Journal of Engineering Science and Technology
Vol. 16, No. 6 (2021) 4920 - 4932

IMPROVISING DATA SECURITY MEASURES USING RAJAN TRANSFORM

ANKIT VISHNOI[1,*], DURGANSH SHARMA[2], MANISH PRATEEK[3]

[1]University of Petroleum and Energy Studies, Dehradun, India
[2]Christ University, Delhi, India
[3]Dev Bhoomi Uttarakhand University, Dehradun, India
*Corresponding Author: avishnoi@ddn.upes.ac.in

Abstract

Data security has always been a concern with the use of a large amount of data in our day-to-day life. There are many methods suggested and presented to secure data during the stages of its preprocessing and post-processing. However, many of them are not following the process of Homomorphism. During the study of Fast Fourier transform (FFT), Hadamard transform (HT) and Rajan transform (RT), this research work encountered a method that uses the cyclic, dyadic and graphical inverse properties of data and encrypts them which makes them homomorphic. This paper is targeting to improvise the data security measures using Homomorphism-based Rajan Transform, a method, which can help in securing data while data processing. The proposed methodology works in such a way that the encrypted data is available for processing without decrypting data into the original form. The performance of the proposed method is described by the efficiency of the algorithm, key size, Block size, and no of rounds required to complete the encryption. It has been found, if we take 512 bits of input data to get 512-bit ciphertext, it takes 9 rounds and generates a 4608-bit key.

Keywords: Cyclic, Data security, Dyadic, Graphical inverse, Homomorphic transform, Rajan transform

1. Introduction

Cryptography has always been instrumental in data security, especially when it comes to device new methods of the encryption scheme. This research aims to find the possible application of homomorphic transform towards the development of an encryption scheme using Rajan Transform. The use of homomorphic transform goes in parallel with the requirement of implementation of any calculation/action to process the data while maintaining its privacy. Homomorphic properties of the Rajan transform make it suitable for the encryption method in this research work, and its implementation is related to the problem statement of research work [1].

With the extensive use of applications, devices, and the internet, most humans and organizations are facing threats involving data security and data breach of their valuable data and information. Almost everyone is a potential target of Phishing, Wishing, Viruses, Ransomware, and various other threats. They are supposed to be two approaches to deal with such situations; Reactive and Proactive approaches. In a reactive approach, we can only try to recover from the disaster or to reduce the degree of impact of the disaster. Nevertheless, in a Proactive approach, we try to secure the data in the system or transit. One way of securing the data is to use the Rajan Transform for the cryptography process[1].

According to the various surveys carried out until now, it has been observed, the cryptographic process is broadly categorized into two categories[2][3]:

- Private Key Cryptography (Symmetric-key)
- Public Key Cryptography (Asymmetric Key)

1.1. Private key cryptography (Symmetric-key)

Private Key cryptography, also known as symmetric-key cryptography, uses the same key for the Encryption and Decryption process. This process involves the same key for both processes. Key generated during the encryption process and transferred to the decryption end for further processing. Private keys can be divided into subkeys to make the process faster in comparison with Public-key Cryptography[4].

1.2. Public key cryptography (Asymmetric key)

Public Key Cryptography, also named asymmetric key cryptography, uses a pair of Keys; the public key and a private key set of the user. When a sender needs to encrypt the data, it uses the public key of the receiver and sends it to the receiver. The key can be transferred by key administration. Besides, the decryption process will require the private key of the user. This is a complex process in comparison with Private Key Cryptography, thus requiring more memory and CPU time to calculate the result. However, Public key Cryptography is considered to be more secure, comparatively[4].

1.3. Cryptographic algorithms

The strength of cryptographic algorithms depends upon the key generation process, the number of rounds, the time is taken in the encryption and decryption process, and the memory required to hold the data during the cryptographic process. The computer system that enables cryptographic algorithms is termed a cryptographic system. Thus we can describe the Cryptographic system, as a combination of

cryptographic algorithms, key management systems, number of rounds, memory, computation power, and hash functions [5-7].

Few such examples are mentioned in Table 1.

Table 1. Comparison of different Encryption Algorithms.

Algorithm	Key Size	Block	Round	Flexible	Features
DES [8]	64 bits	64 bits	16	No	Not Strong Enough
3DES [9]	112 or168	64 bits	48	Yes	Adequate Security
AES [10]	128, 192, 256 bits	128 bits	10,12, 14	Yes	Replacement for DES, Excellent Security
Blowfish [11]	32-448	64 bits	16	Yes	Excellent Security
RC4 [12]	Variable	40-2048	256	Yes	Fast Cipher in SSL
Serpent [13]	128- 256	128 bits	32	Yes	Good Security
IDEA [14]	128 bits	64 bits	8.5	No	Not Strong Enough
RSA [15]	1,024 to 4,096	128 bits	1	No	ExcellentSecurity, low speed
Diffie Hellman [16]	1024 to 4096 bits	512	-	Yes	Many attacks
MD5 [17]	Series of MD	512	4	Hash Function	Not Strong Enough

Any cryptographic algorithms can be compared based on the following characteristics [18, 19]:

- CPU time usage.
- Memory requirement
- Users
- Time to hack
- Time to recover data

Data Security can be increased by using a deep learning mechanism along with a cryptographic process using homomorphic transform. It prevents unauthorized data access and secures the cloud database as well [20].

2. Proposed Method

Rajan transform is a fast algorithm and developed on the lines of Hadamard transform and Fast Fourier transform, however different. Rajan Transform encrypts a number sequence $a(r)$, of length $R=2^o$ where o>0, to cipher sequence $A(o)$. Rajan Transform can encrypt any number sequence and follows isomorphism in this process. One can also find that Rajan Transform follows cyclic, dyadic, and graphical inverse properties of modern Algebra, which makes it Homomorphic. RT covers these different algebraic sequences a(r) into the same $A(o)$ but with different key $E(r)$. In other words, for a set of values that has some different set of cyclic, dyadic, and graphical inverse set of sequences get encrypted into the same sequence of ciphertext, with a different cryptographic key illustrated the reason, why it is viewed as a Transform.

The above-stated statement can be explained mathematically as shared below, describing the entire cryptographic process using Rajan Transform.

2.1. Encryption

Consider a given sequence of r bits, $a(r)$, where r is defined in power of 2. If not, need to add temporary zero bits so that it can easily divided into two equal half for processing. Each having (R/2) points and satisfying the following:

$$p(n) = a(m) + a(m+(R/2)); \; 0 \leq n \leq (R/2); \; 0 \leq m \leq (R/2) \quad (1)$$

$$q(n) = |a(m) - a(m-(r/2)); \; 0 \leq n \leq (R/2); \; 0 \leq m \leq (R/2) \quad (2)$$

Each (R/2) segment is further divided into (R/4) segments until further division is not possible.

$$p_1(o) = p(n) + p(n+(R/4)); \; 0 \leq o \leq (R/4); \; 0 \leq n \leq (R/4) \quad (3)$$

$$p_2(o) = |p(n) - p(n-(R/4))|; \; 0 \leq o \leq (R/4); \; 0 \leq n \leq (R/4) \quad (4)$$

$$q_1(o) = q(n) + q(n+(R/4)); \; 0 \leq o \leq (R/4); \; 0 \leq n \leq (R/4) \quad (5)$$

$$q_2(o) = |q(n) - q(n-(R/4))|; \; 0 \leq o \leq (R/4); \; 0 \leq n \leq (R/4) \quad (6)$$

This process will continue till further division is possible. Anyone from here can divide that the total no of states can be computed as $\text{Log}_2(R)$

2.2. How Encryption is done

To understand, how Rajan Transform is applied for encryption, let us assume sum as '+' and difference as '~'.

2.2.1. Encryption algorithm

Step 1: Consider a(r) number of sequence of length $R=2^o$, where O > o

i.e. (input type: numbers ≥ 0)

Step 2: If the input is not in the power of 2 add extra zeros to make it in the power of 2

Step 3: Divide the input into two equal half

Step 4: Calculate the value of p and q as per the formulas

$p(n) = a(m) + a(m+(R/2)); \quad 0 \leq n \leq (R/2); \quad 0 \leq m \leq (R/2)$

$q(n) = |a(m) - a(m-(r/2)); \quad 0 \leq n \leq (R/2); \quad 0 \leq m \leq (R/2)$

Each (R/2) segments further divided into (R/4) segments until the further division is not possible.

$p_1(o) = p(n) + p(n+(R/4)); 0 \leq o \leq (R/4); \quad 0 \leq n \leq (R/4)$

$p_2(o) = |p(n) - p(n-(R/4))|; 0 \leq o \leq (R/4); \quad 0 \leq n \leq (R/4)$

$q_1(o) = q(n) + q(n+(R/4)); 0 \leq o \leq (R/4); \quad 0 \leq n \leq (R/4)$

$q_2(o) = |q(n) - q(n-(R/4))|; 0 \leq o \leq (R/4); \quad 0 \leq n \leq (R/4)$

Step 5: Repeat step 4 until the value of the input is equal to $\log_2$ number of points. Store output in X(o) (output is again the numbers ≥ 0)

2.2.2. Key generation algorithm

Step 1: After step 3 of encryption, calculate the position to store the additional value of inputs

Step 2: Store the value of the sign as follows:
If the value is +ve then store 0 and if the value is –ve then sore 1
Step 3: Repeat steps 1 and 2 until the value of the input is equal to $\log_2$ no of points
Step 4: Store the final string as Key

2.2.3. Decryption algorithm

Step 1: Store the value of $A(o)$ such as each segment has two values, i.e., (input type: numbers ≥ 0, which is received after encryption process)
Step 2: for positive sample values
$p(2n) = max(A(2o), A(2o+1)) – p(2n+1)$
$p(2n+1) = (A(2o) + A(2o+1))/2$
Step 3: for the negative set of values
$p(2n) = (A(2o) + A(2o+1))/2$
$p(2n+1) = max(A(2o), A(2o+1)) – p(2n)$
Step 3: Repeat step 2 and 3 until no of iteration
Step 4: Store the final value as a(o) (output is again the numbers ≥ 0, which is the same as the input of the encryption process)

For a better understanding of the process, let us assume the values as-
$a(r) = 2, 7, 4, 5, 0, 1, 8, 5$ (see Fig. 1).

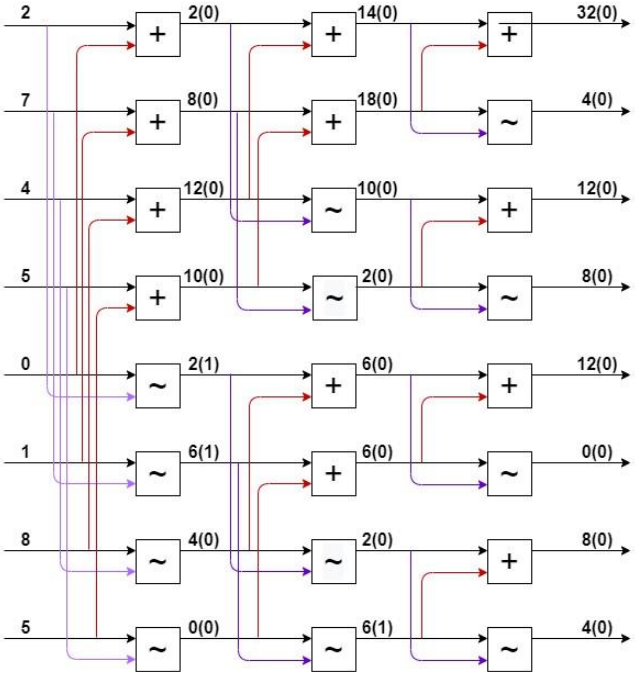

Fig. 1. Encryption process using RT.

Hence sequence a(r) = 2, 7, 4, 5, 0, 1, 8, 5 is encrypted in
Cipher text A(o) = 32, 4, 12, 8, 12, 0, 8, 4
And the encryption key is E(y) = $E_1(y)$ $E_2(y)$ $E_3(y)$
So, E(y)= 000011000000000100010000
Similarly, the Decryption will be like (see Fig. 2.).

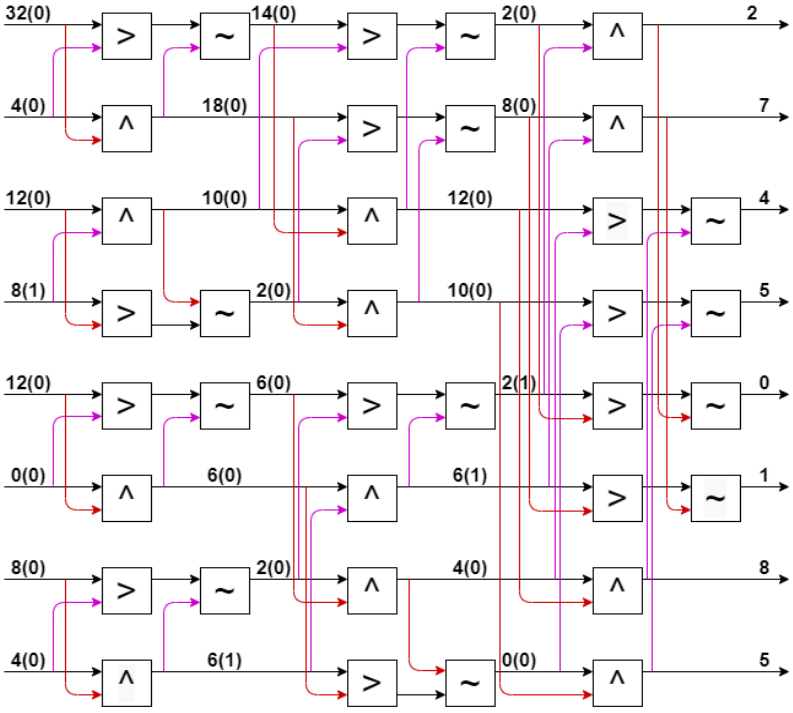

Fig. 2. Decryption process using RT.

Figure 2 illustrates the decryption method one can easily see the decrypted sequence is the same as the input/plain text sequence.

3. Algebraic properties of Rajan Transform

RT uses some of the algebraic properties, which makes it Homomorphic as follows:

3.1. Cyclic Shift

The sequence which we have used to generate the cipher text a(r) = 2, 7, 4, 5, 0, 1, 8, 5 can generate seven cyclic shifted sequences $a_{c1}(r)$ = 7, 4, 5, 0, 1, 8, 5, 2 , $a_{c2}(r)$ = 4, 5, 0, 1, 8, 5, 2, 7 , $a_{c3}(r)$ = 5, 0, 1, 8, 5, 2, 7, 4 , $a_{c4}(r)$ = 0, 1, 8, 5, 2, 7, 4, 5 , $a_{c5}(r)$ = 1, 8, 5, 2, 7, 4, 5, 0 , $a_{c6}(r)$ = 8, 5, 2, 7, 4, 5, 0, 1 , and $a_{c7}(r)$ = 5, 2, 7, 4, 5, 0, 1, 8. Here the interesting fact is that, when we encrypt these new seven cyclic shifted sequences in to cipher text the output will always be the same A(o) = 32, 4, 12, 8, 12, 0, 8, 4 as we did with our original sequence, but only the encryption key E(y) will change.

3.2. Graphical inverse

Same is applicable of the graphical inverse of the original sequence. The graphical inverse of original sequence a(r) = 2, 7, 4, 5, 0, 1, 8, 5 is $a^{-1}(r)$ = 5, 8, 1, 0, 5, 4, 7, 2. Now again the cyclic shift versions stands true for this graphical inverse sequence anyone can calculate the cipher text of $a_{c1}^{-1}(r)$ = 8, 1, 0, 5, 4, 7, 2, 5 , $a_{c2}^{-1}(r)$ = 1, 0, 5, 4, 7, 2, 5, 8 , $a_{c3}^{-1}(r)$ = 0, 5, 4, 7, 2, 5, 8, 1 , $a_{c4}^{-1}(r)$ = 5, 4, 7, 2, 5, 8, 1, 0 , $a_{c5}^{-1}(r)$ = 4, 7, 2, 5, 8, 1, 0, 5 , $a_{c6}^{-1}(r)$ = 7, 2, 5, 8, 1, 0, 5, 4 , and $a_{c7}^{-1}(r)$ = 2, 5, 8, 1, 0, 5, 4, 7 will be the same A(o) = 32, 4, 12, 8, 12, 0, 8, 4 as of the original sequence but with the different encryption key E(y).

3.3. Dyadic shift

Dyadic shift refers to the chance in the sequence in-group of two dyade means a group of two. Let us take the same sequence a(r) = 2, 7, 4, 5, 0, 1, 8, 5. There are two ways to identify the dyadic shift of the original sequence. Let us discus it one by one. The original sequence may be divided into two equal half blocks like $S_d^2[a(r)]$ = 0, 1, 8, 5, 2, 7, 4, 5. Again repeat the same step to get $S_d^4[S_d^2[a(r)]]$ = 8, 5, 0, 1, 4, 5, 2, 7. In continuation to that one can get $S_d^8[S_d^4[S_d^2[a(r)]]]$ = 5, 8, 1, 0, 5, 4, 7, 2. Now if one do the encryption of these dyadic sequences one can get same cipher text A(o) = 32, 4, 12, 8, 12, 0, 8, 4 but different encryption key E(y). Another way of getting the dyadic shift of original sequence a(r) = 2, 7, 4, 5, 0, 1, 8, 5 is to first get $S_d^8[a(r)]$ = 7, 2, 5, 4, 1, 0, 5, 8. Then $S_d^4[S_d^8[a(r)]]$ = 4, 5, 2, 7, 8, 5, 0, 1 , and $S_d^2[S_d^4[S_d^8[a(r)]]]$ = 5, 8, 1, 0, 5, 4, 7, 2. Interestingly $S_d^8[S_d^4[S_d^2[a(r)]]]$ = $S_d^2[S_d^4[S_d^8[a(r)]]]$ and it stand true for all the sequences. One can easily calculate the graphical inverse and cyclic shift of these values and calculate the cipher text which always will come A(o) = 32, 4, 12, 8, 12, 0, 8, 4 as of the original sequence but with the different encryption key E(y).

4. How RT is good for Encryption

The purpose of this study was to identify, how can we describe a method which provides us different encryption keys E(y) using the different possible combinations of an input sequence a(r) = 2, 7, 4, 5, 0, 1, 8, 5 by getting the same output A(o) = 32 4, 12, 8, 12, 0, 8, 4. This satisfied the properties of homomorphic transform, which in turn provides the encrypted value in the same form as the original form. To be precise, if we encrypt the numbers into ciphertext, we will get numbers in return with the different encryption keys. Similarly text to text and image to image. As long as one goes with the process can identify, for an 8-bit input, the 8-bit output is generated and a 24-bit encryption key. The studies have proven in the past that the longer is the size of the key, the more complex it is to crack the encryption process.

The efficiency of the received output is optimum since the size of the key increases as per the input bits. In the last section, it has been discussed, for an 8-bit plain number the output is 8-bit, however, the size of the key is 24 bit. If the size of the input is changed to 32 bit in place of 8 bit, the output size is the same as input 32 bit, however, the key sizes changed to 160 bit. Above all, this 160-bit key generated is unique for only a single 32-bit sequence, and not for all. For every other 32-bit data sequence, the 160-bit key will be unique, which makes this encryption tough to crack

5. Results and Discussion

To test the research outcome of this research few test cases were designed. These test cases included all uppercase alphabets and numeric digits, only. For these test cases, alphabets and numbers are represented in the form of an 8*8 pattern, as shown in Table 2.

Table 2. Matrix representation of alphabets and numeric digits.

0 0 0 0 0 0 0 0 0 0 0 **1 1** 0 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 0 0 0 0 0 0 **A**	0 0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 **1** 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 0 **1 1 1 1** 0 0 0 0 0 0 0 0 0 0 **B**	0 0 0 0 0 0 0 0 0 0 0 **1 1 1** 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 **1 1 1** 0 0 0 0 0 0 0 0 0 0 **C**	0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1 1 1 1 1 1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **D**
0 0 0 0 0 0 0 0 0 0 **1 1 1 1 1** 0 0 0 **1** 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1 1 1 1 1** 0 0 0 0 0 0 0 0 0 **E**	0 0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1 1 1 1 1** 0 0 0 0 0 0 0 0 0 **F**	0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 **1** 0 0 0 0 0 **1 1** 0 0 0 0 0 0 **1** 0 **1** 0 0 0 0 0 **1** 0 0 **1** 0 0 0 0 1 0 0 0 1 0 **G**	0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1 1 1 1 1** 0 0 0 0 0 0 0 0 0 **H**
0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 **1 1** 0 0 **1 1** 0 0 **1** 0 **1 1** 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **I**	0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 **1 1** 0 0 0 **1** 0 0 **1** 0 **1** 0 0 **1** 0 0 **1** 0 0 0 **1** 0 **1** 0 0 **1** 0 0 0 **1 1** 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **J**	0 0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 **1 1 1 1** 0 0 0 0 0 0 0 0 0 0 **K**	0 0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **L**
0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 **1 1** 0 0 **1 1** 0 0 **1** 0 **1 1** 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **M**	0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 0 **1 1 1 1** 0 0 0 0 **1** 0 0 0 **1** 0 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **N**	0 0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 **1 1 1 1** 0 0 0 0 0 0 0 0 0 0 **O**	0 0 0 0 0 0 0 0 0 **1 1 1 1 1** 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 **P**
0 0 0 0 0 0 0 0 0 0 **1 1 1 1** 0 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 **1 1 1 1** 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 **1** 0 0 **Q**	0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **R**	0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 **1** 0 **1** 0 0 0 0 0 **1** 0 **1** 0 0 0 0 0 **1** 0 0 **1** 0 0 0 **1** 0 0 0 0 **1** 0 **1** 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 **S**	0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 **1** 0 **1** 0 0 0 0 0 **1** 0 **1** 0 0 **1** 0 0 **1** 0 **1** 0 **1** 0 **1** 0 **1** 0 **1 1** 0 0 0 **1 1** 0 **1** 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **T**	0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 **1** 0 0 0 **1** 0 0 **1** 0 0 0 0 0 **1 1** 0 0 0 0 0 0 **1 1** 0 0 0 0 0 **1** 0 0 **1** 0 0 0 **1** 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 **U**

U	**V**	**W**	**X**
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
1 0 0 0 0 0 **1** 0	0 **1 1 1 1 1 1** 0	0 0 0 **1 1** 0 0 0	0 0 0 0 **1** 0 0 0
0 **1** 0 0 0 **1** 0 0	0 0 0 0 0 **1** 0 0	0 0 **1** 0 0 **1** 0 0	0 0 0 0 **1** 0 0 0
0 0 **1** 0 **1** 0 0 0	0 0 0 0 0 **1** 0 0	0 0 **1** 0 0 **1** 0 0	0 0 0 0 **1** 0 0 0
0 0 0 **1** 0 0 0 0	0 0 0 0 **1** 0 0 0	0 0 **1** 0 0 **1** 0 0	0 0 0 0 **1** 0 0 0
0 0 0 **1** 0 0 0 0	0 0 0 **1** 0 0 0 0	0 0 **1** 0 0 **1** 0 0	0 0 0 0 **1** 0 0 0
0 0 0 **1** 0 0 0 0	0 0 **1** 0 0 0 0 0	0 0 **1** 0 0 **1** 0 0	0 0 0 0 **1** 0 0 0
0 0 0 **1** 0 0 0 0	0 **1 1 1 1 1 1** 0	0 0 0 **1 1** 0 0 0	0 0 0 **1 1 1** 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Y	**Z**	**0**	**1**
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 **1 1 1 1** 0 0	0 **1 1 1 1 1** 0 0	0 0 0 0 0 **1** 0 0	0 **1 1 1 1 1 1** 0
0 **1** 0 0 0 0 **1** 0	0 0 0 0 0 0 **1** 0	0 0 0 0 **1 1** 0 0	0 **1** 0 0 0 0 0 0
0 0 0 0 0 0 **1** 0	0 0 0 **1 1 1** 0 0	0 0 0 **1** 0 **1** 0 0	0 **1 1 1 1 1** 0 0
0 0 **1 1 1 1** 0 0	0 0 0 0 0 0 **1** 0	0 0 **1** 0 0 **1** 0 0	0 0 0 0 0 0 **1** 0
0 **1** 0 0 0 0 0 0	0 0 0 0 0 0 **1** 0	0 **1 1 1 1 1 1** 0	0 0 0 0 0 0 **1** 0
0 **1 1 1 1 1 1** 0	0 **1 1 1 1 1** 0 0	0 0 0 0 0 **1** 0 0	0 **1 1 1 1 1** 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
2	**3**	**4**	**5**
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 **1 1 1 1 1** 0	0 **1 1 1 1 1 1** 0	0 0 **1 1 1 1** 0 0	0 0 **1 1 1 1** 0 0
0 **1** 0 0 0 0 0 0	0 0 0 0 0 0 **1** 0	0 **1** 0 0 0 0 **1** 0	0 **1** 0 0 0 0 **1** 0
0 **1** 0 0 0 0 0 0	0 0 0 0 0 **1** 0 0	0 0 **1 1 1 1** 0 0	0 0 **1 1 1 1** 0 0
0 **1 1 1 1 1** 0 0	0 0 0 0 **1** 0 0 0	0 **1** 0 0 0 0 **1** 0	0 0 0 0 0 0 **1** 0
0 **1** 0 0 0 0 **1** 0	0 0 0 **1** 0 0 0 0	0 **1** 0 0 0 0 **1** 0	0 0 0 0 0 0 **1** 0
0 0 **1 1 1 1** 0 0	0 0 **1** 0 0 0 0 0	0 0 **1 1 1 1** 0 0	0 0 **1 1 1 1** 0 0
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
6	**7**	**8**	**9**

For analysis, the matrix representation of the letter [A] is considered. Its encrypted matrix is as shown in Table 3.

Table 1. Encryption of 8*8 matrix representing [A] and its cypher sequences.

0 0 0 0 0 0 0 0	14 00 06 00 14 00 06 00
0 0 0 **1 1** 0 0 0	02 00 02 00 02 00 02 00
0 0 **1** 0 0 **1** 0 0	06 00 02 00 06 00 02 00
0 0 **1** 0 0 **1** 0 0	02 00 02 00 02 00 02 00
0 0 **1 1 1 1** 0 0	10 00 02 00 10 00 02 00
0 0 **1** 0 0 **1** 0 0	02 00 02 00 02 00 02 00
0 0 **1** 0 0 **1** 0 0	06 00 02 00 06 00 02 00
0 0 0 0 0 0 0 0	02 00 02 00 02 00 02 00
[A]	Cipher(A)

For study purposes, the encrypted value of the matrix form of [A] has provided us with some valuable results. Its observations show that [A] can be represented in six different forms of the matrix. The analysis of the RT (encrypted) and IRT (decrypted) values of six different matrix forms of [A] represented some errors, as illustrated in Table 4.

Table 2. Error observation of alphabet [A] during different orientations

Input	**Encryption**	**Error**	**Remarks**
0 0 0 0 0 0 0 0	14 00 06 00 14 00 06 00	0 0 0 0 0 0 0 0	Orientation: Normal
0 0 0 **1 1** 0 0 0	02 00 02 00 02 00 02 00	0 0 0 0 0 0 0 0	Scanning: Row-Column
0 0 **1** 0 0 **1** 0 0	06 00 02 00 06 00 02 00	0 0 0 0 0 0 0 0	Error in Encryption=NIL
0 0 **1** 0 0 **1** 0 0	02 00 02 00 02 00 02 00	0 0 0 0 0 0 0 0	Efficiency=100%
0 0 **1 1 1 1** 0 0	10 00 02 00 10 00 02 00	0 0 0 0 0 0 0 0	
0 0 **1** 0 0 **1** 0 0	02 00 02 00 02 00 02 00	0 0 0 0 0 0 0 0	
0 0 **1** 0 0 **1** 0 0	06 00 02 00 06 00 02 00	0 0 0 0 0 0 0 0	
0 0 0 0 0 0 0 0	02 00 02 00 02 00 02 00	0 0 0 0 0 0 0 0	

00000000	14 00 06 00 14 00 06 00	00000000	Orientation: Inverted
00100100	02 00 02 00 02 00 02 00	00000000	Scanning: Row-Column
00100100	06 00 02 00 06 00 02 00	00000000	Error in Encryption=NIL
00111100	02 00 02 00 02 00 02 00	00000000	Efficiency=100%
00100100	10 00 02 00 10 00 02 00	00000000	
00100100	02 00 02 00 02 00 02 00	00000000	
00011000	06 00 02 00 06 00 02 00	00000000	
00000000	02 00 02 00 02 00 02 00	00000000	
00001100	14 00 06 00 14 00 06 00	00000000	Orientation: Translated
00010010	02 00 02 00 02 00 02 00	00000000	Scanning: Row-Column
00010010	06 00 02 00 06 00 02 00	00000000	Error in Encryption=NIL
00011110	02 00 02 00 02 00 02 00	00000000	Efficiency=100%
00010010	10 00 02 00 10 00 02 00	00000000	
00010010	02 00 02 00 02 00 02 00	00000000	
00000000	06 00 02 00 06 00 02 00	00000000	
00000000	02 00 02 00 02 00 02 00	00000000	
00000000	14 02 02 02 16 02 06 02	02428202	Orientation: 90° Rotated
00000000	00 00 00 00 00 00 00 00	20202020	Scanning: Row-Column
01111100	10 02 04 02 02 02 02 02	42424202	Error in Encryption=NIL
00010010	00 00 00 00 00 00 00 00	20202020	Efficiency=71%
00010010	14 02 02 02 06 02 06 02	42024242	
01111100	00 00 00 00 00 00 00 00	20202020	
00000000	10 02 06 02 02 02 02 02	42424202	
00000000	00 00 00 00 00 00 00 00	20202020	
00000000	14 02 02 02 16 02 06 02	02428202	Orientation: 270° Rotated
00000000	00 00 00 00 00 00 00 00	20202020	Scanning: Row-Column
01111100	10 02 04 02 02 02 02 02	42424202	Error in Encryption=NIL
01001000	00 00 00 00 00 00 00 00	20202020	Efficiency=71%
01001000	14 02 02 02 06 02 06 02	42024242	
01111100	00 00 00 00 00 00 00 00	20202020	
00000000	10 02 06 02 02 02 02 02	42424202	
00000000	00 00 00 00 00 00 00 00	20202020	

The error was observed in 2 out of 5 representations of [A]. Similar results were observed with the rest of the alphabets and numbers except [X] and [O]. Thus, one can easily get that this encryption method is giving expected results if we are using the normal, inverted, and translated form of the matrix.

The efficiency of the above-given method can also be summarized by Key size, block size, and the number of rounds for the execution of the process. Rajan Transform has provided flexibility during the use of block size and key size. From the above results, it has been derived that if the user takes the input of 2^n bits then the encryption process will use 2^n*n key size and n number of rounds to generate 2^n bits of data. The time complexity of the above algorithm is of the order of n means O(n).

As shown in Table 5 below, for 512 bits of input data, the Rajan Transform will take 9 rounds to generate ciphertext along with a 4096-bit key.

Table 5. Comparison of different Encryption algorithms with Rajan transform.

Algorithm	Key Size	Block	Round	Flexible	Features
DES [8]	64 bits	64 bits	16	No	Not Strong Enough
3DES [9]	112 or168	64 bits	48	Yes	Adequate Security
AES [10]	128, 192, 256 bits	128 bits	10,12, 14	Yes	Replacement for DES, Excellent Security
Blowfish [11]	32-448	64 bits	16	Yes	Excellent Security
RC4 [12]	Variable	40-2048	256	Yes	Fast Cipher in SSL
Serpent [13]	128- 256	128 bits	32	Yes	Good Security
IDEA [14]	128 bits	64 bits	8.5	No	Not Strong Enough
RSA [15]	1,024 to 4,096	128 bits	1	No	Excellent Security, low speed
Diffie Hellman [16]	1024 to 4096 bits	512	-	Yes	Many attacks
MD5 [17]	Series of MD	512	4	Hash Function	Not Strong Enough
Rajan Transform	4096 bit	512 bit	9	Yes	Excellent

6. Conclusion and Future Scope

The input data considered for this study are alphabets and numeric values, which is generating the results, as expected. Further, the same process shall be implemented for text and image as well, in our future work. Though the results have shown some interesting figures.

Acknowledgment

I would like to thank, the University of Petroleum for providing me the environment to carry on my research. Also Prof. E. G Rajan for blessings and valuable guidance in writing this paper

References

1. Naidu, M.E.; and Rajan, E.G. (2006). Two dimensional object recognition using rajan transform. *Engineering Letters*, 13(3), 268–274.
2. Asif, A.M.A.M.; and Hannan, S.A. (2014). A review on classical and modern encryption techniques. *International Journal of Engineering Trends and Technology*, 12(4), 199-203.

3. Newadkar, S.; Dev, A.; and Sharma, S. (2014). *Data storage/retrieval with access control , security and pre-fetching*. A Project proposal, COEN 241 - Cloud Computing, 1–17.
4. Gupta, A.; and Walia, N.K. (2014). Cryptography algorithms : A review. *International Journal of Engineering Development and Research*, 2(2), 1667-1672.
5. Koko, S.O.A.F.M.; and Mustafa, A.B.A.N. (2015). Comparison of various encryption algorithms and techniques for improving secured data communication. *IOSR Journal of Computer Engineering*, 17(1), Ver-3, 62–69.
6. Sharma, R.; and Bollavarapu, S. (2015). Data security using compression and cryptography techniques. *International Journal of Computer Applications*, 117(14), 15-18.
7. Basharat, I.; Azam, F.; and Muzaffar, A.W. (2012). Database security and encryption: A survey study. *International Journal of Computer Applications*, 47(12), 28–34.
8. .Lide, D.R (2018). *A century of excellence in measurements, standards, and technology: Data encryption standard*. CRC Press Inc., 250-253.
9. Singh, G.; and Kinger, S. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications*, 67(19), 33-38.
10. Mathur, N.; and Bansode, R. (2016). AES Based text encryption using 12 rounds with dynamic key selection. *Procedia Computer Science*, 79, 1036-1043.
11. Ross, B.S.; and Josephraj, V. (2017). Performance enhancement of blowfish encryption using RK-blowfish technique. *International Journal of Applied Engineering Research*, 12(20), 9236–9244.
12. Sriadhi, S.; Rahim, R.; and Ahmar, A.S. (2018). RC4 algorithm visualization for cryptography education," *IOP Journal of Physics: Conference Series*, 1028, 012057.
13. Cayabyab, G.T.; Sison, A.M.; and Hernandez, A.A. (2019). GISKOP: A modified key scheduling operation of international data encryption algorithm using serpent key scheduling. *ICCBD 2019: Proceedings of the 2nd International Conference on Computing and Big Data*, 53-57.
14. Abdullah, D.; Rahim, R.; Siahaan, A.P.U.; Ulva, A.F.; Fitri, Z.; Malahayati, M.; and Harun, H. (2018). Super-encryption cryptography with IDEA and WAKE algorithm. *IOP Journal of Physics: Conference Series*, 1019, 012039.
15. Nisha, S.; and Farik, M. (2017). RSA public key cryptography algorithm - A review. *International Journal of Scientific & Technology Research*, 6(7), 187-191.
16. Boni, S.; Bhatt, J.; and Bhat, S. (2015). Improving the Diffie-Hellman key exchange algorithm by proposing the multiplicative key exchange algorithm. *International Journal of Computer Applications*, 130(15), 7-10.
17. Dhany, H.W.; Izhari, F.; Fahmi, H.; Tulus, T.; and Sutarman (2018). Encryption and decryption using password based encryption, MD5, and DES. *Proceedings of International Conference on Public Policy, Social Computing and Development* 2017 (*ICOPOSDev* 2017), 278–283.

18. Kumari, S.; and Chawla, J. (2014). Comparative analysis on different parameters of encryption algorithms for information security. *International Journal of Innovations & Advancement in Computer Science*, 2(4), 123-129.
19. Al-Hazaimeh, O.M.A. (2013). A new approach for complex encrypting and decrypting data. *International Journal of Computer Networks and Communications*, 5(2), 95-103.
20. Phong, L.T.; Aono, Y.; Hayashi, T.; Wang, L.; and Moriai, S. (2018). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5), 1333-1345.