Publisher homepage: www.universepg.com, ISSN: 2663-7804 (Online) & 2663-7790 (Print)

https://doi.org/10.34104/ajeit.022.045051

Australian Journal of Engineering and Innovative Technology

Journal homepage: www.universepg.com/journal/ajeit

Data Encryption Using Image Processing: A Brief Study

Mahmud Rahman Parag[1]* and Tania Akter Setu[2]

[1&2]Department of Computer Science and Engineering, University of Information Technology & Sciences, Dhaka, Bangladesh.

*Correspondence: 2010352011@uits.edu.bd (Mahmud Rahman Parag, Department of Computer Science and Engineering, UITS, Dhaka, Bangladesh).

ABSTRACT

We can see considerable growth in the field of science and technology over the past few decades. This growth has also brought an enormous threat to the data used by users. Encryption and decryption of data play an important role in safeguarding the data. Many authors have contributed much outstanding research on unauthorized access to user data. In this journal, I have proposed an algorithm for data encryption using images. The main aim is to provide a new and highly customizable approach to protecting data using images and eliminating the use of plaintext. In my research, I will use images to encrypt and decrypt data rather than using plaintext.

Keywords: Cryptography, Data, Encryption, Decryption, Image, Image Pixel, and RGB.

INTRODUCTION:

In this digitalized era, data is everywhere. From daily messaging to storing friends' information onour mobile, from Facebook to criminal information in police office databases, everything is digitalized data. Protecting it from unwanted access is becoming a must to keep our privacy. Cryptography protects this data by changing it into unclear data that can only be accessed via authorized receivers, who then convert the uncertain data into the original textual content (Alemami *et al.,* 2019). By this, we eliminate any kinds of unauthorized access and protect our data.

There are many existing solutions for data encryption. AES and RSA are one of the most used and popular algorithms. AES and RSA heavily relied on plaintext as a key. A long key made encryption much harder to break, but the catch is you have to remember the key. If someone uses different plaintext to encrypt different data then the problem gets much bigger. On top of that, if someone saves their key in a text file then anyone can suspect that text file is a key. On the other hand, AES has a key length limitation. Here, in my thesis, I have shown an alternate approach to eliminate plaintext and use an image as a key for data encryption.This algorithm uses image pixel values as a key and encrypts and decrypts data. On top of that, there is no key length limitation in this algorithm as long as someone has enough disk space and processing power.

Literature Review

Cryptography is a way to conceal confidential information from a third party by implementing keys that are only known by the communicating parties. For example, let us assume that two devices are communicating with themselves. The sender will be A and the receiver will be B. When A will send the message A will the message using a key so that no other than A and B can see the message and when B will receive the

message, B will use the same key as A to decrypt the message to get the original data back. While C who is an eavesdropper won't be able to get the original message as it is encrypted which is shown in **Fig. 1**

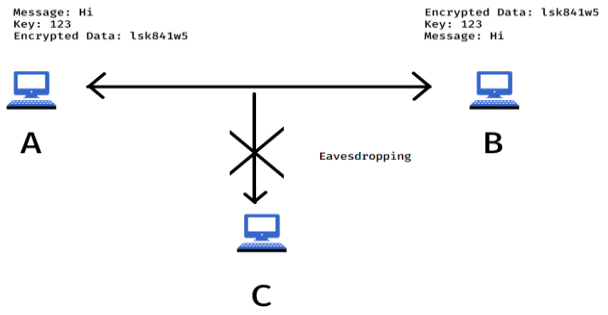


Fig. 1: Secure Data Transfer Between A and B.

Classification

Cryptography is classified into two categories: Symmetricand Asymmetric Key Cryptography. Sometimes we can find a combination of symmetric and asymmetric key cryptography in some applications called hybrid schemes. It is because both types of algorithms have their strength and weakness. When they are combined, they protect each other weaknesses.

Symmetric Key Cryptography

Symmetric Key Cryptography is a type of cryptography where a single secret key is used for both encryption and decryption. When parties send a message to another party, they use a single key to encrypt and decrypt their data.

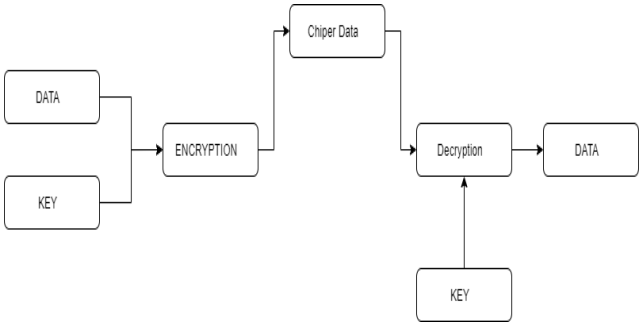


Fig. 2: Symmetric Key Cryptography.

This Symmetric Key Cryptography has two types of structure.

Stream cipher

In this type data is encrypted bit by bit or byte by byte at a single time. The A5 algorithm is the best example of this type which is used in mobile phone calls.

Block cipher

In these types of cipher, data is divided into blocks and then encrypted at the same time. There are plentyof examples of this type of structure. One of the most popular algorithms which are used worldwide is the AES algorithm.

Asymmetric Key Cryptography

Asymmetric Cryptography is also called publickey cryptography. Inasymmetricor public-key cryptography, there are two keys: a private key and a public key are used (Jirwan *et al.,* 2013). When someone wants to encrypt their data, they generate these two keys and used their private key to encrypt their data and use their public key to decrypt. The RSA algorithm is the best example of this kind of cryptography.

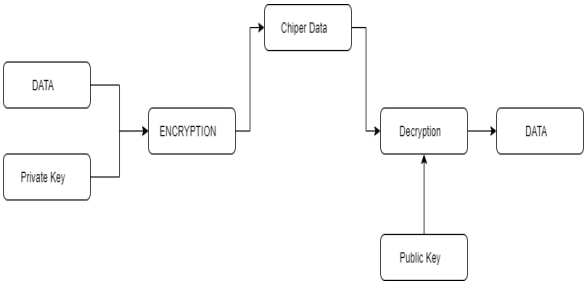


Fig. 3: Asymmetric Key Cryptography.

Encryption and Decryption of Data Using Image Algorithim

Encryption and Decryption of data using images need several steps which are enlisted below:

Encryption Process:

1. Plaintext Processing
2. Key Generation
3. Encryption

Decryption Process:

1. Key Generation
2. Decryption

All designs of theseprocesseswill be explained in detail below. Before entering encryption and decryption 3 key variables should be initialized.

1) NUMBER_OF_COLUMN_TO_MAP
2) NUMBER_OF_ROW_TO_MAP
3) RANDOMIZE_BOX

These three variables should be implemented in the algorithm for encryption and decryption and must be the same for both. If all values of these three are chan-

ged in the algorithm, then all values of these three variables have to be changed and have to be the same in the decryption algorithm also for a given connection. For instance, if a and b want to transfer data to each other then a will encrypt his data and b will decrypta's data using the same value of these three variables.

Encryption Process

Plaintext Processing

It is a mapping-like algorithm. It maps several characters to several boxes. We use two important parameters:

1) Number of characters to map per row (character = byte)
2) The number of rows and columns need to map into

Number of Characters to map per row have to be smaller than or equal to the number of rows needed to map. These two parameters are hardcoded in the algorithm. Now, it takes the plain text and split the text depending on those two parameters if there are any remaining characters it will take those characters after doing the encryption of previous characters. For example, let us assume the plain text is "001122334-45566778899AABBCCDDEEFF", the Number of characters to map per row is 2, the Number of rows needs to map into is 16, and the number of columns needs to map is 10. So, the actionsare,

1) Convert every character to their ascii value. So, the ascii hex value of plaintext will be '48 48 49 49 50 50 51 51 52 52 53 53 54 54 55 55 56 56 57 57 65 65 66 66 67 67 68 68 69 69 70 70'
2) Then, from the parameter, every two ASCII values will be mapped into every 10 rows. It will be like this **Fig. 4**

	1	2	3	4	5	6	7	8	9	10
1	48	48	0	0	0	0	0	0	0	0
2	49	49	0	0	0	0	0	0	0	0
3	50	50	0	0	0	0	0	0	0	0
4	51	51	0	0	0	0	0	0	0	0
5	52	52	0	0	0	0	0	0	0	0
6	53	53	0	0	0	0	0	0	0	0
7	54	54	0	0	0	0	0	0	0	0
8	55	55	0	0	0	0	0	0	0	0
9	56	56	0	0	0	0	0	0	0	0
10	57	57	0	0	0	0	0	0	0	0
11	65	65	0	0	0	0	0	0	0	0
12	66	66	0	0	0	0	0	0	0	0
13	67	67	0	0	0	0	0	0	0	0
14	68	68	0	0	0	0	0	0	0	0
15	69	69	0	0	0	0	0	0	0	0
16	70	70	0	0	0	0	0	0	0	0

Fig. 4: Plaintext mapping into boxes.

Simple MATLAB Code

```
plainText = '00112233445566778899AABBCCDDEEFF';
TOTAL_NUMBER_OF_CHAR = length(plainText);

MAX_NUMBER_OF_CHAR_PER_ROW = 2;

NUMBER_OF_COLUMN_TO_MAP = 10;
NUMBER_OF_ROW_TO_MAP = 16;

PLAINTEXT_OF_BOX = uint8(zeros(NUMBER_OF_ROW_TO_MAP, NUMBER_OF_COLUMN_TO_MAP));

RANDOMIZE_BOX = randperm(NUMBER_OF_COLUMN_TO_MAP);

charCount = 1;
TEMP_MAX_NUMBER_OF_CHAR = MAX_NUMBER_OF_CHAR_PER_ROW;

for i = 1 : NUMBER_OF_ROW_TO_MAP
    for j = 1 : NUMBER_OF_COLUMN_TO_MAP
        if (charCount <= TOTAL_NUMBER_OF_CHAR)
            if (charCount > TEMP_MAX_NUMBER_OF_CHAR)
                if (j == NUMBER_OF_COLUMN_TO_MAP)
                    TEMP_MAX_NUMBER_OF_CHAR = TEMP_MAX_NUMBER_OF_CHAR + MAX_NUMBER_OF_CHAR_PER_ROW;
                end
            else
                PLAINTEXT_OF_BOX(i, j) = plainText(charCount);
                charCount = charCount + 1;
            end
        end
    end
end
```

Fig. 5: Simple MATLAB Code for plaintext processing.

Key Generation

In this part, we take an image as a key. Let us assume, we take a 20 X 20 size image which is 20 * 20 = 400 * 8 = 3200 bit key. We take any channel from R, G, or B and use it as a key.

Fig. 6: A 20 X 20 Size Image as a key.

151	149	157	173	170	178	201	217	250	255	255	255	240	223	229	228	224	231	229	223
148	151	152	163	184	176	199	207	194	195	187	200	222	231	230	226	222	227	228	230
163	153	158	152	166	184	226	174	173	189	169	150	179	223	228	214	228	227	222	223
149	140	145	159	186	213	245	171	167	170	152	128	154	217	242	228	221	229	230	223
152	127	67	72	107	103	102	91	92	73	82	75	61	83	119	144	183	215	234	228
151	112	47	32	45	60	77	93	43	34	62	72	60	74	102	130	158	206	233	225
141	133	152	141	126	141	162	143	90	71	75	79	107	131	118	116	155	203	233	228
133	164	243	243	238	217	215	205	168	154	143	127	144	143	119	145	134	182	222	229
125	184	255	228	225	234	218	197	194	170	170	160	160	153	134	154	137	147	201	232
111	126	119	105	157	225	215	180	185	177	183	174	166	153	143	158	146	142	174	219
88	89	81	81	99	172	219	187	176	180	186	182	167	147	145	155	151	140	155	204
84	89	112	96	87	112	183	196	181	176	172	178	163	143	147	155	146	143	158	188
88	116	131	96	92	81	116	222	209	182	164	174	166	148	154	156	147	148	165	167
72	99	121	107	81	67	65	215	252	209	184	181	168	154	152	153	158	157	168	156
50	45	68	79	62	60	53	185	255	224	205	183	156	150	142	141	157	163	171	172
48	45	50	41	64	59	85	241	248	218	211	176	141	143	132	132	132	143	161	173	198
78	50	18	38	46	49	185	255	230	212	193	166	133	128	128	129	132	143	163	193
112	64	23	28	18	87	247	255	224	204	183	159	132	126	123	118	130	140	161	190
113	84	33	22	17	142	255	233	221	201	180	158	138	129	127	115	125	132	150	179
85	103	34	13	34	180	255	207	224	206	183	160	143	131	134	117	119	123	139	167

Fig. 7: Image Red Channel.

82	77	76	93	97	107	124	123	132	131	134	147	152	155	166	170	165	172	170	164
78	77	71	80	106	97	110	103	67	59	56	84	132	160	167	168	163	168	169	172
93	80	77	66	77	85	114	49	30	43	33	31	86	147	164	155	167	166	162	163
82	72	74	86	105	125	144	59	43	47	40	29	72	147	176	166	159	168	167	163
88	67	18	28	60	52	42	25	23	8	22	18	5	26	57	79	116	150	168	164
86	55	7	0	9	15	23	34	0	0	22	33	13	19	37	62	89	137	165	157
73	73	108	100	75	75	77	55	13	6	23	30	50	67	45	41	82	132	163	158
64	104	198	199	176	135	110	94	71	73	78	69	80	69	37	63	55	107	148	155
53	122	226	179	156	143	101	74	84	76	96	93	85	72	43	62	54	70	125	157
44	66	71	52	82	126	87	43	61	69	94	94	82	66	48	63	61	60	93	138
32	35	33	26	21	67	85	42	37	52	77	87	74	57	53	63	59	49	65	115
27	35	61	40	13	21	68	68	51	52	60	74	66	48	51	57	48	46	62	92
23	53	74	41	37	19	43	139	117	86	66	74	65	45	45	47	42	48	66	69
12	39	65	56	35	23	16	156	178	121	88	82	66	50	44	44	52	56	65	56
9	1	19	32	21	21	10	129	187	131	102	79	57	53	47	45	56	63	69	73
16	12	13	4	29	24	42	183	165	119	104	69	42	51	46	44	49	63	74	100
50	23	0	14	22	16	138	193	141	111	86	61	36	39	44	46	42	50	68	96
87	42	5	11	0	57	195	191	133	100	76	55	35	37	40	35	44	50	65	92
88	62	15	5	0	112	226	163	130	97	73	54	41	40	44	32	38	41	53	80
60	81	16	0	15	150	213	137	133	102	76	56	46	42	50	34	31	32	42	68

Fig. 8: Image Green Channel.

25	18	11	22	18	29	54	69	96	102	105	105	88	72	71	71	73	82	78	70
16	14	6	12	32	22	42	48	32	33	28	45	70	80	72	68	71	78	75	75
21	11	14	5	11	18	50	0	0	18	11	0	29	71	67	53	74	75	66	65
14	11	28	45	58	75	90	13	15	31	29	10	32	87	98	79	72	79	74	67
44	31	1	17	42	25	5	0	8	6	30	24	0	0	16	23	45	68	82	76
44	25	0	0	0	0	0	4	0	0	30	36	3	0	7	15	22	59	82	72
12	23	79	80	48	41	36	15	0	0	9	13	20	29	2	0	13	52	78	72
0	44	157	164	139	95	65	51	38	44	48	32	34	20	0	7	0	26	63	68
0	65	180	138	114	99	58	33	47	38	51	41	28	17	0	11	0	0	37	66
0	16	25	8	42	87	50	7	23	23	38	33	22	12	4	15	0	0	12	56
0	0	0	0	0	35	58	11	0	3	18	23	15	5	12	22	8	0	3	49
7	11	30	7	0	0	41	33	0	0	0	11	13	4	13	22	9	1	12	41
21	46	57	20	16	0	8	87	42	2	0	12	23	12	14	16	10	12	25	24
14	38	52	37	19	0	0	90	83	21	2	17	28	21	6	5	13	14	20	7
3	0	2	12	3	0	0	46	73	12	7	8	15	18	0	0	0	4	7	6
1	0	0	0	7	0	0	84	43	0	10	1	0	12	0	0	0	0	6	27
26	4	0	4	0	0	46	81	25	5	6	3	0	0	0	0	0	0	4	28
57	21	3	4	0	0	94	70	19	1	4	0	0	0	0	0	0	0	5	29
57	39	11	0	0	50	125	42	16	0	1	0	0	0	4	0	0	0	0	21
29	58	12	0	0	86	112	16	19	3	4	1	3	2	13	0	0	0	0	11

Fig. 9: Image Blue Channel.

Let's take the red channel for demonstrating the example. After taking a channel, split it according to the value of the Number column to map. So, we split it on every 10 values until we reached the end of the red channel. If the last split misses some value, we will fill up those values using the beginning value of the red channel. **Fig. 9** will depict the process.

	1	2	3	4	5	6	7	8	9	10
1	151	149	157	173	170	178	201	217	250	255
2	255	255	240	223	229	228	224	231	229	223
3	148	151	152	163	184	176	199	207	194	195
4	187	200	222	231	230	226	222	227	228	230
5	163	153	158	152	166	184	226	174	173	189
6	169	150	179	223	228	214	228	227	222	223
7	149	140	145	159	186	213	245	171	167	170
8	152	128	154	217	242	228	221	229	230	223
9	152	127	67	72	107	103	102	91	92	73
10	82	75	61	83	119	144	183	215	234	228
11	151	112	47	32	45	60	77	93	43	34
12	62	72	60	74	102	130	158	206	233	225
13	141	133	152	141	126	141	162	143	90	71
14	75	79	107	131	118	116	155	203	233	228
15	133	164	243	243	238	217	215	205	168	154
16	143	127	144	143	119	145	134	182	222	229
17	125	184	255	228	225	234	218	197	194	170
18	170	160	160	153	134	154	137	147	201	232
19	111	126	119	105	157	225	215	180	185	177
20	183	174	166	153	143	158	146	142	174	219
21	88	89	81	81	99	172	219	187	176	180
22	186	182	167	147	145	155	151	140	155	204
23	84	89	112	96	87	112	183	196	181	176
24	172	178	163	143	147	155	146	143	158	188
25	88	116	131	96	92	81	116	222	209	182
26	164	174	166	148	154	156	147	148	165	167
27	72	99	121	107	81	67	65	215	252	209
28	184	181	168	154	152	153	158	157	168	156
29	50	45	68	79	62	60	53	185	255	224
30	205	183	156	150	142	141	157	163	171	172
31	48	45	50	41	64	59	85	241	248	218
32	211	176	141	143	132	132	143	161	173	198
33	78	50	18	38	46	49	185	255	230	212
34	193	166	133	128	128	129	132	143	163	193
35	112	64	23	28	18	87	247	255	224	204
36	183	159	132	126	123	118	130	140	161	190
37	113	84	33	22	17	142	255	233	221	201
38	180	158	138	129	127	115	125	132	150	179
39	85	103	34	13	34	180	255	207	224	206
40	183	160	143	131	134	117	119	123	139	167

Fig. 10: Final Processed Key.

Simple MATLAB Code

```
img = imread('1_E.jpg');
RED = img(:,:,1);
RED = RED';
RED_L = RED(:);

MAX_NUMBER_OF_CHAR_PER_ROW = 2;

NUMBER_OF_COLUMN_TO_MAP = 10;
NUMBER_OF_ROW_TO_MAP = 16;
TOTAL_NUMBER_OF_ROUND_FOR_KEY_TO_TRANSFORM_PAILTEXT = ceil((length(RED_L) / NUMBER_OF_COLUMN_TO_MAP));
KEY_OF_BOX = uint8(zeros(TOTAL_NUMBER_OF_ROUND_FOR_KEY_TO_TRANSFORM_PAILTEXT, NUMBER_OF_COLUMN_TO_MAP));

keyCount = 1;

for i = 1 : TOTAL_NUMBER_OF_ROUND_FOR_KEY_TO_TRANSFORM_PAILTEXT
    for j = 1 : NUMBER_OF_COLUMN_TO_MAP
        if (keyCount > length(RED_L))
            keyCount = 1;
            KEY_OF_BOX(i, j) = RED_L(keyCount);
            keyCount = keyCount + 1;
        else
            KEY_OF_BOX(i, j) = RED_L(keyCount);
            keyCount = keyCount + 1;
        end
    end
end
```

Fig. 11: Simple MATLAB Code for Key processing.

Encryption

We begin the encryption by XORing all mapped plaintext by the first key row.

	1	2	3	4	5	6	7	8	9	10
1	151	149	157	173	170	178	201	217	250	255

Fig. 12: First Row of the key.

	1	2	3	4	5	6	7	8	9	10
1	167	165	157	173	170	178	201	217	250	255
2	166	164	157	173	170	178	201	217	250	255
3	165	167	157	173	170	178	201	217	250	255
4	164	166	157	173	170	178	201	217	250	255
5	163	161	157	173	170	178	201	217	250	255
6	162	160	157	173	170	178	201	217	250	255
7	161	163	157	173	170	178	201	217	250	255
8	160	162	157	173	170	178	201	217	250	255
9	175	173	157	173	170	178	201	217	250	255
10	174	172	157	173	170	178	201	217	250	255
11	214	212	157	173	170	178	201	217	250	255
12	213	215	157	173	170	178	201	217	250	255
13	212	214	157	173	170	178	201	217	250	255
14	211	209	157	173	170	178	201	217	250	255
15	210	208	157	173	170	178	201	217	250	255
16	209	211	157	173	170	178	201	217	250	255

Fig. 13: XOR of the first row and all rows of mapped plaintext.

After that, we do a circular right shift to the first row of the result of XOR where the amount is the number of rows needed to map subtract to the current row number.

	1	2	3	4	5	6	7	8	9	10
1	178	201	217	250	255	167	165	157	173	170

Fig. 14: Circular right shift of the first row.

After that, we generate a fixed random matrix/array where the length of it is equal to the number of the column. These random numbers are unique and should not be greater than the number of columns.

	1	2	3	4	5	6	7	8	9	10
1	3	9	10	5	7	8	4	6	2	1

Fig. 15: A Fixed Random Matrix/array.

Then, we randomize the right-shifted row cells according to the random number.

	1	2	3	4	5	6	7	8	9	10
1	170	173	178	165	250	157	255	167	201	217

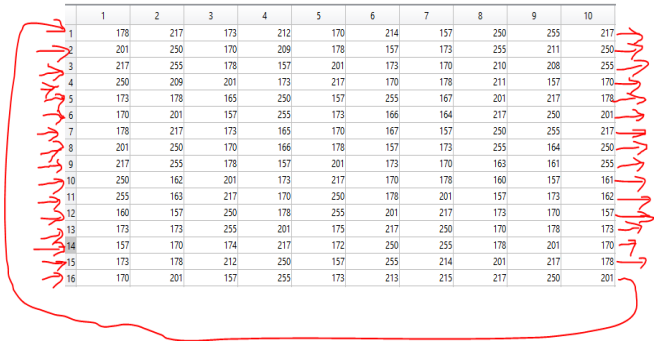
Fig. 16: Randomized Cells of the First Row.

And we continue to do the Circular Right Shift and Randomize the Row until all the rows of the result of XOR are done.

	1	2	3	4	5	6	7	8	9	10
1	170	173	178	165	250	157	255	167	201	217
2	178	170	201	157	255	173	166	164	217	250
3	201	178	217	173	165	170	167	157	250	255
4	217	201	250	170	166	178	157	173	255	164
5	250	217	255	178	157	201	173	170	163	161
6	255	250	162	201	173	217	170	178	160	157
7	161	255	163	217	170	250	178	201	157	173
8	162	160	157	250	178	255	201	217	173	170
9	157	173	173	255	201	175	217	250	170	178
10	173	157	170	174	217	172	250	255	178	201
11	170	173	178	212	250	157	255	214	201	217
12	178	170	201	157	255	173	213	215	217	250
13	201	178	217	173	212	170	214	157	250	255
14	217	201	250	170	209	178	157	173	255	211
15	250	217	255	178	157	201	173	170	210	208
16	255	250	209	201	173	217	170	178	211	157

Fig. 17: Full result of Circular Right Shift, and Randomize of every Row.

Then we again do a circular right shift to the entire matrix/array. Where the amount of right shift is the number of keys subtracted from the row number of the current key.



	1	2	3	4	5	6	7	8	9	10
1	178	217	173	212	170	214	157	250	255	217
2	201	250	170	209	178	157	173	255	211	250
3	217	255	178	157	201	173	170	210	208	255
4	250	209	201	173	217	170	178	211	157	170
5	173	178	165	250	157	255	167	201	217	178
6	170	201	157	255	173	166	164	217	250	201
7	178	217	173	165	170	167	157	250	255	217
8	201	250	170	166	178	157	173	255	164	250
9	217	255	178	157	201	173	170	163	161	255
10	250	162	201	173	217	170	178	160	157	161
11	255	163	217	170	250	178	201	157	173	162
12	160	157	250	178	255	201	217	173	170	157
13	173	173	255	201	175	217	250	170	178	173
14	157	170	174	217	172	250	255	178	201	170
15	173	178	212	250	157	255	214	201	217	178
16	170	201	157	255	173	213	215	217	250	201

Fig. 18: The full matrix of the result of the circular right shift.

Then we repeat the full process until all of the keys are used. In other words, all of the rows of the key matrix are used. And the final matrix will be the chipper text.

	1	2	3	4	5	6	7	8	9	10
1	167	83	69	174	179	182	123	30	38	156
2	147	51	84	4	192	230	133	245	2	36
3	64	38	34	87	145	106	14	245	224	126
4	131	98	170	153	120	194	210	173	128	167
5	164	233	88	13	18	190	23	34	245	26
6	168	108	94	137	163	126	141	172	252	131
7	75	200	25	100	41	12	89	252	60	199
8	225	78	135	41	83	145	82	111	166	220
9	221	59	66	55	72	211	52	230	64	109
10	225	244	196	106	49	69	106	74	83	183
11	224	14	60	103	174	9	159	153	195	57
12	5	9	212	198	174	83	126	211	124	232
13	80	226	43	100	27	11	130	224	222	75
14	105	89	134	220	119	154	233	188	129	129
15	188	229	97	248	94	248	57	75	40	242
16	80	78	61	57	245	79	133	73	79	234

Fig. 19: The Chipper Matrix.

Flow Diagram

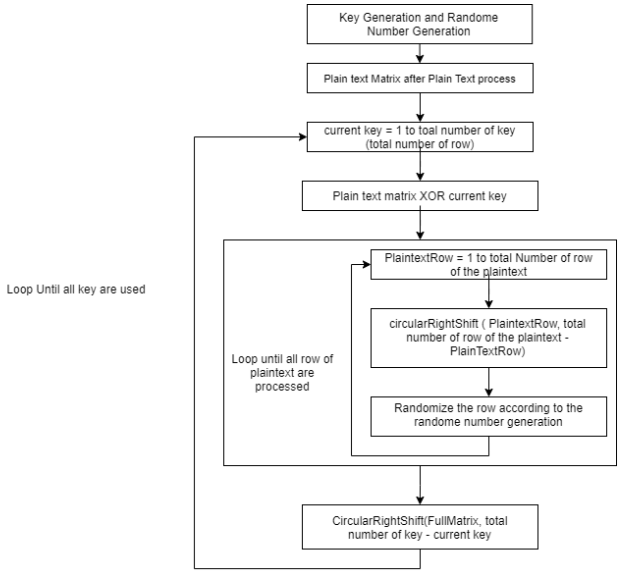


Fig. 20: Flow Diagram of Encryption Process.

Simple MATLAB Code

```
[HeightOfKey, WidthOfKey] = size(KEY_OF_BOX);
[HeightOfPlainText, WidthOfPlainText] = size(PLAINTEXT_OF_BOX);

PlainTextEncryption = PLAINTEXT_OF_BOX;

for ke = 1 : HeightOfKey
    SingleKeyE = KEY_OF_BOX(ke, :);

    PlainTextEncryption = bitxor(PlainTextEncryption, SingleKeyE);

    tempPlainTextEncryption = PlainTextEncryption;

    for pe = 1 : HeightOfPlainText
        tempPlainTextEr = tempPlainTextEncryption(pe, :);
        tempPlainTextEr = circshift(tempPlainTextEr, (HeightOfPlainText - pe));

        tempPlainText = zeros(1, WidthOfPlainText);

        for rne = 1 : NUMBER_OF_COLUMN_TO_MAP
            tempPlainText(RANDOMIZE_BOX(rne)) = tempPlainTextEr(rne);
        end

        tempPlainTextEncryption(pe, :) = tempPlainText(:);
    end

    tempPlainTextEncryption = tempPlainTextEncryption';
    tempPlainTextEncryption = tempPlainTextEncryption(:);
    tempPlainTextEncryption = tempPlainTextEncryption';

    tempPlainTextEncryption = circshift(tempPlainTextEncryption, (HeightOfKey - ke));

    tempPlainTextEncryption = tempPlainTextEncryption';

    tempPlainTextEncryption = reshape(tempPlainTextEncryption, WidthOfPlainText, HeightOfPlainText);

    tempPlainTextEncryption = tempPlainTextEncryption';

    PlainTextEncryption = tempPlainTextEncryption;
end
```

Fig. 21: MATLAB Code for encryption.

Decryption Process

Key Generation

Generating a key in the Decryption processis the same as the key generation processes of Encryption.

Decryption Process

The decryption algorithm is the exact opposite of the encryption algorithm. First, we convert the chipper text to the number of columns needed to map by the Number of rows needed to map the matrix then we do a full circular left shift and then we reconstruct the row from the randomized row and do a circular left shift until all row is done and then XOR it from the last row number or the last key until the first. And the result will be plain text.

	1	2	3	4	5	6	7	8	9	10
1	48	48	0	0	0	0	0	0	0	0
2	49	49	0	0	0	0	0	0	0	0
3	50	50	0	0	0	0	0	0	0	0
4	51	51	0	0	0	0	0	0	0	0
5	52	52	0	0	0	0	0	0	0	0
6	53	53	0	0	0	0	0	0	0	0
7	54	54	0	0	0	0	0	0	0	0
8	55	55	0	0	0	0	0	0	0	0
9	56	56	0	0	0	0	0	0	0	0
10	57	57	0	0	0	0	0	0	0	0
11	65	65	0	0	0	0	0	0	0	0
12	66	66	0	0	0	0	0	0	0	0
13	67	67	0	0	0	0	0	0	0	0
14	68	68	0	0	0	0	0	0	0	0
15	69	69	0	0	0	0	0	0	0	0
16	70	70	0	0	0	0	0	0	0	0

Fig. 22: The Plain text Matrix.

Flow diagram

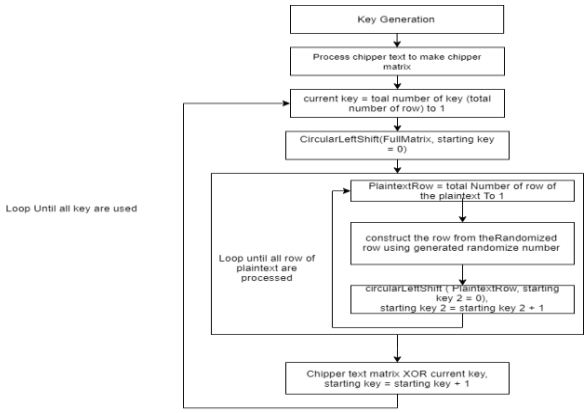


Fig. 23: Flow Diagram of Decryption Process.

Simple MATLAB Code

```
for kd = HeightOfKey : -1 : 1
    tempChiperTextDecryption = ChiperTextDecryption;

    tempChiperTextDecryption = tempChiperTextDecryption';
    tempChiperTextDecryption = tempChiperTextDecryption(:);
    tempChiperTextDecryption = tempChiperTextDecryption';

    tempChiperTextDecryption = circshift(tempChiperTextDecryption, tempShiftSizeForDecryption);

    tempChiperTextDecryption = reshape(tempChiperTextDecryption, WidthOfPlainText, HeightOfPlainText);
    tempChiperTextDecryption = tempChiperTextDecryption';

    tempShiftSizeForDecryptionPH = 0;

    for pd = HeightOfPlainText : -1 : 1
        tempChiperTextDr = tempChiperTextDecryption(pd, :);

        tempRandomizeChiperText = zeros(1, WidthOfPlainText);

        for rnd = 1 : NUMBER_OF_COLUMN_TO_MAP
            tempRandomizeChiperText(rnd) = tempChiperTextDr(RANDOMIZE_BOX(rnd));
        end

        tempPlainTextDr = circshift(tempRandomizeChiperText, tempShiftSizeForDecryptionPH);
        tempChiperTextDecryption(pd, :) = tempPlainTextDr;

        tempShiftSizeForDecryptionPH = tempShiftSizeForDecryptionPH - 1;
    end

    SingleKeyD = KEY_OF_BOX(kd, :);
    tempChiperTextDecryption = bitxor(tempChiperTextDecryption, SingleKeyD);

    ChiperTextDecryption = tempChiperTextDecryption;

    tempShiftSizeForDecryption = tempShiftSizeForDecryption - 1;
end
```

Fig. 24: MATLAB Code for Decryption Process.

Evaluation

Introduction

To implement it in a system, it hasto pass some speed tests. To evaluate it properly we are going to do a speed test for different keys so that we can conclude which key is most appropriate for general use in terms of length.

Speed Measurement

For the speed test, we will use 256-bit text and 3200-bit, 28800-bit, and 20280000-bit images. We are going to five times for every key and going to take their average.

3200-bit KEY:

Plaintext:00112233445566778899AABBCCDDEEFF

Encryption

Table 1: Encryption Time Measurement for 3200 bit KEY.

Number of Run	Time (second)
1	0.051 s
2	0.048 s
3	0.055 s
4	0.045 s
5	0.055 s
Average time	0.051 s

Decryption

Table 2: Decryption Time Measurement for 3200-bit KEY.

Number of Run	Time (second)
1	0.036 s
2	0.050 s
3	0.038 s
4	0.034 s
5	0.039 s
Average time	0.039 s

28800-bit KEY:

Plaintext: 00112233445566778899AABBCCDDEEFF

Encryption

Table 3: Encryption Time Measurement for 28800-bit KEY.

No	Time (second)
1	0.073s
2	0.112 s
3	0.072 s
4	0.089 s
5	0.103 s
Average time	0.090 s

Decryption

Table 4: Decryption Time Measurement for 28800-bit KEY.

No	Time (second)
1	0.096 s
2	0.089 s
3	0.085 s
4	0.091 s
5	0.098 s
Average time	0.092 s

20280000-bit KEY:

Plaintext: 00112233445566778899AABBCCDDEEFF.

Encryption

Table 5: Encryption Time Measurement for 20280-000-bit KEY.

No.	Time (second)
1	17.250 s
2	25.497 s
3	25.487 s
4	25.456 s
5	25.551 s
Average time	21.849 s

Decryption

Table 6: Decryption Time Measurement for 2028-0000-bit KEY.

No.	Time (second)
1	25.456 s
2	25.246 s
3	25.228 s
4	25.524 s
5	25.064 s
Average time	25.304 s

CONCLUSION AND RECOMMENDATIONS:

People use cryptography to safeguard their personal and commercial data. There are many algorithms available for that like AES, and RSA but a problem with those is a limited length of key and every complex algorithm to implement. On top of that, remember plain text key is a tough job. In this research, I have used an image as a key, which opens up a door to remove the plain textkey and introduce a limitless bit of key which doesn't require memorization of the key. It also provides a highly customizable algorithm where we can also use any kind of matrix-based mathematical function which produce an 8-bit int result and has an inverse function.This algorithm has a huge amount of chance for customization. Combining it with AES may create more security than it is now. Also,through some research,there are some possibilities of optimizing this algorithm for the longer keys to reduce time complexity.

ACKNOWLEDGEMENT:

First of all, I would like to express my profound-gratitude to my honorable supervisor, Ms. Tania Akter Setu, for her constant and meticulous supervision, valuable suggestions, patience, and encouragement to complete the project work. I would also like to thank the CSE department of the UITS for providing us with the opportunity to have an industrial-level design experience as part of our curriculum for the master's program. Finally, I would like to thank our families and everybody who supported us and provided us with guidance for thecompletion of this project.

CONFLICTS OF INTEREST:

The authors state that there is no potential conflict of interest in publishing this research article.

REFERENCES:

1) Nitin Jirwan, Ajay Singh, Dr. Sandip Vijay, (2013). Review and Analysis of Cryptography Techniques. *International Journal of Scientific & Engineering Research.* **4**(3), 1-6. https://www.ijser.org/paper/Review-and-Analysis-of-Cryptography-Techniques.html

2) Yahia Alemami, Mohamad Afendee Mohamed, Saleh Atiewi, (2019). Research on Various Cryptography Techniques. *International Journal of Recent Technology and Engineering (IJRTE),* **8** (2S3), 395-405. https://www.ijrte.org/wp-content/uploads/papers/v8i2S3/B10690782S319.pdf

Citation: Parag MR., and Setu TA. (2022). Data encryption using image processing: a brief study. *Aust. J. Eng. Innov. Technol.*, **4**(3), 45-51. https://doi.org/10.34104/ajeit.022.045051