A Real-Time Encryption Algorithm For User Data Preservation In Mobile Computing

Okah C., Matthias D., Nwiabu N.
Department of Computer Science, Rivers State University, Nkpolu-Oroworukwo , Port Harcourt, Nigeria

ABSTRACT

More recently, many businesses are sending real-time Application (RTA) on the Internet and are frequently social networking applications. Therefore, the need to protect data and user restrictions is greater vital than ever. The encryption key is used to shield the security required for the application. But on account that the RTA has high-level data, the historical encryption approach is unsuccessful, considering most RTAs serve online, the encryption and violation protocols will take a little greater time to obtain end-to-end encryption. Generally. This research work developed a new algorithm to enhance encryption time and keep endpoint data to delay time and furnish a excessive level of security in the RTA. The outcomes received from the implementation of the algorithm exhibit that our algorithms are faster and less intrusive and less touchy to contemporary information in contrast to existing algorithms.

Keywords – *Mobile, confidentiality, cloud, privacy*

I. Introduction

With recent evolution of Internet, the world ensures that we are all connected at the touch of the button. Mobile Computing is future of the technology as it allows us to connect to the Internet and the data that demand distribution. Mobile cloud computing techniques legitimates diverse applications. Although mobile cloud computing has several advantages but a little worry about safeguard and confidentiality of data.

Privacy is the major concern in utilizing mobile cloud computing regardless of many services. With increased usage of mobile applications and social media communications, there is huge concern on privacy of user data. As privacy concerned transferring the plain text over the network, mobile users are liable to different kinds of attacks such as Denial of Service attack, malware injection attack and monitoring which results in the loss of data integrity. One of the data security issues is as a result of unencrypted data being passed due to the high volume of data. This situation often results to privacy leakage issues since plaintexts does not restrict or place any constraint for adversaries to capture information in a variety of ways, such as jamming, monitoring, and spoofing.

Many approaches emerged for privacy preserving data in mobile cloud computing. The first approach involved perturbing the input before searching. Though it has the benefit of simplicity it does not provide a formal framework for proving how much privacy is guaranteed. Secure Computation technique (Alex, G. and Ehud, G. 2006) has the advantage of providing a well defined model for privacy using cryptographic techniques and is also accurate. However it is a slower method and also has the drawback of space complexity. i.e., as all data are stored in the Cloud Database Server, it leads to a large memory requirement.

A. Statement of Problem

Despite several existing and proposed encryption algorithms to protect user's data privacy during real time applications in mobile computing, there still exist lots of problems that have kept RTA in mobile computing in difficult times:

B. Objective

The aim of this research is to create and develop an application that will encrypt and protect mobile user's private data in real time applications with the following objectives:

i. To design an algorithm with a higher conversion times and faster than AES with higher security.
ii. To use an object oriented programming language to implement an algorithm that is less complex with less rounds of encryption process that will reduce the packet delay time.
iii. To implement the algorithm using index table generation encryption such that the encryption key length cannot affects the speed of encryption or data privacy.

II. RELATED WORKS

i. Digital Era Encryption

The computerized period of the 1970s featured the significance of a key-based protection framework. Present day PCs comprehend that to send messages online without meeting the beneficiary in any case, they will require frameworks that utilization an alternate key for encryption than for encryption. On the off chance that encryption is contrasted and a PC, this framework will be contrasted with a lock that has a lock key and another key to debilitate the lock. Today, the Diffie-Hellman change is considered. Diffie-Hellman key trade is a sort of security strategy that permits two individuals to make private keys that they don't have before they know one another, which is regularly used to encode them. Whitfield Diffie and Martin Hellman previously adjusted the Diffie-Hellman adjustment in 1976. Key route changes depend on valuable devices and figurings quicker than logarithms. At the point when utilized effectively, the Diffie-Hellman key trade workshop key isn't transferable without copying a key. The quality of this calculation relies upon to what extent it takes to figure the open key exchange calculation (Diffie, H. 1976). Perceiving the failure of Diffie-Hellman key trade to send private data, Ron Rivest, Adi Shamir and Leonard Adleman built up a framework like the Diffie-Hellman framework, then again, actually the data could be encoded and transmitted. The mystery of the RSA, which is named after the names of the creators, depends on the way that duplication and coding are quicker than introductory appraisals. A total arrangement of two principles is utilized.

ii. ***Data Encryption Standard and Advanced Data encryption standard***

As with the previous note, the key to compiling data is to find the so-called Encryption Standard (DES) data. Data entry is a documentary awarded by the National Institute of Standards and Technology (NIST). In any case, it is now covered by another standard known as Advanced Encryption Standard (AES). DES is a closed 64-bit door that seems to extract 64 pieces of information. This contrasts with a sequence that is a bit slow at a time (some small page sessions, for example, one byte). DES was an inappropriate name in a research expedition founded by International Business Machines (IBM) in the late 1960s that brought together a person named LUCIFER. In the mid-1970s, he was chosen to promote LUCIFER with major changes. IBM says it is not only related to this progress that they are seeking specialized intelligence from the National Security Agency (NSA). The LUCIFER reform was adopted as a recommendation for a new privacy management standard proposed by the National Standards Office, as quoted by Dray, J. (2010) in one of his books entitled "Pain Management of Farmers in Java AES." Subsequently, it was adopted and adopted in 1977 as the Encryption Standard (Daemen, et al. 2010).

III. A. METHODOLOGY

Methodology of research is the science of proper modes and orders of procedure. Since this study attempts to explore the history of encryption and the real time application security in mobile computing systems and devices, the historical method will be the most suitable one for this study. Furthermore experimental research **allows cause and effect to be determined. Combining these two research methods will help us to study the past encryption and its challenges as we use the experimental research to propose new system. Therefore a** Hybrid research approach was adopted by combining the best features of historical research method and experimental research method. And the Index table generation encryption for Real-Time Applications is used for our design methodology.

B. Proposed System

Our proposed system is a new cryptographic that will eliminate encryption/decryption time delay that normally occurs in AES. We design an encryption process that will shorten the time it takes to encrypt/decrypt even more better than AES with improved security.

System is made up of two components:
(a) Index Generation Process
(b) Key Insertion Process

In our new system, the major requirement for this encryption application is shared key. The key is key in creating and ensuring better security; the key is 1024-bit long, the key is randomly selected which makes it difficult to hack.

(a) The Index Generation Process

Index generation component consists of:

i. Initial table
ii. Shared value
iii. Circular shifts
iv. Table of Indexes
v. The Extracted Indexes

i. **Initiate Table**; Consists of (16*16) horizontal and vertical as seen fig. 3.2, with table value from 0-F_{16}. the this first table is not confidential and is made public to the sender, receiver and the public.

ii. **Shared Value*:*** Shared value is a very important part that can be generated at random but kept and exchanged secretly; Diffie- Hellman is proposed for its exchange. Diffie Hellman is not an encryption algorithm; rather, it is a key exchange algorithm.

4	B	2	8	A	9	8	6	0	3

Table 3.1 **Shared Key**

iii. **Circular Shifts**: The shift and circular down permutation is performed using the shared key. The first number in the shared key is for left shift followed by another number for the circular down shift until the shared key is exhausted. This is as shown in Tables. 3.3, 3.4, 3.5 and 3.6 using the shared key in Table.3.1

F.	4	6	E.	D.	C.	E.	6	3	F	7	9	1	F,	9	F.
1	D,	1	E.	D.	C.	6	2	2	6	D	0	4	F,	A,	7
2	7	8	9	C.	3	B.	9	2	E	3	4	E,	C,	4	5
4	F.	4	4	9	E.	5	F.	2	2	4	C,	F	3	0	D,
8	3	F,	5	6	9	D.	A.	5	0	7	B,	8	1	7	4
E.	F.	0	0	4	1	D.	E.	B.	5	5	7	B	3	2	F,
8	0	2	A	4	4	E.	1	5	E,	9	5	8	6	A,	4
6	3	E,	9	6	F,	7	8	4	7	A,	A,	0	7	8	6

Table. 3.2: Initial table

Our proposed system will first transfer the first set of data to the other communicating component. This packet is a (16 * 16) rows and columns size table which its entries ranges from 0-F as shown in Table 3.2 and is randomly generated using a mathematical formula. This has a scientific equation previously communicated and known, the equation is used to initialize the same tables' value both at the receiver and sending sides.

9	3	E.	4	1	C	6	E.	2	0	9	1	1	B.	9	B.
3	6	B.	C.	7	C	E.	9	4	E.	D.	C	6	7	7	D.
3	2	D.	8	3	B.	B.	6	F.	3	A.	D.	9	8	5	F
D.	4	B.	C.	6	5	5	B.	3	D	9	A.	C.	B.	E.	2
0	A.	0	E.	0	8	6	D.	2	9	2	6	E.	D.	9	C.
D.	F	4	2	4	0	6	0	B	4	0	7	2	B	1	2
6	2	1	D.	1	B.	F.	A.	0	A.	9	2	9	2	D.	9
C.	0	E.	5	9	5	9	0	2	A.	9	C.	3	F.	7	6
F.	4	6	E.	D.	C.	E.	6	3	F.	7	9	1	F.	9	F.
1	D	1	E.	D	C	6	2	2	6	D	0	4	F	A	7
2	7	8	9	C.	3	B.	9	2	E.	3	4	E.	C.	4	5
4	F.	4	4	9	E.	5	F.	2	2	4	C.	F	3	0	D
8	3	F.	5	6	9	D.	A	5	0	7	B.	8	1	7	4
E.	F.	0	0	4	1	D.	E	B.	5	5	7	B.	3	2	F.
8	0	2	A.	4	4	E.	1	5	E.	9	5	8	6	A.	4
6	3	E.	9	6	F	7	8	4	7	A.	A.	0	7	8	6

Table 3.3 First Circular Left Shifts

Table 3.3 shows the first circular left shifts of the Index Generation Table Shifting with the value 4. The sender shifts the initial table columns circular right followed by a row circular down according to the shared values of table 3.1 between the sender and the receiver.

1	C	6	E.	2	0	9	1	1	B.	9	B.	9	3	E.	4
7	C.	E.	9	4	E.	D.	C.	6	7	7	D.	3	6	B.	C.
3	B.	B.	6	F.	3	A.	D.	9	8	5	F.	3	2	D.	8
6	5	5	B.	3	D.	9	A.	C.	B.	E.	2	D.	4	B..	C.
0	8	6	D.	2	9	2	6	E.	D.	9	C	0	A.	0	E.
4	0	6	0	B	4	0	7	2	B	1	2	D	F	4	2
1	B	F	A	0	A	9	2	9	2	D	9	6	2	1	D
9	5	9	0	2	A	9	C	3	F	7	6	C	0	E	5
D	C	E	6	3	F	7	9	1	F	9	F	F	4	6	E
D	C	6	2	2	6	D	0	4	F	A	7	1	D	1	E
C	3	B	9	2	E	3	4	E	C	4	5	2	7	8	9
9	E	5	F	2	2	4	C	F	3	0	D	4	F	4	4
6	9	D	A	5	0	7	B	8	1	7	4	8	3	F	5
4	1	D	E	B	5	5	7	B	3	2	F	E	F	0	0
4	4	E	1	5	E	9	5	8	6	A	4	8	0	2	A
6	F	7	8	4	7	A	A	0	7	8	6	6	3	E	9

Table 3.4 First Circular down shift

Table 3.4 shows the resulting table from the first circular left shifts of the Index Generator Table Shifting with the value 4 and the row to shift circular down with value of Baccording to the shared values of table 3.1.

iv. **Table of Indexes**: The table of indexes is the result after all rounds of permutations of the initial table using the shared value with a two digits length extracted out from the shared value, then modulo operation is used to perform the shift operations(Shifted rows/columns = Shared value Mod 16).

v. **The Extracted Indexes**: Another table called table of extracted indexes is derived from the Table of Indexes, we select the first octet as the first index and the second octet as the second index and so on, in this case we will have 128 indexes

(b) The Key Insertion Process

The key insertion component consists of:

1. The Plain Text Data
2. The Key Generation:
3. XoR (Encryption) Process:
4. The Key Insertion:

1. **The Plain Text Data**: This is the data being transmitted; which could be of any data type. This does not have restriction of size but it is better fit on one packet size

minus the key size of the Maximum Transfer Unit (MTU); (Plain-Text-Size = (MTU)-(Keysize)).

2. **The Key Generation:** The key is chosen generated randomly by the user. This research, we proposed a key length of 1024 bits though it can be varied over time. Users could change key during the communication session which won't affect the speed of the techniques.

3. **XoR (Encryption) Process:**The level text data is mainly associated with XoRed, the first 1024 bits of data are XoRed and the key, the second 1024 bits of data is XoRed with the same key and continue until there is no other clear text available. Data found.

4. **The Key Insertion**: The keys are inserted into the XoRed table that is derived from step 3 above, the input string is made according to the Index method, the first byte of the key is inserted into the XoRed table using our object as the first index value in the second column. an octet will be placed based on the value of the second index and so on until the end of the key, where the index is set at the appropriate position as shown in the text entry.

C. System Design

System design is the process of defining the architecture, modules, interfaces and data for a system to satisfy specified requirement. It includes the different sub-systems that made up the encryption and their components.

D. Design Method

We shall be using a hybrid approach in our design methodology. This approach will incorporate the best features of model-driven methodology (especially object-oriented analysis) and Rapid Application Development (RAD) methodology.

The Structured Systems Analysis and Design Methodology (SSADM), was once a very popular methodology but it has lost favour as a methodology. As Whitten (2004) had rightly pointed out, "the practice of structured analysis for software design has greatly diminished in favour of object-oriented methods".

While model- driven methodology emphasizes the drawing of pictorial system models that represent either a current reality or a target vision of the system, rapid application development emphasizes extensive user involvement in the rapid and evolutionary construction of a working prototype of a system to accelerate the system development process.

E. Architectural Design

The architectural design of our system is shown in figure 3.1 which shows the detailed encryption process. This architecture is easy but very difficult to guess. The design consists of a combination index generation and key insertion:

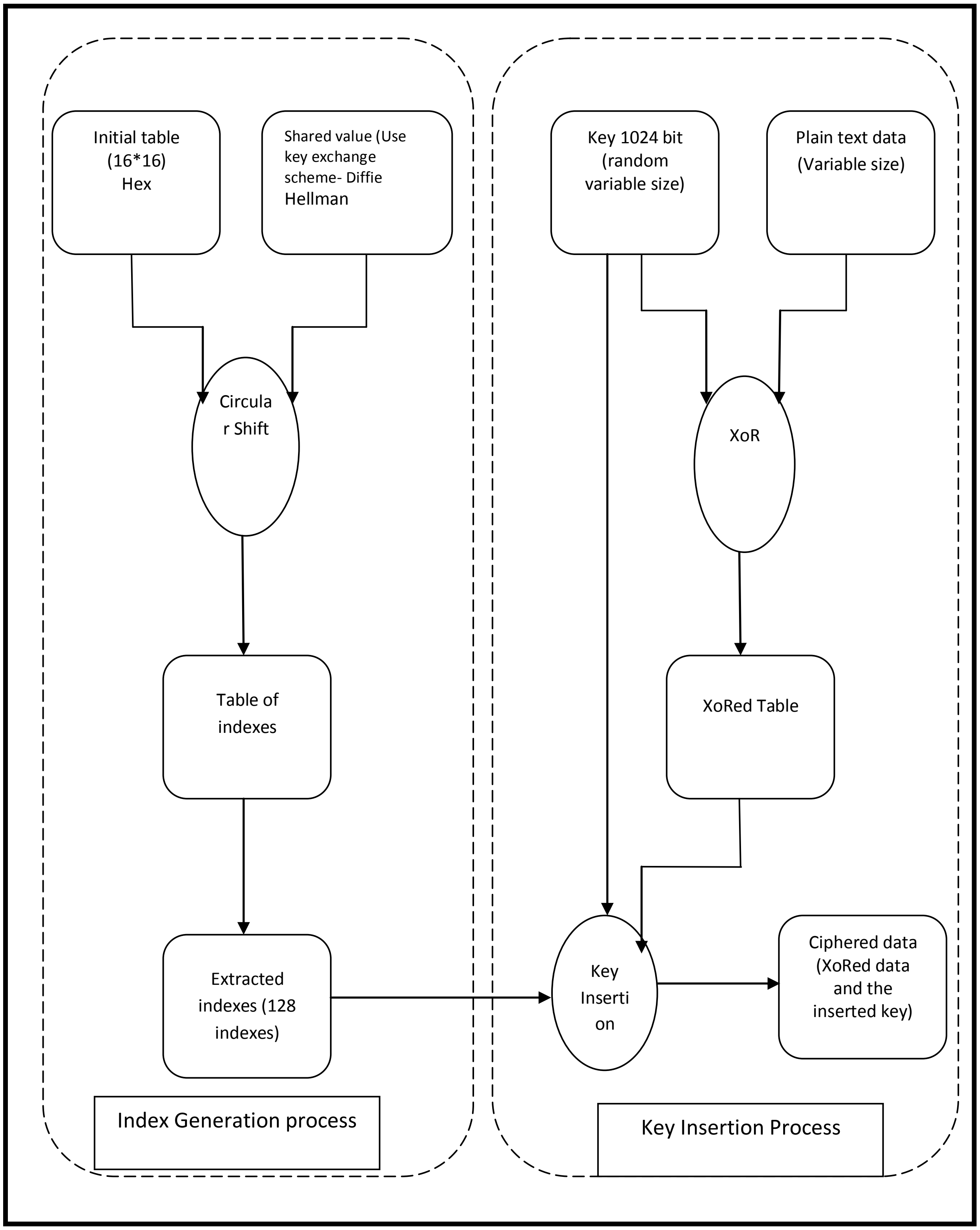

Initial table (16*16) Hex
Shared value (Use key exchange scheme- Diffie Hellman
Circular Shift
Table of indexes
Extracted indexes (128 indexes)
Index Generation process
Key 1024 bit (random variable size)
Plain text data (Variable size)
XoR
XoRed Table
Key Insertion
Ciphered data (XoRed data and the inserted key)
Key Insertion Process

Fig 3.1: Architectural Design of the Propose system

Fig 3.1 represents the entire systems architecture of our proposed system. It is made up of two components joined together to achieve the system. The two components are Index generation component and Key insertion component. The index generation is the component that is responsible for the circular shift and table of index generation which guides the key insertion component on where to insert keys. The key insertion component normally XORed the message to send and the 1024 key selected by the sender which also the receiver knows and the XORed table is generated. The 1024 key is also inserted into the generated XORed table going by the rules generated by the component generating the index as well as inserts the key, the cipher text is sent.

The key insertion component uses the extracted indexes value to decide the location to insert the key in the XORed table.

F. Algorithm Design

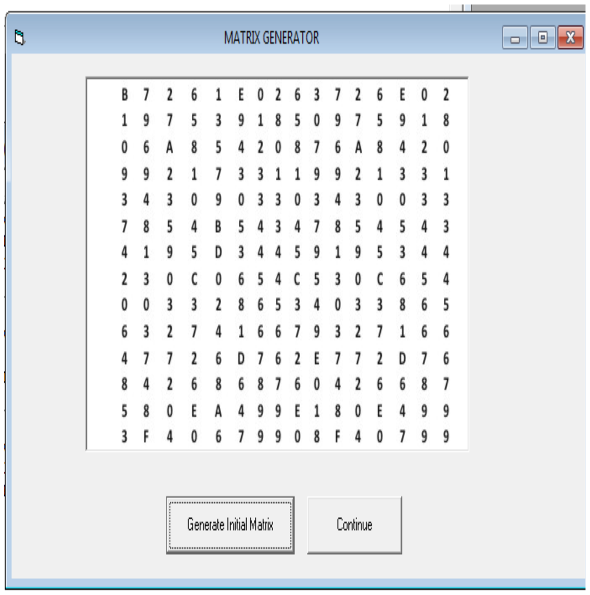
The proposed Algorithm for the sender side and receiver side of our system can be summarized as follows:

At the sender side:

1. Generate 16 * 16 matrix table using mathematical formula whose entries ranges from 0-F
2. Choose randomly key of *1024-bit* long as shared value.
3. Use the shared value to do column right and row down shifting
4. Generate index table and extract indexes from the new table formed
5. Encrypt message XoRed with key.
6. Insert key in the XoRed data using the extracted indexes in 4
7. Send the packet.

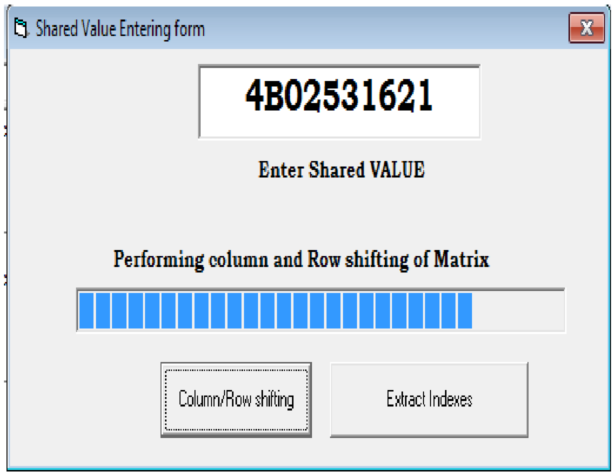
At Receiver side

1. Extract all keys contained in the packet, using the value in the shifted generated table
2. Decrypt the encrypted data by XoRing with the key



```
B 7 2 6 1 E 0 2 6 3 7 2 6 E 0 2
1 9 7 5 3 9 1 8 5 0 9 7 5 9 1 8
0 6 A 8 5 4 2 0 8 7 6 A 8 4 2 0
9 9 2 1 7 3 3 1 1 9 9 2 1 3 3 1
3 4 3 0 9 0 3 3 0 3 4 3 0 0 3 3
7 8 5 4 B 5 4 3 4 7 8 5 4 5 4 3
4 1 9 5 D 3 4 4 5 9 1 9 5 3 4 4
2 3 0 C 0 6 5 4 C 5 3 0 C 6 5 4
0 0 3 3 2 8 6 5 3 4 0 3 3 8 6 5
6 3 2 7 4 1 6 6 7 9 3 2 7 1 6 6
4 7 7 2 6 D 7 6 2 E 7 7 2 D 7 6
8 4 2 6 8 6 8 7 6 0 4 2 6 6 8 7
5 8 0 E A 4 9 9 E 1 8 0 E 4 9 9
3 F 4 0 6 7 9 9 0 8 F 4 0 7 9 9
```

Fig. 4.0(a). 16 x 16 Matrix for Index generation



4B02531621

Enter Shared VALUE

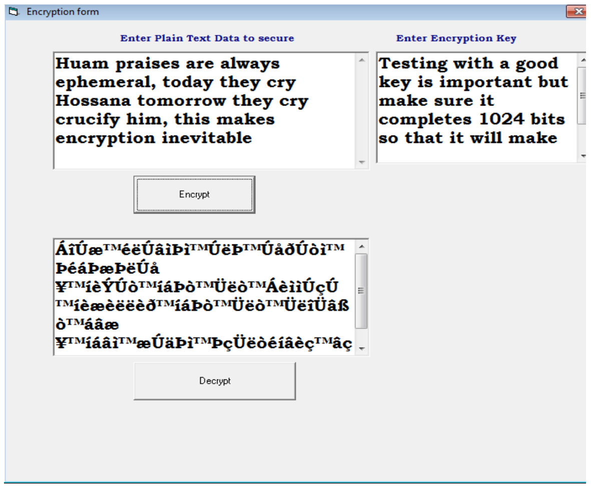
Performing column and Row shifting of Matrix

Fig 4.0 (b) Shared Value for Column Shifting

IV. Results

We tested our work with two different plain text data on 1024bits encryption key with ten (10) digits shared value on 16 x 16 matrix. The matrix and the shared value entered are shown in fig 4.0(a) and fig 4.0 (b)

The result of the encryption performed on the two different plain text data and 1024 bits encryption key is shown in fig 4.1 (a) and 4.1(b)

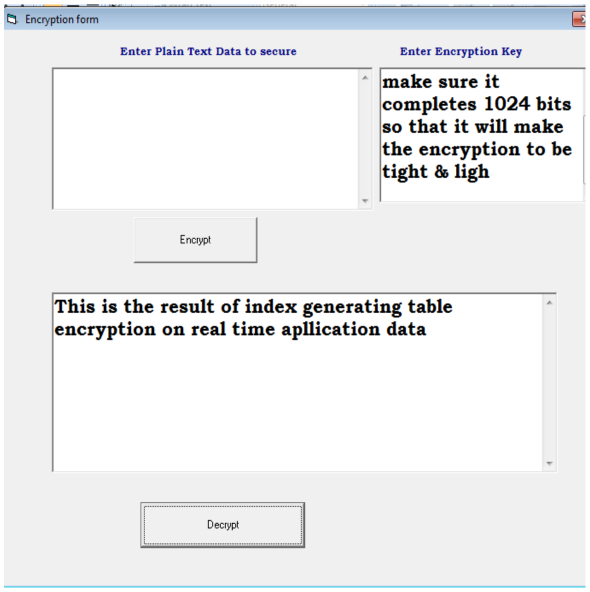


Huam praises are always ephemeral, today they cry Hossana tomorrow they cry crucify him, this makes encryption inevitable

Testing with a good key is important but make sure it completes 1024 bits so that it will make

ÁîÚæ™éëÚâìÞì™ÚëÞ™ÚåðÚòì™ÞéáÞæÞëÚå ¥™íèÝÚò™íáÞò™Üëò™ÁèììÚçÚ™íèæèëëèð™íáÞò™Üëò™ÜëîÜâß ò™áâæ ¥™íáâì™æÚäÞì™ÞçÜëòéíâèç™âç

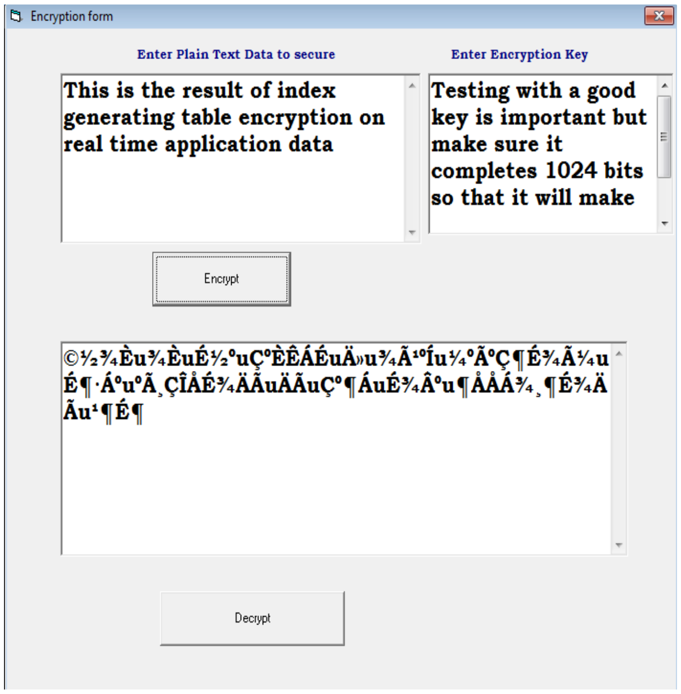
Fig 4.1 (a) Result of the encryption on first plain text data



make sure it completes 1024 bits so that it will make the encryption to be tight & ligh

This is the result of index generating table encryption on real time apllication data

Fig 4.2Result of decryption on second plain text data



This is the result of index generating table encryption on real time application data

Testing with a good key is important but make sure it completes 1024 bits so that it will make

©½¾Èu¾ÈuÉ½ºuÇºÈÊÁÉuÄ»u¾Ã¹ºÍu¼ºÃºÇ¶É¾Ã¼u É¶·ÁºuºÃ¸ÇÎÅÉ¾ÄÃuÄÃuÇº¶ÁuÉ¾Âºu¶ÅÅÁ¾¸¶É¾Ä Ãu¹¶É¶

Fig 4.1 (b) Result of the encryption on second plain text data

We compared our result to the previous work done by authors in terms of time complexity on different plain text data input sizes and also on how better secured the encryption of the plain text data is. Table 4.3 shows the comparative analysis table on the different encryption methods.

Table 4.1: Comparative analysis of encryption Time on different nput data sizes to different encryption methods

Input Size (bytes)	DES	AES	10 x 10 matrix	16 x 16 matrix Proposed approach
20,527	2	4	2	2
36,002	4	6	3	3
45,911	5	8	5	4
59,852	7	11	7	6
69,545	9	13	8	7
137,325	17	26	16	14
158,959	20	30	18	16
166,364	21	31	19	17
191,383	24	36	22	19
232,398	30	44	28	24
Average Time	14	21	13	11
Average Bytes/sec	**7,988**	**5,320**		**10,167**

A comparative analysis of our proposed system with AES-Rijndael as shown in Table 4.2. With the following information:
Data Size/block =1024-byte, and Encryption Key=256-byte.
From the table, it is noted that our proposed techniques achieves the best result, where it is 15 times faster than AES encryption and 6 times faster than AES decryption.
The proposed process is resistant against brute force attacks, where the key is mixed and shuffled strongly inside the XoRed data; it will be very difficult to guess the key.

Table 4. 2 Comparison between AES-Rijindael & Our proposed system

Security Algorithm	**AES-Rijindael**	**Our Proposed system**
Encryption(Ms)	10.884	0.71575
Decryption(Ms)	10.718	1.958125

A. Conclusion

Greater number of the existing encryption/decryption techniques is not perfect for RTA over the Internet since they were initially built for text data, and due to their extensive computations which result to imposition of certain constraint and delay in processing time.

Our Project attempts to develop a new encryption/decryption approach which adds a minimum delay time that makes it appropriate for RTA. In addition, it provides high level of security by choosing a key length of 1024-bit long, another interesting property of the proposed system is its ability of using a new different key for each packet.

The distribution of the encryption keys is usually carried out through a trusted agency; this results in a significant delay before the real-time application starts. Furthermore, this work attempts to provide a new method of key exchange without an intermediate party.

REFERENCES

[1] Atul, M. et al (2011). "*FPGA Implementation of AES Algorithm*", *International Conference on Electronics Computer Technology (ICECT)*, pp. 401-405.

[2] Bassil, C. et al (2005). "*Critical voice network security analysis and new approach for securing Voice over IP Communications*", SETIT 2005, *3rd International Conference: Sciences of Electronic, Technologies of information and Telecommunications,* Tunisia.

[3] Chalermwat, T. et al (2011). "*FPGA Implementation of FOE-Portable hard disk System*", *"The Int. Conf on Information and Communication Technology for Embedded Systems,* Pattaya, Thailand.

[4] Cole, E. et al (2005).*Network Security Bible,* Wiley Publishing Inc, 2005.

[5] Computer Security Objects Register (CSOR): http://csrc.nist.gov/csor/.

[6] Daemen, J. and Rijmen, V. (2009).*AES Proposal: Rijndael*, AES Algorithm Submission, available at http://www.nist.gov/CryptoToolkit

[7] Daemen, J. and Rijmen, V. (2010).*The block cipher Rijndael*, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

[8] Gladman's, B. (2012) AES related home page http://fp.gladman.plus.com/cryptography_technology/.

[9] Hoang, T. and Nguyen, V. (2012). An efficient FPGA implementation of the Advanced Encryption Standard Algorithm IEEE 978-1-4673-0309-5/12.

[10] Lee, A. (2009). NIST Special Publication 800-21, *Guideline for Implementing Cryptography in the Federal Government*, National Institute of Standards and Technology.

[11] Manvi goyal, Jatin Sharma, "Performance Analysis of mRSA for Varying Key Sizes and Data Modulus", SSRG International Journal of Mobile Computing & Application, Volume 2 Issue 2 May to Aug 2015

[12] Menezes, A. et al (2014). *Handbook of Applied Cryptography*, CRC Press, New York, Pp. 81-83.

[13] Nakahara, J. et al (2012).*Square Attack on Extended Rijndael Block Copher*, COSIC Technology Report.

[14] Nechvatal, J. et. al. (2013), Report on the Development of the Advanced Encryption Standard (AES).

[15] Kirti Sapra, Swati Kapoor "Modified Image Encryption Technique", SSRG International Journal of Electronics and Communication Engineering volume1 issue6 August 2014

[16] Omari, A.H. et al (2008). A New Cryptographic Algorithm for the Real-Time Applications, in Proceedings of the 7th International Conference on Information Security and Privacy - (ISP'08), Cairo, Egypt.

[17] Wang W., Chen J. and XU F. (2012). *An Implementation of AES Algorithm Based on FPGA*, IEEE 978-1-4673-0024-7/10.

[18] Yang, J. et al (2010). FPGA based design and implementation of reduced AES algorithm, IEEE 978-0-7695-3972-0/10.