Enhancing Medical Image Security using Coupled Tent Maps

Nayotama Pradipta - 13520089
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520089@std.stei.itb.ac.id

***Abstract*—The privacy and security on medical data is of greatest importance in the healthcare sector. In the digital era, information such as medical images should be sent through secure channel communications and under secure protocols. An additional layer of security can be implemented for patient's medical images using coupled tent maps. This paper seeks to integrate Coupled Tent Maps with Elliptic Curve Diffie-Hellman key exchange algorithm, further enhancing the confidentiality and integrity of patient's data. By leveraging these advanced cryptographic techniques, this paper offers a novel approach to fortifying the security framework around sensitive medical data.**

Keywords—Image Encryption, Chaos Theory, Tent Maps, Elliptic Curve Diffie-Hellman

I. Introduction

Patient confidentiality has been an important matter in the healthcare sector, especially since the start of digital age. Digital medical images, which often contain sensitive personal information, are usually vulnerable to third-party breaches that can cause negative implications for patient privacy and institutional integrity. One way of securing data transfer is using secure channels. However, relying on secure channels alone is not sufficient to prevent interception and tampering. In 2020, it was revealed that over 45 million medical imaging files were leaked due to unsecured servers and medical devices from around the world. These files, containing personally identifiable information (PII) and protected health information (PHI), were accessible without any requirement for a username or password, thereby presenting a straightforward opportunity for ransomware attacks and potential blackmail.

Parallel to the digitalization of healthcare, there has been a rapid advancement in medical imaging technology. Modern imaging techniques such as MRI, CT scans, and PET imaging provide intricate details and high-resolution images that are crucial for accurate diagnosis and treatment planning. However, the increasing resolution and complexity of these images introduces higher data security complexity. This escalation underscores the necessity for encryptions methods that can evolve in tandem with imaging technology to protect patient data effectively.

Further methods like advanced cryptography are needed to encrypt sensitive information in transit. Cryptography plays a crucial role in securing sensitive records through the help of advanced mathematical theories. Encrypted data would remain protected even in the event of a data breach. Attackers would not have the ability to decipher the contents without the specific cryptographic key. It is vital that all data, whether being transmitted through communication channels or stored in databases, be encrypted. This additional layer of security needs to be implemented by healthcare institutions to maintain the integrity and confidentiality of patient data.

Image encryption in cryptography involves passing the image through an encryption algorithm that transforms the pixel values and positions based on an initial parameter. The initial parameter can only be known by the sender and the receiver to prevent an attacker from decrypting the image. Without the appropriate parameter, an encrypted image should be inaccessible and indecipherable.

One of the best encryption methods for images is chaotic encryption. Chaotic encryption uses chaotic maps to utilize their characteristics of sensitivity to initial conditions and parameters, which exhibit deterministic yet unpredictable behavior. Numerous chaotic maps exist, such as the logistic map, tent map, and Arnold's cat map, each defined by its unique mathematical chaos function.

Chaotic maps need to be paired with a secure parameter exchange to ensure thorough security. One such exchange is the Elliptic Curve Diffie-Hellman key exchange which utilizes elliptic-curve cryptography. The ECDH key exchange is a form of key exchange that is used in symmetric encryption algorithms using a shared key. Through a single key exchange, both the sender and receiver can independently generate a shared key that remains consistent for both parties. This shared key is never transmitted through communication channels, making it an ideal candidate for the initial parameters in chaotic maps.

This paper aims to implement advanced cryptographic techniques, specifically chaotic encryption using Coupled Tent Maps and secure key exchange via ECDH. By doing so, it addresses the critical need for robust data protection in healthcare—ensuring both compliance with strict privacy laws and safeguarding against sophisticated cyber threats. The effectiveness of these cryptographic solutions will be demonstrated through a detailed analysis of the implementation results. Ultimately, the application of these methods could be adapted for other types of sensitive information in various sectors.

II. Theoretical Background

A. Medical Image

Medical images play a critical role in the health industry, serving as essential tools for image analysis and pattern recognition in the diagnosis of a patient. There are various types of medical image, ranging from X-rays, CT scans, magnetic resonance images (MRI), retinographies, and ultrasound images. Typically, access to these images is restricted to healthcare providers and the patients themselves. Medical institutions have differing protocols for the distribution of these personally identifiable images: some prefer distributing hard copies, while others opt for digital transmission. These images are classified as PII due to their inclusion of patient-specific details alongside the medical scans.

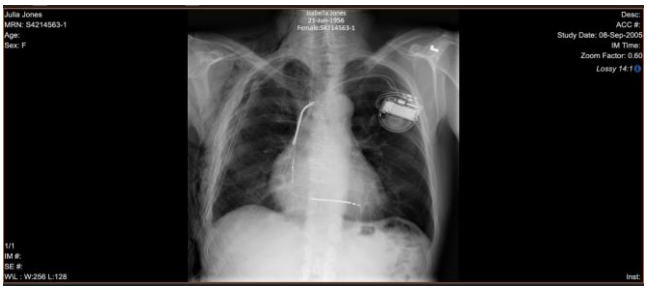

Fig. 1. X-Ray with patient-specific details [4]

Some medical images are inherently grayscale, such as X-rays in Figure 1, while others, like MRI scans, are typically viewed in grayscale but may be color-enhanced for detailed analysis. This difference requires careful consideration of encryption and decryption techniques for each type of image to ensure that there is no loss of information during file transfers.

B. Chaos Theory

Chaos theory is a branch of mathematics that studies complex system where seemingly random states of disorder and irregularities are actually governed by underlying patterns and deterministic laws. These systems are highly sensitive to initial conditions, a phenomenon often referred to as the butterfly effect. This effect manifests in computer science in several ways, particularly in algorithmic sensitivity and error propagation in numerical methods. For instance, floating-point arithmetic in computers can introduce tiny errors in calculation, which, when compounded through multiple operations in a large computational process, can cause significant variance in results.

Chaos theory has been applied in various scientific and engineering disciplines, not just computer science and mathematics. In meteorology, chaos theory is important for understanding weather patterns and climate systems. In fact, the term "butterfly effect" was famously associated with meteorologist Edward Lorenz, who observed that tiny changes in initial conditions of weather could yield drastically different forecasts. In economics, financial markets are often modeled as chaotic systems where small events can trigger significant economic shifts.

In cryptography, chaos theory is pivotal to develop robust encryption methods that are highly sensitive to small changes in the initial parameters. Cryptographic algorithms that utilize chaotic maps capitalize on this property to create encryption schemes that are extremely sensitive to key changes, hence enhancing security. Chaotic systems can be used to generate cryptographically secure pseudo-random numbers, or even to encrypt files. Image encryption using chaotic maps takes advantage of the high sensitivity characteristics to scramble the pixels of an image, thus rendering it unrecognizable without the correct decryption key.

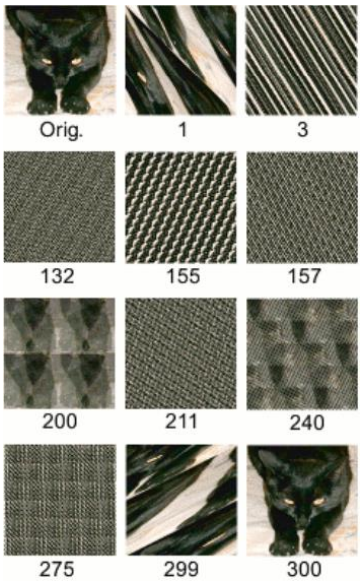

Fig. 2. Chaotic map encryption with different iterations [1]

C. Coupled Tent Maps

The tent map is a simple, yet fundamental example of a one-dimensional chaotic map. It is named for its graph shape that resembles a tent and is also known for demonstrating how simple rules can generate complex, chaotic behavior.

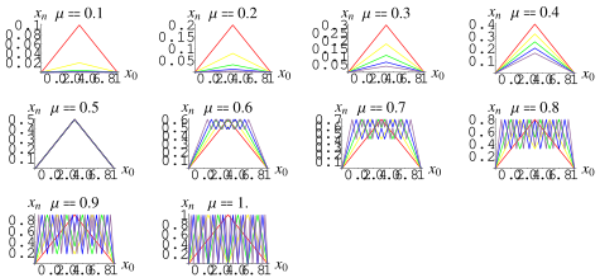

Fig. 3. Tent Maps with different parameters [5]

Tent maps can be used in cryptography to generate pseudo-random sequences. Just like other chaotic maps, tent maps can also be used to encrypt images by transforming pixels in a way that is highly sensitive to initial parameters. The tent map can be mathematically defined with the following function:

$$x_{i+1} = f_\mu(x_i) = \begin{cases} \mu x_i, & x_i < \frac{1}{2} \\ \mu(1 - x_i), & x_i \geq \frac{1}{2} \end{cases}$$

In an image encryption with tent map, there are several steps involved:

1. The process begins by setting the parameters of the tent map, particularly the slope parameter μ.
2. Each pixel value in the image is treated as an input to the tent map. The chaotic map then transforms these pixel values based on the function defined by the tent map.
3. The transformation is typically applied iteratively over several rounds, taking the output of the previous round as the new input.

Coupled tent maps extend the concept of the standard tent map by introducing interactions between two or more tent map systems, which can be used to further enhance the complexity and unpredictability of dynamical behaviors. This coupling can occur in various forms, such as through parameters or the states of the individual maps. In coupled tent maps, the next state of each map depends not only on its current state, but also on the states of other coupled maps. In image encryption, coupled tent maps can be implemented by using the output of one tent map as the input for another, thereby creating a sequence of transformations that increate the system's complexity and security.

D. Elliptic Curve Diffie-Hellman

Elliptic Curve Diffie-Hellman (ECDH) is an anonymous key agreement protocol that allows two parties, each having an elliptic curve public-private key pair, to create a shared secret over an insecure channel. This shared secret can be used to encrypt subsequent communications using a symmetric key cipher. The process involves each party multiplying their private key with the other party's public key. The result is a point on the elliptic curve that both parties can compute independently.

Integrating ECDH with coupled tent maps offers a solid approach to enhancing cryptographic systems. This integration can potentially thwart a range of attack vectors on cryptographic exchanges by making it more difficult for attackers to predict or determine the shared keys from intercepted communications.

One example of a standardized elliptic curve used in cryptography is *secp192r2 also* known as P-192. It falls under the category of curves defined for the 192-bit key size, offering a good balance between security and performance. P-192 is part of the SEC 2 standard for elliptic curve cryptography recommended by the Standards for Efficient Cryptography Group (SECG). This curve is designed to provide adequate protection for sensitive data by leveraging the hardness of Elliptic Curve Discrete Logarithm Problem (ECDLP). Its structure and parameters are chosen to optimize the efficiency of cryptographic operations while maintaining a high level of security, making it suitable for environments requiring rigorous data protection measures.

III. Implementation

A. Data and Setup

For the implementation of the image encryption, the testing utilizes medical X-ray data, which is inherently grayscale. The sample image employed in this study is sourced from the Google Cloud Architecture Documentation, which provides a clear example of typical X-ray data used in medical imaging studies.

Python is employed to conduct the implementation in this study. Libraries used in this study include *os* for random generation, *numpy* for array manipulations, *time* for time analysis, and *cv2* for comprehensive image processing capabilities. Furthermore, a specialized library was made specifically to facilitate ECDH key exchange that includes functions like inverse mod, point addition, and scalar multiplication.

B. ECDH Key Exchange Algorithm

The ECDH key exchange is initiated with the initialization of P-192 curve parameters, generator points, and order of generator. These values are then used in the creation of the key pairs. The values of those parameters used are as follows:

```
p  = 0xfffffffffffffffffffffffffffffffeffffffffffffff
a  = 0xfffffffffffffffffffffffffffffffefffffffffffffc
Gx = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
Gy = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811
n  = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831
```

The provided python code shows two principal functions utilized in the ECDH protocol. The *generate_key_pair* function is responsible for generating a cryptographic key pair for each client, comprising a private key derived from a random 24-byte number, and a corresponding public key computed via elliptic curve scalar multiplication. The *compute_shared_key* calculates a shared secret using a client's private key and another client's public key.

```
def generate_key_pair():
     privateKey = bytes_to_long(os.urandom(24)) % (n - 1) + 1
     publicKey = scalar_multiplication(privateKey, (Gx, Gy), p, a)
     return {
         'privateKey': hex(privateKey),
         'publicKey': (hex(publicKey[0]), hex(publicKey[1]))
     }


def compute_shared_key(privateKey, publicKey):
     privateKey = int(privateKey, 16)
     publicKey = (int(publicKey[0], 16), int(publicKey[1], 16))
     sharedPoint = scalar_multiplication(privateKey, publicKey, p, a)
  return hex(sharedPoint[0])
```

C. Coupled Tent Map Algorithm

The algorithm of the coupled tent map image encryption (grayscale) is as follows:

1. Define initial value x_0 within interval [0, 1] for the first sequence
2. Define initial value y_0 within interval [0, 1] for the second sequence by reversing the shared key
3. Define slope parameter μ
4. For each subsequent value x_i, calculate x_{i+1} using the tent map formula:

$$x_{i+1} = f_\mu(x_i) = \begin{cases} \mu x_i, & x_i < \frac{1}{2} \\ \mu(1 - x_i), & x_i \geq \frac{1}{2} \end{cases}$$

5. Calculate a second sequence based on the value of the first sequence, introducing a new variable α:

$$y_{i+1} = f_\mu(x_i, y_i) = \mu \cdot \left(\begin{cases} y_i + \alpha x_i, & y_i < \frac{1}{2} \\ 1 - (y_i + \alpha x_i), & y_i \geq \frac{1}{2} \end{cases} \right) mod\ 1$$

6. Scale the chaotic sequence result by multiplying it with 255
7. Add integer-scaled chaotic sequence to the original pixel values of the image and mod the result by 256

The following code shows parameter initialization (Step 1-3) of the coupled tent maps:

```
def modify_key(shared_key):
    modified_key = shared_key.strip('0x')[::-1]
    shared_key_int = int(modified_key, 16)
    initial_value2 = shared_key_int % 256 / 255
    return initial_value2


def init_tent_map_params(shared_key):
    shared_key_int = int(shared_key, 16)
    initial_value1 = shared_key_int % 256 / 255
    initial_value2 = modify_key(shared_key)
    slope = 1.9
 return initial_value1, initial_value2, slope
```

The implementation of the coupling sequences is done by creating a function that returns two sequences. These sequences are firstly initialized with zero with the length of the flattened pixels. The code below shows the creation of the first sequence, which is then used by the second sequence. The α value used in the code is 0.1 to ensure that the influence of the first sequence on the second is significant enough to affect the dynamics but not so large as to completely dominate the behavior of the second sequence.

```
def tent_map(length, initial_value1, initial_value2, slope):
        sequence1 = np.zeros(length)
        sequence2 = np.zeros(length)
        sequence1[0] = initial_value1
        sequence2[0] = initial_value2
        alpha = 0.1
        for i in range(1, length):
            if sequence1[i-1] < 0.5:
                sequence1[i] = slope * sequence1[i-1]
            else:
                sequence1[i] = slope * (1- sequence1[i-1])
            # Coupling
            if sequence2[i-1] < 0.5:
                sequence2[i] = slope * (sequence2[i-1] + alpha * sequence1[i-1]) % 1
            else:
                sequence2[i] = slope * (1- (sequence2[i-1] + alpha * sequence1[i-1])) % 1

         return sequence1, sequence2
```

The encryption itself is done by adding the two chaotic sequences with the original pixels, while the decryption is done through the subtraction of the encrypted pixels with the chaotic sequences. The implementation can be seen in the following code:

```
    def encrypt(pixels, chaotic_sequence1, chaotic_sequence2, shape):
        encrypted_pixels = (pixels + (chaotic_sequence1 * 255).astype(int) + (chaotic_sequence2 * 255).astype(int)) % 256
        return encrypted_pixels.reshape(shape)


    def decrypt(pixels, chaotic_sequence1, chaotic_sequence2, shape):
        decrypted_pixels = (pixels - (chaotic_sequence1 * 255).astype(int) - (chaotic_sequence2 * 255).astype(int)) % 256
     return decrypted_pixels.reshape(shape)
```

IV. Results and Analysis

After the completion of the coupled tent implementation, a comprehensive evaluation was performed on Figure 1. The results gained are not only the visual integrity of the encrypted and decrypted images but also the efficiency of the process in terms of time completion. Several tests were also conducted to analyze decryption results using altered shared keys. The results are as follows:

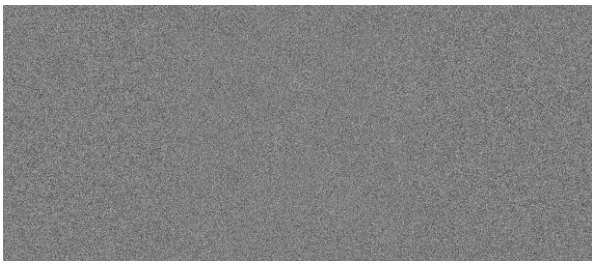

Fig. 4. Encrypted result of Figure 1

Figure 4 illustrates the effective transformation of the original image into a field of random black and white noise. The texture of the encrypted image is uniformly chaotic, with no patterns or structures visible that might allow an unauthorized viewer to infer any details about the underlying content.

When testing the encrypted result using Python's *pillow* library, a substantial difference was noted between the original grayscale image and its encrypted counterpart. Out of a total of 2730000 pixels, 2703762 pixels were found to be different. During the test, a pixel is different if the absolute difference of pixel *a* and *b* is larger than 1. This shows that there is a 99% difference of pixels between the original image and the encrypted image. The code to test the pixel difference between two files can be seen below:

```
    def     compare_images(image_path1,     image_path2,
tolerance=1):
        img1 = Image.open(image_path1).convert('L')
        img2 = Image.open(image_path2)
        if img1.size != img2.size:
            print("Images have different sizes.")
            return
        pixels1 = img1.load()
        pixels2 = img2.load()
        total_pixels = img1.width * img1.height
        print(f"Total    pixels    in    each    image:
{total_pixels}")
        diff_count = 0
        for x in range(img1.width):
            for y in range(img1.height):
                if abs(pixels1[x, y] - pixels2[x, y]) >
tolerance:
                    diff_count += 1
    print(f"The    images    differ    by    {diff_count}
significant pixels with a tolerance of {tolerance}.")
```

Decryption using the shared key of the receiver will result in the original file (grayscale). When testing the decrypted result using Python's *pillow* library, the decrypted image differs by 0 significant pixels with the original image (grayscale). The decryption result can be seen in Figure 5.

Fig. 5. Decrypted version of Figure 1

The encryption and decryption time for the original file (Figure 1), which is 635 KB in size, were 2.63 seconds and 2.57 seconds, respectively. This similarity shows the symmetrical nature of the algorithm used. Based on the time spent for encryption/decryption, the process took a reasonable amount of time, however these numbers are expected to increase with larger file size and incorporation of additional tent maps.

Further tests were conducted on different image sizes and types to analyze the effects of input size on the time needed for encryption and decryption. The results are as follows:

File Size	Encryption Time	Decryption Time
44 KB (PNG)	1.36 seconds	1.37 seconds
1.7 MB (JPG)	2.04 seconds	2.02 seconds
3 MB (PNG)	2.05 seconds	2.08 seconds

Based on the results above, it appears that the time needed for encryption and decryption for larger files remains relatively constant. This suggests that the algorithm is capable of handling large images efficiently, maintaining performance regardless of size. However, there were noticeable white noises in the decrypted JPG image as seen in Figure 6, meaning that the algorithm isn't perfectly suited to JPG inputs. The presence of these artifacts could be attributed to the lossy nature of JPG compression, which can introduce discrepancies when the image is encrypted and then decrypted.

Fig. 6. Noticeable white noise on a large JPG file

One potential solution to address this problem is to involve preprocessing steps to mitigate compression before encryption or use post-processing techniques to clean up the decrypted image.

Additional tests were made for a different shared key to understand the effect of small changes on the result. An initial shared key with value:

0x1762dc4bd0204d0cd9f9283c996d97d077d9e7f3561b3891,

is modified by changing the last hex value into:

0x1762dc4bd0204d0cd9f9283c996d97d077d9e7f3561b3892,
and results in the following decrypted image:

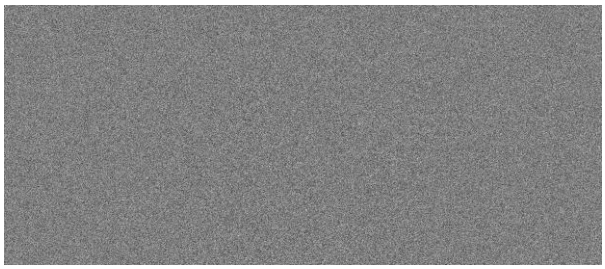

Fig. 6. Decrypted image with wrong shared key

Figure 6 illustrates the efficiency of chaotic maps in image encryption, where the slightest change in the initial parameters results in a completely different decryption image.

V. Conclusion

The implementation of coupled tent maps for image encryption has demonstrated significant success, particularly with PNG images. This success is characterized by the expected behavior of chaotic maps, akin to the butterfly effect, where small changes in the initial conditions can lead to drastically different outcomes. This inherent unpredictability and sensitivity to initial conditions are fundamental properties of the chaotic systems and are essential for ensuring the robustness and security of the encryption algorithm.

Currently, the implementation focuses on outputting grayscale images. While this serves as a proof of concept and demonstrates the algorithm's potential, further development is necessary to extend this capability to full color images. Handling full-color images would involve encrypting each color channel separately or finding a way to incorporate the coupled tent maps into a multi-dimensional color space. Additionally, expanding the implementation to support different types of image files is an important next step. While PNG images have been successfully encrypted and decrypted, further testing on other common formats such as JPEG, BMP, and TIFF also needs to be conducted to ensure no noises are encountered in the decrypted image.

Combining this encryption method with a secure communication channel will lead to a highly secure data transfer system. A secure channel ensures that the encrypted data remains protected during transmission, preventing interception and unauthorized access. When integrated with the coupled tent map encryption, the overall security of the data transfer process is greatly reinforced.

Furthermore, this combination is not limited to image data but is applicable to any type of digital data. The algorithm described in this paper is versatile and reusable across various forms of digital communication, whether it be text, audio, video, or other binary data. This broad applicability makes the coupled tent map encryption a powerful tool in the realm of digital security. Its ability to ensure the confidentiality and integrity of a wide range of data types enhances its utility in numerous fields beyond just medical information.

Code Repository

The implementation of the coupled tent maps can be accessed at GitHub through the following link: https://github.com/NayotamaPradipta/Coupled-Tent-Maps

Acknowledgment

I am deeply grateful to Dr. Ir. Rinaldi Munir, MT. at Bandung Institute of Technology, for the teachings and guidance he has provided. The knowledge and insights from his classes have been instrumental in enabling me to develop and write this paper.

References

[1] Munir, R. (2024). *Pembangkit Bilangan Acak*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/32-Pembangkit-bilangan-acak-2024.pdf

[2] Munir, R. (2024). Elliptic Curve Cryptography (ECC) (Bagian 2). https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/23-ECC-Bagian2-2024

[3] Summers, R. (2020). *X-Ray Images*. CXR8. https://nihcc.app.box.com/v/ChestXray-NIHCC

[4] Google. (n.d.). *De-identification of medical images through the Cloud Healthcare API | cloud architecture center | google cloud*. Google. https://cloud.google.com/architecture/de-identification-of-medical-images-through-the-cloud-healthcare-api?hl=en

[5] Weisstein, E. (2024). *Tent map*. Tent Map – from Wolfram MathWorld. https://mathworld.wolfram.com/TentMap.html

[6] BÎZGĂ, A. (2020). *Leaky database expose over 45 million medical images and patient data*. Hot for Security. https://www.bitdefender.com/blog/hotforsecurity/leaky-databases-expose-over-45-million-medical-images-and-patient-data/

Statement

I hereby declare that the paper I have authored is my original work, not a derivative or a translation of someone else's work, and is not plagiarized.

Bandung, 12 June 2024

Nayotama Pradipta