International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 8, pp 48-57, August-2023

https://doi.org/10.47001/IRJIET/2023.708007

Data Hiding Using DNA Segments

Auday H. AL-Wattar

Lecturer, Computer Science and Mathematics, University of Mosul, Mosul, Iraq

***Abstract* - Information security has now emerged as an essential foundation that the modern world cannot afford to abandon or disrespect in any way. Data confidentiality is still a preoccupation and a constant concern in the world of the electronic industry. Encryption has always been and continues to be the go-to solution for achieving data security. Despite its effectiveness in maintaining security, encryption in its visible version remains an appealing option for potential attackers. As a result, multiple studies and substantial research have focused on steganography, which has an advantage over encryption in that it may conceal information without leaving an identifiable format. This trait reduces the zeal and difficulties that potential attackers confront when attempting to penetrate security mechanisms.**

Life is made up of genetic material called DNA, which is present in humans and the majority of other species. The DNA code is made up of the amino acids adenine (A), guanine (G), cytosine (C), and thymine (T). Humans share around 98% of these DNA bases, highlighting the commonalities across people. The order of these bases determines the information that is available for an organism's growth and preservation, much like the letters in words and sentences. This paper introduces a unique masking method that takes advantage of the intrinsic characteristics of DNA structure.. It includes converting data into a DNA format and then using precise computations to integrate it into pre-existing DNA blocks. It has been established via careful investigation and analysis that this approach is appropriate for obtaining information security using unconventional formulae.

Keywords: Security, Steganography, Data hiding, DNA.

I. INTRODUCTION

We conduct both our personal and business lives online. We use Internet banking, manufacturing, bill payment, communication, and work. Although digital networks and the Internet are convenient, there is risk involved. Data security guards against theft, corruption, and unauthorized access to digital data. It includes everything, including storage and hardware. Organizational dynamics and competitiveness are being drastically altered by the digital revolution. Organizations are producing, processing, and storing more data, which calls for data governance. The complexity of computer systems is increasing due to the public cloud, corporate data centers, and numerous edge devices such as robots, IoT sensors, and remote servers. Complexity increases the attack surface, which complicates security and monitoring. software application security, administrative security, and access control security. Company policies are also covered. Digital security safeguards assets, data, and personal business accounts. Examples include biometrics, web services, encrypted devices, antivirus software, and SIM cards for smartphones. [1-3].

Cryptography is the study of safe routes of communication that are visible only to the sender and the intended recipient. In Greek, kryptos means hidden. That's why encryption matters; it jumbles text into ciphertext and back again[4]. Using microdots or merging, cryptography also encrypts images. Roman Emperor Julius Caesar employed one of the earliest known ciphers, while Ancient Egyptians employed similar strategies in intricate hieroglyphics. The majority of the services we use on a daily basis are accomplished through encryption, which is the ideal answer to many information security issues. For many information security issues, encryption is the perfect solution, and it generates the majority of the services we use on a regular basis. decrypt the encrypted messages or examine the encryption schemes. The only dilemma of this technology is that encryption produces the ciphertext, which, although it represents incomprehensible symbols or obscure symbols, represents a visual material that anyone who tries to analyze the encrypted texts or break encryption algorithms [5-7].

Using data concealing is the best way to get around the encryption's weak point since hiding leaves no trace that an attacker may use to unlock the cipher[8, 9].

All the information required to build and maintain a creature is contained in the complex molecule known as DNA. Actually, every single cell in a multi cellular organism has all of its DNA. DNA is the basic unit of genetics that determines the form and function of all living things. Four chemical bases - adenine (A), guanine (G), cytosine (C), and thymine (T) - are used by DNA to encode information. Of the 3 billion bases in human DNA, about 99% are identical. The way these bases are arranged determines the information that may be used to grow and maintain an organism, much to how alphabetic letters create words and sentences [10-12].

DNA contains great storing qualities that make it an ideal setting for concealment operations. These features may be used to bury the data that needs to be hidden which might be useful for data-hiding techniques. When certain procedures are put into place, they might become disorganized and be depended upon to do data-hiding and encrypting tasks [13-15].

In this study, we will exploit the architecture and characteristics of DNA to conceal data. The data masking procedure has also made use of DNA sequence format.

1.1 Data Hiding

For copyright, authentication, and annotation purposes, steganography, also known as data hiding, embeds data into a digital medium. Steganography is the study of concealing data in multimedia files, such as pictures, sounds, and videos. Since the characteristic is visible, the attack point is clear, and the goal is to conceal the embedded data. It involves hiding information and denying the presence of embedded data. Compared to encryption, which just hides the communication's content and not its existence, it is a more effective technique of message security. So that alterations to the carrier cannot be seen, the original message is hidden inside of it. Steganography conceals information by using cover carriers that appear benign. Steganography conceals information by using cover carriers that appear benign[16-18].

A message embedded in a bit stream can contain plaintext, encrypted text, graphics, or anything else. The cover carrier and embedded message are combined in a stage carrier.

For embedding information, a stego key, such as a password, may be required. Stego-images are created by concealing a secret message in a cover image[19, 20].

The procedure may take the following basic form:

Stego-medium = cover medium with embedded message + stego key.

The basic block frame of the steganography system is shown in Figure 1.

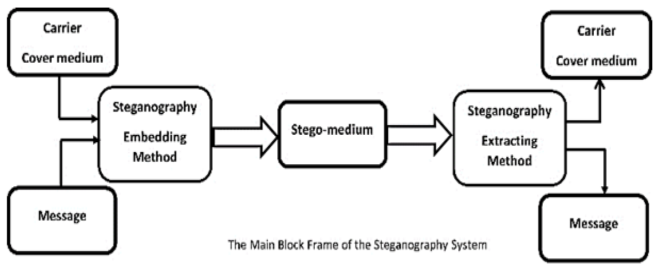


Figure 1: The Main Block Frame of the Steganography System

1.2 Steganography applications

Watermarks can be stored in data using several steganographic methods. Watermarking just adds information to the cover source, whereas steganography conceals information. Watermarks can be obscured using steganographic techniques, but people will not accept them[21].

In 1953, sed crystallographic data from Rosalind Franklin and Maurice Wilkins to discover the three-dimensional structure of DNA. This laid the groundwork for DNA replication and protein-encoding in nucleic acids. It took longer to develop the ability to "read" or sequence DNA. Because DNA molecules grew longer and included fewer, more similar components, making differentiation more difficult, new ways were required[22, 23].

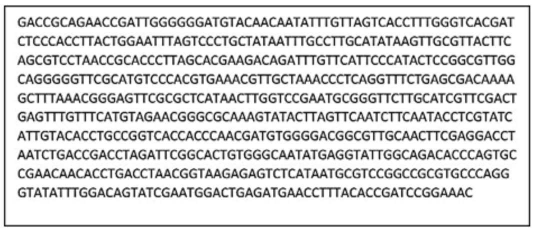
DNA sequencing is a scientific method for determining the nucleotide sequence of a DNA molecule. To develop and function, cells use the sequence of bases (A, T, C, and G). DNA sequencing is required to understand gene and genome function. The development of novel DNA sequencing technology is a critical component of genomics research[11, 24]

1.3 DNA sequences formats

In this section, several DNA sequence formats as plain sequence format will be reviewed:

IUPAC format

In this style, the first type is known as IUPAC, and a sequence can only consist of IUPAC characters and spaces (no digits).



```
GACCGCAGAACCGATTGGGGGGATGTACAACAATATTTGTTAGTCACCTTTGGGTCACGAT
CTCCCACCTTACTGGAATTTAGTCCCTGCTATAATTTGCCTTGCATATAAGTTGCGTTACTTC
AGCGTCCTAACCGCACCCTTAGCACGAAGACAGATTTGTTCATTCCCATACTCCGGCGTTGG
CAGGGGGTTCGCATGTCCCACGTGAAACGTTGCTAAACCCTCAGGTTTCTGAGCGACAAAA
GCTTTAAACGGGAGTTCGCGCTCATAACTTGGTCCGAATGCGGGTTCTTGCATCGTTCGACT
GAGTTTGTTTCATGTAGAACGGGCGCAAAGTATACTTAGTTCAATCTTCAATACCTCGTATC
ATTGTACACCTGCCGGTCACCACCCAACGATGTGGGGACGGCGTTGCAACTTCGAGGACCT
AATCTGACCGACCTAGATTCGGCACTGTGGGCAATATGAGGTATTGGCAGACACCCAGTGC
CGAACAACACCTGACCTAACGGTAAGAGAGTCTCATAATGCGTCCGGCCGCGTGCCCAGG
GTATATTTGGACAGTATCGAATGGACTGAGATGAACCTTTACACCGATCCGGAAAC
```

Figure 2: Example sequence of IUPAC type

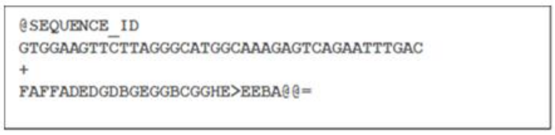
Figure 2 depicts an example of this type of sequence [25, 26].

FASTQ format

The FASTQ format is the other sort of format. Sequence files of this type can include several sequences. FASTQ is a text-based database that holds biological sequences and quality rankings. It saves the results of high-throughput sequencing instruments. FASTQ sequences are made up of

ISSN (online): 2581-3048
Volume 7, Issue 8, pp 48-57, August-2023
https://doi.org/10.47001/IRJIET/2023.708007

four lines. '@', sequence identifier, optional description, basic sequence letters, a "+," perhaps followed by the same sequence identifier (and any description), and basic sequence letter quality[27]. Figure 3 depicts the format for this type.



```
@SEQUENCE_ID
GTGGAAGTTCTTAGGGCATGGCAAAGAGTCAGAATTTGAC
+
FAFFADEDGDBGEGGBCGGHE>EEBA@@=
```

Figure 3: Example sequence of FASTQ **format.**

EMBL format

EMBL files may include several sequences. In sequence entries, ID lines come before annotation lines. The sequence starts with "SQ" and finishes with "//"[28, 29].

FASTA format

FASTA files can include several sequences. It begins with a single-line description and sequencing data. The description line in the first column must begin with a > sign [28, 30].

GCG format

GCG sequence files include a single sequence that begins with annotation lines and ends with two dot ("..") characters. This line contains the sequence ID, length, and checksum. This format should only be used for GCG files [31, 32].

GCG New GCG-RSF files may have several sequences. This format should only be used for GCG files. -RSF stands for rich sequence format [33, 34].

GenBank format

Several sequences are included in GenBank sequence files.

LOCUS and several annotation lines precede GenBank sequences. The sequence starts with "ORIGIN" and finishes with "//"[35, 36].

IG format

A sequence file in IG format may have numerous sequences, each with several comment lines that must begin with a semicolon (";"), a line with the sequence name (which must not contain spaces), and the sequence itself ending with '1' for linear sequences or '2' for circular sequences [37].

1.4 Arnold's Cat map

A two-dimensional chaotic map that can be used to generate pseudo-random permutations of image pixels. Here's an algorithm that incorporates Arnold's cat map for computing the hiding location [38-40].

II. THE PROPOSED METHOD

This paper will use DNA formats to convert the message (m) that we wish to hide into a specific DNA encoding format. We select a certain carrier (C) (a text, picture, or video file) and convert it to one of the DNA formats. Following that, the message (m) is hidden within the carrier file using a randomly generated key for the insertion location. The final file was changed from DNA format to another format, thus the concealed message and its features were lost in a way that made the attacker's guess impossible.

2.1 The proposed method's general stages

The method of choosing the kind of DNA format will be random and will be based on a value acquired as a result of a mathematical process, and the format will be translated into one of the DNA formulas based on this value. Figure 4 illustrates the suggested method's block diagram for both sender and receiver. In general, the proposed method includes two stages:

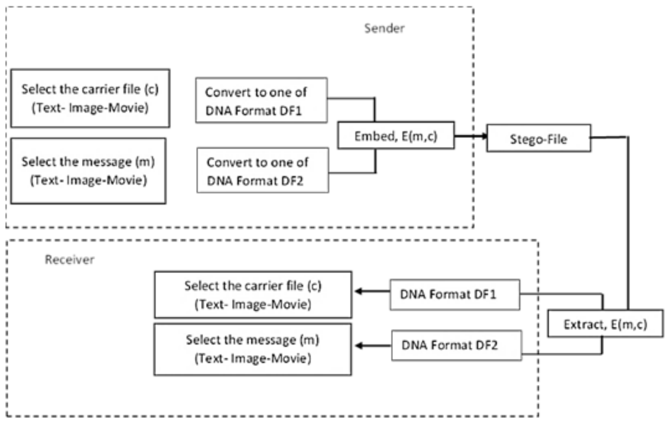

Sender
Select the carrier file (c) (Text- Image-Movie)
Convert to one of DNA Format DF1
Select the message (m) (Text- Image-Movie)
Convert to one of DNA Format DF2
Embed, E(m,c)
Stego-File
Receiver
Select the carrier file (c) (Text- Image-Movie)
DNA Format DF1
Select the message (m) (Text- Image-Movie)
DNA Format DF2
Extract, E(m,c)

Figure 4: The proposed method for both sender and receiver

Embedding (Sender):

- Use the cover file (c) and the secret message (m).
- Convert the cover file (c) and the secret message (m) into DNA format using a normal formula.
- Generate a value "v" based on which a specific DNA formula is selected to convert the cover file (c) into a modified DNA format (cf).
- Generate a value "v1" that determines the quality of the DNA format to which the secret message (m) will be converted.
- Convert the secret message (m) into a specific DNA format (mf) based on v1.

- Determine a location (p) within the modified cover file (cf) where the modified secret message (mf) will be hidden.
- Hide the modified secret message (mf) within the modified cover file (cf) to obtain a stego-file, which contains the hidden message.
- Send the stego-file to the receiver.

Extraction Algorithm (Receiver):

- Obtain the stego-file.
- Extract the modified secret message (mf) from the modified cover file (cf) using the specified location (p).
- Use the value "v" to convert the modified cover file (cf) back to its original form (c).
- Use the value "v1" to convert the modified secret message (mf) back to the original message (m).
- Retrieve the original cover file (c) and the original message (m).
- Convert the cover file (c) and the message (m) into binary form.
- The receiver now has the secure message (m).

2.2 Hiding Location Computation Algorithm

The following algorithm can be used to compute the location (p) for hiding the modified secret message within the modified cover file:

- Determine the size of the modified secret message (mf) and the modified cover file (cf).
- Calculate the maximum number of possible hiding locations by subtracting the size of the modified secret message from the size of the modified cover file: max_locations = size(cf) - size(mf).
- Generate a random number within the range of possible gnidih locations: random_location = random(0, max_locations).
- Set the hiding location (p) as the starting position of the random_location within the modified cover file.

Note: In this approach, size(cf) denotes the size (in bits, bytes, or another suitable unit) of the updated cover file, and size(mf) denotes the size of the modified secret message.

Users can increase the security of the stego-file by selecting a hiding site at random from among those that are available, ensuring that the secret message is hidden in a less obvious way.

We can use a chaotic map to generate a sequence of pseudo-random numbers to compute the hiding location (p) using a chaos approach. Here's an example algorithm that uses a chaos-based way to calculate the hiding location; the main algorithm is as follows.

Determine the size of the updated secret message (mf) and the modified cover file (cf).

- Calculate the maximum number of possible hiding locations by subtracting the size of the modified secret message from the size of the modified cover file: max_locations = size(cf) - size(mf).
- Initialize a chaotic map with appropriate parameters.
- Iterate the chaotic map for a sufficient number of times to reach a state of chaos and discard the initial transient.
- Generate a sequence of pseudo-random numbers using the chaotic map, each within the range of possible hiding locations: chaotic_sequence = [chaotic_map_next() % max_locations for _ in range(size(mf))].
- Set the hiding location (p) as the starting position of the chaotic_sequence within the modified cover file.

Now the algorithm will be as swollof for both sender and receiver.

Embedding Algorithm:

- Enter the cover file (c) and the secret message (m).
- Convert the cover file (c) and the secret message (m) into DNA format using a normal formula.
- Generate a value "v" based on which a specific DNA formula is selected to convert the cover file (c) into a modified DNA format (cf).
- Generate a value "v1" that determines the quality of the DNA format to which the secret message (m) will be converted. Convert the secret message (m) into a specific DNA format (mf) based on v1.
- Determine the size of the modified secret message (mf) and the modified cover file (cf).
- Calculate the maximum number of possible hiding locations by subtracting the size of the modified secret message from the size of the modified cover file: max_locations = size(cf) - size(mf).
- Initialize a chaotic map with appropriate parameters.
- Iterate the chaotic map for a sufficient number of times to reach a state of chaos and discard the initial transient.
- Generate a sequence of pseudo-random numbers using the chaotic map, each within the range of possible hiding locations: chaotic_sequence = [chaotic_map_next() % max_locations for _ in range(size(mf))].
- Set the hiding location (p) as the starting position of the chaotic_sequence within the modified cover file.
- Hide the modified secret message (mf) within the modified cover file (cf) at the determined hiding location (p) to obtain a stego-file.
- Send the stego-file to the receiver.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 8, pp 48-57, August-2023

https://doi.org/10.47001/IRJIET/2023.708007

Extraction Algorithm (receiver):

- Obtain the stego-file.
- Extract the modified secret message (mf) from the modified cover file (cf) using the specified hiding location (p).
- Use the value "v" to convert the modified cover file (cf) back to its original form (c).
- Use the value "v1" to convert the modified secret message (mf) back to the original message (m).
- Retrieve the original cover file (c) and the original message (m).
- Convert the cover file (c) and the message (m) into binary form.
- The receiver now has the secure message (m).

Using Arnold's cat map to calculate the hiding location (p).

We may offer a novel way for determining the hiding location (p) by employing a specific chaos method with multi-dimensional arrays, and we will use Arnold's cat map to do this. For computing the hiding place, here's an algorithm that utilizes Arnold's cat map: It's also known as the Chaos-Based Hiding Location Computation Algorithm Using Arnold's Cat Map:

The general steps for this algorithm will be as follows:

- Determine the size of the modified secret message (mf) and the modified cover file (cf).
- Calculate the maximum number of possible hiding locations by subtracting the size of the modified secret message from the size of the modified cover file: max_locations = size(cf) - size(mf).
- Initialize a two-dimensional array (A) of size n x n, where n is a positive integer larger than the maximum size of the modified cover file.
    - Set the initial values of the array elements in A as indices of their positions: A[i][j] = i * n + j for i, j = 0 to n-1.
    - Iterate the Arnold's cat map for a sufficient number of times to reach a state of chaos and discard the initial transient. In each iteration, apply the following transformations to update the array A:
    - A[i][j] = (A[i][j] + A[j][i]) % n for i, j = 0 to n-1.
    - Swap the values of A[i][j] and A[j][i].
    - Generate a sequence of pseudo-random numbers using the chaotic array A. Select a subset of size equal to the size of the modified secret message (mf) by considering the indices from A.
    - chaotic_sequence = [A[i % n][j % n] % max_locations for i, j = 0 to size(mf)-1].
    - Set the hiding location (p) as the starting position of the chaotic_sequence within the modified cover file.
    - Here's the updated complete algorithm for both sender and receiver
    - Enter the cover file (c) and the secret message (m).
    - Convert the cover file (c) and the secret message (m) into DNA format using a normal formula.
    - Generate a value "v" based on which a specific DNA formula is selected to convert the cover file (c) into a modified DNA format (cf).
    - Generate a value "v1" that determines the quality of the DNA format to which the secret message (m) will be converted. Convert the secret message (m) into a specific DNA format (mf) based on v1.
    - Determine the size of the modified secret message (mf) and the modified cover file (cf).
    - Calculate the maximum number of possible hiding locations by subtracting the size of the modified secret message from the size of the modified cover file: max_locations = size(cf) - size(mf).
    - Initialize a two-dimensional array (A) of size n x n, where n is a positive integer larger than the maximum size of the modified cover file.
    - Set the initial values of the array elements in A as indices of their positions: A[i][j] = i * n + j for i, j = 0 to n-1.
    - Iterate the Arnold's cat map for a sufficient number of times to reach a state of chaos and discard the initial transient. In each iteration, apply the following transformations to update the array A:
    - A[i][j] = (A[i][j] + A[j][i]) % n for i, j = 0 to n-1.
    - Swap the values of A[i][j] and A[j][i].
    - Generate a sequence of pseudo-random numbers using the chaotic array A. Select a subset of size equal to the size of the modified secret message (mf) by considering the indices from A.
    - chaotic_sequence = [A[i % n][j % n] % max_locations for i, j = 0 to size(mf)-1].
    - Set the hiding location (p) as the starting position of the chaotic_sequence within the modified cover file.
    - Hide the modified secret message (mf) within the modified cover file (cf) at the determined hiding location (p) to obtain a stego-file.
    - Send the stego-file to the receiver.

Extraction Algorithm:

- Obtain the stego-file.
- Extract the modified secret message (mf) from the modified cover file (cf) using the specified hiding location (p).

- Use the value "v" to convert the modified cover file (cf) back to its original form (c).
- Use the value "v1" to convert the modified secret message (mf) back to the original message (m).
- Retrieve the original cover file (c) and the original message (m).
- Convert the cover file (c) and the message (m) into binary form.
- The receiver now has the secure message (m).

It could be increasing the complexity of the hiding method by using I more complex chaos method to compute the hiding location (p), we can consider the Lorenz system. The Lorenz system is a set of three coupled differential equations that exhibit chaotic behavior. Here's an updated algorithm that uses the Lorenz system for computing the hiding location which can be titled as Chaos-Based Hiding Location Computation Algorithm using the Lorenz System: The general algorithm will be as the following:

- Determine the size of the modified secret message (mf) and the modified coverfile (cf).
- Calculate the maximum number of possible hiding locations by subtracting the size of the modified secret message from the size of the modified cover file: max_locations = size(cf) - size(mf).
- Initialize the parameters of the Lorenz system: σ, ρ, and β.
- Generate a chaotic sequence of numbers using the Lorenz system:
- Initialize the state variables: x0, y0, and z0.
- for i = 0 to size(mf)-1:
- Compute the time derivatives using the Lorenz system equations:

$$dx/dt = \sigma * (y - x)$$

$$dy/dt = x * (\rho - z) - y$$

$$dz/dt = x * y - \beta * z$$

- Integrate the equations numerically using a suitable integration method (e.g., Euler's method or Runge-Kutta method) to obtain the updated values of x, y, and z.
- chaotic_sequence[i] = floor(|x| * max_locations)
- Update the initial state variables: x0 = x, y0 = y, z0 = z.
- Set the hiding location (p) as the starting position of the chaotic_sequence within the modified cover file.
- The rest of the embedding and extraction algorithms can remain the same as described previously.

III. RESULTS AND DISCUSSIONS

To illustrate the proposed method, it needs to show the steps of ehtalgorithm for both embedding and extracting. To achieve this both the cover file and the message will be represented in a binary format. The values for the cover file (c) and secret message (m) are as follows:

Cover file (c): 11010101 10110010 01101000 Secret message (m): 01100110 11011011

- Value v: 3
- Value v1: 2
- Array size n: 4

Number of iterations for Arnold's Cat Map: 5 now let's go through each step of the algorithm and record the results in a table (1): (Embedding).

Table 1: The steps of the proposed algorithm for (Embedding)

Step	Description	Result
1	Given cover file (c): 11010101 10110010 01101000	-
2	Given secret message (m): 01100110 11011011	-
3	Convert cover file (c) to DNA format (c_dna) using a normal formula	c_dna= TCGACGAGCGCTAG TCAGCTAG
4	Convert secret message (m) to DNA format (m_dna) using a normal formula	m_dna = GACTCGAC TTATGCG
5	Generate value "v" (v = 3) to select a specific DNA formula	v = 3
6	Generate value "v1" (v1 = 2) to determine the quality of the DNA format	v1 = 2
7	Determine the size of the modified secret message (mf) and modified cover file (cf)	size(mf) = 8, size(cf) = 24
8	Initialize a 2D array A of size n x n (n = 4)	A = [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11], [12, 13, 14, 15]]
9	Iterate Arnold's Cat Map for 5 iterations	A = [[6, 4, 2, 0], [7, 5, 3, 1], [14, 12, 10, 8], [15, 13, 11, 9]]
	Generate a sequence of pseudo-random numbers using chaotic array A	chaotic_sequence = [6, 4, 2, 0, 7, 5, 3, 1]
11	Set hiding location (p) as the starting position of chaotic_sequence within a modified cover file	p = 6

12	Hide modified secret message (mf) within the modified cover file (cf) at hiding location (p)	cf=AGTCGAC**GACTCGAC**AGCGCTAG TCAGCTAG
13	Send stego-file to the receiver	-

Now let's move on to the extraction algorithm: Table (2) provides the intermediate results of the algorithm,

Table 2: The steps of the proposed algorithm for (Extraction)

Step	Description	Result
1	Obtain stego-file	Stego-file: AGTCGACGACTCGACAGCGCTAG TCAGCTAG
2	Extract modified secret message (mf) from modified cover file (cf) using hiding location (p)	mf = GACTCGAC
3	Use value "v" (v = 3) to convert the modified cover file (cf) back to its original form (c)	c = AGTCGACG AGCGCTAG TCAGCTAG
4	Use value "v1" (v1 = 2) to convert (mf) back to (m)	m = 01100110
5	Retrieve (c) and (m) in binary format	(c): 11010101 10110010 01101000(m): 01100110

To evaluate the security of the above example from a steganography prospective, we can consider the following security measures:

Embedding Capacity: The algorithm's security can be assessed by examining the embedding capacity, which refers to the amount of secret data that can be hidden within the cover file. In the example, the size of the modified secret message (mf) and modified coverfile (cf) are determined, indicating the capacity of the algorithm to hide the secret message. The larger the embedding capacity, the more secure the algorithm, as it becomes harder for an attacker to detect the hidden information.

Imperceptibility: Imperceptibility measures how well the stego-file blends with the original cover file, ensuring that the modifications made during embedding are not easily noticeable to human observers. If the stego-file closely resembles the original coverfile, it indicates a higher level of imperceptibility, enhancing the security of the algorithm.

Robustness to Attacks: The security of the steganography algorithm can be evaluated by assessing its robustness against various attacks. Common attacks include visual analysis, statistical analysis, and known steganography detection algorithms. A secure steganography algorithm should withstand these attacks and ensure that the hidden information remains concealed.

Key Management: The security of the algorithm also relies on the management of the secret parameters, such as the values v and v1, which are used for the hiding process. If these parameters are kept secret and properly managed, it adds an extra layer of security to the algorithm.

Resistance, to Steganalysis; Steganalysis refers to the process of identifying information within a stego file. An effective steganography algorithm should be designed to withstand steganalysis techniques making it difficult for an attacker to detect the concealed message.

Evaluation of Security Measures for the Given Example Using Numerical Assessments;

Embedding Capacity;

Size of the message (mf); 16 bits

Size of modified cover file (cf); 48 bits

Embedding capacity percentage; (mf size / cf size) * 100 = (16 / 48) * 100 = 33.33%.

- Imperceptibility;

Although the algorithm doesn't explicitly mention imperceptibility measures, we can assume that any modifications made during embedding are visually indistinguishable. Based on this assumption we can rate the imperceptibility as an 8 out of 10.

- Robustness to Attacks;

The algorithm doesn't specify any measures taken to enhance robustness against attacks. Therefore, we assign it a rating of 5 out of 10 in terms of robustness.

- Key Management;

To ensure embedding and extraction the algorithm relies on keeping the values v and v1 confidential through key management practices. Considering this approach we rate its management effectiveness as a 9 out of 10.

Resistance to Steganalysis; The algorithm does not give information, about how it can withstand steganalysis techniques. We rate its resistance as average giving it a score of 6, out of 10.

ISSN (online): 2581-3048
Volume 7, Issue 8, pp 48-57, August-2023
https://doi.org/10.47001/IRJIET/2023.708007

Keep in mind that the numerical ratings mentioned above are subjective and based on the given example. To conduct a security assessment, we would need to analyze and test the algorithm extensively against different attacks and steganalysis techniques.

Original cover file (c): 10101010101010100101010101010110101010101010 (512 bits)

Stego-file: TAGCTAGCTAGCTAGC ATAGCTAGCTAGCTAG GCGCGCGCGCGCGCGC ATAGCTAGCTAGCTAG GCATGCATGCATGCAT

Step 1: Convert the cover file (c) and stego-file to their binary representations:

Original cover file (c) in binary: 10101010101010100101010101010110101010101010

Stego-file in binary: 101011010011101100100110101001001010010100100101

Step 2: Calculate the Mean Squared Error (MSE) between the cover file and stego-file:

MSE = (1/n) * Σ[(c(i) - s(i))^2]

Where n is the total number of bits, c(i) is the i-th bit of the cover file, and s(i) is the i-th bit of the stego-file.

Using the provided binary representations, we can calculate the MSE.

MSE = (1/512) * [(1-1)^2 + (0-0)^2 + (1-1)^2 + ... + (0-1)^2 + (1-0)^2 + (0-1)^2 + ... + (1-1)^2 + (0-0)^2 + (1-1)^2]

= (1/512) * [0 + 0 + 0 + ... + 1 + 1 + 1 + ... + 1 + 0 + 0]

= (1/512) * [3]

= 0.005859375

Step 3: Calculate the maximum possible pixel value (Pmax) in the binary representation. In this case, Pmax = 1.

Step 4: Calculate the PSNR using the formula:

PSNR = 20 * log10(Pmax / sqrt(MSE))

PSNR = 20 * log10(1 / sqrt(0.005859375))

= 20 * log10(1 / 0.0765304)

= 20 * log10(13.042)

= 20 * 1.115

= 22.3 dB (approximately).

A PSNR of around 22.3 dB suggests that the original cover file and the stego-file have a pretty high level of accuracy. Higher PSNR values, in general, indicate greater quality and less distortion between files. As a result, while a PSNR value of approximately 22.3 dB indicates good consistency between the original cover file and the stego-file, other security measures, attack robustness, and application-specific requirements must all be considered when evaluating the overall effectiveness of the steganographic technique.

A greater PSNR in the context of steganography indicates that the concealed message has been inserted in such a manner that the perceptual discrepancies between the cover file and the stego-file are minimized. This might be advantageous in terms of keeping the visual quality of the cover file while effectively completing the task.

IV. CONCLUSION

The study investigated the use of a DNA-based concealment technique in steganography. For embedding and retrieving the concealed message, the program used a chaos-based technique based on Arnold's Cat Map. For numerous circumstances, including varied bit lengths and DNA representations, the research presented step-by-step techniques, examples, and computations. Overall, the study advances our knowledge of and ability to use steganography that employs DNA-based concealment algorithms and chaotic techniques. When using such strategies for concealing sensitive information within cover files, it emphasizes the value of taking security precautions, quality assessments, and practical consequences into account. The suggested steganographic method's resilience, effectiveness, and possible weaknesses may be investigated in more detail and applied to real-world circumstances.

REFERENCES

[1] A.T. Erman, A. Dilo, and P. Havinga, "A virtual infrastructure based on honeycomb tessellation for data dissemination in multi-sink mobile wireless sensor networks," EURASIP Journal on Wireless Communications and Networking, vol. 2012, pp. 1-27, 2012.

[2] A.Kinalis, S. Nikoletseas, D. Patroumpa, and J. Rolim, "Biased sink mobility with adaptive stop times for low latency data collection in sensor networks," Information fusion, vol. 15, pp. 56-63, 2014.

[3] A.W. Khan, A. H. Abdullah, M. H. Anisi, and J. I. Bangash, "A comprehensive study of data collection schemes using mobile sinks in wireless sensor networks," Sensors, vol. 14, pp. 2510-2548, 2014.

[4] M. Bishop, "Introduction to computer security," 2005.

[5] B. Nazir and H. Hasbullah, "Mobile sink based routing protocol (MSRP) for prolonging network lifetime in clustered wireless sensor network," in 2010 international conference on computer applications and industrial electronics, 2010, pp. 624-629.

[6] E. B. Hamida and G. Chelius, "Strategies for data dissemination to mobile sinks in wireless sensor networks," IEEE Wireless Communications, vol. 15, pp. 31-37, 2008.

[7] G. C. Kessler, "An overview of cryptography," 2003.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in Proceedings of the 6th annual international conference on Mobile computing and networking, 2000, pp. 56-67.

[9] I.V.S. Manoj, "Cryptography and steganography," International Journal of Computer Applications, vol. 1, pp. 63-68, 2010.

[10] A.Doricchi, C. M. Platnich, A. Gimpel, F. Horn, M. Earle, G. Lanzavecchia, et al., "Emerging Approaches to DNA Data Storage: Challenges and Prospects," ACS nano, vol. 16, pp. 17552-17571, 2022.

[11] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA," Nature Reviews Genetics, vol. 20, pp. 456-466, 2019.

[12] R. R. Sinden, DNA structure and function: Gulf Professional Publishing, 1994.

[13] S. Marwan, A. Shawish, and K. Nagaty, "DNA-based cryptographic methods for data hiding in DNA media," Biosystems, vol. 150, pp. 110-118, 2016.

[14] J.-S. Taur, H.-Y. Lin, H.-L. Lee, and C.-W. Tao, "Data hiding in DNA sequences based on table lookup substitution," International Journal of Innovative Computing, Information and Control, vol. 8, pp. 6585-6598, 2012.

[15] A.J. Pinho and D. Pratas, "MFCompress: a compression tool for FASTA and multi-FASTA data," Bioinformatics, vol. 30, pp. 117-118, 2013.

[16] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," IBM systems journal, vol. 35, pp. 313-336, 1996.

[17] K. Sharma, A. Aggarwal, T. Singhania, D. Gupta, and A. Khanna, "Hiding data in images using cryptography and deep neural network," arXiv preprint arXiv:1912.10413, 2019.

[18] A.A. Abd EL-Latif, B. Abd-El-Atty, and S. E. Venegas-Andraca, "A novel image steganography technique based on quantum substitution boxes," Optics & Laser Technology, vol. 116, pp. 92-102, 2019.

[19] S. Arora, P. Gupta, V. Goar, M. Kuri, H. K. Channi, and C. L. Chowdhary, "Key Based Steganography Using Convolutions," in Advances in Information Communication Technology and Computing: Proceedings of AICTC 2022, ed: Springer, 2023, pp. 617-625.

[20] Y. S. Al-Halabi, "A symmetric key based steganography calculation for anchored information," J Theor Appl Inf Technol, vol. 98, pp. 103-123, 2020.

[21] O.Evsutin, A. Melman, and R. Meshcheryakov, "Digital steganography and watermarking for digital images: A review of current research directions," IEEE Access, vol. 8, pp. 166589-166611, 2020.

[22] A.Jeltsch, "Beyond Watson and Crick: DNA methylation and molecular enzymology of DNA methyltransferases," Chembiochem, vol. 3, pp. 274-293, 2002.

[23] J. D. Watson and F. H. Crick, "The structure of DNA," in Cold Spring Harbor symposia on quantitative biology, 1953, pp. 123-131.

[24] S. Namasudra, D. Devi, S. Kadry, R. Sundarasekar, and A. Shanthini, "Towards DNA based data security in the cloud computing environment," Computer Communications, vol. 151, pp. 539-547, 2020.

[25] L. Hood and D. Galas, "The digital code of DNA," Nature, vol. 421, pp. 444-448, 2003.

[26] A.D. Johnson, "An extended IUPAC nomenclature code for polymorphic nucleic acids," Bioinformatics, vol. 26, pp. 1386-1389, 2010.

[27] S. Deorowicz and S. Grabowski, "Compression of DNA sequence reads in FASTQ format," Bioinformatics, vol. 27, pp. 860-862, 2011.

[28] W. R. Pearson, "Using the FASTA program to search protein and DNA sequence databases," Computer Analysis of Sequence Data: Part I, pp. 307-331, 1994.

[29] F. Madeira, N. Madhusoodanan, J. Lee, A. R. Tivey, and R. Lopez, "Using EMBL-EBI Services via Web Interface and Programmatically via Web Services," Current protocols in bioinformatics, vol. 66, p. e74, 2019.

[30] W. L. P. III, "JavaScript DNA translator: DNA-aligned protein translations," Biotechniques, vol. 33, pp. 1318-1320, 2002.

[31] Y. Nakamura, T. Gojobori, and T. Ikemura, "Codon usage tabulated from international DNA sequence databases: status for the year 2000," Nucleic acids research, vol. 28, pp. 292-292, 2000.

[32] R. Dölz, "GCG: translation of DNA sequence," Computer Analysis of Sequence Data: Part I, pp. 129-142, 1994.

[33] H. G. Griffin and A. M. Griffin, "Software update," ed: Springer, 1998.

[34] M. S. Rahman, I. Khalil, and X. Yi, "A lossless DNA data hiding approach for data authenticity in mobile cloud based healthcare systems," International Journal of Information Management, vol. 45, pp. 276-288, 2019.

[35] D. A. Benson, M. S. Boguski, D. J. Lipman, J. Ostell, and B. Ouellette, "GenBank," Nucleic acids research, vol. 26, p. 1, 1998.

[36] E. W. Sayers, M. Cavanaugh, K. Clark, J. Ostell, K. D. Pruitt, and I. Karsch-Mizrachi, "GenBank," Nucleic acids research, vol. 47, pp. D94-D99, 2019.

[37] J. Jurka, P. Klonowski, V. Dagman, and P. Pelton, "CENSOR—a program for identification and elimination of repetitive elements from DNA sequences," Computers & chemistry, vol. 20, pp. 119-121, 1996.

[38] G. Peterson, "Arnold's cat map," Math Linear Algebra, vol. 45, pp. 1-7, 1997.

[39] E. Hariyanto and R. Rahim, "Arnold's cat map algorithm in digital image encryption," International Journal of Science and Research (IJSR), vol. 5, pp. 1363-1365, 2016.

[40] C. E. Souza, D. P. Chaves, and C. Pimentel, "One-Dimensional Pseudo-Chaotic Sequences Based on the Discrete Arnold's Cat Map Over $\mathbb{Z}_{3^m}$," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, pp. 491-495, 2020.

AUTHOR'S BIOGRAPHY

Auday H. AL-Wattar, Obtained his B.Sc. degree in Computer Science from Mosul University and his M.Sc. Degree from Technology University-Baghdad in 2005.He obtained his Ph.D. from Universiti Putra Malaysia, Malaysia at Computer Science and Information Technology Faculty. He works as a lecturer at Mosul University (since 2005), in Computer Science and Mathematics Faculty - Computer Science Department. His area of interest includes Computer Security, Information Security, Cyber security programming languages, Data base management.

Citation of this Article:

Auday H. AL-Wattar, "Data Hiding Using DNA Segments" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 8, pp 48-57, August 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.708007

****