CS512 FINAL PROJECT REPORT

Project Report

Analysis of Cryptography Algorithms for Images

Overview:

Implementing RSA and Chaos Maps based image encryption algorithms and analyzing the results.

What is Encryption?

Encryption is the process of transforming a message into a form that can only be read by authorized individuals.

What is Cryptography?

Cryptography is the process of encrypting and decrypting the data.

Why Image Encryption?

In this digital era, images have become the prominent mode of communication. Security for these digitally transmitted images are considered as the highest priority. Hence, by using image encryption technology, only the authorized users with the appropriate keys will be able to access the images. In this project, we have implemented three different Image Encryption algorithms to encrypt and decrypt the images. We have used 4 different images as input and tested three algorithms on these images.

How is data modified?

A single image is given as input. We extract the pixels from the given input image and obtain the Red, Blue, and Green colors from each pixel. Then we apply our image encryption and decryption algorithms on these pixel data. We modify this pixel information to encrypt the input image to obtain the encrypted image. Then we apply the pixel information from the generated encrypted images to regenerate the original image back.

Algorithms:

1) RSA algorithm

RSA algorithm is an asymmetric cryptography algorithm. Since this is asymmetric, nobody else except the decrypter can decrypt the data even if a third party has the public key of the browser.

The idea behind this algorithm:

The RSA algorithm is based on the fact that it is very tough to factorize a very large number into its prime factors. The public key consists of two numbers, the first of which is the result of multiplying two large prime numbers. Also, the private keys are generated with the same two prime numbers. The private key is therefore compromised if we can factorize the huge integer. As a result, the key size completely determines how strong an encryption is, and if we double or triple the key size, the encryption strength improves dramatically. RSA keys are usually 1024 or 2048 bits long.

Time Complexity:

The time complexity of the RSA algorithm used for the image encryption is in O(n^3 * 3). For 3 values of red, green and blue.

The Space Complexity of the RSA algorithm used for the image encryption is in O(n^3 * 3) terms, as we use a different Matrix to store the modified Pixel data.

Advantages of RSA Algorithm:

- **No Key Sharing:** No secret key needs to be shared in order to receive communications from others because RSA encryption relies on using the recipient's public key.
- **Proof of Authenticity:** A receiver cannot intercept a message because they lack the correct private key to decrypt the data because the key pairs are tied to one another.
- **Data Can’t Be Modified:** Since tampering with the data will change how the keys are used, the data will remain secure while it is in transit. Additionally, the information won't be able to be decrypted using the private key, alerting the recipient to manipulation.

Disadvantages of RSA Algorithm:

Due to the fact that RSA only uses asymmetric encryption and both symmetric and asymmetric encryption are necessary for full encryption, it may occasionally fail.

- Due to large numbers involved the rate of data transfer is slow.
- The dependability of public keys occasionally needs to be verified by a third party.
- Decryption requires intensive processing on the receiver's end.

- For public data encryption, such as electoral voting, RSA cannot be utilized.

Code Snippets for RSA Algorithm:

Encryption and Decryption:

```
# Encryption of the Image values present in the 1d vector
ency = []
for i in range(0, len(imgclr_1D)):
    temp = (int(imgclr_1D[i]) ** e) % n
    ency.append(int(temp))

# Reshaping the 1d vector into the m*n*3 sized array
ency_clrimg = np.array(ency).reshape(clrimg.shape[0], clrimg.shape[1], clrimg.shape[2])
imgplot = plt.imshow(ency_clrimg)
plt.show()

# Decryption of the image:
dec_clr = []
for i in range(0, len(ency)):
    temp = (ency[i] ** d)%n
    dec_clr.append(temp)

#Decrypted Image
decy_img = np.array(dec_clr).reshape(clrimg.shape[0], clrimg.shape[1], clrimg.shape[2])
imgplot = plt.imshow(decy_img)
plt.show()

```

Setting the Private and Public Keys:

```
34  # Selecting the two Prime Numbers as Big as possible
35  p1 = 37
36  p2 = 23
37  n = p1*p2
38  totientValue = (p1-1)*(p2-1)
39
40  # Initially selecting a value of e between 1 and the totient value
41  e = random.randrange(1, totientValue)
42  #Using GCD to verify if e and totient(phi(n)) are comprime or not
43  # If not select a new value of e
44  gcdVal = CalculateGcd(e, totientValue)
45  while gcdVal != 1:
46      e = random.randrange(1, totientValue)
47      gcdVal = CalculateGcd(e, totientValue)
48
49  # To find the  D Value
50  def CalculateD(e, phi):
51      d = 1
52      temp = (d*e)%phi
53
54      while (temp != 1):
55          d += 1
56          temp = (d*e)%phi
57
58      return d
59
60
61  d = CalculateD(e, totientValue)
```

Chaotic Maps:

What is Chaotic map encryption?

Chaos implies a state of disorder. Chaotic map image encryption algorithms is the application of chaos theory to the principles of cryptography. A map is a mathematical function that describes how a dynamic system changes over time. If we have n-features then we use n-dimensional maps. Chaos theory states that for any small variations in the initial states results in diverging outcomes from the original result. Some of the characteristics of chaos theory include Nonlinearity, determinism, abnormality, and affectability to initial conditions.

Why Chaotic Maps?

Chaotic maps help to recover from the drawbacks of the RSA algorithm in terms of large computational time and high computational power when encrypting the larger images. Algorithms using chaos methods offer speed, security and minimum computations. Image encryption using chaotic maps reduces the possibility of decryption by the unauthorized parties as the encryption includes randomness.

Different types of Chaotic maps:

Chaotic maps are categorized as one dimensional and multidimensional maps. One dimensional include a logistic map, sine map, tent map, and Arnold cat map. Multidimensional maps include Henon map and 2D logistic map. Any map can be used for key generation but we need to understand dynamic behavior of the map and the control parameter range where the map behaves chaotically.

Arnold Cat map:

Arnold's cat map is a non-linear chaotic map which is named after Vladimir Arnold. He proved the theory using the image of a cat and hence the algorithm is named as Arnold Cat map.
Arnold's map is a process where we rearrange the pixels in a given image without losing any information. Given a square image, we convert it into a matrix. . The position of each pixel in a square with side length N is determined by its x and y coordinates. If the given image gives us a non-square matrix then we pad zero's to make it a square matrix.

Integers in the range of 0, 1,..., N-1 makes up the x and y coordinates.
Image = {(x, y) |x, y = 0, 1, 2...N − 1}

For every iteration, the value of image[x][y] is modified as

[x] = [2*x+y]%n
[y]= [y+x]%n

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} (\mathrm{mod} n)$$

$$\begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} (\mathrm{mod} n)$$

Map is defined as a guide on how the coordinates of an image can be changed.
We set the number of iterations saying how many times the pixel values have to be changed for a given image. The iterations are as follows:

n=0: T(0) = Image[x][y]
n=1: T(1) = T(0)[mod(2*x+y, N), mod(x+y,N)]
.
.
.
n=k: T(k) = T(k-1)[mod(2*x+y, N), mod(x+y,N)]

Mod N helps to get back to the original image from the transformed one. Below is the pseudo code for the above explained algorithm:

Arnold_Transform_Image(image**)**:
    s =>image.shape
    r => number of rows in an image
    c => number of columns in an image
    arnold_transformed_image=>numpy.zeros([r, c,s])
     for x in range(0, r) :
        for y in range(0, c) :
            arnold_transformed_image[x][y] <− [mod(2*x+y, N)][mod(x+y,N)]
return arnold_transformed_image

Arnold_Image_Encryption(image, no. of iterations**)** :
    image => reading a given image
    for i in range(0, no. of iterations) :
        final_image =>Arnold_Transform_Image(image)

return final_image

To summarize, for a given image of size n, Arnold map transforms the image for a specific number of iterations until a threshold point is reached and then the original image is restored.

Time and space complexity:

The time complexity of the Arnold Cat Map algorithm used for the image encryption is in order of O(n^2 * k). Where n is the dimension of the square image and k is the number of iterations

The Space Complexity of the RSA algorithm used for the image encryption is in O(n^2 *k) terms.

Henon Map:

One of the models of discrete time dynamic systems that is most frequently used with chaotic characteristics is the Henon map. It is created by applying the following equation to map the following point from a point (xn, yn).

$x(n + 1) = 1 - \alpha*(x(n) \wedge 2) + y(n)$
$y(n + 1) = \beta * x(n)$

α - 1.4 and β - 0.3 are considered as the classic values which give us a chaotic map.
xn, yn - current positions of the image
xn+1 and yn+1 - next positions

For the first iteration, (xn, yn) are the initial values that determine the consecutive points.
By changing the initial values of (xn, yn) we witness a major change on the map.

A new (xn, yn) value is produced for each iteration, and this value is then converted to a bit value (0 or 1) using a threshold value. The bit value is then changed to gray values for each pixel, and this process continues until the chaotic map reaches the size of m*n.

The threshold value is considered to be
0 - *if* $xi \leq 0.3992$
1 - *if* $xi > 0.3992$
Pseudo code for the above mentioned algorithm is:
for n in range (m^2):
    $x(n + 1) = 1 - \alpha*(x(n) \wedge 2) + y(n)$

$y(n+1) = \beta * x(n)$
bit=0 if xi<=0.3992 else 1

encrypted image=bitwise xor(imagematrix,bitmatrix for R,G,B values in each pixel of image matrix)
decrypted image=bitwise xor(imagematrix,bitmatrix for R,G,B values in each pixel of image matrix)
The initial values used in the Henon map are used to build the bit matrix, and they are the secret keys used for encryption and decryption.

Time and space complexity:

The time complexity of the Henon Map algorithm used for the image encryption is in order of O(n^2 * 3). Where n is the dimension of the square image and 3 is for the red, green, blue colors in the pixels.

The Space Complexity of the Henon Map algorithm used for the image encryption is in O(n^2 *3) terms.

Results:

We have tested our three algorithms on 4 different images. The details of each image is mentioned below. We have calculated the time taken to encrypt and decrypt the 4 images using the three algorithms in the tables below.

Dimensions of images used:

1) Cat Picture: 1591*1591
2) Strawberry Picture: 1943*1943
3) LetterN picture: 600*600
4) Pineapple picture: 700*700

Time taken for Encryption of the 4 images in seconds:

Algorithm used	Cat Picture	Strawberry picture	Letten N picture	Pineapple picture
RSA	68.09	137.3	9.98	20.1
Arnold Cat Map (for k =25 iterations)	26.6	39.8	3.7	5.1
Henon Map	7.45	11.2	1.12	1.46

Time taken for Decryption of the 4 images in seconds:

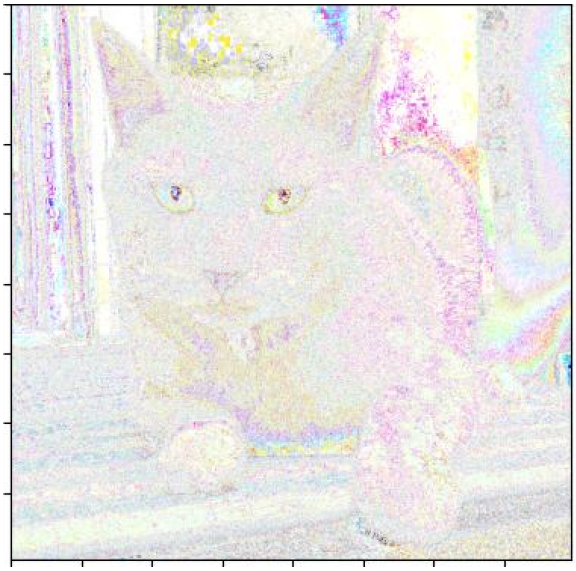
Algorithm used	Cat Picture	Strawberry picture	Letten N picture	Pineapple picture
RSA	23.04	31.4	4.2	24.5
Arnold Cat Map (for k =25 iterations)	25.08	37.7	3.4	4.8
Henon Map	7.9	11.3	1.07	1.45

Examples for Encrypted images for the three algorithms:

1) **RSA:**

Input Image:

Encrypted Image:

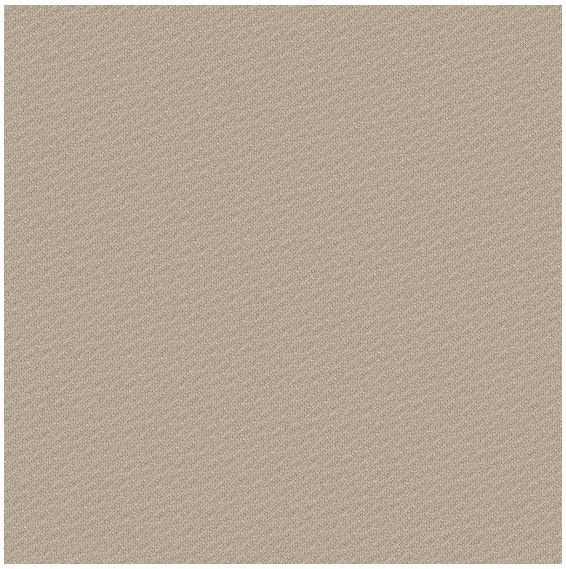


Decrypted Image:

2) **Arnold Cat Map Algorithm:** for k = 25 iterations

Input Image:

Encrypted Image:

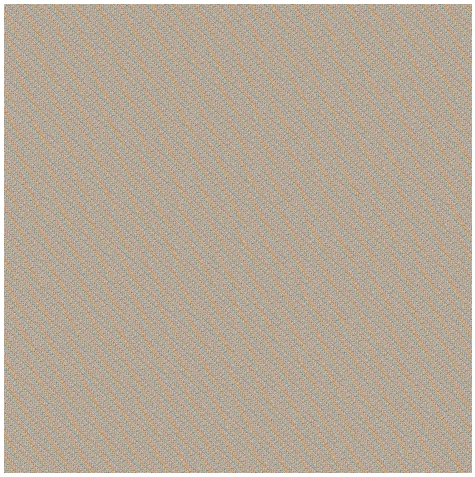


Decrypted Image:

Arnold Cat map encrypted images for various k values:

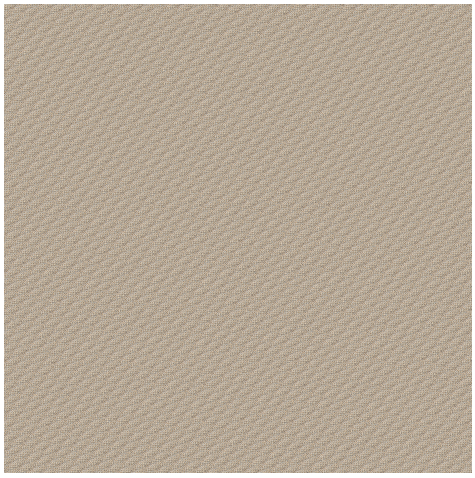
For the cat picture used above,

For k = 1

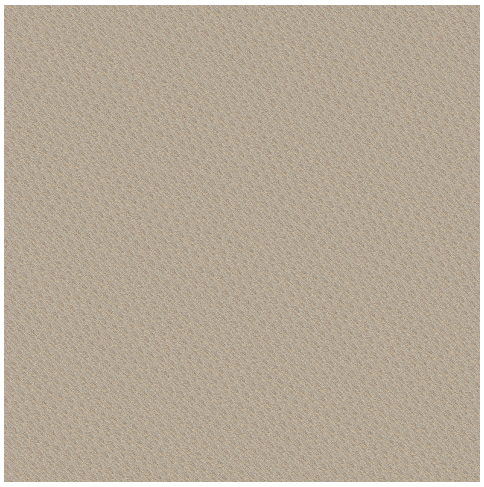


For k = 5

for k = 10



for k = 40



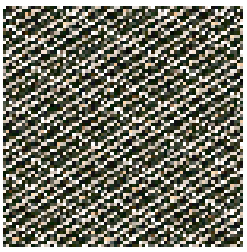
We even explored the periodicity for the Arnold Cat map algorithm with a small picture. As we know, in an Arnold Cat Map algorithm, we get back the original picture after applying the

transformation after a certain number of iterations. In the same way, we got back the original image after 9 Iterations of the Arnold Cat Map Transformation.

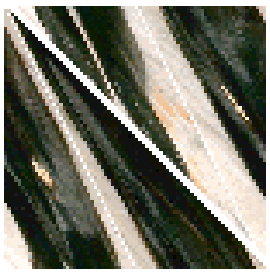
Input picture:

Encrypted for k = 1

Encrypted for k = 4



Encrypted for k = 8



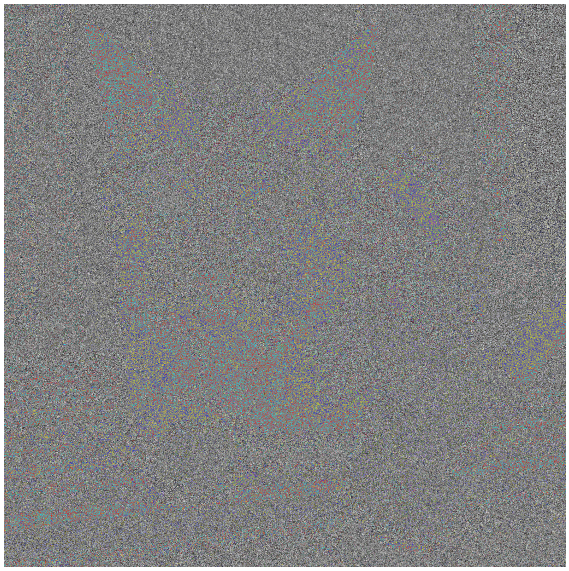
Encrypted for k = 9

We get back the original image after 9 Iterations of the Arnold Cat Map Transformation.

3) Henon Map Algorithm:

Input Image:

Encrypted Image:



Decrypted Image:

Observations:

We have observed that RSA takes a considerably more amount of time to both encrypt and decrypt the images. Whereas the Chaos Maps based algorithms take very little time to encrypt and decrypt the images. This can be observed by the time taken by the various algorithms given in the results table. This fact can also be attested by the fact that the time complexities of both the Arnold Cat map Algorithm and the Henon Map algorithm are lower than the time complexity of the RSA algorithm. The Henon Map algorithm performs way better than the other 2 algorithms in terms of both time taken and extra space used.

However, it is not only the time taken which needs to be analyzed when it comes to cryptographic algorithms. We need to compare other factors like security and the ease of breaking the encryption too.

When we compare the Encrypted images produced from the three algorithms, we observed that the Henon map algorithm and the Arnold cat Map algorithm produced encrypted images which were completely incomprehensible and they could not be easily matched with their original images. The Henon Map Algorithm performs better than the other algorithms in this aspect too as it does not preserve even the colors from the original image in the encrypted image. It produces a completely different image. The Arnold cat map too produces an image which cannot be easily matched with the original image, but it still preserves the colors from the original image in the encrypted image. Whereas, the RSA algorithm produces an image which still preserves many features from the original image, mainly colors and the different edges of the image. So the image produced from the RSA algorithm can be easily matched with the original image. However, this drawback of the RSA algorithm could also stem from the two prime numbers we used in our algorithm. It could produce better results, with bigger prime numbers and if there is a large difference between them. However, using larger primary numbers in turn leads to the worsening of the performance of the algorithm in terms of time taken.

However, if we consider the security aspect, as in the ability to decode the encrypted image, the Arnold Cat map algorithm seems to be the least secure algorithm out of the three. As we saw, the Arnold Cat map algorithm exhibits the periodicity feature, where we get back the original picture from the encrypted picture if we apply the Arnold algorithm on it for a certain number of iterations, this would give us back the original image. So, an image encrypted using the Arnold Cat algorithm can be decoded very easily by running the Arnold Cat Map Algorithm on it for a number of times until we get the original image. Similarly, if we know the initial starting key values for the Henon map, it would be relatively easy to get back the original image from the encrypted image. However, getting this initial key value would be the tough part here. When compared to the Chaos maps based algorithms, RSA will be way tougher to decode, as it would take a lot of time, computing resources and effort to decode the two prime numbers. If we go by the industry standards today, then it is almost impossible to crack the RSA encryption as the

prime numbers are each 1024 or 2048 bits long. So, out of the three algorithms, the RSA algorithm seems to give the best result in terms of security.

Standard encryption algorithms such as RSA have always been the primary choice, but when it comes to image or video encryption, many researchers are recommending chaos-based encryption techniques due to their computational efficiency. This can be observed in our case too, as Chaos based encryption algorithms took a lot less time for encryption and decryption and also gives images which cannot be easily matched with their original images. In recent years, there has been a lot of research being done on chaos-based cryptosystems. As opposed to AES, DES, RSA, etc. These algorithms are not standardized. Traditional ciphers like AES and DES work well for text encryption but not for image encryption since similar images contain repeating information pixels. This problem is fixed by chaos-based encryption methods, which generate uniformly dispersed random keys that cover the image data in the cipher images.
In the future, Encryption algorithms can be created which combine both the standard algorithm like RSA with the Chaos Based algorithms to produce much more efficient and secure encryption algorithms for images.