# DNA Cryptography

Aanya Shantaram
220953438
Department of I&CT
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal- 576104, Karnataka, India
Email: aanya.shantaram@gmail.com

Ritika Salimath
220953428
Department of I&CT
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal- 576104, Karnataka, India
Email: ritika.salimath@gmail.com

Alok Kumar
220953464
Department of I&CT
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal- 576104, Karnataka, India
Email: alx989449@gmail.com

***Abstract*—This paper presents a methodology of cryptographic data security that is inspired by DNA based encryption and decryption. In this method, the techniques used, such as XOR-based encryption, transcription, translation, DNA sequence shifting and intron insertion, are biologically motivated instead of the classical cryptography method. To increase the security aspect, key generation and key exchange have been done with RSA encryption. The experimental analysis involves performance measurement and security criteria evaluation; it reflects that this method is efficient and secure and time-efficient in the process of sensitive data transmission.**

**Keywords: DNA, cryptography, information security, translation, transcription, protein synthesis, encryption, DNA encryption, DNA encryption, RSA key exchange**

## I. Introduction

From a more practical point of view, security and privacy are very important in modern cryptography because attacks are also becoming more complex. Inspired by biological processes such as transcription and translation in DNA, DNA-based cryptography offers a different method by encoding information in sequences similar to genetic material. Information is encoded exactly like in genetic material in DNA-like sequences, then encrypted in such a fashion that it is extremely difficult to attack without knowledge of the key and encoding scheme.

We discuss DNA cryptography through the following key stages in the project: encoding to DNA, DNA sequence shifting, XOR-based encryption, intron insertion, and biological process-inspired transformations like translation and transcription. The key is securely exchanged between the sender and receiver by RSA encryption, which remains confidential.

## II. Requirements for the DNA cryptography system

To achieve a secure DNA encryption and decryption process, the following requirements have to be met:

- Confidentiality: Only the sender and receiver can view the plaintext message.
- Integrity: The message should remain unmodified during transmission.
- Efficiency: The encryption and decryption processes have to be computationally efficient i.e it cannot take an excessively long amount of time.
- Key Security: Secure exchange of encryption keys using RSA to prevent attacks.
- Confusion and Diffusion: Techniques like intron insertion to ensure the encrypted DNA sequence does not resemble the original message, adding complexity.

## III. Literature Survey

A Novel DNA Computing based Encryption and Decryption Algorithm:

The paper referred here mentions an innovative algorithm inspired by DNA processes to secure encryption and decryption. It mentions a unique algorithm transforming plaintext providing enhanced security complex transformations. The gene encoding digital data sequence nucleotide A, T, C, G adds a layer of security crafting natural processes.

Procedure for Encryption:

- Encoding and Sequence Division: Plaintext is encoded into DNA sequences, which are divided into two parts and mapped to unique characters.
- Protein Sequence Transformation: DNA sequences are converted to protein sequences via transcription and translation, using mRNA and tRNA as intermediaries to secure data.
- Intron-Based Transformation : XNOR with intron sequence complexity pattern making decryption by unauthorised entities highly difficult.
- Final Shifts: Shifts and transformations further secure the DNA sequence, resulting in a complex cipher text.

Procedure for Decryption:

- Reverse Transformation: Protein sequences convert back through tRNA and mRNA to DNA, then to plaintext.
- Intron Reversion: Intron sequences are used to revert final transformations, restoring the original message.

Key Findings:

Experiments show the algorithm performs efficiently across encoding, decoding, encoding and decryption with low latency and strong resistance to cryptanalysis. Frequency and character mapping. [1] [2] [3]

## IV. Implementation

Referring to some of the concepts mentioned in the paper surveyed and adding some new concepts to the mix, the implementation of the system is as follows: The encryption and decryption process imitates biological mechanisms for providing security. This process consists of several phases: encoding to DNA, XOR-based encryption,DNA-sequence shifting, transcription, translation and intron insertion, with decryption (which follows the reverse steps). Each stage contributes to data security by increasing confusion and diffusion.

### *A. Encryption Process*

The encryption process has six main steps: encoding, XOR-based encryption, sequence shifting, transcription, translation and intron insertion.

Step-by-Step Description of Encryption Process:

*1) Encoding to DNA:* Each character in the plaintext is mapped to a DNA sequence using an encoding table I. The encoding table provides a 4-base DNA sequence for each character, which allows characters to be represented in DNA format, as one of the 4 nucleotide bases.

Algorithm: For each character in the plaintext, look up the 4-base DNA sequence. Append the DNA sequence to the `dna_sequence` string.

Example:

- Plaintext: ”HELLO”
- DNA Encoding: `"H -> TCGA", "E -> CTGA", "L -> ACTG", "O -> TGCG"`
- Encoded DNA: `TCGACTGAACTGACTG`

| **Character** | **DNA Sequence** | **Character** | **DNA Sequence** |
|---|---|---|---|
| A | AACC | B | AAGG |
| C | TAAT | D | TATG |
| E | TACC | F | TAGA |
| G | CAAT | H | CATG |
| I | CACG | J | CAGT |
| K | GAAG | L | GATA |
| M | GACG | N | GAGG |
| O | AATA | P | AACG |
| Q | TATC | R | TACG |
| S | CATC | T | CACC |
| U | GATT | V | GACC |
| W | ATAA | X | ATTT |
| Y | ATCG | Z | ATGC |
| `space` | GGGG | . | ATCC |
| , | ATTA | ? | TTTA |
| 0 | TTAA | 1 | TTTT |
| 2 | TTCC | 3 | TTGG |
| 4 | CTAT | 5 | CTTG |
| 6 | CTCC | 7 | CTGA |
| 8 | GTAT | 9 | GTTG |

TABLE I: Encoding Table for Characters to DNA Sequences

*2) XOR-Based Encryption:* The DNA sequence undergoes XOR encryption using a shared symmetric key. Each base in the DNA sequence is XOR-ed with a corresponding base in the key, creating an encrypted DNA sequence. XOR provides

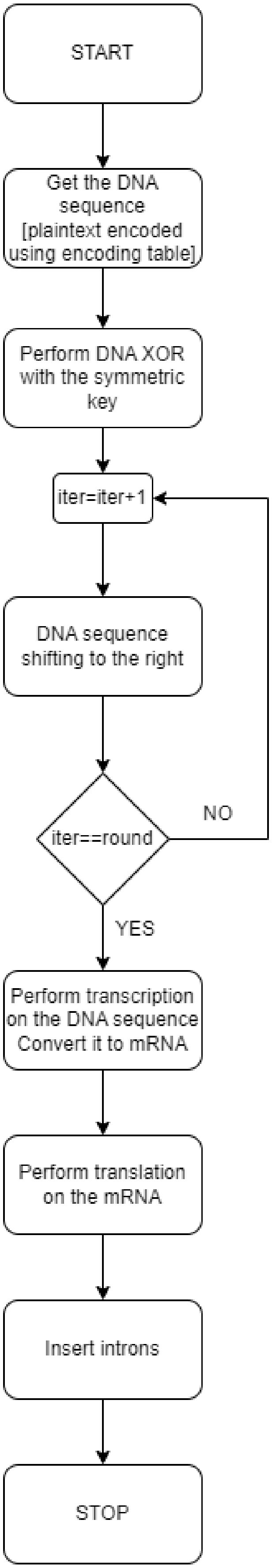
START
Get the DNA sequence [plaintext encoded using encoding table]
Perform DNA XOR with the symmetric key
iter=iter+1
DNA sequence shifting to the right
iter==round
NO
YES
Perform transcription on the DNA sequence Convert it to mRNA
Perform translation on the mRNA
Insert introns
STOP

Fig. 1: DNA Encryption process

cryptographic security by transforming each DNA base in a reversible manner.

Algorithm: For each base in the DNA sequence, use the XOR table II to XOR the DNA base with the corresponding base in the symmetric key. Append the resulting base to `encrypted_dna_sequence`.

| Base 1 | Base 2 | Result | Base 1 | Base 2 | Result |
|---|---|---|---|---|---|
| A | A | A | C | A | C |
| A | C | C | C | C | A |
| A | G | G | C | G | T |
| A | T | T | C | T | G |
| A | U | U | C | U | U |
| G | A | G | T | A | T |
| G | C | T | T | C | G |
| G | G | A | T | G | C |
| G | T | C | T | T | A |
| G | U | U | T | U | U |
| U | A | U | U | C | U |
| U | G | U | U | T | U |
| U | U | A |  |  |  |

TABLE II: XOR Table for DNA Bases

*3) DNA Sequence Shifting (Rightshift) :* The DNA sequence undergoes a basewise right shift by a predefined or randomly chosen number of rounds This shift value must be known for proper decryption. For having a base assumption, we are taking a fixed 16 rounds of sequence shifting.

Algorithm:

Right Shift: Bases are moved to the right by a specified number of positions. Bases that fall off the right end reappear on the left(modular rotation).

*4) Transcription:* The DNA sequence undergoes transcription, which replaces thymine (T) with uracil (U). This step imitates the biological transcription process which converts DNA to mRNA during protein synthesis. By altering one of the bases, transcription further adds confusion to the data, making it harder for attackers to interpret.

Algorithm: For each base in `encrypted_dna_sequence`, replace T with U to obtain `transcribed_sequence`.

*5) Translation:* Translation involves converting an mRNA sequence to a tRNA sequence. In this process, each mRNA base is replaced with its complementary tRNA base, enhancing confusion.

| mRNA Base | tRNA Base |
|---|---|
| A | U |
| U | A |
| C | G |
| G | C |

TABLE III: Base Pair Mapping for Translation

*6) Intron Insertion:* Introns are noncoding sections of an RNA transcript, or the DNA encoding it, that are spliced out before the RNA molecule is translated into a protein. Here we generate introns randomly which are inserted into the middle of the translated mRNA sequence.Introns are used to break up the mRNA sequence, imitating the equivalent biological process.

Algorithm:Determine the middle of the `transcribed_sequence`. Generate a random sequence of bases (A, T, C,G) as the intron. Insert the intron at the middle of the mRNA sequence

Example:

- **Transcribed mRNA**: GCUUACG
- **Intron**: AGCT
- **Final Encrypted Sequence with Intron**: GCUU**AGCT**ACG

### *B. Decryption Process*

*1) Intron Removal:* The inserted intron sequence is removed from the obtained cipher, restoring the original tRNA sequence.

Algorithm: Identify the middle of the sequence. Remove the intron sequence from the middle to get `transcribed_sequence`.

*2) Translation:* Translate tRNA back to mRNA by again replacing bases with their complementary bases. Refer to table III.

*3) Reverse Transcription:* Reverse transcription converts the mRNA sequence back into DNA by replacing U with T. This step is just the reverse of transcription in the encryption process.

Algorithm: Replace every U with T in `transcribed_sequence` to obtain `decrypted_dna_sequence`.

*4) DNA Sequence Shifting (Leftshift):* Here the sequence is shifted back in the opposite direction (left) by the same number of rotations it was shifted in the encryption process. This will restore the original order of bases.

Algorithm:

Left Shift: Bases are moved to the left by a specified number of rotations. Bases that fall off the left end reappear on the right (modular rotation).

*5) XOR-Based Decryption:* The DNA sequence is decrypted by applying XOR operations with the same shared symmetric key which was used during encryption. This reverses the XOR encryption, recovering the original encoded DNA sequence.

Algorithm: For each base in `decrypted_dna_sequence`, XOR the base with the corresponding base in the symmetric key using the XOR table II. Append the result to `decoded_dna_sequence`.

- **Encrypted DNA**: GCTTACG
- **XOR Key**: ACGT
- **Decoded DNA**: TCGACTGA

*6) Decoding:* The decoded DNA sequence is mapped back to the original plaintext using the reverse of the encoding table. Each nucleotide base corresponds to a character, thus forming the plaintext.

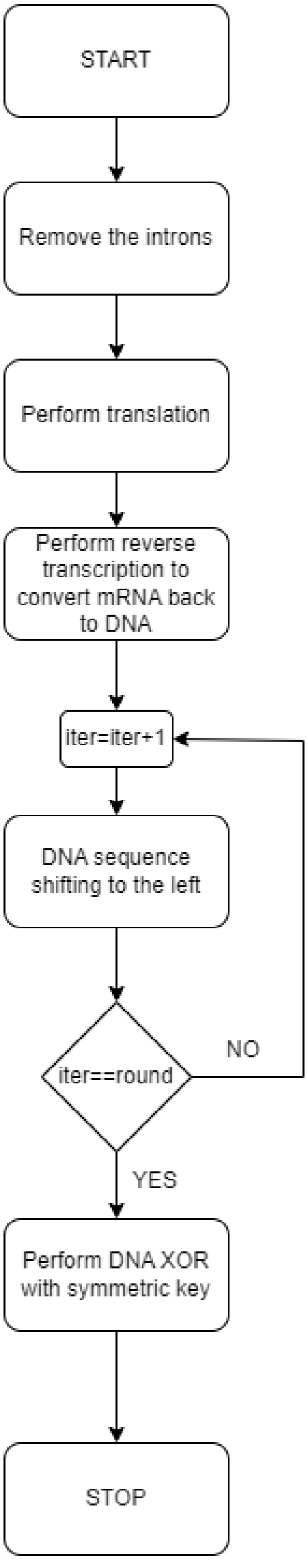
START
Remove the introns
Perform translation
Perform reverse transcription to convert mRNA back to DNA
iter=iter+1
DNA sequence shifting to the left
iter==round
NO
YES
Perform DNA XOR with symmetric key
STOP

Fig. 2: DNA Decryption process

Algorithm: For each 4-base sequence in `decoded_dna_sequence`, find the corresponding character in the reverse encoding table IV. Append each character to form the plaintext.

- **Decoded DNA**: TCGACTGA
- **Plaintext**: HELLO

| DNA Sequence | Character | DNA Sequence | Character |
|---|---|---|---|
| AACC | A | GATA | L |
| AAGG | B | GACG | M |
| TAAT | C | GAGG | N |
| TATG | D | AATA | O |
| TACC | E | AACG | P |
| TAGA | F | TATC | Q |
| CAAT | G | TACG | R |
| CATG | H | CATC | S |
| CACG | I | CACC | T |
| CAGT | J | GATT | U |
| GAAG | K | GACC | V |
| ATAA | W | ATTT | X |
| ATCG | Y | ATGC | Z |
| GGGG | (space) | ATCC | . |
| ATTA | , | TTTA | ? |
| TTAA | 0 | TTTT | 1 |
| TTCC | 2 | TTGG | 3 |
| CTAT | 4 | CTTG | 5 |
| CTCC | 6 | CTGA | 7 |
| GTAT | 8 | GTTG | 9 |

TABLE IV: Decoding Table for Characters to DNA Sequences

### *C. Key Generation and Exchange*

The client generates a random symmetric key which is a DNA sequence of length between 4 to 100. This symmetric key is securely exchanged with the server using RSA encryption and decryption:

- **Server Setup:**
  - The server generates a pair of RSA keys (public and private).
  - It then waits for a connection request from the client.
- **Client-Server Connection:**
  - The client initiates a connection with the server.
  - Upon successful connection, the server sends its RSA public key to the client.
- **Symmetric Key Generation on Client:**
  - The client generates a random symmetric key, which is the DNA sequence key used in the DNA XOR process in the encryption/decryption process.
- **Symmetric Key Encryption:**
  - The client encrypts the symmetric key using the server's RSA public key.
  - By using RSA encryption, only the server, which has the private key, can decrypt and obtain the symmetric key generated by the client.

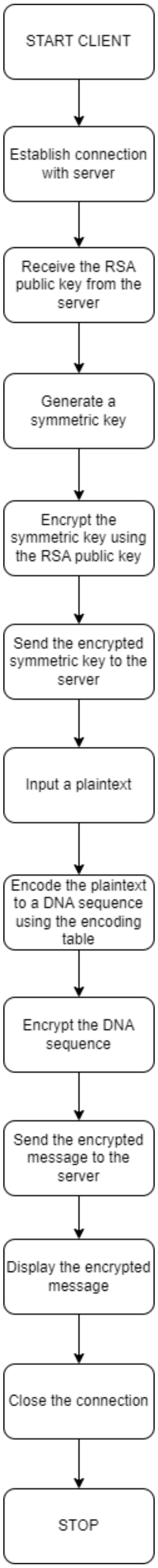
START CLIENT
Establish connection with server
Receive the RSA public key from the server
Generate a symmetric key
Encrypt the symmetric key using the RSA public key
Send the encrypted symmetric key to the server
Input a plaintext
Encode the plaintext to a DNA sequence using the encoding table
Encrypt the DNA sequence
Send the encrypted message to the server
Display the encrypted message
Close the connection
STOP

Fig. 3: Client side process

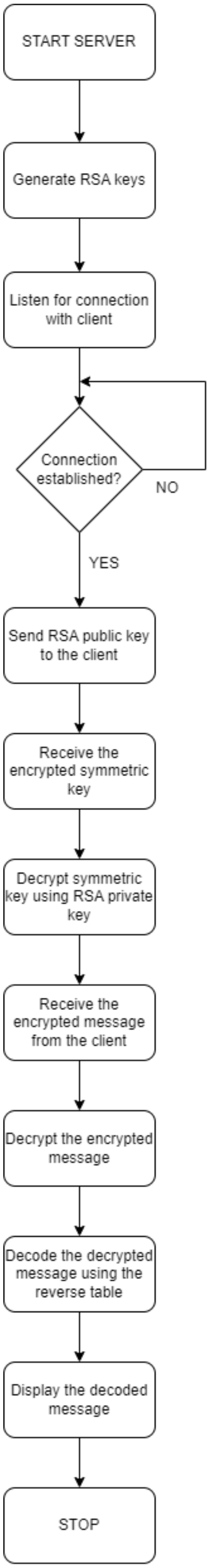
START SERVER
Generate RSA keys
Listen for connection with client
Connection established?
NO
YES
Send RSA public key to the client
Receive the encrypted symmetric key
Decrypt symmetric key using RSA private key
Receive the encrypted message from the client
Decrypt the encrypted message
Decode the decrypted message using the reverse table
Display the decoded message
STOP

Fig. 4: Server side process

- **Sending the Encrypted Key to the Server:**
  - The client sends the RSA encrypted symmetric key to the server.
  - The server receives this encrypted key and decrypts it using its RSA private key, obtaining the symmetric key which is used by the server to decrypt the obtained ciphertext.

## V. Experimental Analysis

### A. Time Analysis

The time taken by the DNA based encryption and decryption was compared to AES encryption and decryption and it was recorded over multiple runs to obtain an average to get an idea of the computational efficiency.

Table V shows the time measurements.

| **Operation** | **Average Time (ms)** |
|---|---|
| DNA Encryption | 0.00014230003580451012 |
| DNA Decryption | 0.0001238000113517046 |
| AES Encryption | 0.004006400005891919 |
| AES Decryption | 0.00011379999341443181 |

TABLE V: Time Taken for Each Operation

### B. Fulfillment of Requirements

- Confidentiality: Achieved through secure XOR-based encryption with key protection.
- Integrity: here is sometimes 1 character in the middle that is off from the original character, but no change other than that.
- Efficiency: The method is not very timetaking, with average encryption and decryption times below 1 ms.
- Key Security: RSA encryption ensures that the symmetric key is securely exchanged.
- Confusion: Intron insertion and transcription help in this.

## VI. Conclusion

The project demonstrates that DNA cryptography can be used as an effective alternative to transmit data securely implementing an interesting crossover of the cryptography practices with biological processes.

This project meets the confidentiality, integrity and efficiency practices. Using functions such as secure key exchange, XOR based encryption and biological processes, a good level of data security is maintained.

## References

[1] Noorul Hussain Ubaidur Rahman, Chithralekha Balamurugan, and Rajapandian Mariappan. A novel dna computing based encryption and decryption algorithm. *Procedia Computer Science*, 46:463–475, 2015. Published under CC BY-NC-ND license.

[2] Guangzhao Cui, Cuiling Li, Haobin Li, and Xiaoguang Li. Dna computing and its application to information security field. In *Proceedings of the 5th International Conference of Natural Computation*, pages Tianjian, China. IEEE, 2009.

[3] Nitya Sree P. Dna-genetic encryption technique. *Vellore Institute of Technology, Chennai*, 2023.