Computer Systems Science & Engineering
DOI:10.32604/csse.2022.022404
Article

Tech Science Press

Binary Multifold Encryption Technique for Complex Cloud Systems

N. Ansgar Mary[1,*] and T. Latha[2]

[1]Department of Information and Technology, St. Xavier's Catholic College of Engineering, Tamil Nadu, 629001, India
[2]Department of Electronics and Communication Engineering, St. Xavier's Catholic College of Engineering, Tamil Nadu, 629001, India
*Corresponding Author: N. Ansgar Mary. Email: ansgarmary123@gmail.com
Received: 06 August 2021; Accepted: 18 September 2021

Abstract: Data security is a major cloud computing issue due to different user transactions in the system. The evolution of cryptography and cryptographic analysis are regarded domains of current research. deoxyribo nucleic acid (DNA) cryptography makes use of DNA as a sensing platform, which is then manipulated using a variety of molecular methods. Many security mechanisms including knowledge-based authentication, two-factor authentication, adaptive authentication, multifactor authentication and single password authentication have been deployed. These cryptographic techniques have been developed to ensure confidentiality, but most of them are based on complex mathematical calculations and equations. In the proposed approach, a novel and unique Hybrid helix scuttle-deoxy ribo nucleic acids (HHS-DNA) encryption algorithm has been proposed which is inspired by DNA cryptography and Helix scuttle. The proposed HHS-DNA is a type of multifold binary version of DNA (MF-BDNA). The major role of this paper is to present a multifold HHS-DNA algorithm to encrypt the cloud data assuring more security with less complexity. The experimentation is carried out and it reduces the encryption time, cipher text size, and improves throughput. When compared with previous techniques, there is a 45% improvement in throughput, 37% fast in encryption time, 54.67% cipher text size. The relevant experimental results and foregoing analysis show that this method is of good robustness, stability, and security.

Keywords: Cryptography; cloud network; deoxy ribo nucleic acid (DNA); encryption; security; cloud service provider (CSP)

1 Introduction

Data security is being the biggest roadblock to cloud computing since the data are complex and of varied mode. Many security methods have been implemented to overcome this issue. Cryptography is particularly advantageous since data may be viewed without authorization but not read. Before storing data in the cloud, data owners must encrypt it. This ensures the confidentiality of sensitive information stored on public cloud storage. There are several cryptographic approaches for ensuring secrecy. However, the majority of them rely on complicated mathematical computations and ratios. Recently big data and cloud computing have been used for secured data processing during the transmission of data to authorized persons. It can give more

efficiency and more storage space to clients securely. Many investors and enterprises are willing to spend their data on a cloud network. A recent report announced 76900 accounts are signing in to the Facebook in a minute, 400360 videos are uploaded to YouTube in a minute, 7.2 million Google searches were updated in a minute, 3.4 million Instagram posts are uploaded [1]. The data owner will upload the data into the cloud network, where all the information from various services is stored. The owner has to encrypt the data and finally, the encrypted data has been given to the cloud network [2]. The data uploaded for each minute is 1.8 ZB, this storage has been provided by cloud service [3]. HHS-DNA is a symmetric key algorithm [4] i.e., the same key is used for encryption as well as for decryption. The idea of this technique is based upon DNA cryptographic technique [5]. According to the proposed scheme, if a data owner wants to upload the data onto the cloud, he has to encrypt the data. Even if the cloud service providers are forced to reveal the data, the data which has been encrypted cannot be read by unauthorized users. The traditional approaches like DES [6], AES, Blowfish [7], RSA [8], DNA cryptography etc. are still in operation for the real-time systems which are consuming enormous computing resources. So, it is required to develop a new cryptographic technique [9] that generates a bridge between the existing and the new technology.

The Cloud computing network has a storage mechanism to store the data in a cloud platform. Cloud computing networks consist of the data owner, data user, and mainly cloud data storage which is shown in Fig. 1. Once the cloud receives the request from the data user, the encrypted data from the owner is sent to the cloud for further processing.

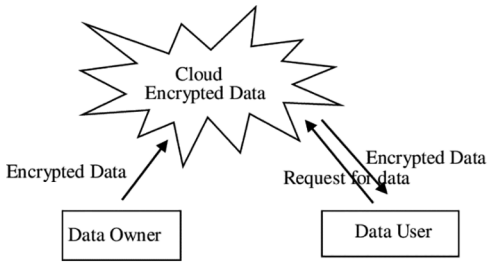

Figure 1: Cloud computing network

When the user wants to accept the data, the user must have the Re-enc key. With the help of the Re-enc key [10], the data user can decrypt the content and view the data. The recent framework demonstrates privacy information of cloud networks and amplified input is given to the multi-fold cloud system. But the major center of this paper is to examine a component of this demonstration which could be a novel Re-enc symmetric key encryption method that has been utilized to encrypt/decrypt the [11] information. Asymmetric encryption techniques employs long keys to ensure the security which is a major drawback of it. Recent encryption techniques utilize more resources such as CPU time, storage, and power. Existing techniques need two keys for encrypt and decrypt the data. Most of the existing algorithms mainly involves complex and arduous numerical expressions. To overcome the hurdles existing in the prior research, the proposed HHS-DNA is a good contender for key encryption. Since this research paper utilizes symmetric key algorithm, it is easier for encrypting and decrypting large amount of data in lesser time. Compared with asymmetric algorithms, symmetric key algorithms provide more security in cloud data over the network. The speed of DNA cryptography is unremarkably 100 times high while compared with a computer. Also DNA needs lesser storage and can be perform using low power. This paper suggests reasonable encryption, which achieves high security over many attacks with the usage of low resources for encryption with less time complexity.

Section 2 progress with the review of the existing methodologies, its measures, and precise problem statements present in each methodology has efficiently emphasized. system model is presented in Section 3. Section 4 explains the proposed cryptographic algorithm and Section 5 has the results and discussion

obtained by simulating the algorithm. Section 6 concludes the paper along with the future research direction of this paper.

2 Literature Survey

Encrypting the data before uploading it onto the cloud prevents unauthorized users from accessing the data. A lot of encryption algorithms have been developed to secure data stored on the cloud. Hiding the messages in an organism is more reversible and scalable compared with other DNA techniques. Tacking of message is negligible using [12] technique, it is more suitable for secret military information. DNA cryptography-based encryption algorithm has been used in [13]. It generates encoding tables randomly which enables the system to become more secure. Using this special encoding technique, the plaintext is converted into a 7-bit character and finally a 5-bit character. The same plaintext provides different cipher text each time; hence it is a more flexible and reliable technique. The multi-fold symmetric DNA encryption key [14] enables the encryption process to transpire on the client side. The result was compared with the previous technique to prove the function of the symmetric key.

Later investigations [15,16] demonstrate that some hidden function is established for DNA. The first research on DNA hiding was achieved in 1997, in which "June 6 night: Normandy" was successfully hidden. It was accomplished by encoding the secret message by substituting three successive characters for one value. For example, Ä was represented as CGE". Hence, a hidden secret may utilize keys as groundwork's of PCR amplification to crack DNA microdots. At later repetitive codes are used to hide the secret message. RNA nucleotides have four distinctive sorts of nucleotides: adenine (A), guanine (G), cytosine (C), and uracil (U). Therefore, there are $4 = 64$ distinctive combinations, with almost twenty essential amino acids.

The multi-fold symmetric-key cryptography technique is based upon DNA cryptography [17]. A consistent view about both data security concerns and privacy issues presented in the survey that are faced by clients in cloud storage environments has been carried out [18]. Further to study the security in a cloud network, Secure Hidden Layer (SHL) and Application Programming Interface (API) for data encryption explained in [19]. The SHL is consisting of two major modules (i) Key Management Server (KMS) and (ii) Share Holder Server (SHS) which is used for storing and sharing cryptographic keys. A server-side encryption algorithm, has been employed which is based on the asymmetric algorithm (RSA and CRT) for providing end-to-end security of multimedia data. A novel DNA-based data encryption scheme was introduced in [20] for the cloud computing environment. Here, a 1024-bit secret key is generated based on DNA computing, user's attributes and Media Access Control (MAC) address of the user, and decimal encoding rule. American Standard Code for Information Interchange (ASCII) value, DNA bases, and complementary rule are used to generate the secret key that enables the system to protect against many security attacks.

A data security based strategy through DNA sequence by the virtue of Steganography has been employed by Raniyah Wazirali and Zenon Chaczko (2016). A new data concealing method based on DNA characteristics has been devised in the research. The encrypted image was represented by a DNA helix. Following, that DNA matrix was transformed to a QR (Quick Response) form, which has a wide range of applications. In addition, the article offers suggestions for selecting the best QR sites in order to get the rightmost position. A novel genetic algorithm-based model has been built [21]. Suyel Namasudra et al. (2017) have established a DNA based approach to overcome the time complexity in terms of data security. A lengthy 512-bit DNA based novel sequence has successfully utilized to improve data security, and it is protected from collision attacks, internal attacks, and other types of attacks. The suggested scheme is assessed in terms of both theoretical and experimental findings, which demonstrate its superiority to current schemes. In the future, an authorization mechanism might be implemented to

improve security towards cybercriminals or bad users [22]. Chirakkarottu and Mathew [23] recommended a encryption method to encrypt the medical images. This paper mainly focused on providing security without minding the storage visual aspect. The images are initially re-arranged randomly and then DNA encrypt the images. Though the method provide security, the execution time is high. Balaraju and Rao [24] implemented Secure Hadoop Cluster using DNA Cryptography (SHCDNA) for securing data as a individual security that keeps the authenticated data in encrypted form. Although, SHCDNA reduces computations encryption of metadata is still questionable which is to be enhanced. Das et al. [25] explained the cryptography algorithm DNA in diverse facets like details about input and outputs and their nature. Yet a detailed implementation results are missing.

Based on the above researches reviewed, it is clear that the asymmetric key encryption algorithms take larger execution time. Moreover, the keys are larger and more complex for asymmetric encryption algorithm when compared to the symmetric encryption. Furthermore, the major drawbacks of existing techniques includes more power consumption, high storage and longer execution time. Hence, this paper presented an HHS-DNA to overcome the aforementioned hurdles. The subsequent Section 3 copes with the system model and structure.

3 System Model

The system model has five components such as Third-Party Examiner (TPE), Data owner (DO), Data User (DU), Trusted Privilege (TP), Cloud Service Provider (CSP). The system model for the proposed methodology is shown in Fig. 2.

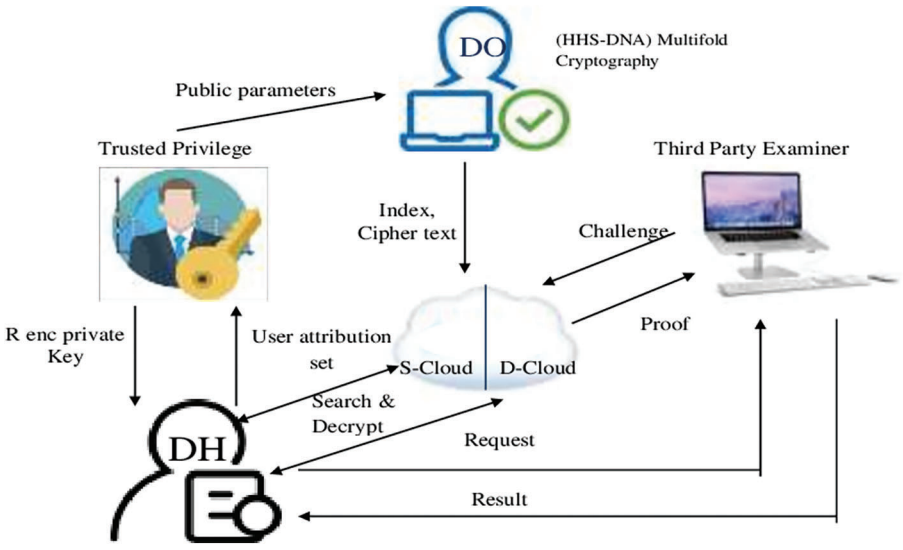

Figure 2: Cloud system model

The major application of the proposed methodology is to provide security for the data owner side. The data owner is responsible for making a key index and data accessing policy. Trusted privilege duty is updating, canceling and distributing the keys for the Data holder and Cloud Service Provider. The data holder is the entity that can outsource the data files. If the data holder has the key to satisfy the terms and policies, it will outsource the data files. CSP has two semi-honest entities such as Storage-CSP and decryption-CSP for storage and qualified decryption of cipher text respectively. When data outsourced requires, it updates its cipher text. The result is checked by a third party examiner either matched or not.

3.1 Trusted Privilege (TP) Set Up

The inputs of TP are 'i' and outputs are private R encrypt key (R_{pi}) and master R encrypt key is (R_{mi}), explained in this section (input k: output R_{pi}, R_{mi}).

3.1.1 R-Key

It takes a user's role $role_{u_i}$as input, and returns a role key $rkey_j$, $rkey_j = role_{u_i}.rolekey$. The specific user u_i is associated with the role key $rkey_j$. Upon receiving an authorization request with the user's role, it returns the role key $rkey_j$. Each role is associated with a unique role key and has the corresponding privilege.

3.1.2 Encryption

The user executes the algorithm to encrypt the data blocks with the convergent key. It takes a set of file blocks $\{b_i\}_{i\in[1,n]}$ as input, and returns the cipher text Cm = {Pm + KA × PB}. The user runs the hash over level blocks $\{b_i\}_{i\in[1,n]}$ to obtain the convergent key $\{k_i\}_{i\in[1,n]}$, $k_i = h(b_i)$. Then, the user symmetrically encrypts the blocks with the convergent public key, PB = KB × M to obtain the cipher text Cm = {Pm + KA × PB}.

The subsequent section deals with the comprehensive emphasize of the proposed system and functions.

4 Proposed Methodology

The proposed methodology is based on three levels as user level, security level, cloud storage level which are shown in Fig. 3. At the user level, encryption key generation and encryption processes have been done. At the security level, encrypted data has been uploaded to the cloud and the encryption process has been done based on key level management. At the same time, the receiver receives the encrypted data along with key management. The receiver decrypts the data based on their requirement.

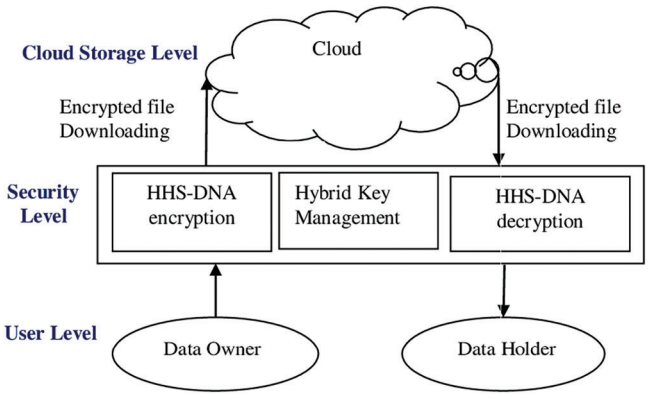

Figure 3: Different cloud levels

4.1 User Level

At the data user level, the data is transferred with the correct key for the authentication process. The key selection is processed based on node selection and Re-enc key processing. For node selection, we define three parameters as hold, children, and grade for TP. In between TP and DH, many nodes are present. So, we have to select the best node for the transmission of keys over the network. In holder stores the parameters of the nodes which is defined as V= $\{v_1|v_2\}$ (left to right index). Children nodes contain child node f, corresponding nodes such as left, right, and middle nodes. The grade represents the number of elements present in the nodes. If grade = 2 means, it will save 2 elements, and order 3 saves 3 elements.

Algorithm 1: Check Z_k

Input: Correspondent node Z_k

Output: inhold (Z_k), grade (Z_k)

1: obtain grade (Z_k)

2. **if** (grade (Z_k)=2) **then**

3. get inhold (Z_k)=$\{z_t||z_{t+1}\}$

4. return inhold (Z_k) and grade (Z_k)

5. **end if**

2. **if** (grade (Z_k)=3) **then**

3. get inhold (Z_k)=$\{z_t||z_{t+1}||z_{t+2}\}$

4. return inhold (Z_k) and grade (Z_k)

5. **end if**

Step 1: Trusted authority gets in hold and grade of current nodes between the networks.

Step 2: if (grade $(Z_k) = 2$) then trusted privilege checks the node value with x_i, x_{i+1} of node X_i.

Step 3: if (grade $(Z_k) = 3$) then trusted privilege checks the node value with x_i, x_{i+1}, x_{i+2} of node X_i.

Step 4: These steps are continued until the child node Z_k is obtained. From algorithm. 1 the Trusted privilege gets grade = 2, then, trusted privilege has two nodes such as $\{z_t||z_{t+1}\}$ and returns inhold (Z_k), grade (Z_k)

Step 5: From algorithm. 1, whether Trusted Privilege (TP) gets grade = 3, then, TP had three nodes such as $\{z_t||z_{t+1}||z_{t+2}\}$ and returns inhold (Z_k), grade (Z_k).

Re-enc key: Keys are classified as private Re-enc key and master Re-enc keys that are generated for public parameters. R_{pi} and R_{mi} are the keys that are used for accessing the Data Holder (DH) from Trusted Privilege (TP). Model description algorithm for key generation is described in algorithm 2. User means token value or $h(h(f))$ function. In each process, a specific user token is uploaded and corresponding key is generated by $rekey_j = aspect_{u_i}.aspectkey$ formula. Each user generates an authorized request. After the request is received, TP checks the node z_k for Re enc key. Each grade has unique keys for different TP. User aspect $aspect_{u_i}$is given to Re enc key for key generation. Initially, $aspect_{u_i}$ is used instead of Upload or download. *aspectkey* is a constant key generated by each node z_k. Constant key and aspect value of user is multiplied and obtained a $rekey_j$.

Algorithm 2: Re-enc key

Input: Users aspect $aspect_{u_i}$

Output: $rekey_j = aspect_{u_i}.aspectkey$

1. user $u_i = token = h(h(f))$

2. Upload specific user token aspect =$aspect_{u_i}$

3. authorized request from user's aspect

4. After receive the request, TP checks the node

z_k for Re-enc key.

5. Each grade has unique keys for different TP.

4.2 Security Level

The implementation of the encryption process was developed by two encoding tables, the 14-bit encryption key and number N. This encryption process is more dynamic and depends on the n value. Based on n values, the value in the tables is shifted. Even though the intermediate persons have the access to encoding tables, they can't access the data. First table will move the data continuously until it stops. First table will keep on moving, so it is unbreakable.

After shifting the table values, the data owner changes the plain text into a binary sequence. If the intruder has access, they can decrypt the sequence. It is a secured technique compared with the previous technique. After the conversion, the binary sequence is converted into encryption data using Tab. 1. Finally, the cipher text is obtained using an encryption table, and the original cipher data is obtained using the table. On the receiver side, if the user has the key to access the data, the user can perform the decryption process based on a random number N.

Table 1: Final cryptography code

000000-F	000010-O	000100-c	000110-+
001000-g	001010-t	001100-W	001110-N
010000-Z	010010-X	010100-y	010110-v
011000-I	011010-3	011100-6	011110-R
100000-r	100010-V	100100-G	100110-8
101000-l	101010-7	101100-0	101110-B
110000-C	110010-J	110100-u	110110-4
111000-j	111010-l	111100-z	111110-x
000001-S	000011-a	000101-P	000111-U
001001-b	001011-i	001101-k	001111-2
010001-L	010011-M	010101-5	010111-/
011001-w	011011-s	011101-T	011111-H
100001-Q	100011-Y	100101-e	100111-p
101001-o	101011-h	101101-q	101111-E
110001-A	110011-D	110101-K	110111-f
111001-m	111011-n	111101-9	111111-d

The proposed methodology uses 14-bit data for encryption before using 14 bits it generates a 7-bit encryption key for data processing. It is derived from the first bit of the encryption key. Further, it improves the proposed technique; hence multi-fold cryptography methodology is accomplished.

4.2.1 Multi-Fold Encryption

Proposed encryption is a client-based encryption process, hence there is no encryption process in the cloud. The encryption data will be stored in the cloud while the Input is 14 bits of data. The plain text is converted into a binary sequence. When the plaintext is entered into a table, the text values are shuffled. The shuffled 14 bits (input) are generated. If the initial bit is 1, then the odd values are extracted,

otherwise even bits are extracted. If the initial bit is 0, the even values of the bits are extracted. After the extraction process, the 7-bit key is an original key, which is processed for further operation. Then XOR operation is performed between 7-bit actual key with plaintext. The XOR-operated bits are divided into several blocks and each block has 7 bits. The position of each bit is calculated. The bits are divided into two halves, left half and right half. These two halves are interchanged and the resulting bit sequence is converted into cipher text characters which are represented in Tab. 1.

Algorithm 3: Multifold Encryption Algorithm

Start

Input: The plain text

Random Number N

1. If (N≠ 0)

Shuffle the plaintext values.

N value decrement

2. End
3. Plaintext is converted into 7 bits key
4. Input the 14-bit binary value
5. If (initial bit = 0)

Extract the even binary value

6. Else

Extract odd binary value

7. XOR Operation for 7 bits key with extracted key (7bit)
8. Find position of each block in i-th bit
9. Substitute each block with the block located at (127-i) the position
10. Partition the binary bits into two halves
11. Left half is first part, right half is second part
12. Interchange the two halves (left and right)
13. Divide each binary bit into blocks, each block contains 6 bits
14. If (Resulting is not divided equal)

Add padding at last

15. After the partition, cipher text is applied according to Tab. 1.

4.2.2 Multi-Fold Decryption

The decryption process is performed on the user side. If the user has access only to decrypt the data, they can access the data alone. For authentication, the Re-enc key is transmitted to the data user. If the user has this key on the client-side, the decryption operation will be performed. The data from the cloud is in the encrypted format, it has to be decrypted. Hence the user must have the Re-enc key, N random number, and access rights. The values in Tab. 1 are shuffled according to cipher text, finally, a binary bit sequence is obtained. The obtained binary sequence is further processed to obtain the original text with the help of Tab. 1. Binary sequences from First table is divided into two halves, right half and left half. The two halves are

interchanged. Finally, the binary bits are divided into seven blocks. Each block is replaced with 127bits. The resultant binary sequence is XOR with an actual 7-bit key. The binary sequence is processed and the plaintext is retrieved from a binary sequence. Multi-fold Decryption algorithm is shown below,

Algorithm 4: Multifold Decryption Algorithm

Start

Input: Cipher text

Random Number N

Actual key

1. Partition the binary bits into two halves
2. Interchange the two halves (left and right)
3. Divide each binary bit into seven blocks, each block contains 6 bits
4. Substitute each block with the block located at (127-i) th position in Tab. 1
5. Find position of each block in i-th bit
6. XOR Operation for 7-bit actual key with extracted key (7bit)
7. If (initial bit-=0)

Extract the even binary value

8. Else

Extract odd binary value

9. Generate plaintext

4.3 Cloud Storage Level

The cloud storage level has a storage mechanism to store the data in the cloud platform. A Cloud interface is used to upload the data to a cloud storage system. When a user wants to accept the data, the user must have the Re-enc key. With the help of the Re-enc key, data requests from the cloud and encryption will be processed by the security framework. The verification and analysis has been carried out in the following section.

5 Verification and Analysis

For searching and verification Index I_n, user token (ω_{ui}) is used, while output access rights for decrypt the data. For ($1 < i < n$), many users have the right to access, hence i is defined as $1 < i < n$, where n is the last user. If ($\omega_{ui} = \omega'_{ui}$), the user has access rights, otherwise the user has the right to decline the access request. Fig. 4 shows the signature-based verification scheme and search methodology.

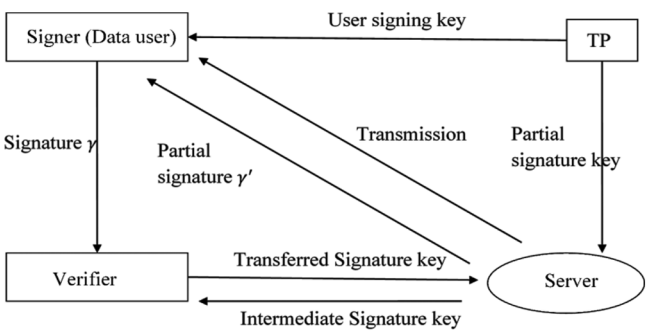

Figure 4: Signature-based verification theme and search methodology

TP: The attribute authority (TP) takes input as the security algorithm β, and then it returns the master secret key MSK and the public parameter PK.

Extract: The AA enters the master secret key MSK, the public parameter PK, and a signing predicate Γ as input, this algorithm outputs the partial signing key p_{sk} and the user signing key u_{sk}.

Server: The server takes the partial signing key psk, the public parameter PK as an attribute set S and the message m as input, and returns the partial signature γ' to the signer.

Signer (Data Signer): This algorithm takes input as the user signing key usk, the partial signature γ', the attribute set S, the message m and the public parameter PK, and then the signer computes and outputs the signature €.

Transform: The verifier randomly selects the Transferred signature key t_k, and inputs the signature σ and the public parameter PK. This algorithm outputs the transformed signature $\hat{€}$.

Server: This algorithm takes input as the transformed signature € and the public parameter PK and returns the intermediate signature $\hat{€}$.

Verifier: The verifier takes input as the intermediate signature $\hat{€}$, the transformation key t_k and the public parameter PK, then this algorithm outputs true or false.

5.1 Configuration

TP extracts the user signing key and the public parameter (PP) for generating partial signing key and user signing key. A clipping algorithm can be performed to generate partial signing and user signing key. The server takes input as a partial signing key, attribution set, public parameter as input, and produced output as a partial signature. Data user takes partial signature, attribution set, and public parameter and message data as input and produces original signature for verification. Verifier randomly selects the transformation key t_k, attribution set and public parameter to generate the transferred signature key. The server takes a transferred signature key, attribution set and public parameter for generating intermediate signature key $\hat{€}$. Verifier takes the transferred signature key and an intermediate signature key as input and this algorithm returns true or false.

5.2 Clipping Algorithm

This algorithm is used to generate a partial signing key and user signing key. The algorithm is explained with LSSS structure (D,ω)

1) Define an LSSS matrix M with size $n_r \times n_c$. For each row k ∈ [1, n_r] in M, select a labeling function (k) $\omega \in V_P$, which corresponding to an attribute in UA.
2) Vector $\vec{v} = \{\rho_1, v_1, \ldots, v_n\}$. Compute, $\varphi_{\omega(k)} = \overrightarrow{D_k}\overrightarrow{v_t}$. Where $\vec{D}$ denotes the k-th row of the matrix D.
3) TP computes $f_k = m^{\varphi_{\omega(k)}}$

Similarly, AA calculates $f_e = m^{\rho_1}(r_0.r_e)^{x_e}$ and $f'_e = m^{x_e}$. Hence partial signature key under the server is expressed as PK = $\{f_k, f'_k, f''_k\}_{k\in[1,n_r]}$. For user key UK = $\{f_e, f'_e\}$.

5.2.1 Signing

Partial signature produces with the help of message d, and attribution set S which satisfies (D, ω). Define K=$\{k\}_{p(k)\in S}$, select set of constant $\sum_{k\in K} \mu_k \varphi_{\omega(k)} = 1$.

Partial signature after an attribution set S is expressed as $\gamma' = \{\gamma'_0, \gamma'_1, \gamma'_2, d\}$.

5.2.2 User Signer

This algorithm is performed after the reception of partial sign from the server, γ'. The signer calculates, $\gamma_0 = \gamma'_0.f_e.(r_e,\ r_o)^{x'_e}.H(d)$, $\gamma_1 = \gamma'_1.g^{x'_e}$ and $\gamma_2 = \gamma'_2.g$.

The signer output the signature such as $\gamma = \{\gamma_0, \gamma_1, \gamma_2, d\}$ of message d, with attribution set S.

The signature γ is converted into the transferred signature.

Randomly select transferred key $t \epsilon V_p$

Compute $\widehat{\gamma_0} = \gamma_0^t$ and $\widehat{\gamma_1} = \gamma_1^t$

Transferred signature $\hat{\gamma} = \{\widehat{\gamma_0},\ \widehat{\gamma_1},\ d\}$

5.2.3 Intermediate Key

This algorithm is designed to delegate the server for heavy computations of verification. The server performs the following computations and obtains the intermediate key $\breve{\gamma}_1$

$$\widetilde{\gamma_1} = \frac{e(\widetilde{\gamma_0},\ m)}{e(\widetilde{\gamma_1},\ r_0 \prod_{i \varepsilon S} r_i)} \tag{1}$$

5.2.4 Verification

After receiving an intermediate key from the server, the signature verification is processed.

First it checks,

$$\widetilde{\gamma_2} = e(\gamma_2^t,\ H(d)).V^t) \tag{2}$$

Check whether the equation $\widetilde{\gamma_2} = \widetilde{\gamma_1}$

If $\widehat{\gamma_2} = \widehat{\gamma_1}$ then the signature is valid and true otherwise the signature is invalid or false. In this way, the authentication process is accomplished with the SKVS algorithm.

5.3 Analysis of SKVS

For SKVS, two parameters are needed which are the public parameter and master key. The input of the first setup is security parameter β and multiplicative group such as M, M_T with prime number is taken as output.

5.3.1 Set Up

Bilinear pairing: $M \times M = M_T$. The multiplicative group (M) of integers modulo n is the group under multiplication of the invertible elements. Where ρ_1, ρ_2 are the parameter which is used for security purpose.

Algorithm 5: Verification & Search

Input: Index I_n, user token (ω_{ui})

Output = Access Accept (AA)

1: user u_i = token = h(h(f))

$$\widetilde{\gamma_1} = \frac{e(\widetilde{\gamma_0},\ m)}{e(\widetilde{\gamma_1},\ r_0 \prod_{i \varepsilon S} r_i}$$

$$\widetilde{\gamma_2} = e(\gamma_2^t,\ H(d)).V^t)$$

2. *if* $(\widehat{\gamma_2} = \widehat{\gamma_1})$

(Continued)

3. user has access rights
4. User hold the state
5. else
6. decline the access request
7. return

Steps:

Step.1. Randomly choose $r_0, uA_0, uA_1, \ldots, uA_n$ from G and collision resistance has function is defined as $H(d) = uA_0 \prod_{j=1}^{n} uA_j^{d[j]}$, $d[j]$ is j th bit of data (d).

Step. 2. Set the attributes set as A = {1, 2… t}, for each attribute i ∈ A, r ∈ M.

Step. 3. Set additional attributes (Ω), attribution set denoted as r ∈ Ω.

Step. 4. Randomly select some parameter (ρ_1, ρ_2), M∈m, and calculate $V = e(m,\ m)^{\rho_1 + \rho_2}$

Public parameter = $\{m, e, M, M_T, H, V, r_0, u_0, u_1, \ldots, u_n\}$, where master key is expressed as MK=$\{\rho_1, \rho_2\}$.

5.3.2 File Corruption Check

Initialize – Start Holder, and Call TPA

Holder – file ids and Hash values

TPA – Log in Cloud

File id 1: number of files File

Check files id and reference id

If (file id)

H = Check file has value

K=user given hash value of file

If (H==K)

Send (the file is ok)

Else

If (H≠K)

Send (File is corrupted)

End

TPA-Next File-User

Section 4 comprises the result and discussion of the proposed methodology.

6 Result and Discussion

In the result section, the proposed methodology is compared with the previous technique based on experimental and theoretical aspects. Computational cost and storage overhead are the parameters that are used to calculate the proposed methodology. We have compared and evaluated symmetric algorithms for different encryption and encoding techniques and came up with the conclusion that the recent encryption techniques utilize more resources such as CPU time, storage, and power. Also, HHS-DNA is a good

contender for key encryption. The proposed encryption achieves high security over many attacks and usage of low resources for encryption.

6.1 Encryption Time

Encryption time is defined as the time required to convert plaintext into cipher text. The efficiency of the encryption technique is evaluated based on encryption time. If the plaintext size is large, then encryption time will also increase. At the plaintext size of 5Kb, encryption time is increased by 21%. At the plaintext size of 30 kb, the encryption time will also increase to 56.78%. The encryption time calculated using different algorithms is shown in Fig. 5.

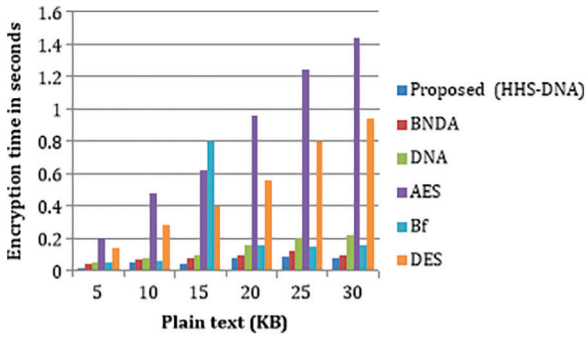

Figure 5: Comparison of encryption time with the *vs.* plaintext (Kb)

Whenever the number of plaintexts increases, the cipher text will also increase linearly. When compared to the proposed method with BDNA, the cipher text is decreased to 9% at 5Kb of plaintext while compared with AES cipher text is reduced to 15.5%. When compared to the proposed method with DES, the cipher text is decreased to 7.45% at 5Kb of plaintext while compared with DNA cipher text is reduced to 5.5%. Fig. 6 shows the performance of plaintext *vs.* cipher text.

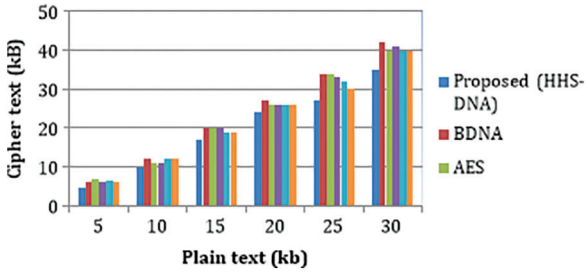

Figure 6: Cipher text (Kb) *vs.* plain text (Kb)

6.2 Throughput

Based upon the recorded encryption time, the throughput for the proposed and the existing techniques (BDNA, AES, Bf, DES) have been calculated using the formula,

$$Throughput = \frac{Plaintext\ size}{Encryption\ Time} \tag{3}$$

The throughput of HHS-DNA is higher than all the other techniques which are shown in Fig. 7. Hence, it can be concluded that the proposed method performs better than other symmetric key algorithms. Higher throughput gives better performance.

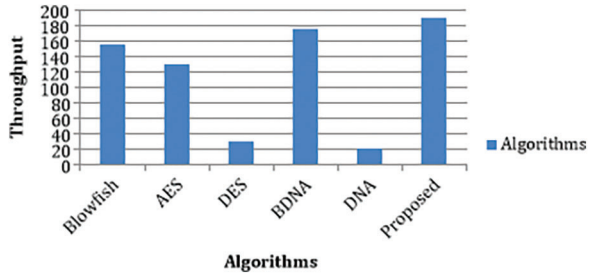

Figure 7: Throughput

From the above results, it is obvious that the proposed HHS-DNA encryption algorithm outperforms other algorithms. The proposed algorithm has high throughput which indicates that the HHS-DNA has high performance efficiency than the other methods. Likewise, higher throughput symbolizes the reduction of power consumption, since throughput is inversely proportional to power consumption. Besides, the lesser encryption time determines that HHS-DNA's speed in encrypting the given data.

7 Conclusion

In the proposed approach, a novel and unique Hybrid Helix Scuttle - Deoxyribo nucleic Acids (HHS-DNA) encryption algorithm have been proposed. HHS-DNA encryption reduces the encryption time, cipher text size and improves the throughput. The reduction of time complexity and improved data security is comprehended in the proposed strategy. The simulation contrasts with the proposed algorithm, DNA, and other previous techniques. When compared with previous techniques, there is a 45% improvement in throughput, 37% in encryption time, 54.67% cipher text size. Simulation outputs showed that the proposed HHS-DNA technique is a proficient scheme for encrypting the data in the cloud and improving data security. The relevant experimental results and foregoing analysis show that this method is of good robustness, stability, and security. The performance measures of the proposed scheme are correlated with the prior art techniques and acquired a performance improvement of 0.9% to 0.99%. Future research will be focused on reducing CPU processing time and CPU workload.

Acknowledgement: The authors would like to thank Anna University and also we like to thank Anonymous reviewers for their so-called insights.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Xu, E. C. Chang and J. Zhou, "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage," in *Proc. ACM SIGSAC*, IEEE, Hangzhou, China, pp. 195–206, 2013.

[2] J. Xiong, Y. Zhang, S. Tang, X. Liu and Z. Yao, "Secure encrypted data with authorized deduplication in cloud," *IEEE Access*, vol. 7, no. 2, pp. 75090–75104, 2019.

[3] Z. Yan, L. Zhang, W. Ding and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," *IEEE Transactions on Big Data*, vol. 5, no. 3, pp. 393–407, 2019.

[4] S. Wang, Y. Wang and Y. Zhang, "Block chain-based fair payment protocol for deduplication cloud storage system," *IEEE Access*, vol. 7, pp. 127662–127668, 2019.

[5] Y. Zhao, Y. Li, Q. Mu, B. Yang and Y. Yu, "Block chain based fair payment with reputation for reliable cyber physical systems," *IEEE Access*, vol. 6, no. 8, pp. 12295–12303, 2018.

[6] H. Li, M. Dong, X. Liao and H. Jin, "Deduplication-based energy efficient storage system in cloud environment," *the Computer Journal*, vol. 58, no. 6, pp. 1373–1383, 2015.

[7] J. Li, J. Li, D. Xie and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2386–2396, 2015.

[8] B. Mao, H. Jiang, S. Wu and L. Tian, "Leveraging data deduplication to improve the performance of primary storage systems in the cloud," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1775–1788, 2016.

[9] J. Hur, D. Koo, Y. Shin and K. Kang, "Secure data deduplication with dynamic ownership management in cloud storage," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 11, pp. 3113–3125, 2016.

[10] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo *et al.*, "Secure and efficient cloud data deduplication with randomized tag," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 3, pp. 532–543, 2016.

[11] Y. Wang, Q. Han, G. Cui and J. Sun "Hiding messages based on DNA sequence and recombinant DNA technique," *IEEE Transactions on Nanotechnology*, vol. 18, pp. 299–307, 2016.

[12] M. Sohal and S. Sharma, "Enhancement of cloud security using DNA inspired multifold cryptographic technique," *International Journal of Security and Its Applications*, vol. 11, pp. 15–26, 2017.

[13] S. Marwan, A. Shawish and K. Nagaty, "DNA-Based cryptographic methods for data hiding in DNA media," *Biosystems*, vol. 150, pp. 110–118, 2019.

[14] H. Houa, J. Yu and R. Hao, "Cloud storage auditing with deduplication supporting different security levels according to data popularity," *Journal of Network and Computer Applications*, vol. 134, pp. 26–39, 2019.

[15] Y. Miaoa, J. Maa, X. Liub, Q. Jianga, J. Zhanga *et al.*, "Verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings," *Pervasive and Mobile Computing*, vol. 40, pp. 205–219, 2017.

[16] C. M. Yu, S. P. Gochhayat, M. Conti and C. S. Lu, "Privacy aware data deduplication for side channel in cloud storage," *IEEE Transactions on Cloud Computing*, vol. 14, no. 8, pp. 1–13, 2015.

[17] M. Sohal and S. Sharma, "BDNA-A DNA inspired symmetric key cryptographic technique to secure cloud computing," *Journal of King Saud University-Computer and Information Sciences*, vol. 7, no. 1, pp. 11–32, 2018.

[18] N. Kaaniche and M. Laurent, "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Comupter Communications*, vol. 111, no. 1, pp. 120–141, 2017.

[19] K. Sinhaa, A. Priyaa and P. Paul, "K-RSA: Secure data storage technique for multimedia in cloud data server," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 3, pp. 3297–3314, 2020.

[20] S. Namasudra, D. Devi, S. Kadry, R. Sundarasekar and A. Shanthini, "Towards DNA based data security in the cloud computing environment," *Computer Communications*, vol. 151, no. 1, pp. 539–547, 2020.

[21] R. Wazirali, Z. Chaczko and L. Carrion, "Bio-informatics with genetic steganography technique," in *Computational Intelligence and Efficiency in Engineering Systems*, 1st ed., vol. 595, Springer, Cham, Studies in Computational Intelligence, pp. 333–345, 2015.

[22] S. Namasudra, P. Roy, P. Vijayakumar, S. Audithan and B. Balusamy, "Time efficient secure DNA based access control model for cloud computing environment," *Future Generation Computer Systems*, vol. 73, pp. 90–105, 2017.

[23] S. Chirakkarottu and S. Mathew, "A novel encryption method for medical images using 2D zaslavski map and DNA cryptography," *SN Applied Sciences*, vol. 2, no. 1, pp. 1–10, 2020.

[24] J. Balaraju and P. P. Rao, "Investigation and finding a DNA cryptography layer for securing data in hadoop cluster," *International Journal of Advance Soft Computing Applications*, vol. 12, no. 3, pp. 55–64, 2020.

[25] A. Das, S. K. Sarma and S. Deka, "Data security with DNA cryptography," in *Transactions on Engineering Technologies*, 1st ed., vol. 3, Springer, Singapore, Transactions on Engineering Technologies, pp. 159–173, 2021.