primary limitation of the proposed system. This may not be suitable for environments with limited resources or real-time applications that need minimal latency. Hence, further enhancements must focus on refining the shuffling algorithm to minimize computational cost and investigating alternative lightweight cryptographic primitives to enhance efficiency. Additionally, testing the model in several real-world environments, like IoT networks and large-scale cloud systems, will offer a better understanding of its practical use and effectiveness. Enhancing the model to address emerging threats or optimizing it for specific applications could render additional improvements. Furthermore, future research could explore incorporating post-quantum cryptographic algorithms to safeguard the encryption system from potential quantum computing risks. The cloud scenario with a different number of users and the utilization of memory and CPU will be the subject of additional in-depth investigation.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] S. Kolasani, "Innovations in Digital, Enterprise, Cloud, Data Transformation, and Organizational Change Management Using Agile, Lean, and Data-Driven Methodologies," International Journal of Machine Learning and Artificial Intelligence, vol. 4, no. 4, pp. 1-18, 2023.

[2] A. Saravanan and S. Sathya Bama, "A Review on Cyber Security and the Fifth Generation Cyberattacks," Oriental Journal of Computer Science and Technology, vol. 12, no. 2, pp. 50-56, 2019.

[3] A. Saravanan, S. Sathya Bama, S. Kadry, and L. K. Ramasamy, "A New Framework to Alleviate DDoS Vulnerabilities in Cloud Computing," International Journal of Electrical and Computer Engineering, vol. 9, no. 5, pp. 4163-4175, 2019.

[4] A. Saravanan and S. Sathya Bama, "CloudSec (3FA): A Multifactor With Dynamic Click Colour-Based Dynamic Authentication for Securing Cloud Environment," International Journal of Information and Computer Security, vol. 20, no. 3-4, pp. 269-294, 2023.

[5] Q. Zhang, "An Overview and Analysis of Hybrid Encryption: The Combination of Symmetric Encryption and Asymmetric Encryption," 2nd International Conference on Computing and Data Science, pp. 616-622, 2021.

[6] A. E. Adeniyi, K. M. Abiodun, J. B. Awotunde, M. Olagunju, O. S. Ojo, and N. P. Edet, “Implementation of a Block Cipher Algorithm for Medical Information Security on Cloud Environment: Using Modified Advanced Encryption Standard Approach,” Multimedia Tools and Applications, vol. 82, no. 13, pp. 20537-20551, 2023.
[7] K. K. Reddy, A. R. Chadha, P. S. Nikhil, and S. Sountharrajan, “Hybrid Cryptography Techniques for Data Security in Cloud Computing,” IEEE International Conference on Computing, Power and Communication Technologies, pp. 1836-1842, 2024.
[8] B. P. Kavin and S. Ganapathy, “A New Digital Signature Algorithm for Ensuring the Data Integrity in Cloud Using Elliptic Curves,” International Arab Journal of Information Technology, vol. 18, no. 2, pp. 180-190, 2021.
[9] Y. Alghofaili, A. Albattah, N. Alrajeh, M. A. Rassam, and B. A. S. Al-Rimy, “Secure Cloud Infrastructure: A Survey on Issues, Current Solutions, and Open Challenges,” Applied Sciences, vol. 11, no. 19, article no. 9005, 2021.
[10] A. Orobosade, T. A. Favour-Bethy, A. B. Kayode, and A. J. Gabriel, “Cloud Application Security Using Hybrid Encryption,” Communications on Applied Electronics, vol. 7, no. 33, pp. 25-31, 2020.
[11] S. Arumugam, “An Effective Hybrid Encryption Model using Biometric Key for Ensuring Data Security,” International Arab Journal of Information Technology, vol. 20, no. 5, pp. 796-807, 2023
[12] R. Akter, M. A. R. Khan, F. Rahman, S. J. Soheli, and N. J. Suha, “RSA and AES Based Hybrid Encryption Technique for Enhancing Data Security in Cloud Computing,” International Journal of Computational and Applied Mathematics & Computer Science, vol. 3, pp. 60-71, 2023.
[13] R. Nadaf, N. B. Sumangala, M. Mandi, and A. Konnur, “Symmetric and Asymmetric Cryptographic Approach Based Security Protocol for Key Exchange,” International Conference on Applied Intelligence and Sustainable Computing, pp. 1-6, 2023.
[14] S. Rehman, N. Talat Bajwa, M. A. Shah, A. O. Aseeri, and A. Anjum, “Hybrid AES-ECC Model for the Security of Data Over Cloud Storage,” Electronics, vol. 10, no. 21, article no. 2673, 2021.
[15] D. Kumar Shukla, O. Khalaf, R. Vallabhaneni, S. Kumar Srivastava, and S. Algburi, “A Three-Phase Hybrid Cryptography Algorithm: Utilized in Public Sensor Network for Data Security With an Enhancement of Hashing Algorithm,” International Journal of Computing and Digital Systems, vol. 15, no. 1, pp. 1-13, 2024.
[16] A. Dutta, R. Bose, S. Roy, and S. Sutradhar, “Hybrid Encryption Technique to Enhance Security of Health Data in Cloud Environment,” Archives of Pharmacy Practice, vol. 14, no. 3, pp. 41-47, 2023.
[17] S. K. Maddila and N. Vadlamani, “A Novel Efficient Hybrid Encryption Algorithm Based on Twofish and Key Generation Using Optimization for Ensuring Data Security in Cloud,” Journal of Information & Knowledge Management, vol. 23, no. 01, article no. 2350062, 2024.
[18] A. Almalawi, S. Hassan, A. Fahad, and A. I. Khan, “A Hybrid Cryptographic Mechanism for Secure Data Transmission in Edge AI Networks,” International Journal of Computational Intelligence Systems, vol. 17, no. 1, article no. 24, 2024.
[19] Anjana and A. Singh, “An Enhanced Three Layer Cryptographic Algorithm for Cloud Information Security,” International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 17s, pp. 615-627, 2024.
[20] H. Abroshan, “A Hybrid Encryption Solution to Improve Cloud Computing Security Using Symmetric and Asymmetric Cryptography Algorithms,” International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, pp. 31-37, 2021.