DNA-BASED SECURE DATA HIDING SYSTEM

Ensuring Confidentiality, Integrity, and Authenticity

Under the guidance of
Prof. Ashok K Bhateja

By
Vaddi Jhancy
2024JCS2040

Date: 12-05-2025

Introduction

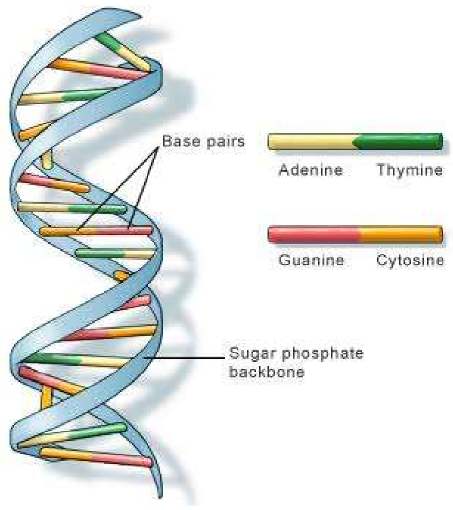
- Cryptography is the science of securing communication by transforming (encrypting) data so that only authorized parties can access it.
- Steganography is the practice of hiding secret information within a cover medium (like images, videos, or DNA) in such a way that the presence of the hidden information is undetectable to unauthorized users.
- Deoxyribonucleic acid (DNA) is a molecule that contains the genetic instructions necessary for the growth, development, and reproduction of all living organisms.
- DNA consists of four nucleotide bases:

    Adenine (A)

    Thymine (T)

    Guanine (G)

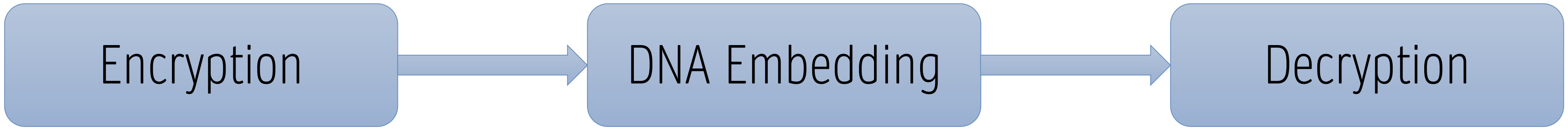
    Cytosine (C)



Introduction

- Secure communication using DNA sequences.
  DNA sequences serve as a unique medium to embed and transmit secret messages securely. For example, a short DNA sequence is “ATCGGACT”. Generally, any organism’s DNA sequence contains more than a billion bases in it.
- Combines cryptography, compression, and DNA encoding.
  Cryptography: Ensures the message remains confidential and authentic using encryption techniques like AES and digital signatures such as ECDSA or RSA.
  Compression: Reduces the secret message's size with Huffman coding, making it more efficient to embed.
  DNA Encoding: Transforms the encrypted and compressed message into DNA nucleotides (A, T, G, C), allowing it to blend seamlessly into a original DNA sequence.
  A -> 00, T -> 01, G -> 10, C -> 11

System Overview



Encryption Process

- Involves compression, encryption, error correction, and digital signatures.

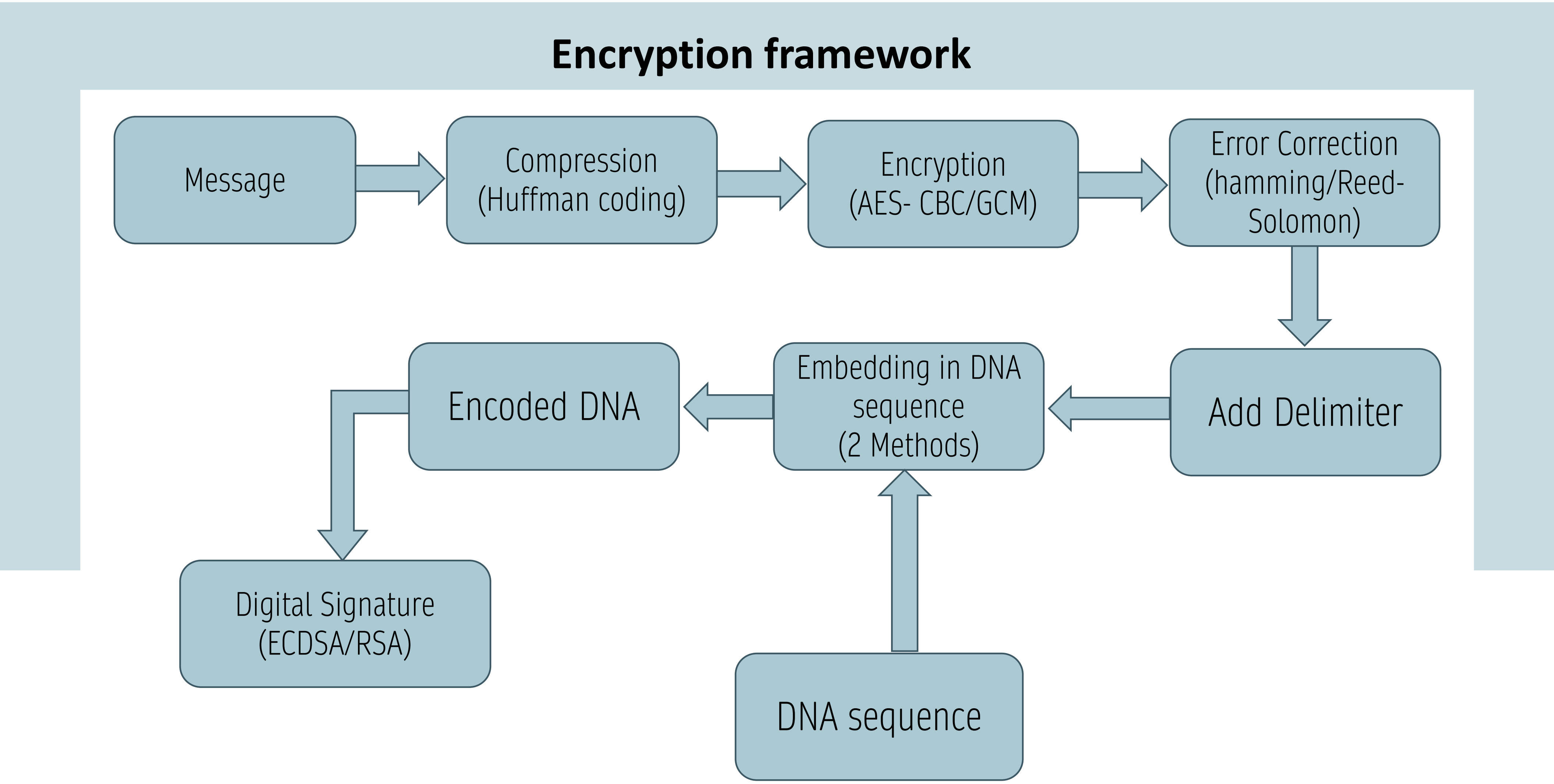
DNA Embedding

- Integrates the processed message into a DNA sequence using one of these methods:
- Method 1: Binary insertion into the DNA binary.
- Method 2: Codon-based embedding with XOR and substitution rules.

Decryption Process

•Extracts the embedded message from the DNA sequence.

•Reverses the encryption steps to recover the original secret message.


Encryption framework
Message
Compression (Huffman coding)
Encryption (AES- CBC/GCM)
Error Correction (hamming/Reed-Solomon)
Add Delimiter
Embedding in DNA sequence (2 Methods)
Encoded DNA
Digital Signature (ECDSA/RSA)
DNA sequence

Encryption Process

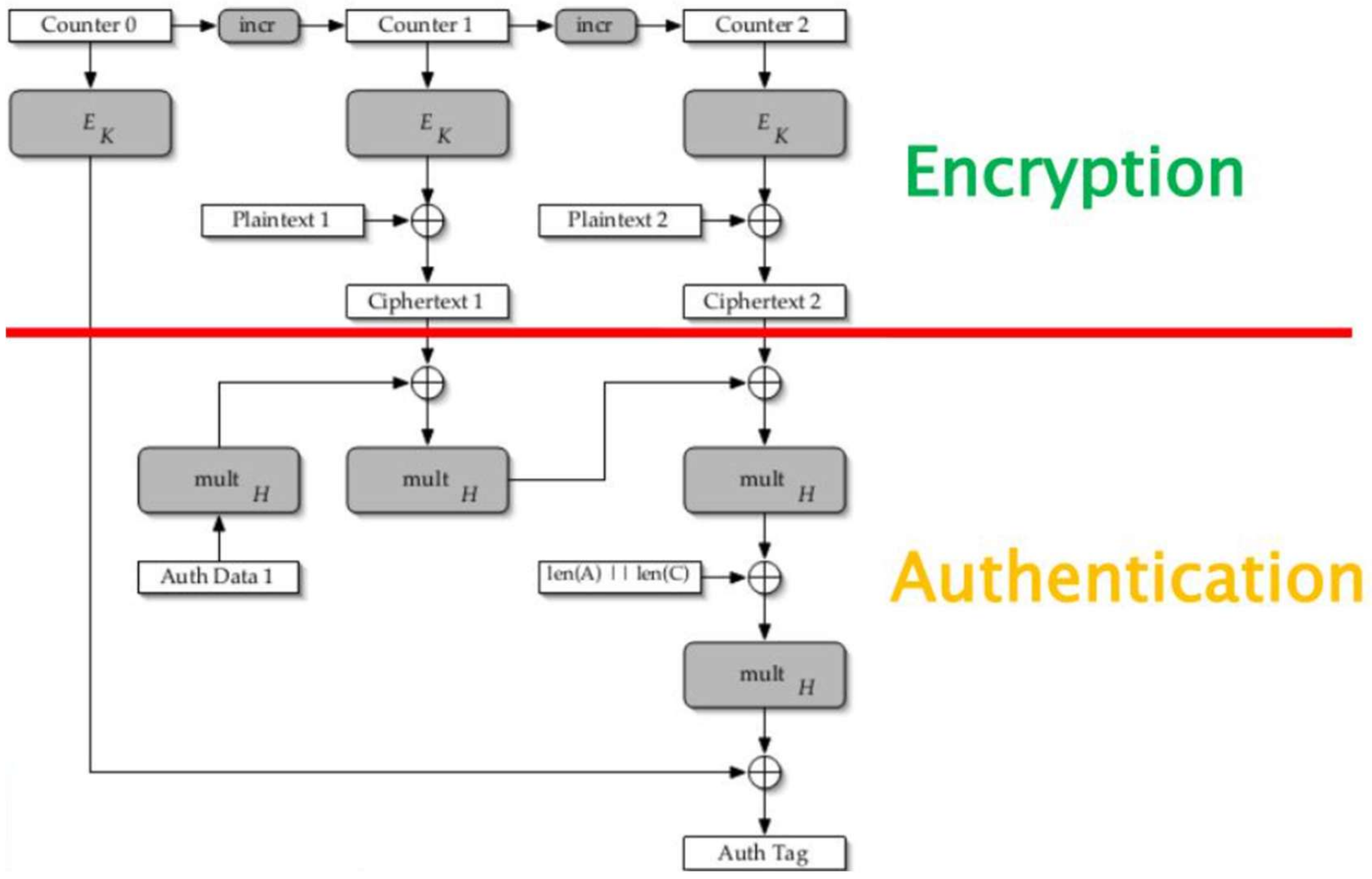
Steps:

1) **Key Generation (ECDSA/RSA)**
   Generates private and public key pairs used for digital signatures.
   ECDSA: Utilizes the SECP256k1 curve to create secure and efficient keys.
   RSA: Produces 2048-bit keys with a public exponent of 65537 for strong security.

2) **Huffman Compression**
   Compresses the secret message using Huffman coding.
   Constructs a frequency-based tree to assign variable-length binary codes to characters.
   Reduces the message size to optimize embedding into DNA.

3) **AES Encryption (CBC/GCM)**
   Encrypts the compressed binary data using AES with a 256-bit key.
   CBC Mode: Applies padding and an initialization vector (IV) for secure block encryption.
   GCM Mode: Uses a nonce and provides an authentication tag to ensure data integrity.

GCM mode


Counter 0
incr
Counter 1
incr
Counter 2
E K
E K
E K
Encryption
Plaintext 1
Plaintext 2
Ciphertext 1
Ciphertext 2
mult H
mult H
mult H
Auth Data 1
len(A) || len(C)
Authentication
mult H
Auth Tag

Encryption Process

Steps:

4) Error Correction (Reed-Solomon/Hamming)
Adds redundancy to the encrypted data for error detection and correction.
Reed-Solomon: Encodes data with additional symbols to correct multiple errors.
Hamming: Inserts parity bits into each byte for single-error correction.

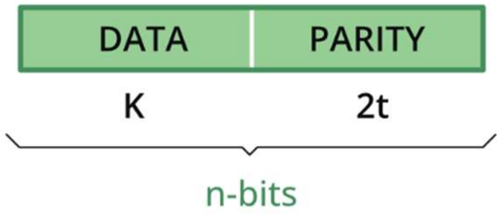
5) Delimiter Addition
Adds a fixed 56-bit delimiter(7-Bytes) before and after the error-corrected binary data.
Ensures accurate extraction of the embedded message during decryption.

Error Correction Codes

Reed–Solomon Code

- (n, k) code is used to encode each symbol of m bit.
- Block length(n) is given by (2^m)-1 symbols.
- In Reed-Solomon codes, message size is given by (n-2t) where t= number of errors corrected.
- Parity check size is given by = (n-k) or 2t symbols.

Representation on n-bits solomon codes



$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)......(x-\alpha^{2t})$$

Generator function

$$p(x) = m_{k-1}x^{k-1} + \ldots + m_1x + m_0$$

Transmitted polynomial s(x)

$$s(x) = p(x) \times x^{2t} - s_r(x)$$

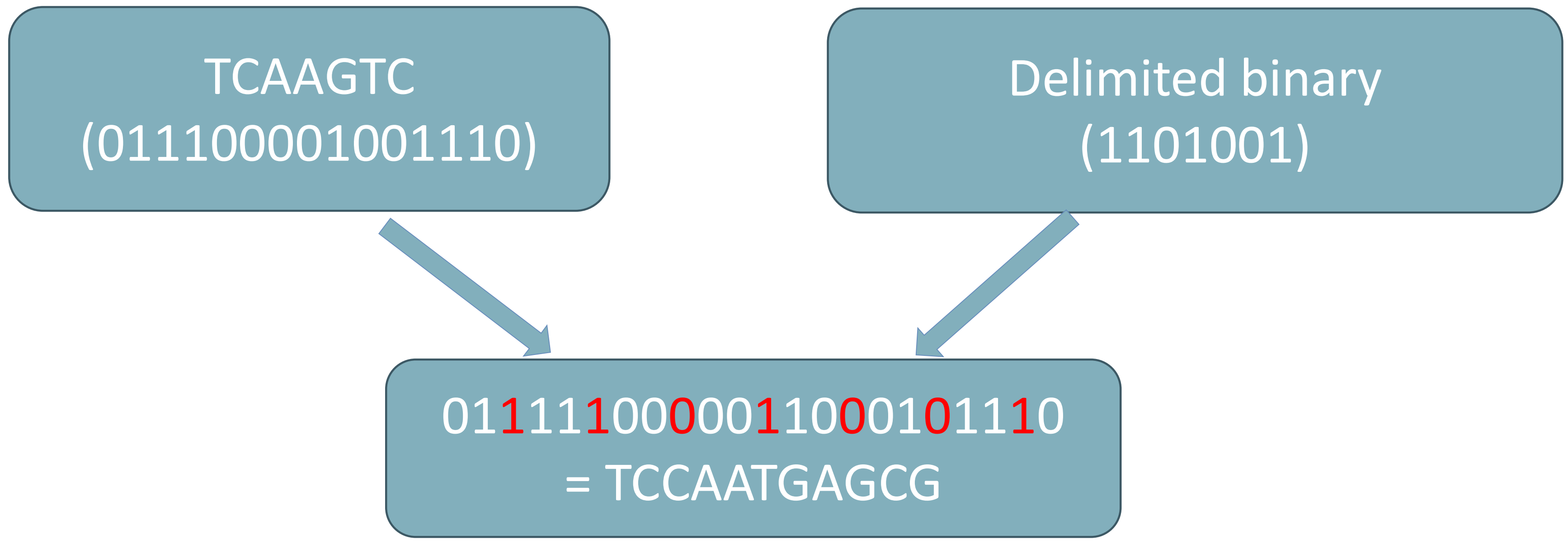
- Find $S_r(x) = (p(x) * x^{2t}) \bmod g(x)$
- s(x) is evenly divisible by g(x), let s(x) = q(x) *g(x)
- Sender sends s(x) = q(x) * g(x), and sends over the coefficients of s(x)
- Receiver receives r(x).
- If s(x) = r(x), then g(x) divides r(x) with no remainder
- Otherwise, r(x) = q(x) * g(x) + e(x), i.e., r(x) = s(x) + e(x) where e(x) is an error polynomial.

DNA Embedding

Steps:

Adenine (A) -> 00, Thymine (T) -> 01, Guanine (G) -> 10, Cytosine (C) -> 11

Method 1: Converts human DNA to binary, inserts the delimited binary every third bit, then converts back to DNA.



DNA Embedding

Adenine (A) -> 00, Thymine (T) -> 01, Guanine (G) -> 10, Cytosine (C) -> 11

Method 2:

->Convert encrypted binary to DNA codons (e.g., "ACT" represents 6 bits) using codons table. (number of codons = 2^6 = 64)

->XOR the codon sequence with a second DNA sequence for additional security.

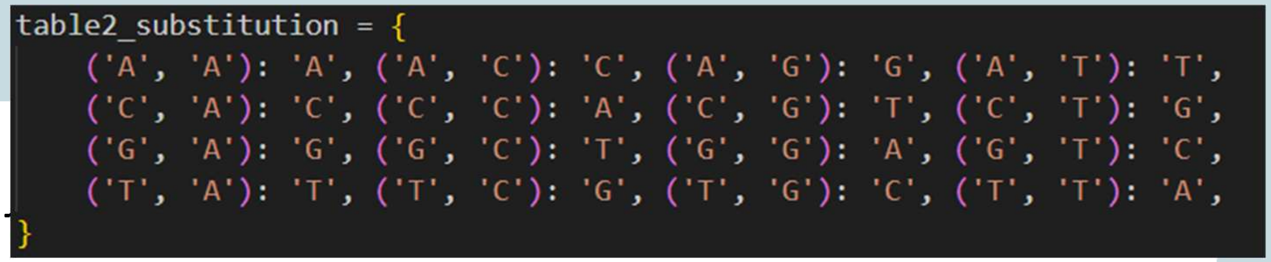
->Embed the result into the host DNA at positions with adjacent repeated nucleotides (e.g., "AA") using a substitution table.

DNA Embedding

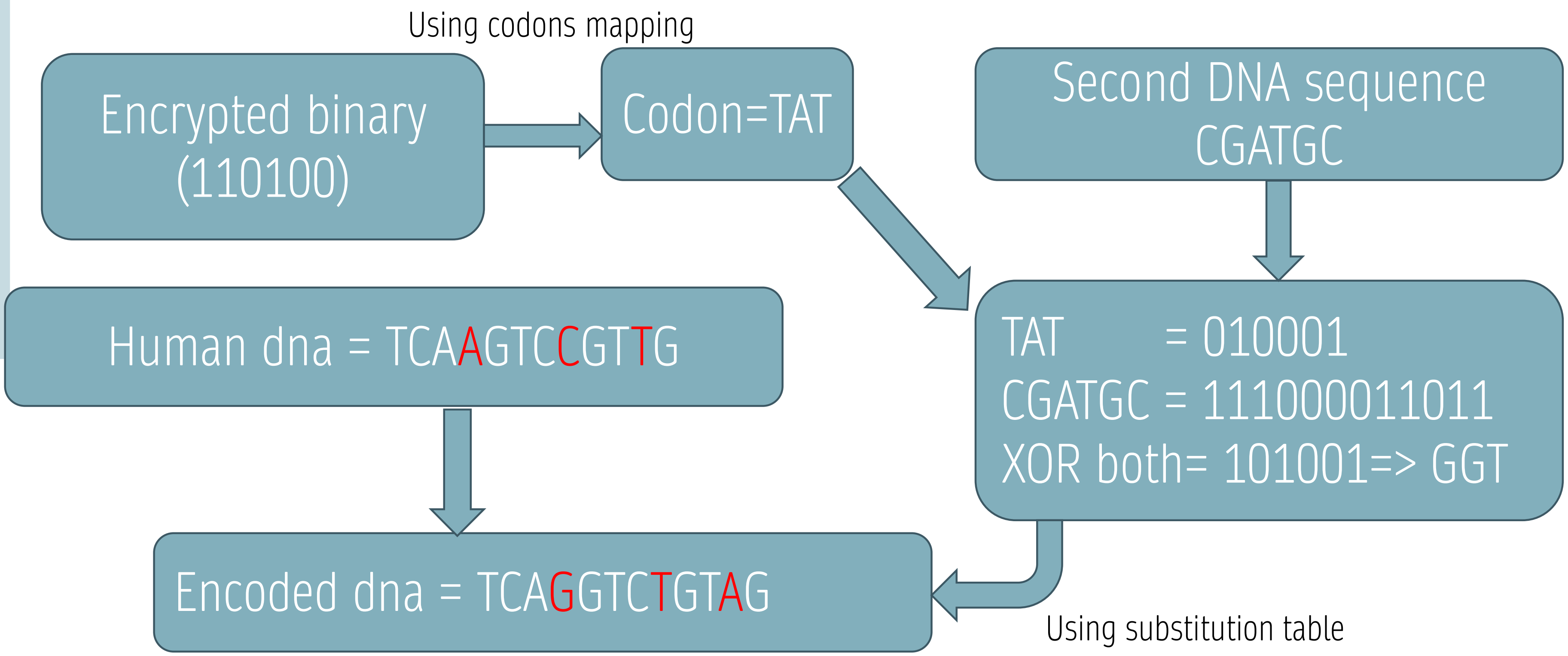
Steps:

Adenine (A) -> 00, Thymine (T) -

Method 2:



```
table2_substitution = {
    ('A', 'A'): 'A', ('A', 'C'): 'C', ('A', 'G'): 'G', ('A', 'T'): 'T',
    ('C', 'A'): 'C', ('C', 'C'): 'A', ('C', 'G'): 'T', ('C', 'T'): 'G',
    ('G', 'A'): 'G', ('G', 'C'): 'T', ('G', 'G'): 'A', ('G', 'T'): 'C',
    ('T', 'A'): 'T', ('T', 'C'): 'G', ('T', 'G'): 'C', ('T', 'T'): 'A',
}
```



DNA Embedding

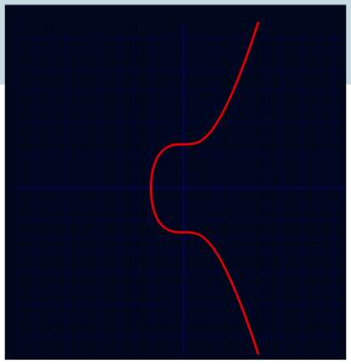
Steps:

Digital Signature
Computes a SHA-256 hash of the encoded DNA.
Signs the hash using the private key (ECDSA or RSA) to verify authenticity and integrity.

Signature Embedding
Generates a 10-character hash of the signature.
Embeds it after the first 10 nucleotides of the encoded DNA for additional verification during decryption.

ECDSA

Creates ECDSA private and public key pair (SECP256K1 curve).



Signs the hash with ECDSA private key.

•Message has **not been altered** (integrity)

•Sender is **authentic** (non-repudiation)

$$Z_{2^{256}-2^{32}-977}$$

The curve is designed to work over a Galois finite prime field which means the X and Y coordinates are 256-bit integers modulo a large number.

ECDSA is an algorithm used to sign transactions with secp256k1 as the underlying curve.

A private key is randomly generated as a 256-bit integer to create a new key pair.

The corresponding public key is then derived by multiplying the private key by the secp256k1 base point G (a predefined point on the curve).

Decryption Process

1. **Signature Verification:** Use the public key to verify the signature against the DNA hash.
2. **Signature Hash Verification:** Check if the embedded hash matches the signature’s hash.
3. **Message Extraction:** Retrieve the hidden binary from the DNA using the chosen embedding method.
4. **Error Correction:** Correct errors in the extracted binary.
5. **AES Decryption:** Decrypt the binary using the AES key.
6. **Huffman Decompression:** Decompress the binary to recover the original message.

Decryption Process

1. **Signature Verification:** Use the public key to verify the signature against the DNA hash.

   Compute the SHA-256 hash of the encoded DNA (excluding the embedded hash).

   Use the public key to verify the signature against this hash.

   Outcome:

   If valid, proceed with decryption.

   If invalid, halt the process (indicating potential tampering).

Decryption Process

3. **Message Extraction:** Retrieve the hidden binary from the DNA using the chosen embedding method.

   Method1

   Convert the encoded DNA to binary.

   Extract bits inserted at regular intervals (e.g., every third bit).

   Remove delimiters to obtain the error-corrected binary.

   Example:

   Encoded DNA: "TGACT" → "$01\underline{1}00\underline{0}11\underline{0}1$“

   Extract Every Third Bit: "100"

Decryption Process

3. **Message Extraction:** Retrieve the hidden binary from the DNA using the chosen embedding method.

   Method2

   Identify positions with adjacent repeated nucleotides.

   Use the reverse substitution table to extract embedded nucleotides.

   XOR the extracted sequence with the second DNA sequence to recover codons.

   Convert codons to binary and remove delimiters.

   Example: Extract "TGA" from repeated positions.

   XOR "TGA" with "ACG" → "GTC“ Convert "GTC" to Binary: "11010110"

Decryption Process

4. **Error Correction:** Correct errors in the extracted binary.

   Reed-Solomon: Decodes the binary to recover the original encrypted data.

   Hamming: Corrects single-bit errors in each 12-bit block.

   Process: Apply the appropriate decoding algorithm to fix errors.

5. **AES Decryption:** Decrypt the binary using the AES key.

   CBC: Uses the IV to decrypt and removes padding.

   GCM: Uses the nonce to decrypt and verifies the authentication tag.

   Process: Decrypt the binary using the AES key.

   Output: Compressed binary ready for decompression.

Decryption Process

6. **Huffman Decompression:** Decompress the binary to recover the original message.

   Rebuild the Huffman tree from the codebook.

   Traverse the tree using the binary string to decode characters.

   Example:

   Binary: "01" → "0" = "H", "1" = "i" → "Hi"

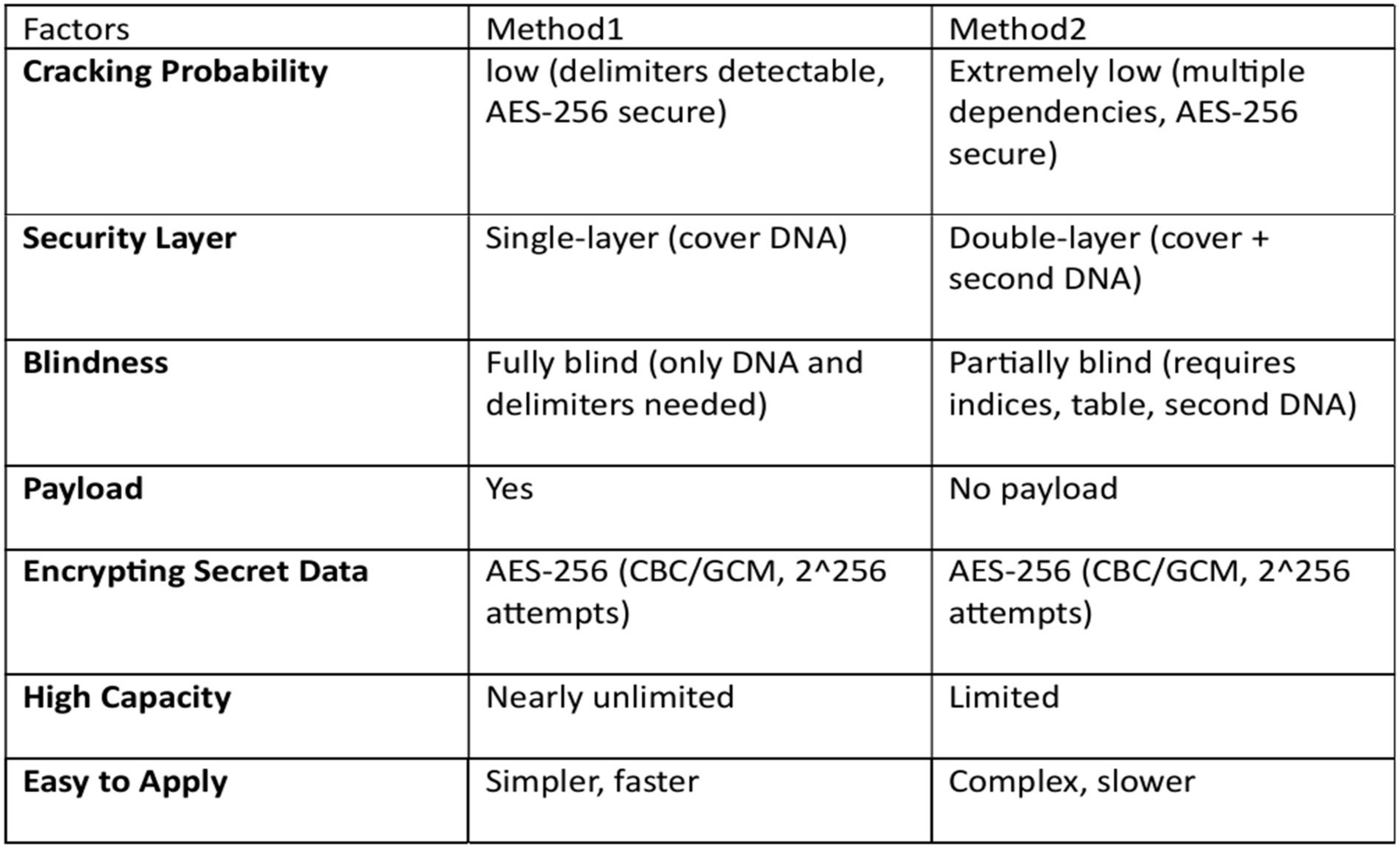
Evaluation Factors and Security analysis

- **Cracking probability**
  As the length of the DNA sequence increases the probability of cracking decreases.
  AES 256-bit key encryption- > probability to crack this key is1/(2^256).
  Quantum resistance also.
- **Layer of Security**
  method2 is more secured than method1 because we are using double layer.
- **Blindness**
  method1 has more blindness compare to method2 because in method2 the receiver need to use the original DNA sequence while decrypting the secret message.
- **Modification rate**
  In both the methods there is less modification rate only.

Evaluation Factors and Security analysis

- **Preserving original DNA functionality** is that function of producing proteins is not affected.
- **Confidentiality**
  Encryption with AES-256 key will provide confidentiality
- **Integrity**
  Hashing will ensure integrity
- **Authentication**
  digital signatures will ensure authenticity
- **Correctness**
  with Reed-Solomon code or Hamming code
- **Avalanche effect**
  depending on the ratio of number of DNA sequence bits to the number message bits.

Comparative analysis



Factors	Method1	Method2
Cracking Probability	low (delimiters detectable, AES-256 secure)	Extremely low (multiple dependencies, AES-256 secure)
Security Layer	Single-layer (cover DNA)	Double-layer (cover + second DNA)
Blindness	Fully blind (only DNA and delimiters needed)	Partially blind (requires indices, table, second DNA)
Payload	Yes	No payload
Encrypting Secret Data	AES-256 (CBC/GCM, 2^256 attempts)	AES-256 (CBC/GCM, 2^256 attempts)
High Capacity	Nearly unlimited	Limited
Easy to Apply	Simpler, faster	Complex, slower

Comparison

- **ECDSA vs. RSA:**

  ECDSA is faster and more compact, while RSA offers broader compatibility.

- **CBC vs. GCM:**

  GCM provides integrity, making it ideal for untrusted environments like cloud storage; CBC is simpler but lacks built-in integrity.

- **Reed-Solomon vs. Hamming:**

  Reed-Solomon corrects multiple errors, suiting DNA’s error-prone nature; Hamming is lightweight but limited.

Shared between Sender and Receiver secretly

- Codebook -> compression mapping
- AES key -> encryption key
- Embedding key-> for method1
- Delimiters
- Codons mapping-> for method2
- Substitution table-> for method2

Public keys

- Signature
- Human DNA
- Encoded DNA

Conclusion

- **New system** integrating steganography, cryptography, and error correction for secure data hiding in DNA.
- **Steganographic Security:** Hides data in natural DNA patterns (e.g., repeated nucleotides), minimizing detection risk.
- **Security:** Ensures confidentiality (AES-256), integrity (RSA/ECDSA signatures), and reliability (Reed-Solomon/Hamming codes).
- **Flexibility:** Supports multiple methods (Method 1: insertion, Method 2: codon-based) and algorithms (CBC/GCM, RSA/ECDSA).

References

- K. Ning, “New field of cryptography: DNA cryptography,” Chinese Sci. Bull., vol. 49, no. 15, pp. 1539–1540, Aug. 2004
- NIST, “Advanced Encryption Standard (AES),” Federal Inf. Process. Standards Publ. 197, Nov. 2001
- I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” J. Soc. Ind. Appl. Math., vol. 8, no. 2, pp. 300–304, Jun. 1960
- R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, Feb. 1978

THANK YOU