Branch: CSE

Name: Pooja K

Reg. No: 1CD21CS109

Email: poojak.21cse@cambridge.edu.in

GitHub Repository Link: https://github.com/pooja123reddy/DNA_CRYPTOGRAPHY.git

Team members:

Name	Reg. No.	Email
Pragathi	1CD21CS112	Pragathi.21cse@cambridge.edu.in
Pravalika P V	1CD21CS114	Pravalika.21cse@cambridge.edu.in
Pooja R P	1CD21CS110	Pooja.21cse@cambridge.edu.in

1: Problem introduction

DNA cryptography leverages the biological properties of DNA to encode information in a secure format. The encryption algorithm provided here uses DNA sequences to encode plaintext data into a DNA-based cipher. This code demonstrates a multi-step encryption process that converts ASCII characters to DNA sequences, applies transformations based on DNA and mRNA structures, and introduces XOR logic with introns for added security. Given the increasing demand for secure data transfer, DNA cryptography offers a novel approach to achieving high security through biological encoding.

2: Proposed solution

The encryption process involves multiple steps designed to mimic biological transcription and translation processes. Here are two key solutions explored within the code:

1. ASCII to DNA Sequence Mapping and Transformation:
    - ASCII characters are mapped to DNA sequences using a predefined mapping. The DNA sequences then undergo a series of transformations, including binary conversion and right shifting, to enhance security.
2. XNOR-Based Cipher with Introns**:**
    - An XNOR operation using an intron sequence is applied to convert binary sequences into DNA bases. This XNOR-based approach adds another security layer by introducing additional complexity, making decryption more challenging for unauthorized entities.

Both solutions rely on DNA's complex structure, utilizing it as a secure medium for data encryption and transmission.

3: Review of paper

Upon reviewing literature in DNA cryptography, several key studies explore methods such as DNA steganography, molecular encryption with OTP, and public key DNA-based encryption. Notably:

- DNA Steganography: This method hides messages within DNA strands, using biological complexity as a security layer.
- Molecular Encryption Using OTP: Studies reveal that a one-time pad (OTP) with DNA sequences can create highly secure encryption by ensuring each encryption instance is unique.

Merits and Methods:

- Merits: DNA cryptography's vast storage potential and complex encoding allow for efficient data hiding and high security.
- Methods: The literature generally involves encoding data into DNA strands, applying PCR for encoding, and using substitutions based on amino acid mappings for enhanced security.

Gaps and Limitations:

- Costly and complex DNA synthesis and sequencing hinder large-scale implementation.
- Requires specialized lab equipment for encoding and decoding, which may limit practical use.

Adopted Solution and Differences: In our code, ASCII values are directly converted into DNA sequences without using lab-based DNA synthesis, thus enhancing accessibility. The main difference from existing literature lies in the computational approach, as our method relies on software simulation rather than physical DNA.

4: Derive your claim

This section outlines the activities carried out in the code for encryption:

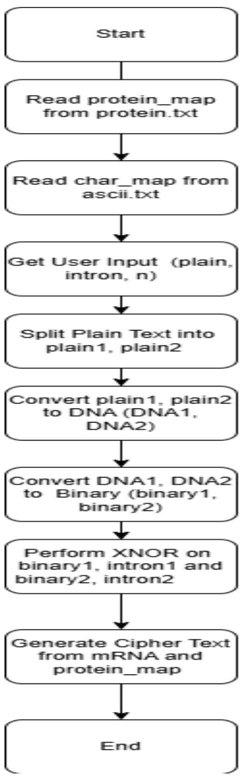
- Conversion from ASCII to DNA: ASCII characters are mapped to DNA sequences.
- Binary Encoding: DNA sequences are converted to binary, representing nucleotide bases as binary values (A=00, T=01, C=10, G=11).
- XNOR Operation: Binary DNA sequences undergo XNOR logic with an intron sequence, introducing additional encryption.
- mRNA and tRNA Transformation: DNA sequences are converted into mRNA and tRNA, simulating biological transcription and translation processes.
- Complementary DNA Calculation: The DNA strands are complemented and right-shifted to further alter the sequence.
- Protein Mapping: The final encrypted text undergoes mapping to produce a protein-based ciphertext.
- Applications: This technique could be adapted for secure communication in fields requiring high encryption standards, such as medical records and military data.

5: Design your evaluation

The DNA cryptography approach in this code is designed to maximize security using biological transformations. The use of ASCII-to-DNA mapping, binary encoding, XNOR logic, and mRNA/tRNA transformations provides a robust, multi-layered encryption method. Since the process only requires software-based simulation, it is resource-efficient and accessible. The layered approach—combining ASCII-to-DNA encoding, binary transformations, and biological simulations—makes it a strong candidate for secure data transmission.

6: Visual elements

Flow chart:

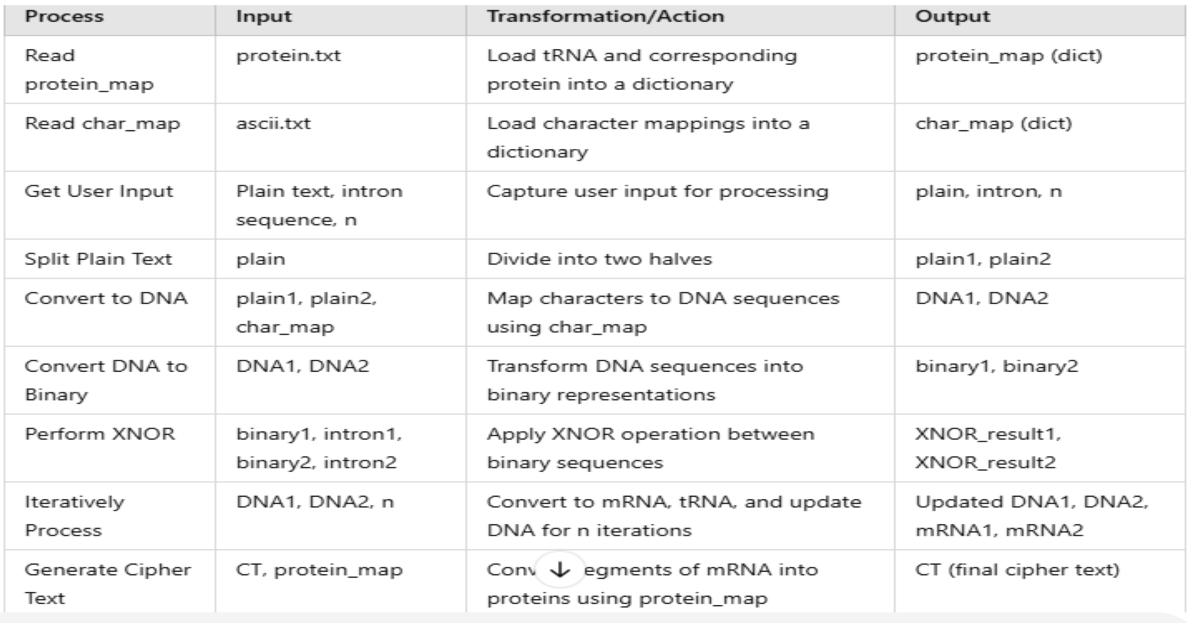

Start
Read protein_map from protein.txt
Read char_map from ascii.txt
Get User Input (plain, intron, n)
Split Plain Text into plain1, plain2
Convert plain1, plain2 to DNA (DNA1, DNA2)
Convert DNA1, DNA2 to Binary (binary1, binary2)
Perform XNOR on binary1, intron1 and binary2, intron2
Generate Cipher Text from mRNA and protein_map
End

pseudocode representation of the main functionality:

```
START PROGRAM

FUNCTION read_protein_map()
   OPEN "protein.txt" AS f
   INITIALIZE protein_map AS EMPTY DICTIONARY
   FOR EACH line IN f
      IF i % 2 == 0 THEN
         tRNA = line.strip()
      ELSE
         protein = line.strip()
         protein_map[tRNA] = protein
      END IF
      i += 1
   END FOR
   CLOSE f
END FUNCTION

FUNCTION read_char_map()
   OPEN "ascii.txt" AS f
   INITIALIZE char_map AS EMPTY DICTIONARY
```

```
    FOR EACH line IN f
        SPLIT line BY WHITESPACE INTO parts
        char_map[parts[0]] = parts[1]
    END FOR
    CLOSE f
END FUNCTION

FUNCTION convert_to_DNA(plain_text)
    INITIALIZE DNA AS EMPTY STRING
    FOR EACH char IN plain_text
        IF char IN char_map THEN
            DNA += char_map[char]
        ELSE
            PRINT "Character not found"
        END IF
    END FOR
    RETURN DNA
END FUNCTION

FUNCTION XNOR_and_convert_to_DNA(binary, intron)
    INITIALIZE xnor AS EMPTY STRING
    FOR EACH bit IN binary
        IF bit == intron THEN
            xnor += "1"
        ELSE
            xnor += "0"
        END IF
    END FOR
    RETURN convert_xnor_to_DNA(xnor)
END FUNCTION

FUNCTION main()
    CALL read_protein_map()
    CALL read_char_map()
    GET plain_text, intron, iterations FROM USER
    SPLIT plain_text INTO plain1 AND plain2
    DNA1 = convert_to_DNA(plain1)
    DNA2 = convert_to_DNA(plain2)
    FOR i FROM 0 TO iterations
        mRNA1 = convert_to_mRNA(DNA1)
        tRNA1 = complement_tRNA(mRNA1)

        DNA1 = convert_to_DNA(tRNA1)

        DNA1 = right_shift(DNA1)

    END FOR




    CIPHER_TEXT = generate_cipher_text(DNA1)

    PRINT CIPHER_TEXT

END FUNCTION

END PROGRAM
```

Data flow table:



Process	Input	Transformation/Action	Output
Read protein_map	protein.txt	Load tRNA and corresponding protein into a dictionary	protein_map (dict)
Read char_map	ascii.txt	Load character mappings into a dictionary	char_map (dict)
Get User Input	Plain text, intron sequence, n	Capture user input for processing	plain, intron, n
Split Plain Text	plain	Divide into two halves	plain1, plain2
Convert to DNA	plain1, plain2, char_map	Map characters to DNA sequences using char_map	DNA1, DNA2
Convert DNA to Binary	DNA1, DNA2	Transform DNA sequences into binary representations	binary1, binary2
Perform XNOR	binary1, intron1, binary2, intron2	Apply XNOR operation between binary sequences	XNOR_result1, XNOR_result2
Iteratively Process	DNA1, DNA2, n	Convert to mRNA, tRNA, and update DNA for n iterations	Updated DNA1, DNA2, mRNA1, mRNA2
Generate Cipher Text	CT, protein_map	Conv[illegible]egments of mRNA into proteins using protein_map	CT (final cipher text)

Evaluation

Marks distribution				
Your nearest neighbour (20)	Claims (20)	Clarity and Conciseness (20)	Visual Elements & Formatting (20)	Accuracy & Precision (20)