**ASSIGNMENT REPORT: STREAM CIPHER**

|  |  |  |
| --- | --- | --- |
| **Instructor** | **PhD. Nguyễn Ngọc Tự** | |
| **Student** | Mai Nguyễn Phúc Minh | 23520930 |

**Stream Cipher**

1. **Arnold Cat Map**

* **Description: Arnold's cat map** is a **chaotic map** often used for pixel manipulation. It applies a transform on the image that essentially shuffles the pixels by stretching and folding the image. When an optimal number of iterations of the transformation is applied onthe image, the resulting image becomes incomprehensible and hence encrypted.
* **Function:** The transformation applied on the image is:

**R([x,y]) = [(x + y) mod n, (x + 2y) mod n]**

where n is the dimensions of the image

* **Note**: The image must be in **square dimension**, otherwise it cannot be decrypted.
* **Encryption code**:

# Arnold Cat Map function

def ArnoldCatTransform(img):

    rows, cols, ch = img.shape

    n = rows

    img\_arnold = np.zeros\_like(img)

    for x in range(n):

        for y in range(n):

            new\_x = (x + y) % n

            new\_y = (x + 2 \* y) % n

            img\_arnold[new\_x, new\_y] = img[x, y]

    return img\_arnold

# Encryption function

def ArnoldCatEncryption(imageName, key):

    img = np.array(Image.open(imageName).convert("RGB"))  # Convert image to RGB

    for \_ in range(key):

        img = ArnoldCatTransform(img)

    Image.fromarray(img.astype('uint8')).save(imageName.split('.')[0] + "\_ArnoldcatEnc.png")

    return img

* **Decryption code**

def ArnoldCatInverseTransform(img): # Inverse Arnold Cat Map function

    rows, cols, ch = img.shape

    n = rows

    img\_inverse = np.zeros\_like(img)

    for x in range(n):

        for y in range(n):

            new\_x = (2 \* x - y) % n

            new\_y = (-x + y) % n

            img\_inverse[x, y] = img[new\_x, new\_y]

    return img\_inverse

# Finding the period of Arnold's cat map

def find\_period(img\_size):

    img = np.arange(img\_size \* img\_size).reshape((img\_size, img\_size, 1))

    original = img.copy()

    count = 0

    while True:

        img = ArnoldCatTransform(img)

        count += 1

        if np.array\_equal(img, original):

            return count  # The cycle repeats after 'count' iterations

ecryption function

def ArnoldCatDecryption(imageName, key):

    img = np.array(Image.open(imageName).convert("RGB"))  # Convert image to RGB

    period = find\_period(img.shape[0])  # Find Arnold Map Period

    key = key % period  # Adjust key to be within the correct period

    for \_ in range(key):

        img = ArnoldCatInverseTransform(img)

    Image.fromarray(img.astype('uint8')).save(imageName.split('\_')[0] + "\_ArnoldcatDec.png")

    return img

* **Example**:



Figure : Original Picture

A close up of a fabric

AI-generated content may be incorrect.

Figure : Encrypted Picture using key 17

A ghost in a cemetery

AI-generated content may be incorrect.

Figure : Decrypted Picture with key 17

A close up of a black and white surface

AI-generated content may be incorrect.

Figure : Decrypted Picture with key 18

* **Notice**: This chaotic map algorithm implement is not secured where there are chances that it can be brute force with some key to view the original picture. This is mainly caused by the function, which only substitution the pixel with the given formula.

1. **Cryptanalysis on Stream Cipher**

* **Brief description:** A stream cipher encrypts data one bit or one byte at a time rather than in fixed-size blocks. It generates a keystream that is combined with the plaintext to the produce ciphertext. Stream ciphers are made for the scenarios where data needs to be encrypted in the continuous stream making them suitable for real-time applications.
* **Strengths:**
  + **Continuous Encryption:** The data is encrypted in a stream that runs continuously, a bit or byte at a time
  + **Keystream Generation:** To create encryption keys, the Stream ciphers use a pseudorandom keystream generator.
  + **Efficiency:** Stream ciphers are generally more efficient for encrypting data of variable length and in the streaming applications.
* **Weaknesses:**
  + Less secure than block ciphers when the same key is used multiple times.
  + Reverse encrypted text is easy.
* **Conclusion:** In this report, I just conducted a type of chaotic map “Arnold Cat Map”, where coincidently a weak crypto algorithm due to the ability of breakable using brute-force techniques. However, taking into the context of stream cipher, it still be used in today-world as to encrypt and decrypt in real time applications despite its weakness. The cryptography turns out to be more complex but still be insecure if the keys are leaked and used by the attacker. Fortunately, nowadays, there are too many types of chaotic maps using different algorithms, loops, dimensions that make the attackers hardly conduct a vulnerability to their target.