Cluster Comput (2018) 21:1411–1437
https://doi.org/10.1007/s10586-017-1368-4

Enhanced DNA and ElGamal cryptosystem for secure data storage and retrieval in cloud

M. Thangavel[1] · **P. Varalakshmi**[2]

Received: 17 February 2017 / Revised: 17 June 2017 / Accepted: 10 November 2017 / Published online: 22 November 2017

Abstract Cloud computing enables the access of the resources such as network hardware's, storage, applications and services that are configurable based on the demand in a network especially specific to the operations on the data. The need for data security in the cloud is progressively higher as the abundant sensitive data in the cloud are transferred among various stakeholders for data operations leads to loss of data confidentiality. To maintain data confidentiality in the cloud, the data need to be encrypted with cryptographic algorithms. Existing cryptographic algorithms face the challenges of key management, dynamic encryption, and computational complexity. In this paper, a novel variant of DNA cryptosystem is proposed to secure the original data within the DNA nucleotides providing greater storage space, reduced overhead and dynamic operations. The significance of DNA is incorporated in the proposed Novel DNA cryptosystem, which encrypts the data transferred between the Data Owner and the Data User in the cloud. Enhanced ElGamal cryptosystem is the proposed asymmetric cryptosystem used to address key management issues in the cloud, by securely transferring the key file between the Data Owner and the Data User. Enhanced ElGamal cryptosystem provides better user authentication and performance with respect to the security accomplishment against attacks. At the same time, Novel DNA cryptosystem achieves better performance, reduced the complexity of implementing the properties of DNA and embarks upon a standardized algorithmic approach among the existing DNA cryptographic methodologies. The performance analysis, mathematical proof as well as security analysis forms the security metrics and it meets out the proposed objectives. Thus, on utilizing the proposed Novel DNA and Enhanced ElGamal cryptosystems (i.e) both symmetric and asymmetric cryptosystems, enhances the security and performance of data storage and retrieval in the cloud.

Keywords Confidentiality · Secure cloud storage · Secure data retrieval · DNA cryptosystem · ElGamal cryptosystem · Cryptanalysis

1 Introduction

Cloud has become a prominent technology in almost every business organization that handles a large amount of data for its business operations. It raises the eventual objective of the organization to safeguard the data from security breaches for its successful growth. The goal to accomplish data confidentiality in cloud prevails with lots of challenges to be resolved [1]. Cloud computing is the base technology for any real-time applications. Cloud services can be portrayed as "X as a Service (XaaS)", where X can be everything relevant to computing like hardware, software, platform, etc... In cloud computing, three entities play the major roles, namely Cloud Server (CS), Data Owner (DO) and Data User (DU). CS provides storage services for application owners and distributed users. DO can create, store and update any kind of data on the cloud server. DU can access the data stored in the cloud through proper authentication. It is hard to choose cloud-based storage or non-cloud based storage at the time of storing confidential data. By the continuous development of cloud technology, many security problems arise in deploy-

✉ M. Thangavel
thangavelmuruganme@gmail.com

P. Varalakshmi
varanip@gmail.com

[1] Department of Information Technology, Thiagarajar College of Engineering, Madurai, India

[2] Department of Computer Technology, Madras Institute of Technology, Anna University, Chennai, India

ment and usage. Data Security is the primary requirement for any kind of data in the cloud, because of internet based service. DO will face a major problem if their sensitive or confidential data has been accessed by any unauthorized users.

Data confidentiality [2] is an essential component for data owners to store and retrieve the cloud data in a secure manner. In order to maintain data confidentiality, data need to be encrypted in the cloud using cryptographic techniques, while stored and transferred. The management of cryptographic keys [3] is the challenging problem of the implementation of cryptographic techniques in the cloud. The significance of cryptographic techniques is to scramble the content of the data and make the data in unreadable or meaningless forms, during storage and data transmission. Cryptography can be categorized into two types, namely, (i) Symmetric cryptosystem, (ii) Asymmetric cryptosystem. The variation between two cryptosystems lies in the usage of encryption and decryption keys. In a symmetric cryptosystem, DO need to share a secret key for DU, which is utilized for both encryption and decryption. In Asymmetric cryptosystems, DO and DU needs to generate the public and private keys individually. Then, using DU's public key, data will be encrypted by DO and using DU's private key, data will be decrypted by the DU itself. The parameters and key length are high in an asymmetric cryptosystem. If the key length is high, it is difficult to break the cryptosystem. Throughput is high in symmetric cryptosystems. If the throughput is high, power consumption will be low. In real time scenario [4], the data communication will take place between the User's web browser and the cloud service provider such as Google, Amazon, etc. The data is encrypted using symmetric key cryptosystem and the keys to authenticate DO and DU is managed through public key cryptosystem. This concept is adopted in the proposed cloud framework with novel algorithms to serve the purpose. A novel cryptosystems need to be proposed by maximizing throughput and data security.

To solve the complex computational problems such as Hamilton path problem—an NP-complete problem, Adleman [5] used DNA molecules and solved it, which leads to a new field of computing known as DNA Computing. In order to increase throughput and security of the cryptosystems [6], a unique field of cryptosystem using the concept of Deoxyribonucleic acid (DNA) known as DNA cryptosystem came into existence.

DNA has a massive power to store nearly 700 terabytes of data in its one single gram. DNA molecules hold the genetic information for the growth and development of the living organisms. It can be visualized as a double helix structure, which is two strands of polynucleotides. Adenine (A), Guanine (G), Cytosine (C) and Thymine (T) are four nucleotides of the DNA molecule. Adenine is complementary paired with Thymine, Guanine is complementary paired with Cytosine and vice versa. Binary values are assigned to DNA nucleotides, Adenine (A)—00, Guanine (G)—01, Cytosine (C)—10, and Thymine (T)—11. These binary values are used in the transformation of DNA sequences. Using a defined character set, amino acids in protein synthesis are used to convert intermediate ciphertext into collated ciphertext.

Table 1 The properties for an efficient DNA cryptosystem

Complete character set encoding	To provide encoding of every unique sequence for the complete ASCII characters sets entries.
Dynamic Encoding table generation	To generate a distinct encoding table for every data transfer between the Data Owner and the Data User.
Unique sequence for character encoding	The sequences are encoded with different characters in every data transfer in every session.
Robustness of encoding	Randomness is ensured by varying the intron sequence, collating values and encoding table values for each access.
Biological process simulation	To involve the biological properties of DNA in encryption, decryption and encoding table generation.
Dynamic encryption process	To produce different ciphertext for a plaintext through every unique generation of the encoding table.

Existing research works on DNA cryptosystems concerns with the requirements of DNA cryptosystems, storage space, computational speed and security against intruders in a digital world. Through many rigorous studies and research, DNA cryptography has been evolved where the confidential data are hidden inside the DNA molecules to enhance data security. DNA sequences like NCBI, DDBJ and EBI databases are available. Based on the data user, DNA sequences can be selected at random. These DNA sequences are used in DNA cryptosystems to provide uncertainty for the intruders. The biological process of DNA is hard to break. It reduces the level of complexity in computation. Thus, these factors depict the need of DNA Cryptography in cloud computing, where fast computation and increasing complexity in cryptanalysis are achieved. In recent years, researchers proposed variants in DNA cryptosystems to solve the confidentiality issues, but each cryptosystem differs from one another based on the usage of the biological and arithmetic operations. Still, there has not been a standardized method for evaluating existing variants of DNA cryptosystems to prove the results of measured metrics. Noorul Hussain et al. [7] identified the six properties for the security measures of an efficient DNA cryptosystem, as shown in Table 1. Thus, the Novel DNA cryptosystem forms the symmetric key cryptography that can be used for data encryption [8]. The significance of DNA molecules has been proved by maximizing computational speed, minimizing computational power and efficient stor-

age. The key file of DNA cryptosystem needs to be shared secretly between DO and DU. While public key cryptography is used to encrypt as well as transfer the keys securely through an insecure channel between DO and DU [9].

In 1985, ElGamal [10] proposed a public key cryptosystem, which had a wide range of attention these days. The key idea behind the cryptosystem is the use of the discrete logarithm problem. It is hard to find the solution for a discrete logarithm problem. The ElGamal algorithm is described as follows:

User A - Receiver:
Key Generation:
Choose q, a large prime number, and α, primitive root of q
Choose X, a random integer such that 1 < X < q-1
Compute Y as $\alpha^X \ mod \ q$
Private Key: X, Public Key: {q, α, Y}

User B - Sender:
Encryption:
Represent message as integer m, such that 0 ≤ m ≤ q-1
Choose a random integer k, such that 1 ≤ k ≤ q-1
Compute one-time key, $K = Y^k \ mod \ q$
Encrypt message 'm' as a pair of integers (C_1,C_2) Where
$$C_1 = \alpha^k \ mod \ q, \ C_2 = K.m \ mod \ q$$

User A - Receiver:
Decryption:
Recover key by computing $K = {C_1}^X \ mod \ q$
The message, m can be retrieved as
$$m = C_2.K^{-1} \ mod \ q$$

In ElGamal cryptosystem, if the private key X is derived then the entire system can be broken easily as the message can be retrieved as $m = C_2 \cdot C_1^{-X} \ mod \ q$. The security of the ElGamal cryptosystem lies in the difficulty of the discrete logarithm problem. But, many algorithms [11,12] like naive, pollard's rho method and baby-step/giant-step methods [13] has been proposed to solve the discrete logarithm problem.

Enhanced ElGamal Cryptosystem has been proposed and used in the cloud framework to address the key management issues. Enhanced ElGamal cryptosystem provides data integrity and security by increasing the complexity in deriving the private key of the DU by the intruder.

The rest of the paper is organized into various sections as follows: Sect. 2 describes the related work in variants of ElGamal cryptosystem and DNA cryptosystem, Sect. 3 describes the proposed framework, Sect. 4 describes an Implementation and results of the Novel DNA cryptosystem, as well as the Enhanced ElGamal cryptosystem and Sect. 5, describes the security analysis of the proposed cryptosystems. The conclusion is summarized in Sect. 6.

2 Related work

In cloud computing, ensuring data confidentiality and managing cryptographic keys avoids the events of data breaches or data loss. Since there are many malicious users in the cloud, the data security may be at risk. So, Cloud Server needs to ensure data confidentiality to avoid unauthorized data access in the cloud storage.

The incidents of data breaches or data loss have become quite common these days. In 2014, the leakage of confidential information on Sony such as email exchanges among the Sony employees. It revealed the Personally Identifiable Information (PII) which helps to identify, contact or to track a particular person. It caused an expense of 15 million dollars to address the damages. Similarly, Codespaces, an online hosting provider was hacked and the impact was most of their customer data were compromised. Based on the Cost of Data Breach study of 2015 by Ponemon Institute and IBM, the cost incurred for the loss of sensitive and confidential information has increased from 201 to 217 dollars. These data breaches have occurred due to the vulnerabilities of data security that caused malicious attacks and process failures. At 2016, the data breach of the health care records has gone high. The loss of health care records is due to lack of security measures in cloud storage environment. As the volume of data is increasing the data confidentiality methodologies need to be enhanced to suit the needs.

The various research works on modified ElGamal and DNA cryptosystems are analyzed to enhance the cryptosystems for real-time implementation in the cloud framework. Key management issues can be addressed through Public Key Cryptosystems. The security features like confidentiality and authentication are achieved through ElGamal cryptosystem [14]. To improvise this system many variants were proposed by the researchers, but the security of the algorithm depends on the level of difficulty of solving the discrete logarithm problem remains unchanged.

Shiang et al. [15] proposed an ElGamal-based cryptosystem for enciphering the plaintext based on Diffie–Hellman key exchange [16]. The computational complexity was comparatively less to ElGamal cryptosystem and it relies on the discrete logarithm problem. Similarly, modified ElGamal cryptosystem algorithm (MECA) [17,18] was proposed to enhance the existing ElGamal cryptosystem, but still, it depends on integer factorization problem along with the discrete logarithm problem. Since it was based on a one-way function with increased execution time, it was unable to be used for authentication purpose. A variant of ElGamal cryptosystem with less time-consuming hash functions was used to encrypt short length messages such as passwords, PIN codes and details of credit card [19]. It proved secure encryption against Chosen – Ciphertext Attack. A modified ElGamal cryptosystem was also proposed to encrypt gray and color images in MATLAB depending on the discrete logarithm problem [20,21]. These variants of the ElGamal cryptosystem used different algorithms [13] to solve discrete logarithm problems such as naive, pollard's rho method and

baby-step/giant-step method. But research works to improve security by using hybrid concepts like combining any two public key cryptosystems and modifying the algorithmic concept making it as a novel approach. Most of the research work was combining the ideas of RSA and ElGamal cryptosystems [10,22–24]. Among which there was a digital signature algorithm based on the same discrete logarithm problem and the integer factorization problem making it harder to break the cryptosystem [25,26].

Few research works were based on elliptic curve cryptography, which is used for key distribution and authentication of user's identity. But the drawbacks were the size of the ciphertext was doubled and increased computation for a key generation [27]. A variant methodology operated on the hexadecimal representation that reduced the above drawbacks, but the security analysis of the cryptosystem was unproven [28]. In this methodology, the time taken for encryption and decryption is quite lesser than ElGamal cryptosystem, but still, it depends on the difficulty of integer factorization and discrete logarithm problems. Based on the related works, the Enhanced ElGamal cryptosystem needs to be designed, where security of the cryptosystem must not depend only on the discrete logarithm problem, but also on the level of randomness [29]. This increases the computation complexity for attackers to break the proposed cryptosystem.

In order to provide a secure data storage and retrieval in the cloud, data confidentiality need to be ensured through data encryption. Variants of DNA cryptosystems are analyzed based on the requirements to the fulfillment of the efficient DNA cryptosystem [7]. Yunpeng et al. [30] proposed a scheme of symmetric key cryptography where the XOR operation is performed between the plaintext and the key. The resulting sequence is mapped to a reference sequence and the positions are indexed to form the ciphertext. This system mainly depends on arithmetic operations without involving the biological process of DNA. Likewise, Abbasy et al. [31] initially converts the plaintext to binary, then finds the complement of the sequences and finds the index in the reference string which forms the ciphertext. The chances of occurring unique sequences are quite less and there is less need for an encoding table in this framework.

The concept of using codebooks to encrypt and decrypt is by finding the matches of the key file with the sequences in the codebook [32]. But here only the prefabricated messages can be communicated. The codebook remains static until the Data Owner and Data User decides to change the codebook. Thus encoding in a different manner has been performed where the biological processes are less involved.

In another approach, a substitution array is created, the plaintext is converted into ASCII and the values of the array is divided by the ASCII value, the quotient and remainders are converted into DNA nucleotides and then into amino acid sequences [33]. It is dynamic and follows few biological properties, but the encoding process for the unique sequence is less emphasized.

Mandge et al. [34] proposes an encryption scheme which involves a XOR operation, mini cipher generation and conversion of values into DNA sequences and then to amino acid sequences using a toolbox in MATLAB. It does not have any encoding table and so dynamicity of encoding table is not achieved. Majumder et al. [35] contributed towards block based encryption where the blocks of 256-bit plaintext are converted to 64-bit blocks and performed a XOR, a straight D-Box permutation and later the binary values are mapped to form the ciphertext. Thus, it incorporates very few biological processes.

Jain et al. [6] proposed a methodology of converting the plaintext into binary through ASCII values. The binary values are converted into decimal values which are mapped to a DNA sequence belonging to the DNA sequence dictionary. It uniquely maps for the given number of 0 to 255 with the corresponding DNA sequences. It involves very few steps and that contributes less towards the biological process.

A DNA cryptosystem involves generating a random key which is a DNA sequence adjusted to the binary length of the plaintext with its binary sequence values [36]. It is split into odd and even parts that are replaced with 0's in the first part and all 1's in the second part. It forms a dummy key with the initial bit as '1' and remaining with the replaced odd and even parts. It is done a XOR operation along with the original adjusted key to form an OTP key. This OTP key is used to encrypt the plaintext through XOR operations. Finally, the key and the ciphertext are converted into alphabet form. The usage of encoding tables to encode the character set of 96 elements of unique DNA sequences dynamically forms a robust system [7]. The encoding table becomes dynamic as it is generated with two new DNA sequences for every process. It involves the biological process of DNA like transcription, translation and protein synthesis. It operates completely on DNA sequences. But the computational complexity is high as it involves too many preprocessing and multiple rounds of algorithmic steps.

Majumder et al. [37] proposes a cryptosystem that divides the plaintext to blocks of 256 bits, each block is split into blocks of 64 bits. It is then performed four rounds of the XOR operation with the subkeys that get shifted by one block of values in each round. Finally, the resulting sequences are converted into DNA sequences and are appended with a random DNA sequence as a front primer and end primer. It does not involve an encoding table but performs dynamic encryption. The method of using biological equipment and processes like PCR amplification along with the usage of primers and codons is less feasible to be used in real time scenario [38]. In general, it uses DNA sequences and complementary pairs in the process of encryption. Similarly, another DNA cryptosystem involves cellular automata to support the robustness of

Table 2 Analysis on various DNA cryptosystems

Authors	Complete character set encoding	Dynamic encoding table generation	Unique sequence for character encoding	Robustness of encoding	Biological process simulation	Dynamic encryption process
Yunpeng [30]	*	×	*	✓	×	✓
Gao [32]	×	×	×	×	×	×
Abbasy [31]	×	×	×	✓	×	✓
Dhawan [33]	*	×	*	✓	*	✓
Mandge [34]	*	×	*	✓	✓	✓
Cui [38]	*	×	*	✓	✓	✓
Majumder [37]	*	×	×	✓	×	✓
Jain [6]	×	×	✓	*	×	*
Aich [36]	*	*	✓	✓	*	×
Rahman [7]	✓	✓	✓	✓	✓	✓
Majumder [35]	*	×	×	✓	×	✓
Sundaram [39]	✓	✓	✓	*	*	×
Aich [40]	*	*	✓	✓	✓	✓
Gugnani [41]	*	×	×	*	×	✓
Our Cryptosystem	✓	✓	✓	✓	✓	✓

✓ Follows to a greater extent, × follows to a least extent, * follows to an average extent

the system [39]. Here, the plaintext is converted into a DNA sequence using an encoding table with 66 elements. It is then converted into binary and performed XOR operation with the keys generated and the final binary sequence is applied with a pattern of rule 51 in cellular automata. Then it is transformed to DNA sequence and performed a transition using automata to generate the ciphertext. Here, it forms a static encoding table known as a codebook.

The DNA cryptosystem proposed by Aich et al. [40] uses Diffie – Hellman technique to generate and share a secret key which is converted to DNA sequence. The plaintext is converted into DNA sequence and appended with primers and then converted to binary and performed DNA hybridization with the key to forming the ciphertext. Here, the encoding table is static.

Gugnani et al. [41] applies DNA cryptography in XML-SOAP file encryption. Initially, the important data like Account PIN numbers, passwords, etc. present in the file are extracted and converted into binary. It is then converted into DNA bases, complementary pairs are replaced with the bases. A DNA reference string is then hybridized with this sequence and outputs the position values as the ciphertext string. It does not generate a unique character encoding as the reference sequence bases repeat multiple times in the sequence.

The DNA cryptosystem [23] is unique in handling the UNICODE characters and has been utilized for secure data transfer in the cloud. The plaintext can be in any language. Initially, the plaintext is converted from UNICODE to ASCII character considering each eight bits. It is converted to a binary sequence using hexadecimal values. The DNA sequence is encoded based on the key combination table that maps two DNA nucleotides with four binary digits. This becomes the ciphertext. Another DNA cryptosystem uses a key to convert the plaintext into a DNA ciphered sequence using its molecules [1]. This is implemented for secure data transfer in the AWS cloud environment.

From the above-studied DNA cryptosystems, it is very clear that very few cryptosystems are dynamic and others work with static keys, encoding tables, code books, etc. The utilization of biological properties of DNA is very less among the proposed cryptosystems which are less secure for cloud data security. Thus, our proposed DNA cryptosystem is made dynamic in encoding table generation as well in encryption. It utilizes the biological properties of DNA like complementary pairs, translation, transcription, and amino acid processes. It satisfies the six properties which have been framed in [7] and also overcomes the larger permutations without compromising data security in the cloud.

The study on various DNA cryptosystems based on the six properties is shown in Table 2.

3 Proposed work

The proposed framework involves major stakeholders who operate on the data in a cloud environment such as the Data Owner who sends the data to the Data User when he/she requests for. Initially, the Data Owner converts the unique ID of the Data User into a DNA sequence known as Data

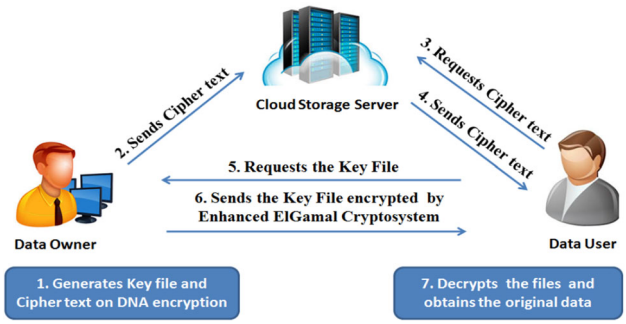

Fig. 1 Architecture of data storage and retrieval from cloud

User sequence. The Data Owner generates a random DNA sequence known as Data Owner sequence. Then Data Owner encrypts the data with both the DNA sequences using Novel DNA cryptosystem and outputs a key file and ciphertext file. The ciphertext file is stored in the cloud. The key file is then encrypted with the public keys of the Data User generated using the novel Enhanced ElGamal cryptosystem. When the Data User requests the ciphertext to cloud it checks the Data User and sends the corresponding ciphertext as shown in Fig. 1.

The Data User then requests for the key file to the Data Owner. Data Owner sends the encrypted key file which is decrypted with the private keys of the Data User using Enhanced ElGamal cryptosystem. It then decrypts the key file and the ciphertext file using Novel DNA cryptosystem to obtain the original data.

Thus, the proposed framework consists of two cryptosystems namely, symmetric key cryptosystem to encrypt the data is a Novel DNA cryptosystem and the public key cryptosystem to authenticate the user and to encrypt the key file is an Enhanced ElGamal cryptosystem.

The Enhanced ElGamal cryptosystem (EEC) security lies on the randomness and the discrete logarithm problem. The increase of randomness leads to increase in the security of the cryptosystem. It also authenticates the Data Owner and Data User with their private keys and public keys. The key file could be decrypted only by the intended Data User using his private key. Thus, the confidentiality of the data is maintained among the stakeholders in a cloud environment.

3.1 Novel DNA cryptosystem

The Novel DNA Cryptosystem upholds the data confidentiality in a cloud environment with utmost security. It makes it harder to break up by involving the biological processes of DNA, which are highly randomized in the proposed mechanism.

DNA Cryptography causes less computational time, but its robust process increases the attack time. This essential tradeoff is least achieved by most of the traditional cryptographic approaches proving the significance of the proposed Novel DNA Cryptosystem. The proposed Novel DNA cryptosystem comprises of three phases,

1. Novel DNA encoding table generation
2. Novel DNA encryption algorithm
3. Novel DNA decryption algorithm

3.1.1 Novel DNA encoding table generation

The generated encoding table is based upon the process of protein synthesis by the amino acids of DNA forming the vital part of our framework. Initially, the sequence of both the Data Owner and the Data User is generated. The unique ID of Data User is transformed to form a DNA sequence of having four unique nucleotides known as Data User sequence and Data Owner generates a random DNA sequence known as Data Owner sequence. These sequences are converted into mRNA sequence and then to tRNA sequence. A 4*4 matrix is formed by considering the sequences as row and column. The entries are formed by combining the row and column nucleotides. It is extended to a 16*16 matrix in a similar manner. It forms the amino acid table, where the values are collated with the collating amino value. The ASCII character set is extended to 256 values and it is collated with the collating character value. The collating value is a number that can be from 1 to 256. Both the amino sequences and the character set are mapped to form the DNA encoding table as shown in Fig. 2.

The pseudo code for the generation of encoding table is,

Encoding_table (DO,DU,ca,cc)
Input: Data Owner Sequence **DO**, Data User Sequence **DU**, Collate_amino **ca**, Collate_character **cc**
Output: Encoding Table $\mathbf{E_T}$
Method Variables: ASCII character set $\mathbf{cs_{94}}$, Extended character set $\mathbf{cs_{256}}$, mRNA sequence of DO $\mathbf{DO_m}$, mRNA sequence of DU $\mathbf{DU_m}$, tRNA sequence of DO $\mathbf{DO_t}$, tRNA sequence of DU $\mathbf{DU_t}$, 4 * 4 matrix of DO_t * DU_t $\mathbf{D_4}$, 16 * 16 matrix of D_4* D_4 $\mathbf{D_{16}}$
Procedure:
convert **DO**, **DU** into mRNA sequence $\mathbf{DO_m}$, $\mathbf{DU_m}$ respectively
convert $\mathbf{DO_m}$, $\mathbf{DU_m}$ to tRNA sequence $\mathbf{DO_t}$, $\mathbf{DU_t}$ respectively
compute $\mathbf{D_4}$
compute $\mathbf{D_{16}}$
collate $\mathbf{D_{16}}$ with **ca**
generate $\mathbf{cs_{94}}$ and extend $\mathbf{cs_{94}}$ to $\mathbf{cs_{256}}$
collate $\mathbf{cs_{256}}$ with **cc**
map $\mathbf{D_{16}}$ and $\mathbf{cs_{256}}$ to form $\mathbf{E_T}$

The steps to generate the encoding table is as follows:

Step 1: Initially, the Data Owner DNA sequence and Data User DNA sequence, each having four nucleotide bases of DNA are taken in a unique order.
Step 2: Convert both the sequences into mRNA sequence.
Step 3: Convert the resulting mRNA sequences into tRNA sequences which form the input for the encoding table.

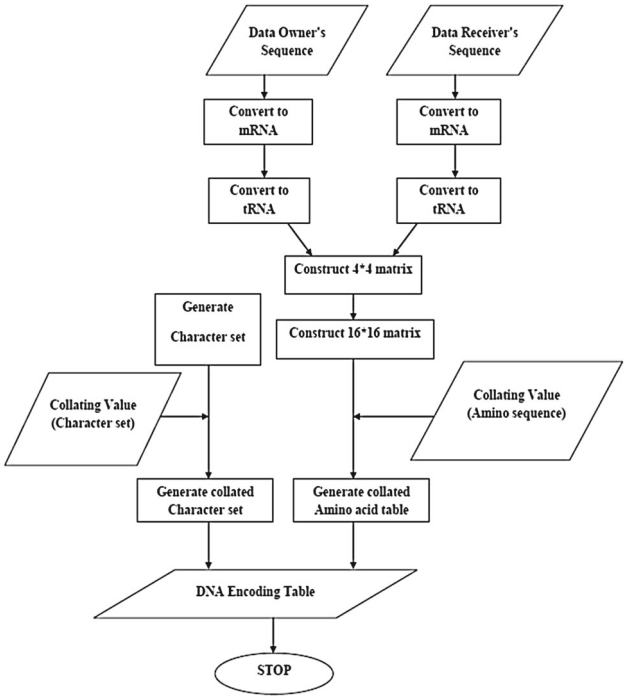

Fig. 2 Encoding table generation

Step 4: Compute the 4*4 matrix of sequences using the tRNA sequences as row and column.
Step 5: The values of the above matrix are taken as row and column to generate a 16*16 matrix of sequences that forms the amino acid table.
Step 6: A collating value is chosen to circularly shift the generated amino sequences.
Step 7: The entire ASCII character set of printable 94 elements is extended to 256 elements by having the prefix of the character 'D' for the first time on the character set, then it is prefixed with the character 'N' and for the remaining elements it is prefixed with the character 'A'.
Step 8: A collating value is chosen to circularly shift the generated 256 character set elements which are then mapped to the amino sequences to form the encoding table.

3.1.2 Novel DNA encryption algorithm

The encryption process is done as shown in Fig. 3. The data needed by the Data User would be as a plaintext to the Data Owner. It is converted into binary form through ASCII and it has performed an XNOR operation with the concatenated binary sequence of the Data Owner and the Data User.

It is further split into N blocks based on the value of a number of bits per block which can be 16, 32, 64, 128, 256, 512 or 1024. The odd blocks are performed a XOR operation with the intron sequence made to the length of the block by either appending or splitting it. The even blocks are performed a XOR operation with the reverse of the intron sequence. The obtained block is then reversed and all blocks are concatenated. It is converted into DNA sequence, then to mRNA sequence and further to tRNA sequence. The tRNA sequence is considered with sequences of four nucleotides and mapped to the encoding table. The encoded values are split into odd digits converted into special characters by mapping with the characters such as D-$, N-*, A-@ which forms the ciphertext along combined with even digits. The key file holds the collating values, DNA sequences, bits per block value and mapping with special characters. The pseudo code for the encryption process is as follows:

DNA_Encryption (pt, DO, DU, N, D, M, Y, H, Mi, S, RN, E_T)
Input: plaintext **pt**, Data Owner Sequence **DO**,
Data User Sequence **DU**, bits per block **N**, date **D**,
month **M**, year **Y**, hour **H**, minute **Mi**, second **S**,
random number **RN**, Encoding Table $\mathbf{E_T}$, Collate_amino **ca**,
Collate_character **cc**
Output: key file **cl**, ciphertext **ct**
Method Variables: binary plaintext $\mathbf{pt_b}$, binary DO $\mathbf{DO_b}$,
binary DU $\mathbf{DU_b}$, block $\mathbf{N_b}$, intron sequence **iseq**,
binary intron sequence $\mathbf{iseq_b}$, DNA sequence $\mathbf{DNA_{seq}}$,
mRNA sequence $\mathbf{mRNA_{seq}}$, tRNA sequence $\mathbf{tRNA_{seq}}$,
odd position $\mathbf{o_p}$, even position $\mathbf{e_p}$
Procedure:
convert **pt** into ASCII values
transform ASCII values to binary plaintext to form $\mathbf{pt_b}$
convert **DO, DU** to binary $\mathbf{DO_b}$, $\mathbf{DU_b}$
concatenate $\mathbf{DO_b}$, $\mathbf{DU_b}$ to form $\mathbf{DO_b.DU_b}$
XNOR $\mathbf{DO_b.DU_b}$ with $\mathbf{pt_b}$
split into **N** blocks $\mathbf{N_b}$
generate **iseq** with **D, M, Y, H, Mi, S, RN**
convert **iseq** to binary $\mathbf{iseq_b}$
for each $\mathbf{N_b}$
  if odd then XOR $\mathbf{N_b}$ and $\mathbf{iseq_b}$
  if even then XOR $\mathbf{N_b}$ and reverse of $\mathbf{iseq_b}$
  reverse $\mathbf{N_b}$
combine all $\mathbf{N_b}$ and concatenate $\mathbf{iseq_b}$
convert to $\mathbf{DNA_{seq}}$, then convert to $\mathbf{mRNA_{seq}}$
convert to $\mathbf{tRNA_{seq}}$, then encode with $\mathbf{E_T}$
split odd position $\mathbf{o_p}$ and even position $\mathbf{e_p}$
map $\mathbf{o_p}$ to $,*,@ for D,N,A
combine $\mathbf{o_p}$ and $\mathbf{e_p}$ as **ct**
append **N**, **ca, cc, DO, DU** as **cl**
have **cl &** send **ct**

The Data Owner encrypts the plaintext by the following steps:

Step 1: Convert the plaintext into ASCII code which is further converted into a binary sequence.
Step 2: Convert the DNA sequences of Data Owner and Data User into binary sequences and concatenate them as a single binary sequence.

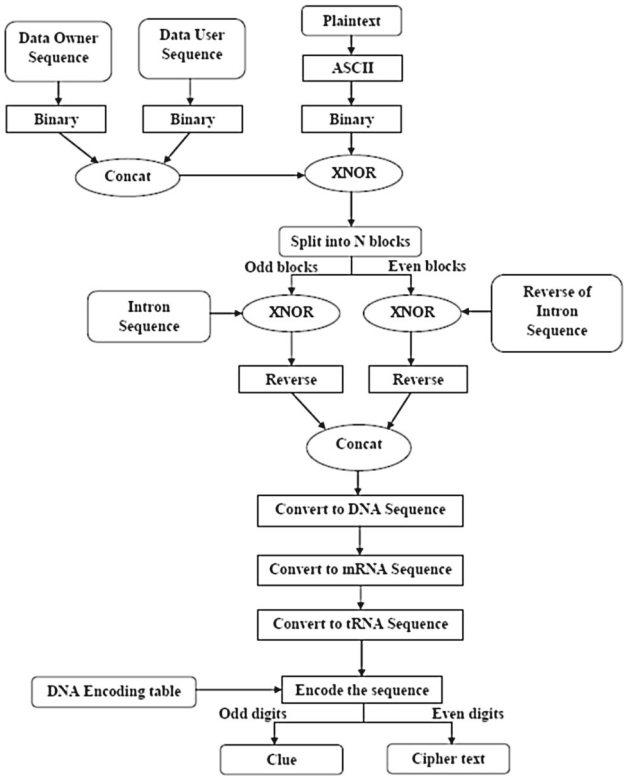

Fig. 3 DNA encryption process

Step 3: Perform XNOR operation on the plaintext binary sequence with the above resulting sequence.
Step 4: The resulting binary sequence is split into N bit blocks, where N varies as 16, 32, 64, 128, 256, 512 and 1024 based on the N-value given by the Data Owner.
Step 5: The intron sequence is generated from the values for Date, Month, Year (last two digits), Hour, Minute, Second each having two digits and a four digit random number.
Step 6: The values of the intron sequence are converted into binary for each digit, resulting in an intron binary sequence which is either split or concatenated depending on the block size.
Step 7: For every block of binary plaintext,

> Step 7.1: If they are odd blocks, perform the XOR operation for the odd blocks with the binary intron sequences.
> Step 7.2: If they are even blocks, perform the XOR operation for the even blocks with the reverse of the binary intron sequences.

Step 8: Reverse the bits for every resulting block and combine them as a single binary sequence. The binary intron sequence is concatenated with this binary sequence of the blocks.
Step 9: Convert the binary sequence into a DNA sequence by mapping binary values with the DNA nucleotides shown in Table 3.

Table 3 DNA nucleotide to binary value mapping

Nucleotide	Binary value
A	00
T	11
C	01
G	10

Step 10: The DNA sequence is converted into mRNA sequence where the Thymine (T) is replaced with Uracil (U).
Step 11: The mRNA sequence is then converted into tRNA sequence by taking the complement of each of the nucleotides.
Step 12: The obtained tRNA sequence is now mapped with the encoding table sequence values to form the encoded sequence.
Step 13: The encoded sequence is split into odd position digits and even position digits.
Step 14: The odd position digits will be of characters as D, N, and A. It is then replaced with the special characters as $, * and @.
Step 15: The DNA sequences of Data Owner and Data User, the bits per block size value, the collating value for the amino sequences and the collating value of the character set elements are combined to form the key file. The odd position values and even position values are combined to form the ciphertext.

The final key file is encrypted with the public key of the Data User is using EEC and kept by the Data Owner and the ciphertext is sent to the cloud storage on completion of the encryption process.

3.1.3 DNA decryption algorithm

The ciphertext is received from the cloud storage server and the key file received from the Data Owner and the ciphertext is decrypted using the private key of the Data User by EEC and both of these files are used for the DNA decryption process as shown in Fig. 4. The key file is synthesized to get the various values combined together in the file such as collating values, etc. It is combined with the ciphertext to form a sequence. It is converted into tRNA sequence, then converted to mRNA sequence and further converted into a DNA sequence. The intron sequence is obtained by removing the last bits of the block size value obtained from the key file. Split it into N blocks and perform the XOR operation with the intron sequence for odd blocks and perform the XOR operation with the reversed intron sequence for even blocks. Combine all the blocks and do the XOR operation with the concatenated binary sequence of Data Owner and Data User.

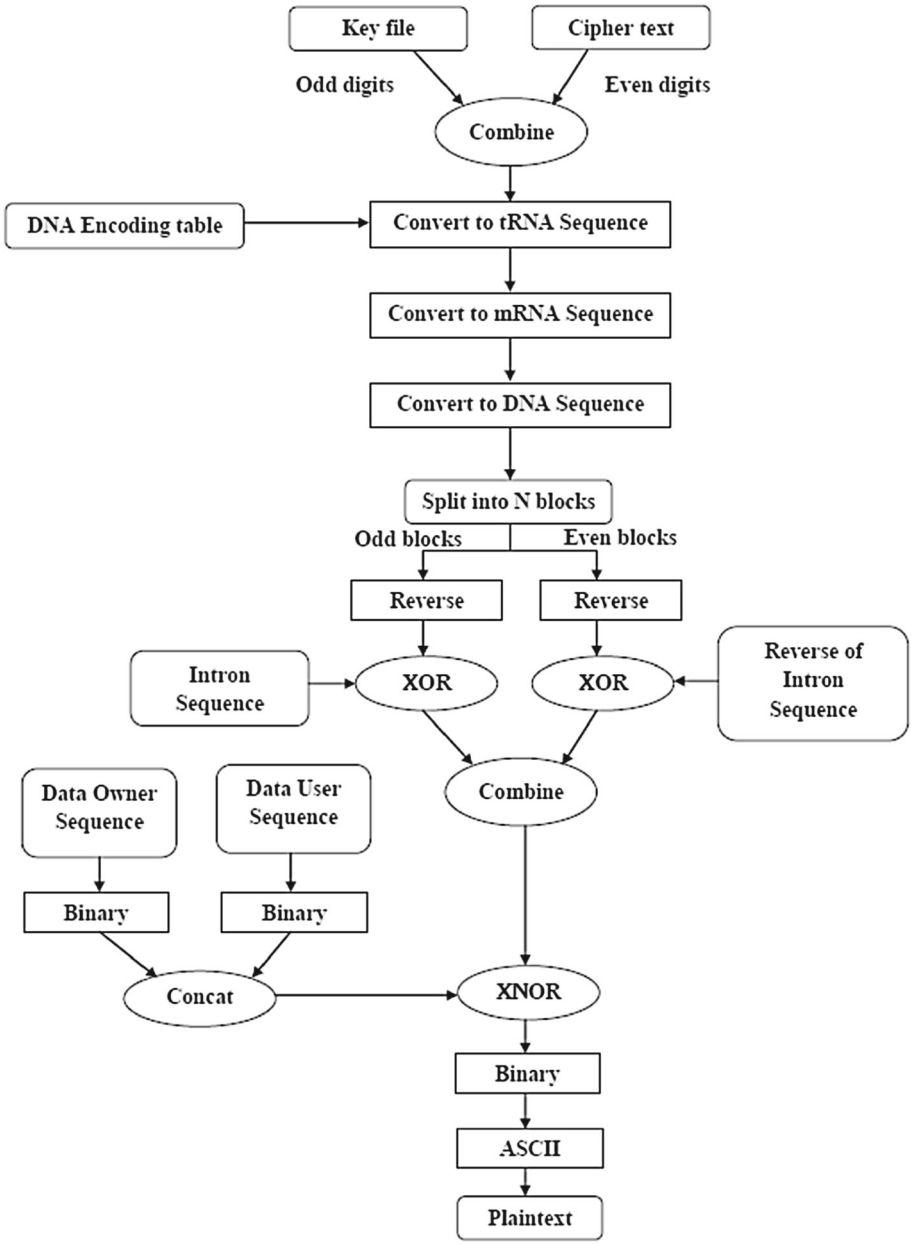

Fig. 4 DNA decryption process

The resulted sequence is converted into ASCII and then to the original plaintext sent by the Data User.

The procedure to perform DNA decryption process is as follows:

Step 1: The DNA sequences of Data Owner and Data User, the bits per block size value, the collating value for the amino sequences and the collating value of the character set elements are split from the key file.
Step 2: The odd position digits will be of characters as D, N, and A. It is then replaced with the values as $, * and @.
Step 3: The ciphertext is processed to form the encoded sequence by combining the odd position digits and even position digits.
Step 4: The encoded sequence is mapped with the encoding table values to form the tRNA sequence.
Step 5: The mRNA sequence is obtained by taking the complement of the tRNA sequence nucleotides.
Step 6: The mRNA sequence is then converted into a DNA sequence by replacing Uracil (U) with Thymine (T).
Step 7: The resulting DNA sequence is transformed into the binary sequence.
Step 8: The bits per block value is obtained from the key file and that a block size of bits is removed from the binary sequence to obtain the binary intron sequence.

Step 9: The sequence is split further to obtain the blocks and each block are reversed to obtain the final blocks of values.
Step 10: For every block obtained,

Step 10.1: If they are odd blocks, perform the XOR operation for the odd blocks with the binary intron sequences.
Step 10.2: If they are even blocks, perform the XOR operation for the even blocks with the reverse of the binary intron sequences.

Step 11: The blocks are combined together to form a single binary sequence.
Step 12: The Data Owner DNA sequence and the Data User DNA sequence are converted into binary sequences and concatenated to form a single sequence.
Step 13: The binary string of the blocks and the above resulting sequence are XNORed and it results in a binary sequence of the plaintext.
Step 14: The binary sequence is converted into ASCII values.
Step 15: The ASCII values are converted to form the final original plaintext.

Thus, the Data User obtains the original plaintext sent by the Data Owner securely.

The pseudo code for DNA decryption is as follows:

DNA_Decryption (cl, ct, E_T)
Input: key file **cl**, ciphertext **ct**, Encoding Table E_T
Output: plaintext **pt**
Method Variables: Data Owner Sequence **DO**, Data User Sequence **DU,** bits per block **N,** binary plaintext pt_b, binary DO DO_b, binary DU DU_b, block N_b, intron sequence **iseq**, binary intron sequence $iseq_b$, DNA sequence DNA_{seq}, mRNA sequence $mRNA_{seq}$, tRNA sequence $tRNA_{seq}$, odd position o_p, even position e_p
Procedure:
split **cl** into **N**, **ca, cc, DO, DU**
map $,*,@ to D,N,A for o_p
combine odd position o_p and even position e_p
encode with E_T
convert to $tRNA_{seq}$
convert to $mRNA_{seq}$
convert to DNA_{seq}
split all N_b and $iseq_b$
reverse all N_b
for each N_b
  if odd then XOR N_b and $iseq_b$
  if even then XOR N_b and reverse of $iseq_b$
combine all N_b to form pt_b
convert **DO, DU** to binary DO_b, DU_b
concatenate DO_b, DU_b to form $DO_b.DU_b$
XNOR $DO_b.DU_b$ with pt_b
transform pt_b to ASCII values
convert ASCII values into **pt**

3.2 Enhanced ElGamal cryptosystem (EEC)

In the EEC, the Data User computes the public key and private key of the system. Then, the Data Owner selects the random keys and the key file to be communicated. The encryption function is invoked by the Data Owner to encrypt the key file with the computed keys. The ciphertext will be sent to the Data User through any kind of networks. The Data User will decrypt the ciphertext to retrieve the key file by computing the random key.

The Enhanced ElGamal cryptosystem works as follows: The key generation procedure involves choosing a large prime number, q and finding its primitive roots, α, β and perform the modular inverse for the multiple values of primitive root which is denoted as 'd'. It also involves the selection of a random number for setting up a private key X, which should be $1 < X < q - 1$ and determine the public key from the random number Y.

The encryption procedure involves by choosing 2 random numbers as k_1, k_2 and selecting a shared secret key, k_3. Using these 2 random numbers and a secret key, compute one time secret key K, to encrypt the key file. The key file is encrypted as $C = (C_1, C_2, C_3)$. The decryption procedure involves retrieving the one time secret key K. The ciphertext is decrypted by computing the value of K, and the private key $\{X, d\}$.

The proposed EEC algorithm is given as pseudo code.

Data User:
Key Generation:
Choose q, a large prime number and α, β primitive roots of q
Find $d = (\alpha.\beta)^{-1} \bmod q$
Choose X, a random integer such that $1 < X <$ q-1
Compute Y as $Y = (\alpha.\beta)^X \bmod q$
Private Key: {X, β,d}, Public Key: {q, α, Y}
Data Owner:
Encryption:
Represent message as integer m such that $0 \le m \le$ q-1
Choose 2 random integers k_1, k_2, such that $1 \le k_1, k_2 \le$ q-1
Select shared secret key k_3, $1 \le k_3 \le$ q-1
Compute one-time key $K = (k_1)^{k_2}.Y^{k_3} \bmod q$
Compute $C_1 = \alpha^{k_3} \bmod q$
Compute $C_2 = k_1^{k_2} \bmod q$
Compute $C_3 =$ K. m. Y $\bmod q$
Ciphertext $C = (C_1, C_2, C_3)$ send to the Data User.
Secretly share k_3 to the Data User
Data User:
Decryption:
Recover one time key by computing
$K = C_1^X.C_2.\beta^{k_3.X} \bmod q$
Find $K^{-1} \bmod q$
Retrieve message m = $K^{-1}.C_3.d^X \bmod q$

The proposed algorithm can be proved mathematically by the following way: The decryption equation m =

$K^{-1}.C_3.d^X \ mod \ q$ is taken and the proof involves getting back the original message from it.

$$
\begin{aligned}
m &= K^{-1}.C_3.d^X \ mod \ q \text{ (substitute K value)} \\
&= C_1^{-X}.C_2^{-1}.\beta^{-k_3 \cdot X}.C_3.d^X \ mod \ q \\
&\quad \text{(substitute } C_1, C_2 \text{values)} \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3 \cdot X}.C_3.d^X \ mod \ q \\
&\quad \text{(substitute } C_3 \text{value)} \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3 \cdot X}.K.m.Y.d^X \ mod \ q \\
&\quad \text{(substitute K value)} \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3 \cdot X}.k_1^{k_2}.Y^{k_3}.m.Y.d^X \ mod \ q \\
&\quad \text{(substitute Y value)} \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3 \cdot X}.k_1^{k_2}.\alpha^{k_3.X}.\beta^{k_3.X}.m.Y.d^X \ mod \ q \\
&\quad \text{(inverses cancel each others)} \\
&= m.Y.d^X \ mod \ q \text{ (substitute Y value)} \\
&= m.\alpha^X.\beta^X.d^X \ mod \ q \text{ (substitute d value)} \\
&= m.\alpha^X.\beta^X.\alpha^{-X}.\beta^{-X} \ mod \ q \\
&\quad \text{(inverses cancel each other)} \\
&= m
\end{aligned}
$$

Thus, the proposed algorithm has been proved mathematically.

3.3 Detailed example

The plaintext is taken as "**mRNA**". The random Data Owner's sequence is ATCG. The Data User's sequence generated from his unique ID is CTAG (Let the unique ID be 72. Converting to DNA sequence through binary values of each digit forms the DNA sequence CTAG using Table 3).

3.3.1 Novel DNA encoding table generation

Step 1: The Data Owner's sequence **ATCG** and the Data User's sequence**CTAG** is taken.
Step 2: Both the sequences are converted into mRNA sequence, where T is replaced with U.

ATCG => **AUCG**
CTAG => **CUAG**

Step 3: Convert the resulting mRNA sequences into tRNA sequences which form the input for the encoding table. (Replace A with U, C with G and vice-versa).

AUCG => **UAGC**
CUAG => **GAUC**

	G	A	U	C
U	UG	UA	UU	UC
A	AG	AA	AU	AC
G	GG	GA	GU	GC
C	CG	CA	CU	CC

Fig. 5 4*4 matrix formation

Step 4: Compute 4*4 matrix of sequences using the tRNA sequences as row and column as shown in Fig. 5.
Step 5: The values of the above matrix are taken as row and column to generate a 16* 16 matrix of sequences such as in Fig. 6.
Step 6: The collating amino value is chosen as **3** and the amino sequences are circularly shifted in the generated 16* 16 matrix entries.
Step 7: The entire character set of 94 elements is extended to 256 elements by having the prefix of the character 'D' for the first time on the character set, then it is prefixed with the character 'N' and for the remaining elements it is prefixed with the character 'A'. It is shown in Fig. 7.
Step 8: The collating character value is chosen as **3** and it is used to circularly shift the generated 256 character set elements which are then mapped to the 16* 16 sequences generated to form a complete encoding table as shown in Fig. 8.

3.3.2 Novel DNA encryption algorithm

For simplicity, the plaintext of the Data Owner is taken as "**mRNA**".

Step 1: The plaintext into ASCII code which is further converted into a binary sequence.

mRNA => **109 82 78 65 109 82 78 65**
=> **01101101010100100100111001000001**

Step 2: The Data Owner's and Data User's DNA sequences **ATCG**,**CTAG** are converted into binary sequences and are concatenated.

=> **0011011001110010**

Step 3: XNOR operation in the plaintext binary sequence with the above resulting sequence is performed and the result is,

=> **10100100110111111000011111001100**

	UG	UA	UU	UC	AG	AA	AU	AC	GG	GA	GU	GC	CG	CA	CU	CC
UG	UGUG	UGUA	UGUU	UGUC	UGAG	UGAA	UGAU	UGAC	UGGG	UGGA	UGGU	UGGC	UGCG	UGCA	UGCU	UGCC
UA	UAUG	UAUA	UAUU	UAUC	UAAG	UAAA	UAAU	UAAC	UAGG	UAGA	UAGU	UAGC	UACG	UACA	UACU	UACC
UU	UUUG	UUUA	UUUU	UUUC	UUAG	UUAA	UUAU	UUAC	UUGG	UUGA	UUGU	UUGC	UUCG	UUCA	UUCU	UUCC
UC	UCUG	UCUA	UCUU	UCUC	UCAG	UCAA	UCAU	UCAC	UCGG	UCGA	UCGU	UCGC	UCCG	UCCA	UCCU	UCCC
AG	AGUG	AGUA	AGUU	AGUC	AGAG	AGAA	AGAU	AGAC	AGGG	AGGA	AGGU	AGGC	AGCG	AGCA	AGCU	AGCC
AA	AAUG	AAUA	AAUU	AAUC	AAAG	AAAA	AAAU	AAAC	AAGG	AAGA	AAGU	AAGC	AACG	AACA	AACU	AACC
AU	AUUG	AUUA	AUUU	AUUC	AUAG	AUAA	AUAU	AUAC	AUGG	AUGA	AUGU	AUGC	AUCG	AUCA	AUCU	AUCC
AC	ACUG	ACUA	ACUU	ACUC	ACAG	ACAA	ACAU	ACAC	ACGG	ACGA	ACGU	ACGC	ACCG	ACCA	ACCU	ACCC
GG	GGUG	GGUA	GGUU	GGUC	GGAG	GGAA	GGAU	GGAC	GGGG	GGGA	GGGU	GGGC	GGCG	GGCA	GGCU	GGCC
GA	GAUG	GAUA	GAUU	GAUC	GAAG	GAAA	GAAU	GAAC	GAGG	GAGA	GAGU	GAGC	GACG	GACA	GACU	GACC
GU	GUUG	GUUA	GUUU	GUUC	GUAG	GUAA	GUAU	GUAC	GUGG	GUGA	GUGU	GUGC	GUCG	GUCA	GUCU	GUCC
GC	GCUG	GCUA	GCUU	GCUC	GCAG	GCAA	GCAU	GCAC	GCGG	GCGA	GCGU	GCGC	GCCG	GCCA	GCCU	GCCC
CG	CGUG	CGUA	CGUU	CGUC	CGAG	CGAA	CGAU	CGAC	CGGG	CGGA	CGGU	CGGC	CGCG	CGCA	CGCU	CGCC
CA	CAUG	CAUA	CAUU	CAUC	CAAG	CAAA	CAAU	CAAC	CAGG	CAGA	CAGU	CAGC	CACG	CACA	CACU	CACC
CU	CUUG	CUUA	CUUU	CUUC	CUAG	CUAA	CUAU	CUAC	CUGG	CUGA	CUGU	CUGC	CUCG	CUCA	CUCU	CUCC
CC	CCUG	CCUA	CCUU	CCUC	CCAG	CCAA	CCAU	CCAC	CCGG	CCGA	CCGU	CCGC	CCCG	CCCA	CCCU	CCCC

Fig. 6 Amino acid table generation

D!	D"	D#	D$	D%	D&	D'	D(	D)	D*	D+	D,	D-	D.	D/	D0
D1	D2	D3	D4	D5	D6	D7	D8	D9	D:	D;	D<	D=	D>	D?	D@
DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN	DO	DP
DQ	DR	DS	DT	DU	DV	DW	DX	DY	DZ	D[	D\	D]	D^	D_	D`
Da	Db	Dc	Dd	De	Df	Dg	Dh	Di	Dj	Dk	Dl	Dm	Dn	Do	Dp
Dq	Dr	Ds	Dt	Du	Dv	Dw	Dx	Dy	Dz	D{	D|	D}	D~	N!	N"
N#	N$	N%	N&	N'	N(	N)	N*	N+	N,	N-	N.	N/	N0	N1	N2
N3	N4	N5	N6	N7	N8	N9	N:	N;	N<	N=	N>	N?	N@	NA	NB
NC	ND	NE	NF	NG	NH	NI	NJ	NK	NL	NM	NN	NO	NP	NQ	NR
NS	NT	NU	NV	NW	NX	NY	NZ	N[	N\	N]	N^	N_	N`	Na	Nb
Nc	Nd	Ne	Nf	Ng	Nh	Ni	Nj	Nk	Nl	Nm	Nn	No	Np	Nq	Nr
Ns	Nt	Nu	Nv	Nw	Nx	Ny	Nz	N{	N|	N}	N~	A!	A"	A#	A$
A%	A&	A'	A(	A)	A*	A+	A,	A-	A.	A/	A0	A1	A2	A3	A4
A5	A6	A7	A8	A9	A:	A;	A<	A=	A>	A?	A@	AA	AB	AC	AD
AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT
AU	AV	AW	AX	AY	AZ	A[	A\	A]	A^	A_	A`	Aa	Ab	Ac	Ad

Fig. 7 Character set

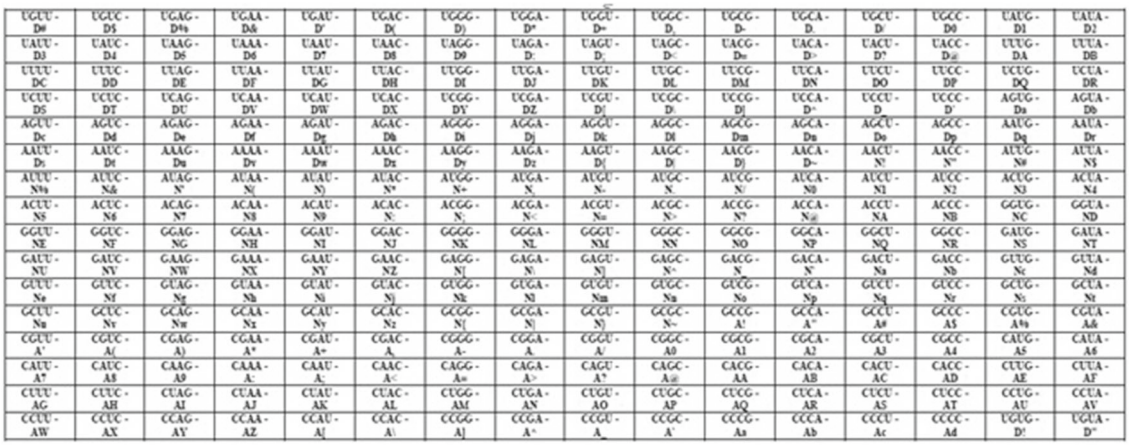

Fig. 8 Encoding table

Step 4: The resulting binary sequence is split into N bit blocks, where N is taken as **16**.
Step 5: The intron sequence is generated from the values as given in Table 4.
Step 6: The values of the intron sequence are converted into binary for each digit, resulting in an intron binary sequence which is split to form a 16-bit sequence.

=> **0000011100000110**

Table 4 Input values for intron sequence generation

Date = 07	Hour = 01
Month = 06	Minute = 37
Year = 89	Second = 01
Random number = 1234	

Step 7: The blocks of binary plaintext (step 3) are,

B1 = 1010010011011111 B2 = 1000011111001100

Step 7.1: The odd blocks are XORed with the binary intron sequence.
Step 7.2: The even blocks are XORed with the reverse of the binary intron sequence.

Step 8: Every block is reversed and combined them as a single binary sequence and it is combined with intron sequence as,

=> 1001101111000101001101001111001110000011100000110

Step 9: The binary sequence is converted into a DNA such as,

=> GCGTTACCATCATGCTAACTAACG

Step 10: The DNA sequence is converted into an mRNA sequence where the Thymine (T) is replaced with Uracil (U)

=> GCGUUACCAUCAUGCUAACUAACG

Step 11: The mRNA sequence is then converted into tRNA sequence. **=> CGCAAUGGUAGUACGAUUGAUUGC**
Step 12: The obtained tRNA sequence is now mapped with the encoding table sequence values (Fig. 8), to form the encoded sequence as, **=> A2N + D; N < DJDL**
Step 13: The encoded sequence is split into odd position digits and even position digits.

Odd position digits **=> ANDNDD**
Even position digits **=> 2+; < JL**

Step 14: The odd position digits will be of characters as D, N and A. It is then replaced with the special characters such as $, * and @ respectively.

Odd position digits => @*$*$$
Even position digits **=> 2+; < JL**

Step 15: The key file and the ciphertext are obtained as,

Key file **=> 000300160003ATCGCTAG$D*N@A**
Ciphertext **=> @2*** + $;* < **$J$L**

The final key file is then encrypted with the public key of the Data User using Enhanced ElGamal cryptosystem and the ciphertext is stored in the cloud. When the Data User requests the corresponding file Data Owner sends the encrypted key file.

3.3.3 Novel DNA decryption algorithm

The key file after decrypting through Enhanced ElGamal cryptosystem is taken along with the ciphertext for the decryption process by the Data User as follows:

Step 1: The key file and the ciphertext are,

Key file **=> 000300160003ATCGCTAG$D*N@A**
Ciphertext **=> @2*** + $;* < **$J$L**

Step 2: The odd position digits are replaced with the values as D, N and A.

Odd position digits **=> ANDNDD**
Even position digits **=> 2+; < JL**

Step 3: The odd position digits and the even position digits are combined to form the encoded sequence.

=> A2N + D; N < DJDL

Step 4: The encoded sequence is mapped with the encoding table values to form the tRNA sequence.

=> CGCAAUGGUAGUACGAUUGAUUGC

Step 5: The mRNA sequence is obtained by taking the complement of the tRNA sequence nucleotides.

=> GCGUUACCAUCAUGCUAACUAACG

Step 6: The mRNA sequence is then converted into a DNA sequence by replacing Uracil (U) with Thymine (T).

=> GCGTTACCATCATGCTAACTAACG

Step 7: The resulting DNA sequence is transformed into the binary sequence.

=> 1001101111000101001101001111001110000011100000110

Step 8: The bits per block value obtained from the key file is **16** and the binary intron sequence is,

=> 0000011100000110

Step 9: The sequence is split further to obtain the blocks and each blocks are reversed to obtain the final blocks of values.
Step 10: For every blocks obtained,

Step 10.1: The odd blocks are XORed with the binary intron sequences.
Step 10.2: The even blocks are XORed with the reverse of the binary intron sequences.

They final blocks obtained are,

B1 = 1010010011011111
B2 = 1000011111001100

Step 11: The blocks are combined together to form a single binary sequence.

=> **10100100110111111000011111001100**

Step 12: The Data Owner's DNA sequence obtained from the key file and the Data User's DNA sequence is converted into binary sequences and concatenated to form a single sequence.

=> **0011011001110010**

Step 13: The binary string of the blocks and the above resulting sequence are XNORed and its results as,

=> **01101101010100100100111001000001**

Step 14: The binary sequence is converted into ASCII values.

=> **109 82 78 65**

Step 15: The ASCII values are converted to form the final original plaintext.

=> **mRNA**

Thus, the original plaintext is obtained.

3.3.4 Enhanced ElGamal cryptosystem

Data User:
Key Generation
Take a small prime (for testing purpose), q = 101 and its primitive root is taken as α = 2, β = 72
Compute $d = (\alpha.\beta)^{-1} \ mod\ q$, d = 47
Choose a random integer such that 1 < X < $q - 1$, X = 10
Compute $Y = (\alpha.\beta)^X \ mod\ q$, $Y = 95$
Private Key: {X, β, d}, Public Key: {q, α, Y}
Private Key: {10, 72, 47}, Public Key: {101, 2, 95}

Data Owner:
Encryption
Choose 2 random integers k_1, k_2 such that $1 \leq k_1, k_2 \leq q-1, k_1 = 25, k_2 = 16$
Choose k_3, such that $1 \leq k_3 \leq q-1, k_3 = 7$
Compute one time key $K = (k_1)^{k_2}.Y^{k_3} \ mod\ q$, K = 54
Compute $C_1 = \alpha^{k_3} mod\ q$, $\boldsymbol{C_1} = 27$, $C_2 = k_1^{k_2} mod\ q$, $\boldsymbol{C_2} = 52$
Convert the key file characters into ASCII values as,
Key file

=> **000300160003ATCGCTAG$D*N@A**
=> {484848514848495448484851658467716784657136684278646 5}

Let the message m = {48 48 48 51 48 48 49 54 48 48 48 51 65 84 67 71 67 84 65 71 36 68 42 78 64 65},
Compute C_3 = K.m.$Y mod\ q$, C_3 = {2 2 2 40 2 2 82 78 2 2 2 40 49 54 7 24 7 54 49 24 52 87 27 79 70 49}
Ciphertext $C = (C_1, C_2, C_3)$ are send to the Data User.
Secretly share $k_3 = 7$ to the Data User.
Data User:
Decryption:
Recover keys by computing $K = C_1^X.C_2.\beta^{k_3.X} mod\ q$, K = 54
Retrieve message m = $K^{-1}.C_3.d^X \ mod\ q$,

m = {48 48 48 51 48 48 49 54 48 48 48 51 65 84 67 71 67 84 65 71 36 68 42 78 64 65}

Convert the ASCII values into characters to obtain the original key file as,

=> {48 48 48 51 48 48 49 54 48 48 48 51 65 84 67 71 67 84 65 71 36 68 42 78 64 65}
Key file => **000300160003ATCGCTAG$D*N@A**

4 Implementation and results

The implementation of the proposed framework has been done in a private cloud using Eucalyptus is running on a 2.50 GHz Intel ®Core™ i5-3120 M Processor and 16 GB RAM. Twelve clusters are formed with the sufficient number of nodes (owners/users) for representing each cluster. The node which initiates to upload a file will be the owner, who encrypts and shares the key file to the user. The Data Owner encrypts the data and sends to the Data User. The Data User decrypts the data in order to access it. With this cloud setup, the performance analysis of the proposed framework has been made. The various metrics are analyzed for

the novel cryptosystems, Enhanced ElGamal cryptosystem and DNA cryptosystem.

4.1 Performance analysis of enhanced ElGamal cryptosystem

In order to commit that the Enhanced ElGamal cryptosystem is more secure than the traditional ElGamal cryptosystem, the private key term needs to be made complex against cryptanalysis. Here, X is a private key, β is a primitive root and d is the inverse of primitive root multiples, which is kept secret (i.e.,) not shared with the Data Owner.

In the case of cryptanalysis, X need to be computed by solving discrete logarithm problem and β, d needs to be selected through randomness. In the ElGamal cryptosystem, $m = C_2.C_1^{-X} \, mod \, q$, so that cryptanalysis can be easily done by randomly identifying the private key 'X'.

But in the Enhanced ElGamal cryptosystem $m = \left(C_1^X \cdot C_2\right)^{-1} \cdot C_3 \cdot d^X \, mod \, q$, so the cryptanalysis has made complex by introducing private key more than once and the modular inverse of the public terms, while performing the decryption. So basically the amount of time taken to break the Enhanced ElGamal cryptosystem will be higher compared to the ElGamal cryptosystem.

In the ElGamal cryptosystem, (i) for encryption, two power modulus and one multiplication modulus operations need to be performed, and (ii) for decryption, one power modulus, one multiplicative inverse and one multiplicative modulus operations need to be performed. But in Enhanced ElGamal cryptosystem, (i) for encryption, four power modulus and two multiplication modulus operations need to be performed, and (ii) for decryption, three power modulus, one multiplicative inverse and four multiplicative operations need to be performed. So, as comparatively both encryption time and decryption time of an Enhanced ElGamal cryptosystem will be higher than ElGamal cryptosystem .

4.2 Performance analysis of enhanced DNA Cryptosystem

Based on the survey [42], we have identified that a standardized DNA cryptosystem with experimental analysis is an emerging research area. The time taken to encrypt and decrypt the data is dependent on the size of the plaintext whereas the time taken for the generation of the encoding table is the same always and it is independent of the plaintext always. Similarly, the various metrics analyzed in our proposed algorithm and the results obtained are emphasized below.

Table 5 Character count and time taken to encrypt and decrypt

Character count	Encryption time (ms)	Decryption time (ms)
4	47	15
8	47	15
16	47	15
32	47	15
64	47	15
128	62	21
256	78	46
512	125	63
1024	202	125
2048	437	312
4096	905	826
8192	4227	3183
16,384	15,194	11,544

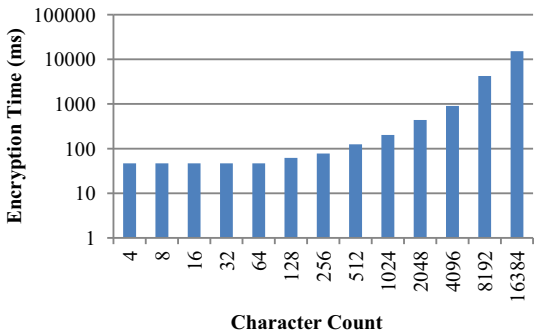

Fig. 9 DNA—character count—encryption time

4.2.1 Time taken to encrypt and decrypt a range of characters

For the characters of varying count, the encryption time and decryption time is computed as shown in Table 5.

From the results, it is found that the encryption time increases linearly with increase in the count of characters, but the time taken to decrypt is lesser than the encryption time. Similarly, when the character count increases, the time taken for encryption and decryption also increases. It proves that the computational complexity is less, as shown in Figs. 9 and 10.

4.2.2 Time taken to encrypt and decrypt a range of words

The encryption and decryption time for the corresponding word count is shown in Table 6.

The time taken to decrypt is lesser than the time taken to encrypt thus enables faster retrieval of the original data by the Data User in the cloud (Fig. 11).

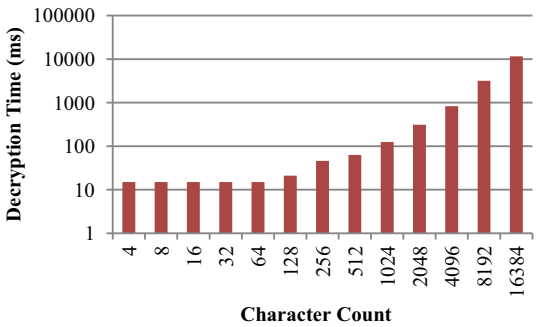

Fig. 10 DNA—character count—decryption time

Table 6 Word count and time taken to encrypt and decrypt

Word count	Encryption time (s)	Decryption time (s)
4	0.062	0.015
8	0.062	0.015
16	0.063	0.031
32	0.078	0.031
64	0.109	0.072
128	0.125	0.078
256	0.49	0.43
512	0.5	0.4
1024	2	1.6
2048	7	6
4096	27	25
8192	118	114
16,384	483	330

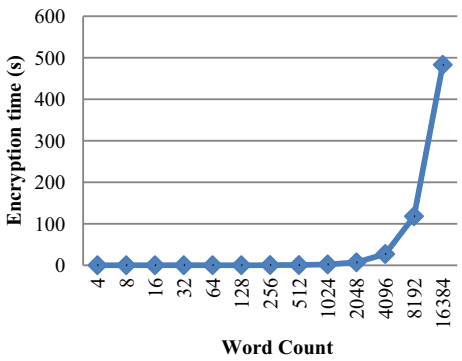

Fig. 11 DNA—word count—encryption time

4.2.3 Impact of block size

The length of the ciphertext varies for the number of bits per block of plaintext. The plaintext bits are fixed in 256. Thus, the change in block size hides the plaintext in different ciphertext with different length enabling better security as shown in Table 7 (Fig. 12).

Table 7 Ciphertext length for corresponding plaintext varying in block size

Plaintext length (count of characters)	Block Size (in bits)	Ciphertext length (count of characters)
256	16	516
	32	520
	64	528
	128	544
	256	576
	512	640
	1024	768

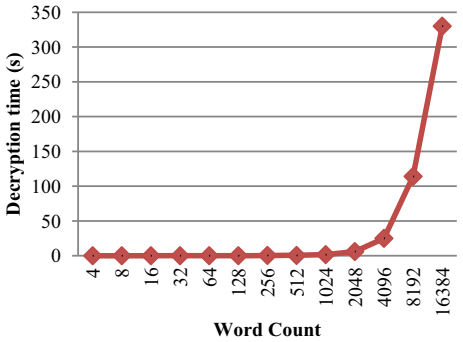

Fig. 12 DNA—word count—decryption time

Table 8 Comparison of files size

Plaintext file size (KB)	Key file file size (KB)	Ciphertext file size (KB)
4	0.028	8.0
8	0.028	16.1
16	0.028	32.2
32	0.028	64.6
64	0.028	129.0

4.2.4 Comparison of file size

The key file data is constant in its structure and so the file size has not been changed, whereas the file size of ciphertext is linearly increased. It supports in optimal space utilization, reducing the space complexity as shown in Table 8.

Thus, the tradeoff between time and space complexity is well balanced without compromising security.

4.2.5 Frequency analysis

The frequency analysis of the ciphertext reveals the correlation between the ciphertexts compared. It should be distinct from the fact that no two ciphertexts are the same for the given plaintext. This minimal correlation decreases the chances of breaking the ciphers. It is achieved with the factors like Data

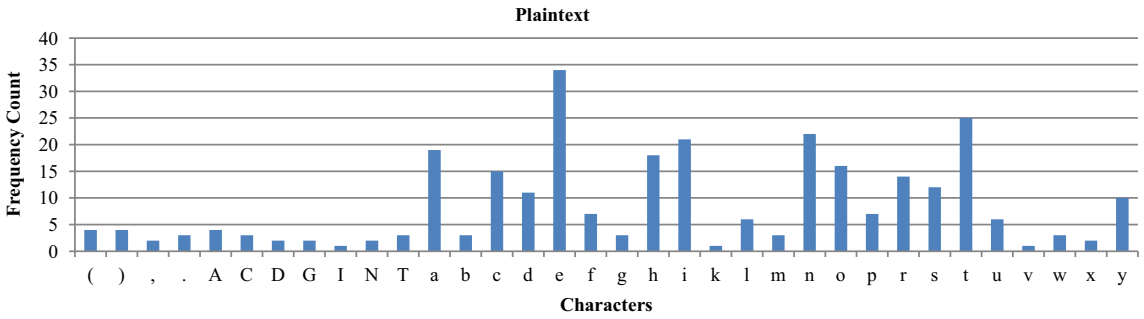

Fig. 13 Frequency of plaintext characters

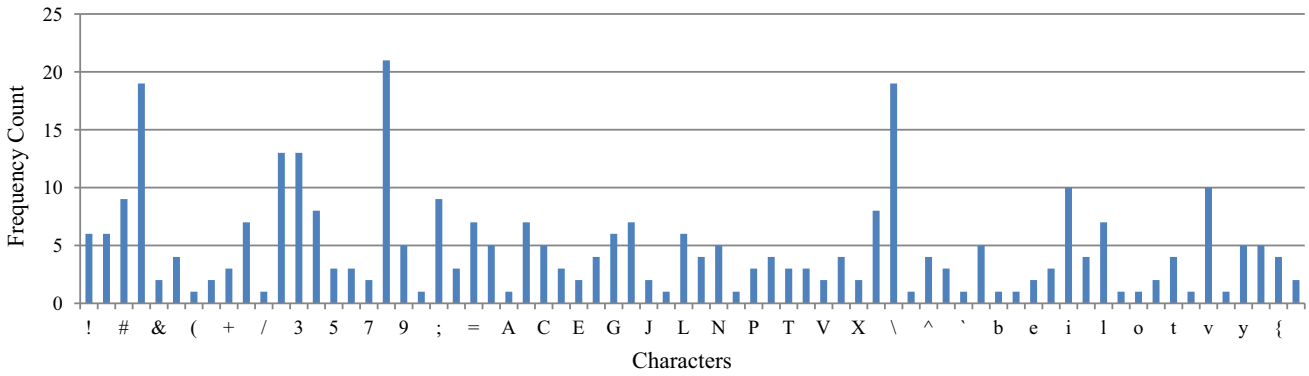

Fig. 14 Ciphertext 1—ciphertexts generated using same encoding tables

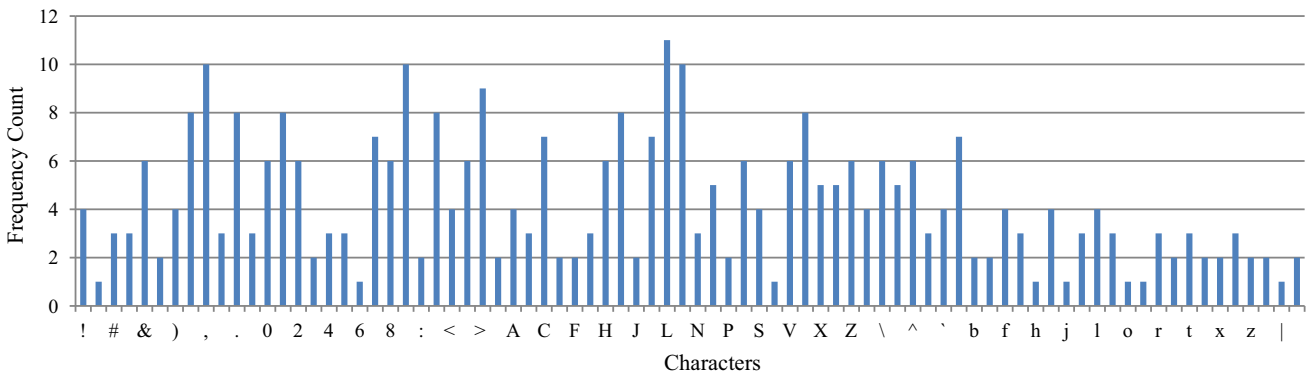

Fig. 15 Ciphertext 2—ciphertexts generated using same encoding tables

Owner's sequence, Data User's sequence, Intron sequence, Encoding Table and Collating values of our framework.

The plaintext taken for frequency analysis is:

DNA Cryptography is the secret to achieve faster and highly robust encrypted communication. The four nucleotides Adenine (A), Thymine (T), Guanine (G) and Cytosine (C) are the backbone of DNA cryptography which hides the entire data within itself and exposes only few ciphertext characters. It enhances confidentiality of the data in cloud.

The frequency of the characters occurring in the plaintext is shown in Fig. 13. This plaintext is kept fixed, whereas the other factors are made dynamic to analyze the correlation among the varying ciphertexts. The frequency of the characters occurrence rather than the occurrence of mapped special characters is displayed.

For all the upcoming graphs, the x-axis value represents the characters and y-axis value represents the frequency count.

(a) Ciphertexts generated using same encoding tables

The ciphertext generated for the same plaintext and the same encoding table is analyzed as shown in Figs. 14 and 15. From the results obtained, it is proved that the ciphertexts are not correlated to each other.

(b) Ciphertexts generated using different intron sequences

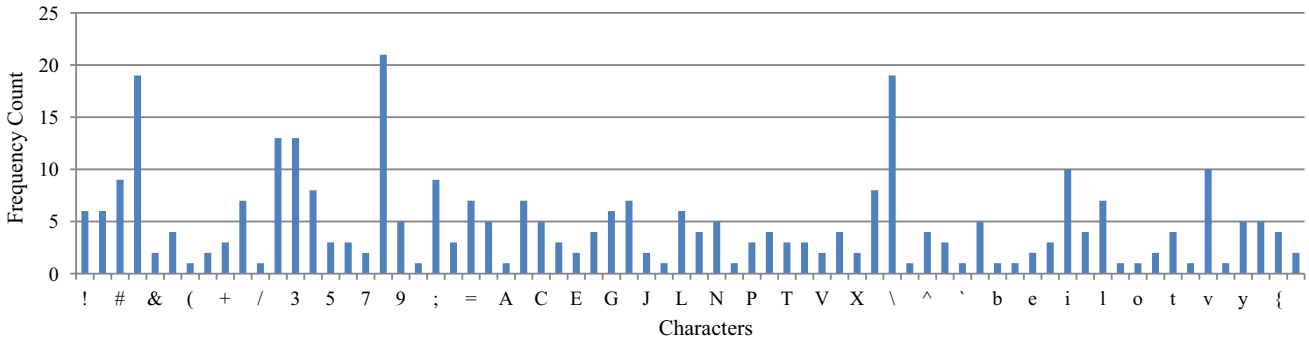

Fig. 16 Ciphertext 1—ciphertexts generated using different intron sequences

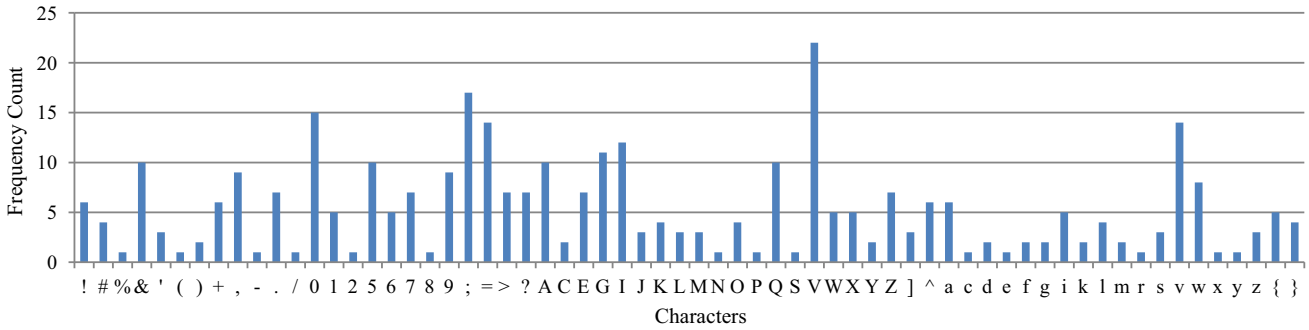

Fig. 17 Ciphertext 2—ciphertexts generated using different intron sequences

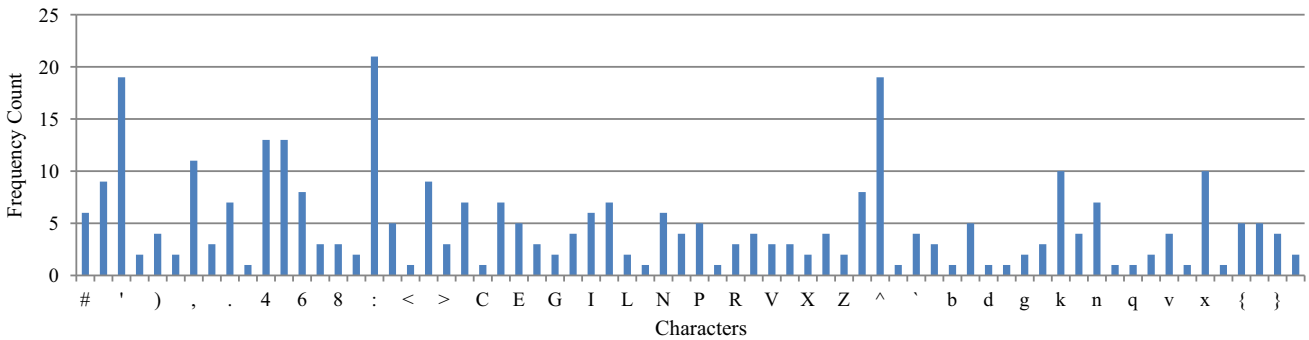

Fig. 18 Ciphertext 1—ciphertexts generated using different collating values

The graphs in Figs. 16 and 17, shows the correlation among the two ciphertexts generated with different intron sequences but having the remaining values as same.

(c) Ciphertexts generated using different collating values

When the collating values are varied, keeping the other values same, the ciphertext differs as shown in Figs. 18 and 19.

(d) Ciphertexts generated using different data owner sequence and data user sequence

When the sequences of the Data Owner and the Data User are changed, the ciphertext eventually changes leading to the dynamic encryption process. It is shown in Figs. 20 and 21.

(e) Ciphertexts generated using different sequences and collating values

When all the dynamic factors are changed with their values, the ciphertext ultimately changes as shown in Figs. 22 and 23.

(f) Ciphertexts generated for different plaintext having same sequences and collating values

Here, two different plaintexts are considered with the varying frequency of occurrence of characters as shown in Figs. 24 and 26. Though the sequences and the collating values are kept same, but still it produces two different ciphertexts as shown in Figs. 25 and 27. Thus, cryptanalysis is quite harder to perform.

(g) Ciphertexts generated for different plaintext having different sequences and collating values

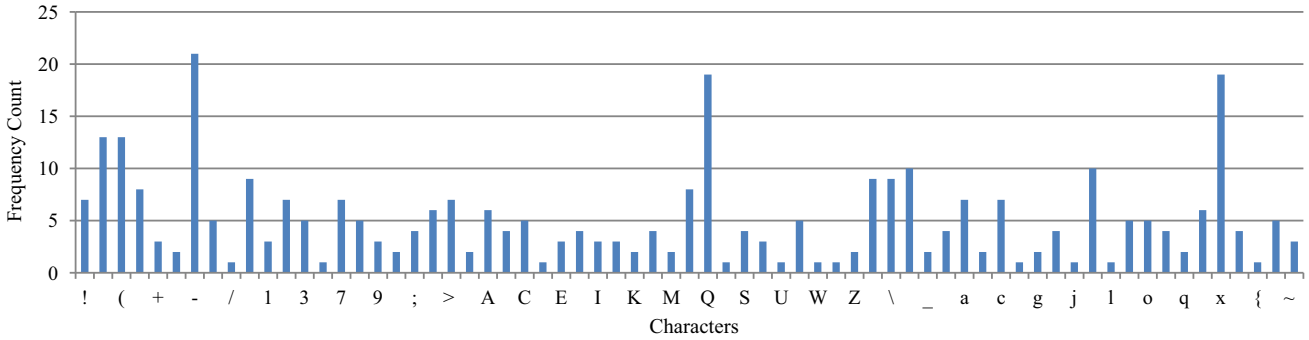

Fig. 19 Ciphertext 2—ciphertexts generated using different collating values

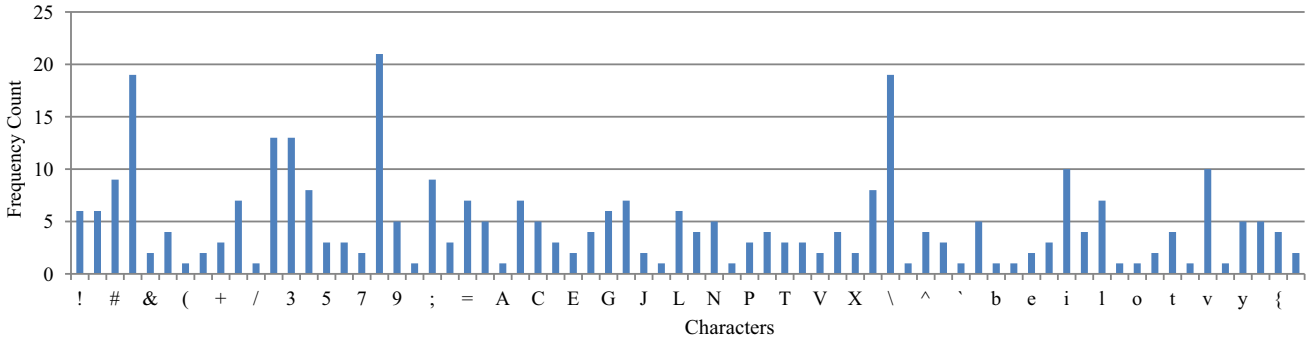

Fig. 20 Ciphertext 1—ciphertexts generated using different data owner sequence and data user sequence

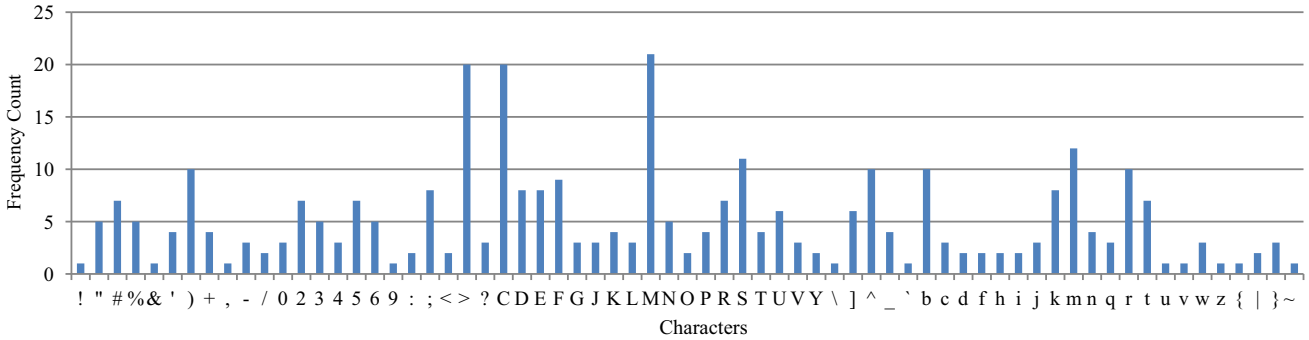

Fig. 21 Ciphertext 2—ciphertexts generated using different data owner sequence and data user sequence

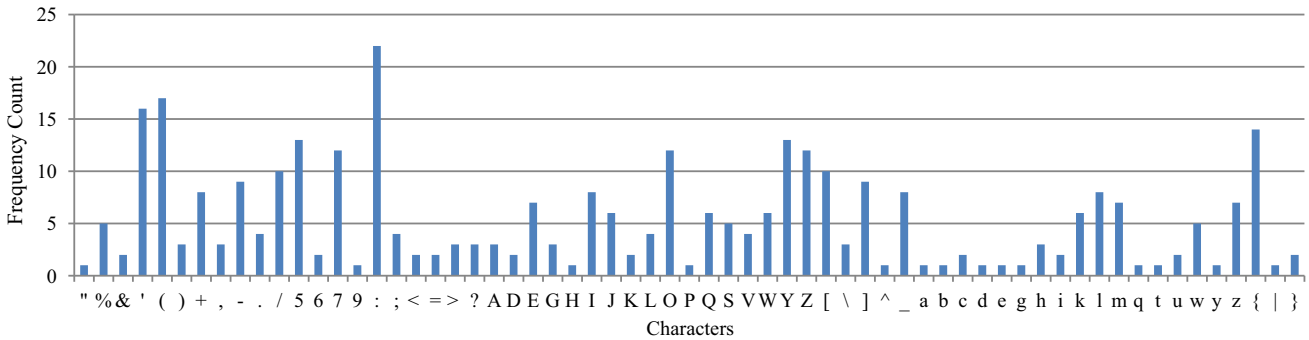

Fig. 22 Ciphertext 1—ciphertexts generated using different sequences and collating values

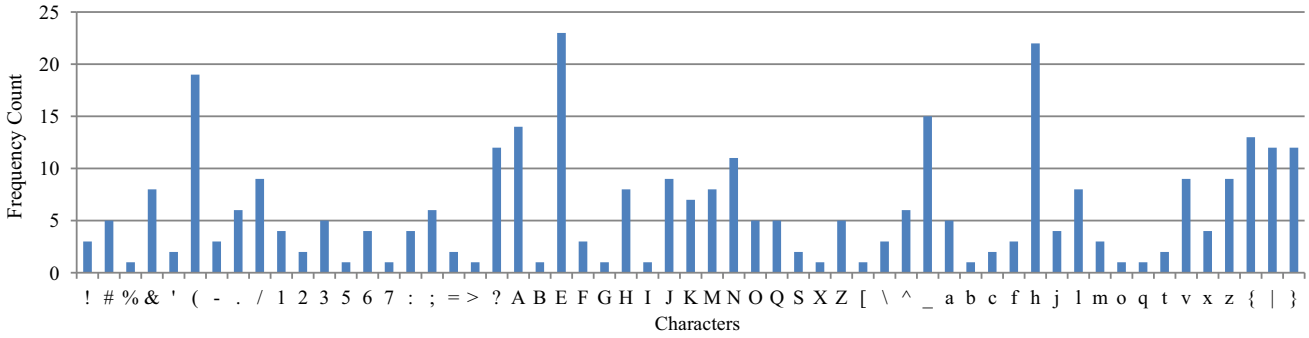

Fig. 23 Ciphertext 2—ciphertexts generated using different sequences and collating values

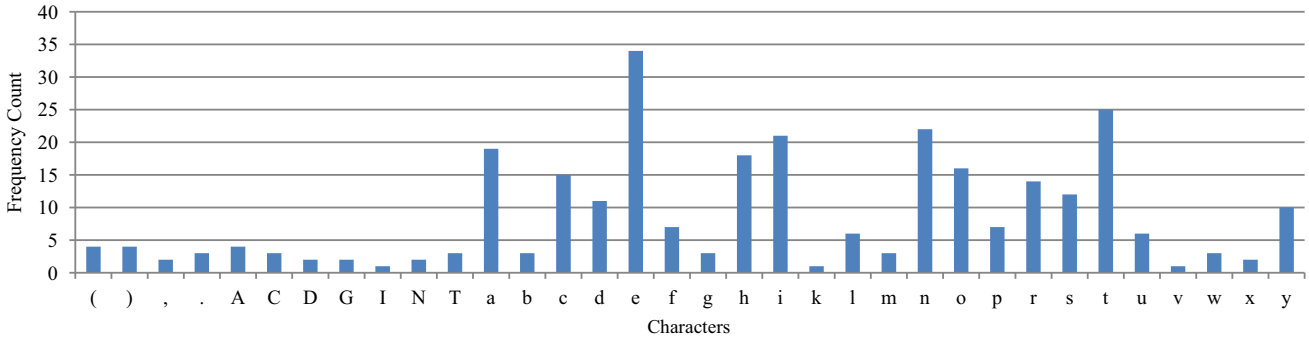

Fig. 24 Plaintext 1—ciphertexts generated for different plaintext having same sequences and collating values

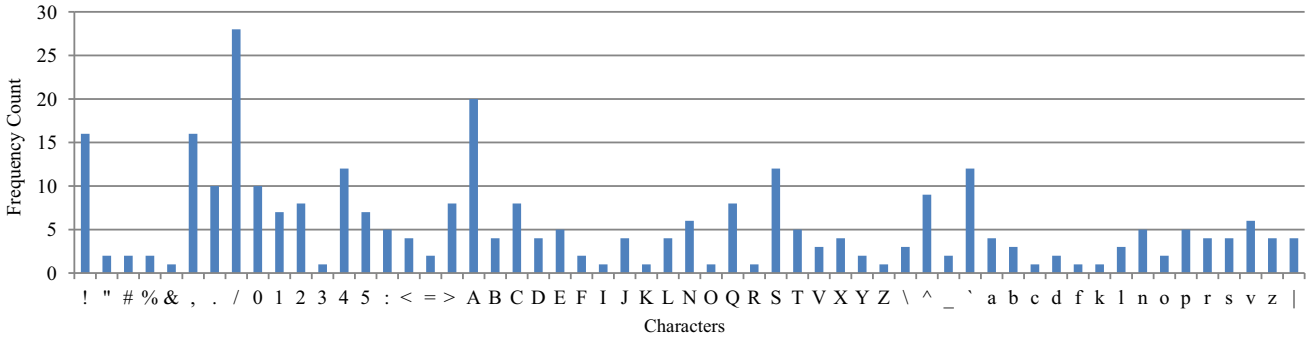

Fig. 25 Ciphertext 1—ciphertexts generated for different plaintext having same sequences and collating values

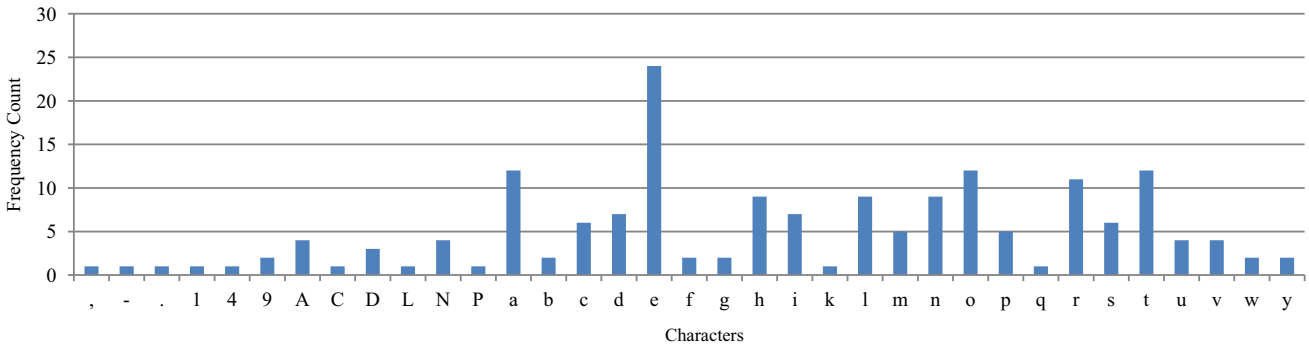

Fig. 26 Plaintext 2—ciphertexts generated for different plaintext having same sequences and collating values

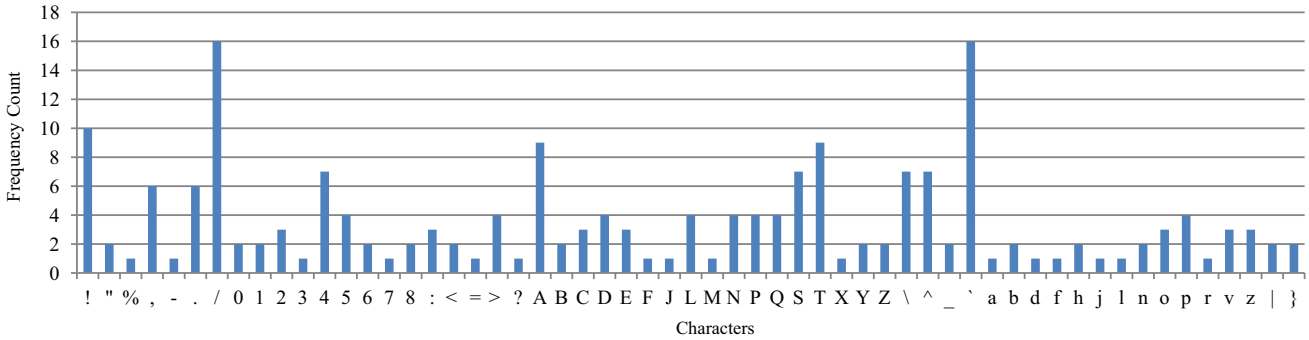

Fig. 27 Ciphertext 2—ciphertexts generated for different plaintext having same sequences and collating values

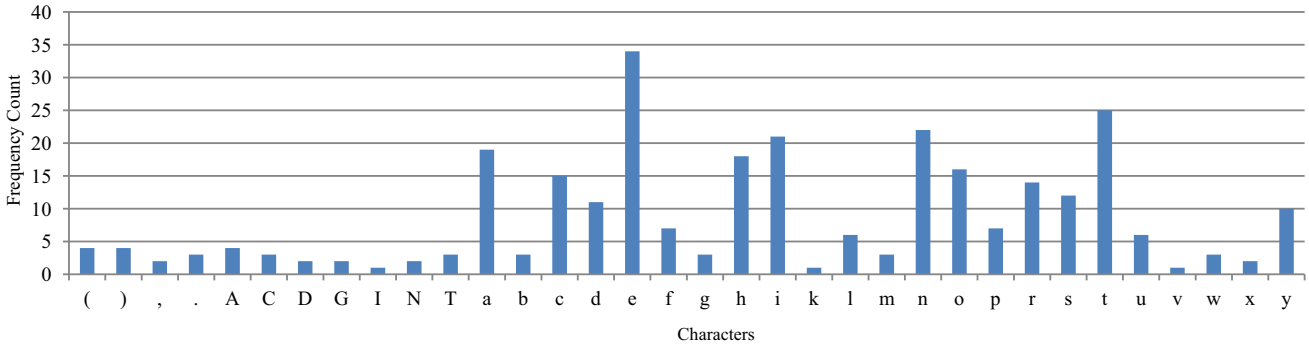

Fig. 28 Plaintext 1—ciphertexts generated for different plaintext having different sequences and collating values

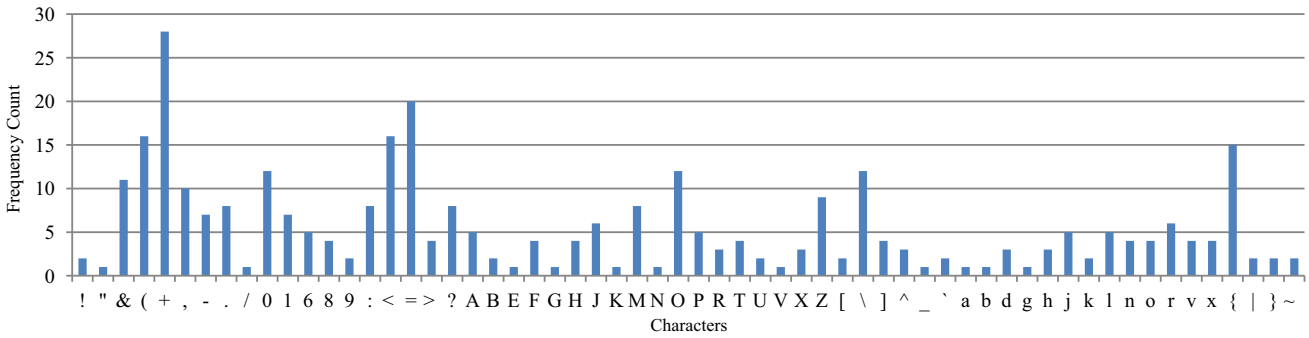

Fig. 29 Ciphertext 1—ciphertexts generated for different plaintext having different sequences and collating values

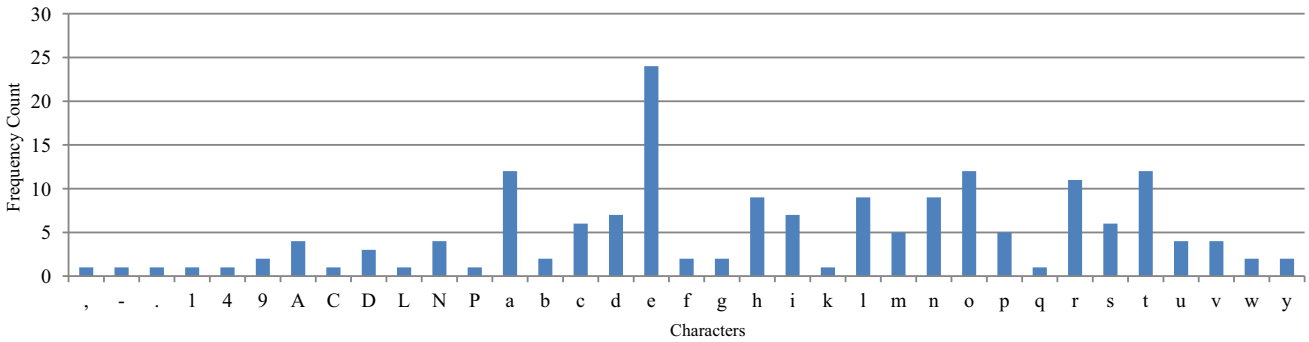

Fig. 30 Plaintext 2—ciphertexts generated for different plaintext having different sequences and collating values

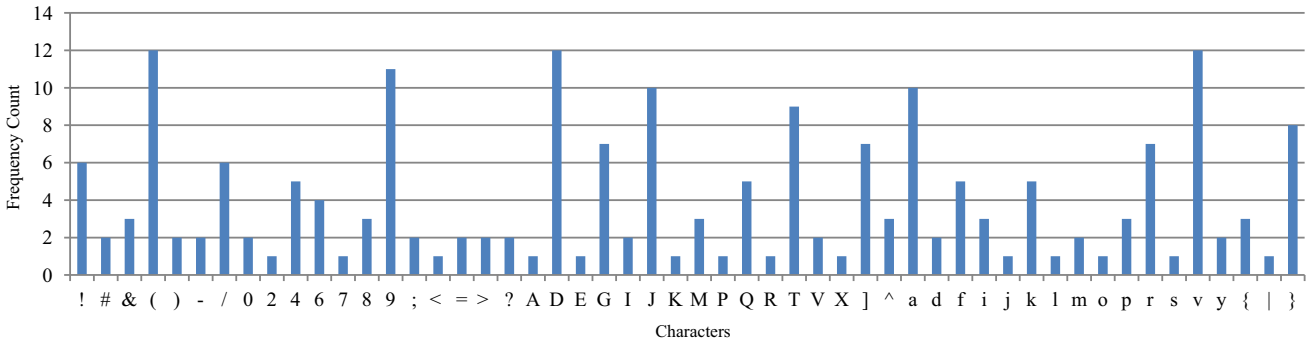

Fig. 31 Ciphertext 2—ciphertexts generated for different plaintext having different sequences and collating values

When the two different plaintexts as shown in Figs. 28 and 30 with different sequences and different collating values are performed encryption, the two different ciphertexts are generated as shown in Figs. 29 and 31.

5 Security analysis

The need to provide security to the data in a cloud environment has the equal need and importance to protect it from recovering the plaintext through cryptanalysis. The security analysis for both the proposed cryptosystems are as follows:

5.1 Novel DNA cryptosystem

In specific to DNA cryptosystem, security can be analyzed through the six properties as metrics and they are,

5.1.1 Complete character set encoding

The encoding table generated provides sequences to be encoded with the complete character set. The character set contains all the 94 ASCII characters which are extended to 256 elements by prefixing the first set of ASCII characters with the alphabet 'D', the second set of ASCII characters with the alphabet 'N' and the remaining characters are prefixed with the alphabet 'A'. Thus, unique sequences are encoded with a unique character set.

5.1.2 Dynamic encoding table generation

The Encoding table has to be dynamic since the plaintext has to be transformed into different ciphertext for every access of data in the cloud. It has been fulfilled by using distinct values for Data Owner sequence, Data User sequence, collating values for the encoding table and amino acid table for every access of data in the cloud. The encoding table is used only once for a particular session between the Data Owner and the Data User enabling confidentiality in the cloud.

5.1.3 Unique sequence for character encoding

Each sequence is encoded with a unique character and thereby it prevents from security issues like cipher attacks and frequency analysis. The uniqueness of encoding is supported for every generation of encoding tables in our algorithmic approach.

5.1.4 Robustness of encoding

It is strengthened by the randomness involved in the usage of intron sequence, collating values and encoding table generation. Thus, it makes harder to perform cryptanalysis.

5.1.5 Biological process simulation

Our algorithm inherits the major biological processes of DNA such as transcription (conversion of DNA to mRNA sequence), translation of DNA to amino acid sequence, complementary pairs of DNA and it is also exhibited in encryption process as well as decryption process.

5.1.6 Dynamic encryption process

The process of encryption is made dynamic by the encoding table, unique values given for the encryption process resulting in unique generation of ciphertext. The ciphertext generated is eventually distinct for every encryption process carried out. Thus, every requirement has been met out in our algorithm to strengthen our framework.

In the Table 9, by considering cryptosystem factors, the enhanced DNA cryptosystem has been compared with AES Symmetric cryptosystem and RSA Asymmetric cryptosystem.

The inference from the Table 9: Enhanced DNA Cryptosystem rounds and key file can be decided by the users before the communication. The small variants in the key file reflects an major changes in the ciphertext generation. With the minimal key size, Enhanced DNA cryptosystem

Table 9 Factors comparison of EDNAC, AES and RSA cryptosystems

Factors	Enhanced DNA cryptosystem	AES—symmetric cryptosystem	RSA—asymmetric cryptosystem
Key size	104 bits	128/192/256 bits	128/256/512/1024/2048 bits
Ciphering and deciphering key	Same	Same	Different
Key used	Minor variations in key causes complete change in the ciphertext.	Changes completely	Changes completely
Algorithm	Partially symmetric and asymmetric type	Symmetric type	Asymmetric type
Encryption	Faster	Moderate	Slower
Decryption	Faster	Moderate	Slower
Power consumption	Very low	Low	High
Security	High secure	High secure	Less secure
Deposit of keys	Needed	Needed	Needed
Inherent vulnerabilities	Brute force attack	Brute force attack	Brute force, Timing and Oracle attack
Rounds	Not limited to 1 round, depends on the users	10/12/14	1
Trojan horse	No	Not proved	No
Ciphering and Deciphering algorithm	Same	Different	Same

Table 10 Performance of EDNAC, AES and RSA cryptosystems

Word count	EDNAC		AES		RSA	
	E (s)	D (s)	E (s)	D (s)	E (s)	D (s)
4	0.062	0.015	0.2	0.1	0.9	0.6
8	0.062	0.015	0.3	0.2	1.3	1.0
16	0.063	0.031	0.4	0.3	1.5	1.2
32	0.078	0.031	0.6	0.5	1.8	1.4
64	0.109	0.072	0.8	0.5	2.2	1.8
128	0.125	0.078	1.3	0.8	4.7	3.5
256	0.49	0.43	1.5	1.1	9.5	6.4
512	0.5	0.4	1.8	1.3	11.5	8.5
1024	2	1.6	2.5	1.7	18.7	10.7

E encryption time, *D* decryption time

provides very high security for the data transmission and storage.

Table 10 illustrates the performance comparison between enhanced DNA, AES—symmetric and RSA—asymmetric cryptosystems for the variable word counts in the file.

The inference from the Table 10: compared to AES and RSA cryptosystems, enhanced DNA cryptosystem are computationally fast. The time complexity is less compared to symmetric and asymmetric standard cryptosystems.

5.2 Enhanced ElGamal cryptosystem

The security of the Enhanced ElGamal cryptosystem is analyzed against chosen plaintext attack, chosen ciphertext attack and brute force attack.

5.2.1 Chosen plaintext attack

The EEC algorithm provides more security against Chosen Plaintext Attack (CPA) than ElGamal cryptosystem, which is due to the fact that the proposed algorithm involves two random integers in order to compute encryption key.

In order to apply CPA on the proposed cryptosystem, the adversary chooses an arbitrary m and having access to the encryption oracle obtains the corresponding Ciphertext C. The adversary wins, if the assumption on C is correct. Table 9 is based on the assumption that the adversary's guess on k_1, k_2, k_3, β, d and X are correct. Recall that in EEC, $C_1 = \alpha^{k_3} \bmod q$, $C_2 = k_1^{k2} \bmod q$ and $C_3 = K.m.Y \bmod q$, where m is chosen by the adversary at random. Now chosen that $Y = (\alpha.\beta)^a \bmod q$, $C_1 = \alpha^c \bmod q$ and $C_2 = b^d \bmod q$, where β, a, b, c, and d are taken at random. C_3 is chosen at random, but gives a valid encryption of m as $C_3 = (b)^d .Y^c.m.Y \bmod q$

Now $K = \frac{C_3}{m.Y} = \frac{(b)^d.Y^c.m.Y}{m.Y} = (b)^d .Y^c$

Using the derived key K, the adversary can access the encryption oracle to encrypt the chosen m and obtains C_3.

Table 11 Performance against chosen plaintext attack

KeySize—size of q (in bits)	ElGamal cryptosystem (in s)	Enhanced ElGamal cryptosystem (in s)
2	0.353	0.86
4	0.412	0.991
8	0.582	1.231
16	0.62	1.8
32	1.25	45.89
64	32.72	357.15
128	628.65	7996.57
256	3247.12	35781.54
512	52558.25	486375.98
1024	6662587.14	15987332.74
2048	258963147.27	3579514574.22

Table 12 Performance against chosen ciphertext attack

KeySize—size of q (in bits)	ElGamal cryptosystem (in s)	Enhanced ElGamal cryptosystem (in s)
2	0.519	1.379
4	0.811	1.615
8	1.22	2.891
16	1.836	3.34
32	8.15	98.54
64	56.14	789.45
128	6547.12	35894.35
256	12355.25	753158.24
512	417224.64	9852348.45
1024	5714625.11	77588965.46
2048	95214478.21	485585446.74

The adversary wins if the guess on C_3 is correct. The major difficulty faced by the adversary is to predict all keys X, β, d, k_1, k_2 and k_3 correctly as involved in the communication. Since, all the values are in the limit q-1 and q is the large prime number, it is too difficult for the adversary to predict the key values.

The implementation results of performing CPA on the ElGamal cryptosystem and Enhanced ElGamal cryptosystem are shown in Table 11 shows that it is competitively difficult to perform CPA on Enhanced ElGamal cryptosystem.

5.2.2 Chosen ciphertext attack

The proposed algorithm also provides more security against Chosen Ciphertext Attack (CCA), where it consumes more time compared to ElGamal Cryptosystem. Applying CCA on the proposed cryptosystem involves that the adversary chooses an arbitrary C' (say $2C$) which is related to a Ciphertext C. The adversary is able to access the decryption oracle, but not able to request the decryption of C. The adversary provides the decryption oracle with C' and obtains m'(say $2m$). From which the adversary retrieves original plaintext m. The adversary computes C' as (C_1, C_2, $2C_3$) and accesses the decryption oracle to decrypt C' and obtains $2m$. Then the adversary computes $\frac{m}{2}$ which gives m as,

$$\begin{aligned}
\frac{m}{2} &= \frac{2C_3.C_1^{-X}.C_2^{-1}.\beta^{-k_3.X}.d^X}{2} \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3.X}.C_3.d^X \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3.X}.K.m.Y.d^X \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3.X}.k_1^{k_2}.Y^{k_3}.m.Y.d^X \\
&= \alpha^{-k_3.X}.k_1^{-k_2}.\beta^{-k_3.X}.k_1^{k_2}.\alpha^{k_3.X}.\beta^{k_3.X}.m.Y.d^X \\
&= m.Y.d^X \\
&= m.\alpha^X.\beta^X.\alpha^{-X}.\beta^{-X} \quad = \quad m
\end{aligned}$$

The major difficulty faced by the adversary is to predict all the keys and plaintext values correctly as involved in the communication. Since all the values are at the limit of q, which is the large prime number; it is too difficult for the adversary to predict all the parameters involved in the encryption and decryption. The implementation results of performing CCA on the ElGamal Cryptosystem and Enhanced ElGamal Cryptosystem are shown in Table 12 shows that it is competitively difficult to perform CCA on Enhanced ElGamal cryptosystem. For a secured communication, the encryption and decryption key will be computed periodically. If one time the key is generated, it can be utilized several times for the purpose of encryption and decryption. In order to increase the security, the key generation of EEC has been introduced with randomness to bring the complexity for the attacker.

5.2.3 Brute force attack

For brute force attack, in ElGamal cryptosystem, the private key {X} value alone needs to be tried at random to obtain the plaintext. In Enhanced ElGamal cryptosystem, private keys {X, β, d} and secret key k_3 need to try as combinations to obtain the plaintext. Analyzing the EEC algorithm for brute force attack it results in better performance, which is shown in Table 13. From the table, it is very clear that the time was taken to apply brute force attack on EEC is far greater than the time taken to apply brute force attack on ElGamal Cryptosystem. So, EEC is comparatively difficult to break. As the key size increases, brute force attacking time is also increased for EEC compared to the ElGamal cryptosystem.

Table 13 Brute force attack time

KeySize—size of q (in bits)	Time taken to do brute force attack	
	ElGamal (in s)	Enhanced ElGamal (in s)
2	0.164	100.282
4	0.253	192.378
8	10.502	287.184
16	35.981	496.721
32	451.975	5877.016
64	6548.57	78952.015
128	15789.17	125756.85
256	3557885.52	7532159.47
512	74588966.12	85245667.98
1024	654412544.45	7531598565.24
2048	4521896321.57	44887566354.45

Moreover, for security purposes, normally key size will be higher for the communication.

Since EEC scheme based on the discrete logarithm problem and randomness, it is very difficult for an unauthorized user to compute the private key X from the equation $Y = (\alpha.\beta)^X \bmod q$. It is also difficult to find the two random numbers k_1 and k_2 from the encryption equations $C_2 = k_1^{k2} \bmod q$ and $K = (k_1)^{k_2} .Y^{k_3} \bmod q$. The difficulty of the cryptanalysis relies on solving discrete logarithm problem.

Even though, if the intruder solves the discrete logarithm problem, it is computationally infeasible to break the EEC unless primitive β and d values are obtained. The primitive root β has been selected purely based on the randomness. Based on primitive roots α, β, the value of 'd' is calculated. The security of EEC lies both on the discrete logarithm problem and randomness. Since the prime number 'q' is large, it is difficult to succeed in identifying exact random values for cryptanalysis. So the amount of time taken to break EEC by solving the discrete logarithm problem and randomness is too high compared to ElGamal cryptosystem. Thus, EEC is secure against chosen plaintext attack, chosen ciphertext attack and brute force attack. From the literature survey, it is prominent that all the research works challenges to increase the throughput. But, we considered security as a major factor for the improvisation of ElGamal cryptosystem.

6 Conclusion

In this paper, an Enhanced ElGamal cryptosystem is proposed. The proposed work improvises the randomization for key generation, encryption, and decryption from the ElGamal cryptosystem. Consequently, for the proposed algorithm, key generation is a time-consuming one, since it will be done periodically, it is tolerable. And also it proves the user authentication of the Data Owner and the Data User thereby resulting in secure transfer of the key file between the Data Owner and the Data User. From the experiments, it is proved that the system is highly secure and hard to perform a brute force attack and cryptanalysis attacks like CPA and CCA as compared to ElGamal cryptosystem. And EEC security relies on the difficulty of randomness and the discrete logarithm problem. Similarly, the proposed DNA cryptosystem provides data confidentiality for the data transferred between the Data Owner and the Data User in a cloud environment. The DNA nucleotides are used to completely hide the original data for a secure communication. The dynamic generation of encoding table and intron sequence reduces the possibility of cryptanalysis and also enhances the security of data. The biological properties of DNA make the system yet more randomized and a prudent system as well as becomes the efficient system in practice while most of the DNA cryptosystems are theoretical. The possibility for an attack on the cloud environment for cryptanalysis is hard due to the dynamicity of our proposed cryptosystem. Thus, the proposed hybrid cryptosystems are novel as well as efficient in terms of performance and security. Further, on implementing the proposed framework with both the cryptosystems for real-time applications could result in future enhancements towards the efficiency of the system.

References

1. Pant, V., Kumar, A.: DNA cryptography an new approach to secure cloud data. Int. J. Sci. Eng. Res. **7**(6) (2016)
2. Arockiam, L., Monikandan, S.: Efficient cloud storage confidentiality to ensure data security. In: IEEE International Conference on Computer Communication and Informatics (ICCCI) (2014)
3. Chandramouli, R., Iorga, M., Chokhani, S.: Cryptographic Key Management Issues & Challenges in Cloud Services. National Institute of Standards and Technology. U.S, Department of Commerce (2013)

4. European Payment Council: Guidelines on Cryptographic Algorithms Usage and Key Management. https://www.europeanpaymentscouncil.eu/document-library/guidance-documents/guidelines-cryptographic-algorithms-usage-and-key-management
5. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science **266**, 1021–1024 (1994)
6. Jain, S., Bhatnagar, V.: A novel DNA sequence dictionary method for securing data in DNA using spiral approach and framework of DNA cryptography. In: IEEE, ICAETR, pp. 1–5 (2014)
7. Ubaidur Rahman, N.H., Balamurugan, C., Mariappan, R.: A novel DNA computing based encryption and decryption algorithm. In: Proceedia Computer Science, International Conference on Information and Communication Technologies, pp. 463 – 475 (2015)
8. Stallings, W.: Cryptography and Network Security Principles and Practice, 5th edn. Pearson Education (2013)
9. Agarwala, A., Saravanan, R.: A public key cryptosystem based on number theory. 978-1-4673-0255-5/12, IEEE (2012)
10. ElGamal, T.: A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
11. Boneh, D., Joux, A., Nguyen, P.: Breaking plain ElGamal and plain RSA encryption. https://www-almasty.lip6.fr/~joux/pages/papers/PlainRSA.pdf
12. Bryce, D.A.: Implementing several attacks on plain ElGamal encryption. http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=2577&context=etd
13. Katz, J., Lindell, Y.: Introduction to Modern Cryptography (Chapter 9), 2nd edn. CRC Press, Boca Raton (2016)
14. McCurley, K.: The discrete logarithm problem. In: Proceedings of Symposia in Applied Mathematics, vol. 42 (1990)
15. Hwang, M.-S., Chang, C.-C., Hwang, K.-F.: An ElGamal-like cryptosystem for enciphering large messages. IEEE Trans. Knowl. Data Eng. **14**(2), 445–446 (2002)
16. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inform. Theory **22**(6), 644–654 (1976)
17. Mohit, P., Biswas, G.P.: Design of ElGamal PKC for encryption of large messages. In: INDIACom, International Conference. IEEE (2015)
18. Sharma, P., Sharma, S, Dhakar, R.S.: Modified elgamal cryptosystem algorithm (MECA). In: International Conference on Computer & Communication Technology, pp. 439–443 (2011)
19. Asbullah, M.A., Ariffin, M.R.K.: A proposed CCA-secure encryption on an ElGamal variant. In: 7th International Conference on Computing and Convergence Technology (ICCCT), pp. 499–503 (2012)
20. Nguyen, M.T., Nguyen, B.: Some hybrid crypto systems constructed on discrete logarithm problem. In: ATC, International Conference. IEEE (2015)
21. Hashim, H.R., Neamaa, I.A.: Image encryption and decryption in a modification of ElGamal cryptosystem in MATLAB. Int. J. Sci. **14**(2), 141–147 (2014)
22. Alam, K., Alam, K.R., Faruq, O., Morimoto, Y.: A comparison between RSA and Elgamal based untraceable blind signature schemes. In: NSysS, International Conference. IEEE (2016)
23. Prajapati, A., Barkha, P.: Implementation of DNA cryptography in cloud computing and using socket programming. In: ICCCI. IEEE (2016)
24. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
25. Malhotra, M.: A new encryption scheme based on enhanced RSA and ElGamal. Int. J. Emerg. Technol. Comput. Appl. Sci. (IJETCAS) **8**(2), 138–142 (2014)
26. Madhur, K., Yadav, J.S., Vijay, A.: Modified ElGamal over RSA digital signature algorithm (MERDSA). Int. J. Adv. Res. Comput. Sci. Softw. Eng. **2**(8), 289–293 (2012)
27. Wang, H., Sun, Z.: Study on the improvement of ELGamal cryptosystem based on elliptic curve. J. Netw. **9**(11), 3025–3029 (2014)
28. Dawahdeh, Z.E., Yaakob, S.N., Sagheer, A.M.: Modified ElGamal elliptic curve cryptosystem using hexadecimal representation. Indian J. Sci. Technol. **8**(15) (2015)
29. Gennaro, R.: Randomness in cryptography. IEEE Secur. Priv. **4**(2), 64–67 (2006)
30. Yunpeng, Z., Yu, Z., Zhong, W., Sinnott, R.O.: Index-based symmetric DNA encryption algorithm. In: Image and Signal Processing (CISP), pp. 2290–2294 (2011)
31. Abbasy, M.R., Nikfard, P., Ordi, A., Torkaman, M.R.N.: DNA base data hiding algorithm. In: IJNCAA, pp. 183–192 (2012)
32. Gao, Q.: A few DNA based security techniques. In: Systems, Applications and Technology Conference, pp. 1–5. IEEE, Long Island (2011)
33. Dhawan, S., Saini, A.: A new DNA encryption technique for secure data transmission. In: IJETCAS, pp. 36–42 (2012)
34. Mandge, T., Choudhary, V.: A DNA encryption technique based on matrix manipulation and secure key generation. In: ICICES, pp. 47–52. IEEE (2013)
35. Majumder, A., Majumdar, A., Podder, T., Kar, N., Sharma, M.: DNA-based cryptographic approach toward information security. In: Advances in Intelligent Systems and Computing, vol. 308, pp. 209–219. Springer (2015)
36. Aich, A., Sen, A., Dash, S.R., Dehuri, S.: A symmetric key cryptosystem using DNA sequence with OTP key. In: Advances in Intelligent Systems and Computing, pp. 207–215 (2015)
37. Majumder, A., Majumdar, A., Podder, T., Kar, N., Sharma, M.: Secure data communication and cryptography based on DNA based message encoding. In: ICACCCT. IEEE (2014)
38. Cui, G., Han, D., Wang, Y.: An improved method of DNA information encryption. In: BIC-TA, CCIS, pp. 73–77. Springer (2014)
39. Sundaram, G.S., Pavithra, S., Arthi, A., Bala, B.M., Mahalakshmi, S.: Cellular automata based DNA cryptography algorithm. In: ISCO. IEEE (2015)
40. Aich, A., Sen, A., Dash, S.R., Dehuri, S.: Deoxyribonucleic acid (DNA) for a shared secret key cryptosystem with Diffie hellman key sharing technique. In: CCCIT, pp. 1–6 (2015)
41. Gugnani, G., Ghrera, S.P., Gupta, P.K., Malekian, R., Maharaj, B.T.J.: Implementing DNA . In: Advances in Intelligent Systems and Computing, vol. 381. Springer (2016)
42. Thangavel, M., Varalakshmi, P., Sindhuja, R.: A Comparative study on DNA cryptosystem. In: IEEE International Conference on Recent Trends in Information Technology (ICRTIT) (2016)

M. Thangavel is an Assistant Professor, Department of Information Technology at Thiagarajar college of Engineering, Madurai. He is pursuing Ph.D. under the faculty of Information and Communication, Anna University. He completed his M.E. in Computer Science & Engineering at J.J College of Engineering. His research interests include Cloud Computing, Information Security, Cryptography, Ethical Hacking, Compiler Design and Data Structures.

P. Varalakshmi is an Associate Professor at Department of Computer Technology, Madras Institute of Technology, Anna University. She received her Ph.D. degree under Information & Communication Engineering-Trust Management in Grid Computing at Anna University. Her research interests include Compiler Design, Network security in Grid and Cloud Computing. She has published many research papers in international and national journals and conference proceedings with very high citation index of about 25 papers so far.