IMAGE ENCRYPTION AND DECRYPTION USING SDES ALGORITHM

A PROJECT REPORT

Submitted by
Hingrajia Nimit (17BCB0074)
Deep Patel (17BCE2036)
Shane Doyal Jose (17BCE2198)

Course Code: CSE4003
Course Title: Cyber Security
Project title: IMAGE ENCRYPTION USING SDES ALOGRITHM

Under the guidance of
Dr. Navaneethan C
Professor, SITE,
VIT University, Vellore

INTRODUCTION:

In this project we implement image encryption using SDES algorithm. An image will be uploaded by a user to be encrypted or decrypted. The encryption and decryption process itself will be done using the SDES (Simplified Data Encryption Standard) algorithm implemented using Python3. An image will be uploaded by the user for encryption or decryption Using SDES algorithm we encrypt or decrypt the image. Image encryption is a process of changing pixel data inside of an image based on an algorithm (usually two way). This algorithm makes use of individual pixel data, and changes it to another value so it can be decrypted later. A serial algorithm would have to pass through every pixel in sequential order, and perform the encryption on the pixel data. This takes a considerable amount of time, just like performing any other processing on an image. An image will be uploaded by a user to be encrypted or decrypted. The encryption and decryption process itself will be done using the DES (Data Encryption Standard) algorithm implemented using JavaScript. A main server will be hosted and other server instances will be created using Clustering Module. The work will be divided amongst different servers. The load amongst the server instances will be balanced. The image data will be divided and will be sent to different server instances. The encryption/decryption process will be done parallel on different child servers and then, the computed data will be returned back to the main server which will combine the data. The processed image can then be downloaded by the user.

LITERATURE REVIEW SUMMARY TABLE

Authors and Years	Title	Concept/ Theoretical	Methodology Used implementation	Dataset Details/	Relevant finding	Limitation/ future
1.X Y Yu1 , J Zhang1, Year-2006	Chaotic Image Scrambling Algorithm Based on SDES	A dual image encryption algorithm	Through Matlab simulation	Analyzing from the quantities of the key, the parameters of the	When given all parameters, the	The same methodology will be used

		based on S-DES and Logistic map is proposed.	experiments, the key quantities will attain 1017 and the encryption speed of one image doesn't exceed one second.	Arnold cat and the number of iteration may be regarded as the key, and the parameter and the initial value of the Logistic map also can be regarded as the key, then the total quantities of the key.	encryption speed doesn't exceed one second, and it can meet the requirement of normal encryption.	to encrypt images for all other existing algorithms
2.Saurabh V. Joshi Ajinkya A. Bokil Year-2012	Image Steganography Combination of Spatial and Frequency Domain	A steganographic technique which makes use of an image file as a carrier to hold the secret message and transfer it over the communication medium	LSB substitution technique for spatial domain embedding and Discrete Wavelet Transform (DWT) for transform domain embedding. The paper also proposes technique to combine cryptography with the proposed image steganography technique.	experimental results show that all four approaches have very low MSE values for different cover images.	To further security, instead of using LSB replacement technique performs a bit-exor of bit from pixel of message image with the 7th bit of pixel from carrier image and the exored result is hidden in the LSB i.e. 8th bit. At the receiving end the 7th and 8th bits of the stego image are exored and the result obtained is treated as the corresponding message bit.	The future work for this project Will be try to habe an MSB approach to the probelm
3.Sanjay Kumar, Sandeep Srivastava Year-2014	Image Encryption using Simplified Data Encryption Standard (S-DES)	try to implement Image encryption using S-DES (Simplified Data Encryption Standard).	make a chaotic map of the image using S-DES. Then use that chaotic image as a key for encrypting the image using S-DES. Thus in	After getting the chaotic image of the original image we need to make the binary image of the chaotic image, this binary conversion of the	Chaotic map make large random and key quantities and it will make this encryption more secure and gives fast	The key size is low in this algorithm. 2. Due to low key size, the security of S-DES algorithm is reduced. 3. If

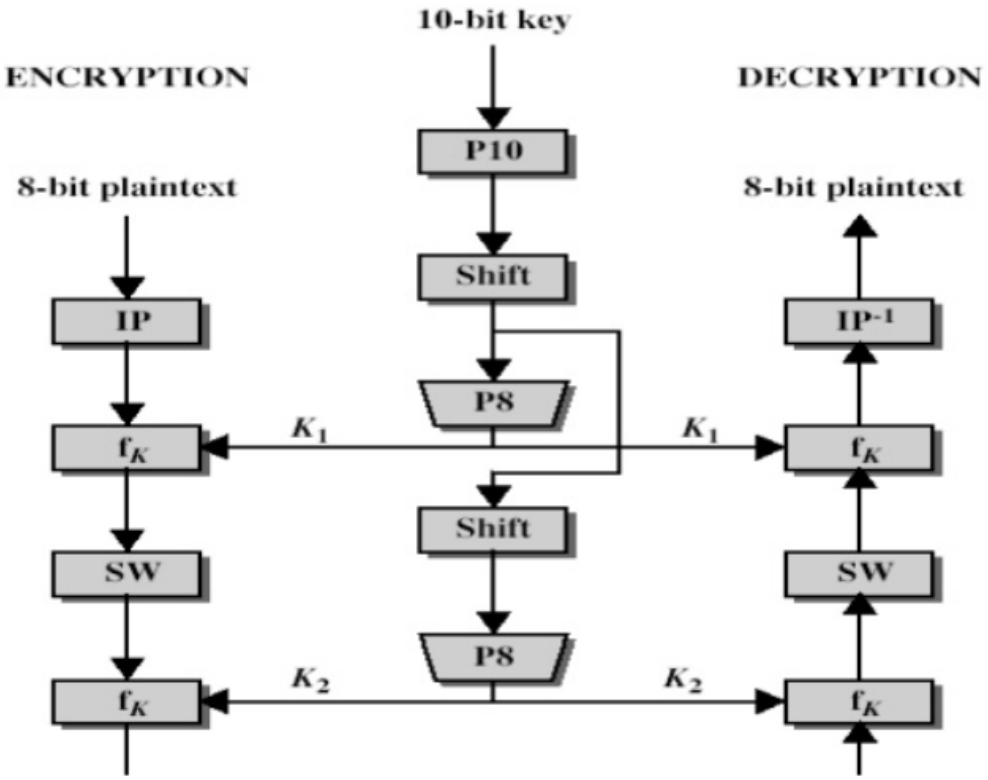
			this paper select the key when encrypt the image and use a chaotic image as a key not any other text.	chaotic image will be treated as the key for S-DES to encrypt the original image	operation speed	we use lots of data such as an image, then that algorithm can't satisfy the encryption requirement.
4.G.Sateesh1 , E.Sai Lakshmi2 Year-2016	Assured Data Communication Using Cryptography and Steganography	This paper proposes Assured Data Communication by Using Cryptography and Steganography together. The combination of these two techniques can provide robust platform for secured data communication System.	Here, a Cipher text (Encrypted) of text message Using Cryptography Techniques and then we hide the Cipher text into Multimedia Using Steganography techniques. SDES algorithm in Cryptography for data Encryption and Decryption, LSB Method of Steganography to hide Cipher text into image	The LSB steganography technique was developed in it based on embedding the secret message into the sharper edge regions of the image to ensure its resistance against image steganalysis based on statistical analysis	In this case, the particular bases used are determined by the degree of local variation of the pixel magnitudes in the host image. Due to this related works are fail to overcome the problems Changes stegoimage like colour changes in the image	
5. Dr.S.Kiran, R. Pradeep Kumar Reddy Year-2015	MULTI-STAGE ENCRYPTION USING SEEDED SDES	This paper presents a multistage encryption algorithm. At the end of each stage an intermediate cipher is produced. The key is generated by using SEEDED SDES algorithm	The proposed algorithm uses seeded sdes key generation, railference and local binary pattern provides more complexity for the text. The advantage of high complexity, the unauthorized person face very difficult to understand the original plain text.	Local binary pattern,arithmetic and logical operations ,railference technique and substitution methods are applied on final cipher text and produces the original plain text.	The key is generated through random number generation with 64 bit sequence. The railference and local binary pattern methods give complexity to the intermediate cipher text.High processing speed.	The proposed algorithm uses local binary pattern, railference, substitution method and seeded SDES key generation. In the future work, this algorithm will be extended to use the UNICODE system support.

OBJECTIVE OF THE PROJECT:

In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating cipher text that can be read only if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users. We will use encryption on image bitmap data to test our computing performance across distributed systems, by the DES (Data Encryption Standard) algorithm.

SDES:

The overall structure of the simplified DES. The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and a 10-bit key as input and produces an 8-bit block of ciphertext as output. The S-DES decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used to produce that ciphertext as input and produces the original 8-bit block of plaintext.



WORK DONE AND IMPLEMENTATION:

1. **Encryption and Decryption**

```
def encryption( self, event ):
        path=self.file_path.GetPath();
        pos_arr=findAllOccurences("\\",path)
        path=list(path)
        for i in range(len(pos_arr)):
            path.insert(pos_arr[i]+i,"\\")
        path="".join(path)
        image.encrypt_image(path)
    def decryption( self, event ):
        try:
            path=self.file_path.GetPath();
            pos_arr=findAllOccurences("\\",path)
            path=list(path)
            for i in range(len(pos_arr)):
                path.insert(pos_arr[i]+i,"\\")
            path="".join(path)
            image.decrypt_image(path)
        except Exception:
            print("Error in Decryption");
```

2. Image Encryption and Decryption

```
def encrypt_image(path):
    c=0
    img_matrix=cv2.imread(path)
    for i in range(0,len(img_matrix)):
            for j in range(0,len(img_matrix[i])):
                    for k in range(len(img_matrix[i][j])):
                            c=c+1
                            print(c)
                            original_bin=dec_to_bin.binary(img_matrix[i][j][k])
                            encrypted_bin=sdes.encrypt(original_bin)
                            img_matrix[i][j][k]=bin_to_dec.decimal(encrypted_bin)
    print(c)
    cv2.imshow("image",img_matrix)
    cv2.imwrite("encrypted.png",img_matrix)
    cv2.waitKey()
def decrypt_image(path):
    c=0
    img_matrix=cv2.imread(path)
    for i in range(0,len(img_matrix)):
            for j in range(0,len(img_matrix[i])):
                    for k in range(len(img_matrix[i][j])):
                            c=c+1
                            print(c)
                            original_bin=dec_to_bin.binary(img_matrix[i][j][k])
                            decrypted_bin=sdes.decrypt(original_bin)
                            img_matrix[i][j][k]=bin_to_dec.decimal(decrypted_bin)
    cv2.imshow("image",img_matrix)
    cv2.waitKey()
```

3. SDES

```
def permutation(perm, key):
   permutated_key = ""
   for i in perm:
      permutated_key += key[i-1]
   return permutated_key
def generate_first_key(left_key, right_key):
   left_key_rot = left_key[1:] + left_key[:1]
   right_key_rot = right_key[1:] + right_key[:1]
   key_rot = left_key_rot + right_key_rot
   return permutation(P8, key_rot)
def generate_second_key(left_key, right_key):
   left_key_rot = left_key[3:] + left_key[:3]
   right_key_rot = right_key[3:] + right_key[:3]
   key_rot = left_key_rot + right_key_rot
   return permutation(P8, key_rot)
def F(right, subkey):
```

```
    expanded_cipher = permutation(E, right)
    xor_cipher = bin( int(expanded_cipher, 2) ^ int(subkey, 2) )[2:].zfill(8)
    left_xor_cipher = xor_cipher[:4]
    right_xor_cipher = xor_cipher[4:]
    left_sbox_cipher = Sbox(left_xor_cipher, S0)
    right_sbox_cipher = Sbox(right_xor_cipher, S1)
    return permutation(P4, left_sbox_cipher + right_sbox_cipher)
def Sbox(input, sbox):
    row = int(input[0] + input[3], 2)
    column = int(input[1] + input[2], 2)
    return bin(sbox[row][column])[2:].zfill(4)
def f(first_half, second_half, key):
     left = int(first_half, 2) ^ int(F(second_half, key), 2)
     return bin(left)[2:].zfill(4), second_half
def encrypt(cipher):
     key = "0111111101"
     p10key = permutation(P10, key)
     left = p10key[:len(p10key)//2]
     right = p10key[len(p10key)//2:]
     first_key = generate_first_key(left, right)
     key1=first_key
     second_key = generate_second_key(left, right)
     key2=second_key
     permutated_cipher = permutation(IP, cipher)
     first_half_cipher = permutated_cipher[:len(permutated_cipher)//2]
     second_half_cipher = permutated_cipher[len(permutated_cipher)//2:]
     left, right = f(first_half_cipher, second_half_cipher, key1)
     left, right = f(right, left, key2)
     return permutation(IPi, left + right)
def decrypt(encrypted):
     key = "0111111101"
     p10key = permutation(P10, key)
     left = p10key[:len(p10key)//2]
     right = p10key[len(p10key)//2:]
     first_key = generate_first_key(left, right)
     key1=first_key
     second_key = generate_second_key(left, right)
     key2=second_key
     permutated_cipher = permutation(IP, encrypted)
     first_half_cipher = permutated_cipher[:len(permutated_cipher)//2]
     second_half_cipher = permutated_cipher[len(permutated_cipher)//2:]
     left, right = f(first_half_cipher, second_half_cipher, key2)
     left, right = f(right, left, key1)
     return permutation(IPi, left + right)
```

METHODLOGY ADAPTED:

Simplified-Data Encryption Standard (S-DES) is a reduced adaptation of the Data Encryption Standard (DES) algorithm. It was designed as a test block cipher for learning about modern cryptanalytic techniques such as linear cryptanalysis, differential cryptanalysis and linear-differential cryptanalysis. It is a variation of basic DES. In Simplified-DES, the same key is used for encryption and decryption in fig1.1.

Advantages of S-DES

1. It is simpler than Data Encryption Standard.

2. It takes smaller block of plaintext and use small key in encryption than DES.

3. Its execution speed is faster than DES.

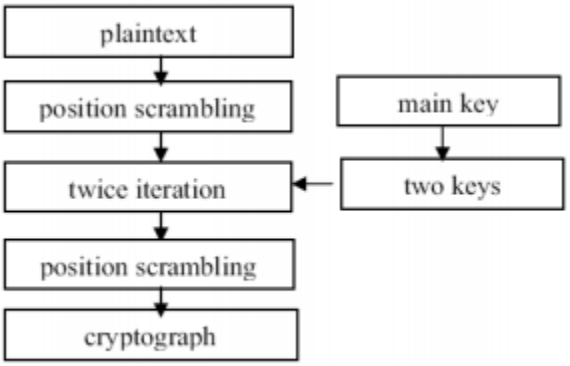


Fig.1. S-DES Structure

Limitations of S-DES

1. The key size is low in this algorithm.

2. Due to low key size, the security of S-DES algorithm is reduced.

3. If we use lots of data such as an image, then that algorithm can’t satisfy he encryption requirement.

REQUIREMENTS – HARDWARE AND SOFTWARE

Software

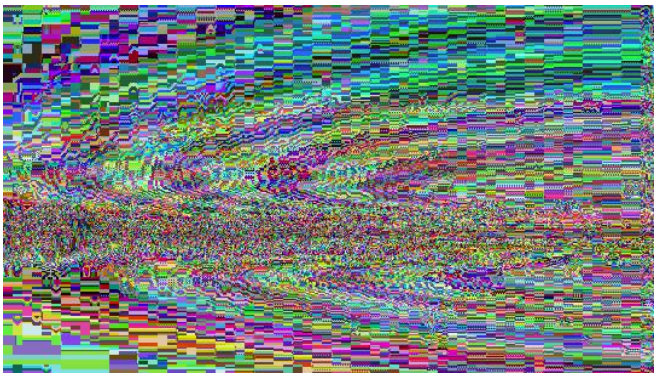
● Image Encryption and Decryption using SDES algorithm implemented on Python

● wxPython, CV2, numpy and math libraries are used.

Hardware

● Any machine that can compile python3

RESULTS AND SCREEN SHOTS:



REFRENCES:

Python environment: https://www.anaconda.com/
SDES algorithm: https://www.c-sharpcorner.com/article/s-des-or-simplified-data-encryption-standard/
https://s3.amazonaws.com/academia.edu.documents/52063962/pxc3899070.