Double image encryption algorithm based on compressive sensing and elliptic curve final project

111753156 陳羽暉 , 111753115 黃紹禎 , 111753147 羅　揚

Abstract:

現在大家對於資料傳輸不再只侷限於訊息的傳輸，影像的傳輸變得越來越普遍。就如同訊息傳輸的過程當中需要保障資料的安全性，在影像傳輸的過程當中也會希望影像中的隱密信息不會被其他人得知，因此影像加密的演算法不斷的被提出，而我們的期末報告所選的主題就是影像加密，希望能透過一些影像的前處理之後再進行加密演算法進而達到只有指定接收者可以解密得到原本的影像。此演算法結合了壓縮感知和公鑰橢圓曲線加密，在減少多影像加密的數據量與傳輸量同時，保持多影像加密的大部分優點，從而提高影像傳輸數據的安全級別，抵禦各種攻擊。

Introduction:

隨著科技的發展，我們應該採取一些必要的措施來確保影像和視頻的安全傳輸，以維護個人和組織的隱私權。然而，現有的影像加密演算法存在一些問題。舉例來說，在安全性方面，某些現有的演算法可能存在安全漏洞或被破解的風險。而從複雜性和效能來看，有些演算法可能過於複雜，或是會產生大量的冗餘信息，導致加密和解密過程耗時且需要大量計算資源。此外，影像加密演算法需要使用密鑰來執行加密和解密操作，密鑰的生成、分發、保存和更新等過程對於確保加密系統的安全性至關重要，當有大量影像時也會導致傳輸成本提高。針對上述問題，作者根據三維混沌系統（Bsys）[1]推出改進的三維連續混沌系統（ImproBsys），並利用 ImproBsys 提出了一種基於壓縮感知和公鑰橢圓曲線的雙影像加密演算法。該系統表現出增強的混沌行為，在保持安全性的同時亦減少了數據傳輸。

Related work:

混沌系統和混沌映射由於具有遍歷性、不可預測性、隨機性和對初值極度敏感等特點，被廣泛應用於影像加密，[2]基於分數階混沌系統和 RSA 公鑰密碼體制提出了一種非對稱影像加密演算法。而超混沌系統因具有更複雜的動態特性，可以提升安全性，故比一般的混沌系統更受影像加密研究者的喜愛，[3]提出了一種基於超混沌系統和 Fibonacci Q 矩陣的影像加密演算法。

根據 Nyquist 理論，在模擬信息或數位訊號的轉換過程中，當採樣率大於訊號中最高頻率的兩倍時，採樣後的數位訊號將能夠完整保留原始訊號中的信息，但這對數位影像採樣來說，會產生大量的多餘信息。壓縮感知[4]作為一種新的採樣理論，其採樣率遠低於 Nyquist 採樣率，通過隨機採樣獲得訊號的離散樣本後，利用非線性重構方法可以近乎完美地重構原始訊號。在影像加密領域，已經許多人採用壓縮感知方法對影像進行壓縮，以減少加密數據，加快加密過程，[5]設計了一種基於壓縮感知和超混沌系統的明文相關影像加密演算法。

大多數基於壓縮感知的單影像加密方法需要傳輸測量矩陣作為密鑰，這導致額外的傳輸，尤其隨著影像數量的增加，傳輸成本顯著上升。因此，多影像加密演算法引起了研究人員的關注，它是將多個影像組合在一起進行同步加密，使加密後的影像相互關聯，從而提高密文對各種攻擊的抵抗力，並減少密鑰量，[6]利用壓縮感知和 Schur 分解設計了一種多影像視覺意義的加密演算法。

Preliminaries:

・ImproBsys chaotic system:

$$\begin{cases} x_0 = ayz \\ y_0 = bx - cy \\ z_0 = d - exy - f(xy) \end{cases}$$

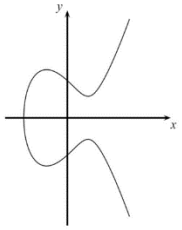
where $f(x) = B \cdot \mathrm{sign(x)} + \mathrm{B} \cdot \mathrm{sign(x - 2A)} + \mathrm{B} \cdot \mathrm{sign(x + 2A)},\ A = 2,\ B = 0.25,\ a = 1,\ b = 1,\ c = 1,\ d = 2,\ e = 1.$

$$sign(x) = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}.$$

・Elliptic curve:

Let F_p denotes the field of prime numbers. The elliptic curve over it is defined as the following,

$E: y^2 = x^3 + ax + b\ (mod\ p)$ where $p > 3$ and $4a^3 + 27b^2 \neq 0\ (mod\ p)$.



・Elliptic Curve Discrete Logarithm Problem (ECDLP):

Given a primitive element P and another element T on an elliptic curve E. The ECDL problem is finding the integer d, such that $P + P + \cdots + P = dp = T$.

・Advanced Encryption Standard (AES):

It is a block encryption standard adopted by the US federal government. This standard is used to replace the original DES, has been analyzed by many parties and is widely used all over the world. After a five-year selection process, the Advanced Encryption Standard was published by the National Institute of Standards and Technology (NIST) in FIPS PUB 197 on November 26, 2001, and became an effective standard on May 26, 2002. Today, the Advanced Encryption Standard is one of the most popular algorithms for symmetric key cryptography.

The AES encryption process operates on a 4×4 byte matrix, which is also called "state", and its initial value is a plaintext block (the size of an element in the matrix is one of the plaintext blocks). Byte). When encrypting, each round of AES encryption (except the last round) includes 4 steps:

AddRoundKey: In the AddRoundKey step, the round key will be merged with the original matrix. In each encryption cycle, a round key will be generated by the master key (generated through the Rijndael key generation scheme). The size of this key will be the same as the original matrix, and each corresponding Bytes are mutually exclusive or (⊕) addition.

SubBytes: In the SubBytes step, the bits in the matrix are transformed through an 8-bit S-box.

ShiftRows: ShiftRows describes the row manipulation of a matrix. In this step, each row is rotated left by some offset. In AES (128-bit block size), the first line remains unchanged, and each byte in the second line is rotated one space to the left. Similarly, the leftward cyclic shift offsets of the third row and the fourth row are 2 and 3 respectively. The cyclic shift modes of the 128-bit and 192-bit blocks at this step are the same. After ShiftRows, each vertical column in the matrix is composed of elements in each different column in the input matrix.

MixColumns: In the MixColumns step, the four bytes of each column are combined with each other by a linear transformation.

Method:

Encryption process:

In our method, Step 1 to Step 4 are the same as the paper[7]. But in Step 4, we get a matrix V with the size $H \times H$. Because of the V we get in Step 4, we will get a new matrix Y with the size $2H \times H$. And Step 6 is used to normalize the matrix Y in to the range $0 \sim 255$. Then we merge Step 7 to Step 10 in the paper into only one step, Step 7, we use Advanced Encryption Standard (AES) to encrypt the processed image, and use Elliptic Curve Cryptography (ECC) to encrypt the AES shared key.

Step 1: Given two images D and F, then do the three-layer discrete wavelet transform on each of the image, and get the coefficient matrices J and I. In this phase, we use the python pywt package.

Step 2: Take the absolute values of all elements in J to get a matrix N. Then sorted in descending order to get an array O. We just deal with the first 15% of the date, we use a threshold to get a matrix Q that if J_{ij} is less than threshold then $Q_{ij} = 0$, otherwise $Q_{ij} = J_{ij}$.

Step 3: Do the same thing in Step 2 to the matrix I and get a matrix R.

Step 4: Doing the ImproBsys chaotic system with the initial values

$$x_0 = \big((entropy(D) \times 10^{10})\ mod\ 1 + 0.36255\big)\ mod\ 1.$$
$$y_0 = \big((entropy(F) \times 10^{10})\ mod\ 1 + 0.45368\big)\ mod\ 1.$$
$$z_0 = (x_0 + y_0 + 0.24142)\ mod\ 1.$$

And get a chaotic sequence S with length H. Then sort the chaotic sequence S to get the index sequence T. Next generate a Hadamard matrix U with size $H \times H$ and create another matrix V such that $V = U\big(T(1{:}H), T(1{:}H)\big)$

Step 5: Let $W = V \times Q$ and $X = V \times R$ with size $H \times H$. And the matrix Y is the matrix W and X merge together.

Step 6: Normalize the matrix Y into $0{\sim}255$.

$$Z_{ij} = \Big(Y_{ij} - min(Y)\Big)\,\%255,\ i = 1,2,\dots,H,\ j = 1,2,\dots H.$$

In this phase, because the difference of the maximum and the minimum of the matrix Y is too big, if we use the method in the paper, it will cause big error when we want to restore the matrix, so we use the above method. Moreover, we also save the quotient after we do modulo to each element to make sure we can restore the matrix correctly.

Step 7: In this phase, we use python cryptography package to do all the encryption method. First we generate the ECC key pair and the AES shared key, and save the ECC key pair to file 'public_key.pem', 'private_key.pem', respectively. Second we read the matrix Z into the encrypt function and use the public key to get the encrypted image and the initial value of AES, and save encrypted image and initial value to file 'encrypted_image.dat' and 'iv.dat', respectively.

Decryption process:

The decryption process is the inverse of the encryption process, we just have to use the private key to decrypt the encrypted shared key, and use the decrypted shared key to decrypt the encrypted image. After we get the decrypted image, we do the inverse method from Step 6 to Step 1.

Inverse Step 7: First we load the private key and the encrypted shared key, then get the shared key. Second, we use the shared key to decrypt the encrypted image.

Inverse Step 6: As we said in Step 6, we save the quotient after doing modulo, so we just use the quotient matrix to get the matrix $Y_{restored}$ that is not normalized.

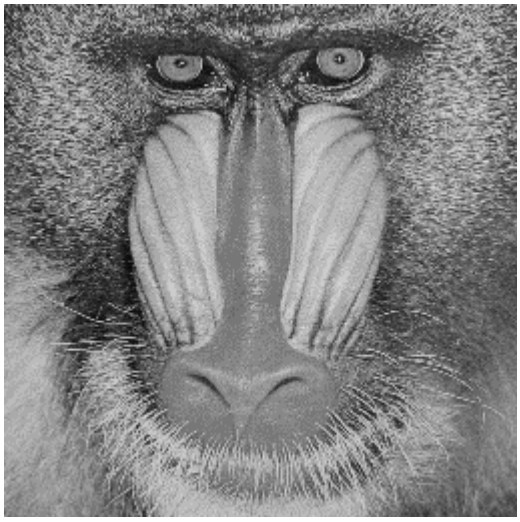
Inverse Step 5: Whether other people know matrix V or not, the method is still secure, so we just make the matrix to be a public parameter. So we calculate the inverse of the matrix V and the two matrices that split from the matrix $Y_{restored}$ are multiply by the inverse V. And get the two matrices $Q_{restored}$ and $R_{restored}$.

Inverse Step 4: Since we get the matrices $Q_{restored}$ and $R_{restored}$, so the inverse Step 4 is not necessary to do.

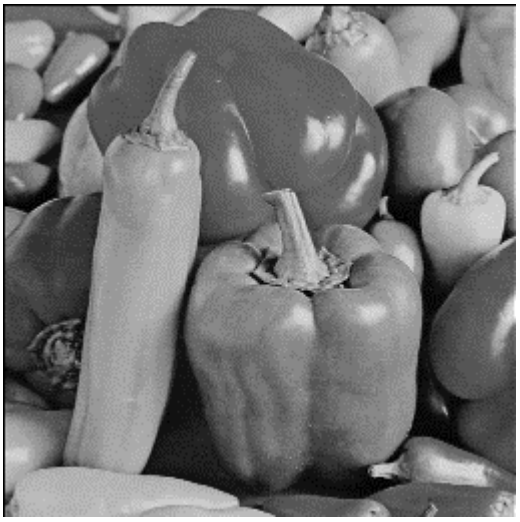
Inverse Step 3 to Step1: Now we use the inverse function of the three-layer discrete wavelet transform in the pywt package to the matrices $R_{restored}$ and $R_{restored}$ and get the restored image. Since in the step 1 to step 3 we only deal with the first 15% of the data that after sort, the restored image will not be exactly the same as the original image.

Experiment:

Our experiments are all run on python environment.

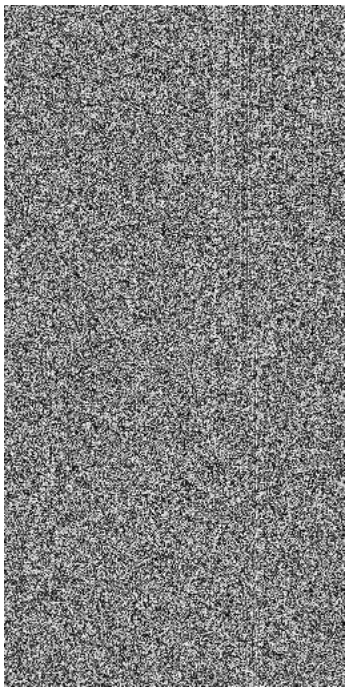
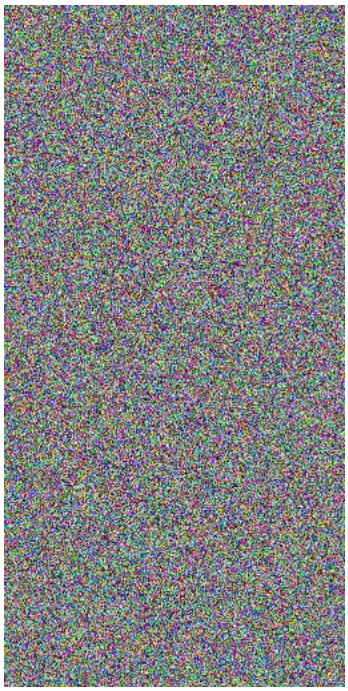
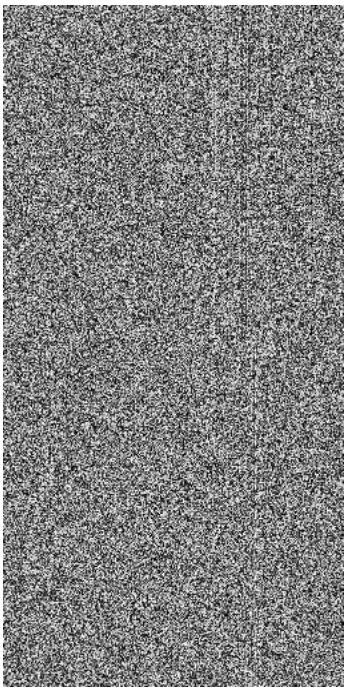


(a) Baboon (256×256)



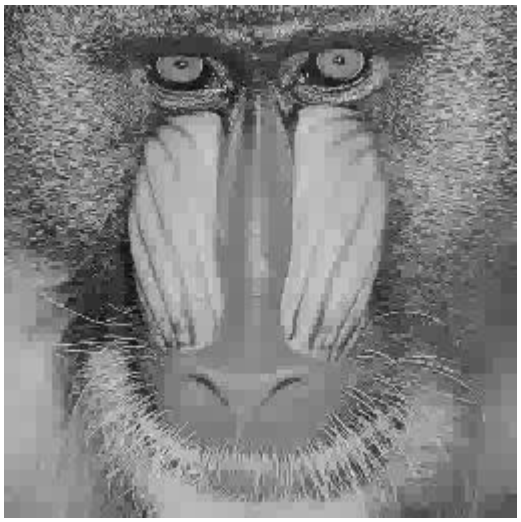
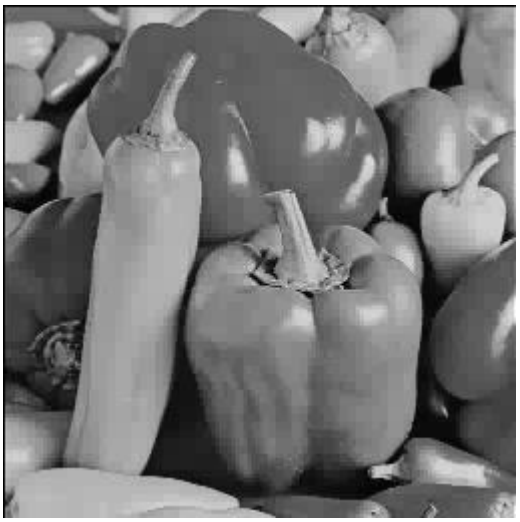
(b) Peppers (256×256)

First we use the baboon image (a) and peppers image (b) with the same size to do our experiment.

(c) Preprocessed image (d) Encrypted image (e) Decrypted image

The images above (c) (d) (e) are the image after we do Step 1 to Step 6, Step 7 and inverse Step 7 to Step 5, respectively.

(f) Restored baboon (256×256) (g) Restored peppers (256×256)

The image (f) and (g) are the image that we restored from the decrypted image.

After we get (f) and (g), we calculate the PSNR between the original image (a) (b) and restored image (f) (g).

Then we get the PSNR of (a) and (f) is 25.06, and the PSNR of (b) and (g) is 34.90.

Conclusion:

- The first 15% of the three-layer discrete wavelet transform retains the relatively important information of the image and reduces the amount of calculation.
- After doing ImproBsys, we can ensure that the processed images are random and sensitive enough, so that we can avoid someone who wants to tamper with the data or get any information of the original data.
- Using ECC and AES image encryption is theoretically faster than using ECC alone, because AES is a symmetric key encryption and ECC is more efficient in key exchange.

Actually, we didn't figure out how to use ECC to the image directly, so we can not show the time cost difference between only ECC and ECC with AES. And our experiment result is using PSNR to decide whether our scheme is good our not, but as you can see, our PSNR value are around 30, it means that we can perceive the difference between the restored image and the original image by human's eyes, we thinks it is because that we only deal with the first 15% of the data after doing the three-layer discrete wavelet transform. However, the first 15% retains the relatively important information of the image, that's why we can still believe that the content of the restored image is the same as the original image.

Reference:

[1] J.C. Sprott, Some simple chaotic flows, Phys. Rev. E 50 (2) (1994) R647–R650.
[2] G. Ye, K. Jiao, H. Wu, C. Pan, X. Huang, An asymmetric image encryption algorithm based on a fractional-order chaotic system and the RSA public-key cryptosystem, Int. J. Bifurcat. Chaos 30 (15) (2020) 2050233, https://doi.org/10.1142/S0218127420502338.
[3] K.M. Hosny, S.T. Kamal, M.M. Darwish, G.A. Papakostas, New image encryption algorithm using hyperchaotic system and fibonacci Q-matrix, Electronics 10 (9) (2021) 1066.
[4] E.J. Candes, M.B. Wakin, An introduction to compressive sampling, IEEE Signal Process. Mag. 25 (2) (2008) 21–30.
[5] W. Xiao-Qing, Z. Hao, S. Yu-Jie, W. Xing-Yuan, A plaintext- related image encryption algorithm based on compressive sensing and a novel hyperchaotic system, Int. J. Bifurcat. Chaos 31 (02) (2021) 2150021, https://doi.org/10.1142/S0218127421500218.

[6] G. Ye, C. Pan, Y. Dong, K. Jiao, X. Huang, A novel multi- image visually meaningful encryption algorithm based on compressive sensing and Schur decomposition, Trans. Emerg. Telecommun. Technol. 32 (2) (2021), https://doi.org/10.1002/ett.v32.210.1002/ett.4071.

[7] Double image encryption algorithm based on compressive sensing and elliptic curve - G Ye, M Liu, M Wu - Alexandria Engineering Journal, 2022 - Elsevier