International Journal of Scientific Research in Computer Science and Engineering
Research Paper Volume-3, Issue-4 ISSN: 2320-7639

Construction of Cryptographic e-Tags using Chronological Binary Transforms

Monday O. Eze[1*], Chekwube G. Nwankwo[2]

[1*]*Department of Computer Science, Federal University, Ndufu-Alike , Ikwo, Nigeria*
[2] *Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University, Nigeria*

Available online at www.isroset.org

Received: 07 Jul 2015 Revised: 18 Jul 2015 Accepted: 20 Aug 2015 Published: 30 Aug 201

***Abstract*—** A cryptographic tag generator is a computational algorithm designed and implemented to synthesize e-tags for system identification. Some of the digital resources that could be tagged are data files, digital images, video clips, and so on. As a best practice procedure, the tags are usually encrypted before insertion into the internal structure of the identified object, and also decrypted whenever there is a need to read the tags. This research presents a new algorithm for cryptographic tag generation using a series of chronological transformations. The dataset for the experimental run consists of 200 standard identification tags that could be applied for securing the patients' registration data in a typical hospital information system. The scope of this research covers the synthesis of the tags cum implementation of the cryptographic system, but not the insertion of the cipher tags into the internal structure of the host media. This research is important for a number of reasons. One is that it contributes to the emerging computational field of evolving hybrid technologies that combine steganographic and cryptographic techniques in system security.

***Keywords*—**Cryptographic Tags, Computational Algorithm, Cryptographic Array, Cryptanalysis, Steganography.

I. INTRODUCTION

The main aim of this research is to present a new computational methodology for generating electronic tags using cryptographic techniques. The cryptographic tag generating algorithm is based on a series of chronological transforms that convert binary datasets into cipher-text, and vice versa. At the fundamental level, the proposed algorithm can be viewed as composed of four major generic compartments A, B, C and D in Fig. 1. The compartments A and D are special data formats used as inputs into the encryption and decryption engines respectively. The other compartments B and C are procedures and reverse-procedures for encryption and decryption respectively.

It is necessary to introduce some important concepts which are related to the subject matter of this research. The term cryptography is defined as the art and science of design and implementation of ciphers [1]. As pointed out by the Advanced Learners' dictionary [2], a cipher is a secret or disguised way of writing. Cryptanalysis is defined as the science of breaking ciphers [3].

The term cryptology is the study of both cryptography and cryptanalysis [4]. A plaintext is the original input into a cryptographic system while a cipher text is the encrypted output [5]. Some of the basic building blocks in cryptography termed as cryptographic primitives are stream ciphers, hash functions and block ciphers [6]. When a block cipher has a singular key for both encryption and decryption, it is said to be symmetric [7]. On the other hand, if it uses a separate key for encryption and another for decryption, it is said to be asymmetric or public key [8].

Corresponding Author: *Monday O. Eze (PhD), eze_monday@yahoo.com*
Department of Computer Science, FUNAI, Ebonyi State, Nigeria

II. RELATED RESEARCH

A comprehensive treatment of the application of number-theoretic algorithms to the design and implementation of large prime number-based cryptographic schemes is [9]. In the recent time, a number of new cryptanalysis algorithms have evolved.

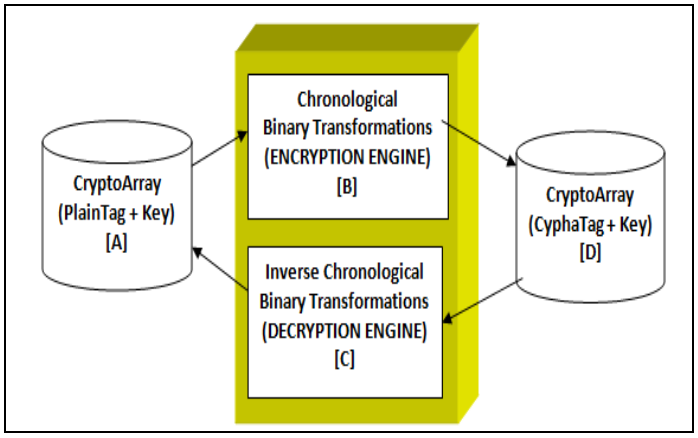

Fig. 1: System Dynamics

One of them is the higher order differential algorithm [10] used to launch an attack against a symmetric block cipher using brute-force key search.

A research [11] has applied the field of automated reasoning to cryptography as part of a larger field termed logical cryptanalysis. The work viewed cryptology from the angle of a satisfiability (SAT) problem with the ultimate aim of measuring the strength of the system in terms of its ability to satisfy a set of criteria. Another interesting area of research is the implementation of hardware-based cryptographic systems. A recent work by [12] reported that with the advent of Field Programmable Gate Arrays (FPGA's), it is now possible to build special purpose hardware for computationally intensive cryptographic applications. According to the authors, the Data Encryption Standard (DES) was used as a proof of concept [13].

Breakthroughs in multimedia cryptography are already common. A research in this category is [14], which implemented a new block cipher called the fast encryption algorithm (FEAL) on gray scale digital images. Application of cryptography to other multimedia such as video has also been reported by [15]. The field of artificial intelligence has also been applied in cryptography. Some recent works in this regard are in genetic algorithm [16], neural networks [17], and evolutionary algorithm [18], among others.

It is evident that active researches exist in Financial Cryptography. For instance, the use of cryptography as a security measure in inter-bank fund transfer was proposed by [19], while [20] focused on security protocols and evidences in e-payment systems. Other important applications of cryptography are in wireless communication [21] and court proceedings [22], among others.

III. RESEARCH METHODOLOGY

The generation of cryptographic tags as tackled in this work takes a number of steps. The aim of this section is to explain the proposed algorithm in more details. The workflow diagram for implementing this system is shown in Fig. 2.

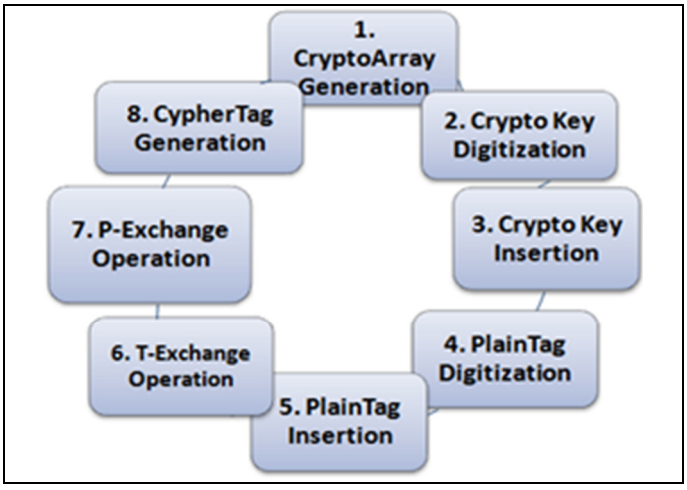

Fig. 2: Cryptographic Tag Generation Workflow Diagram

It is important to mention that the programming tool used in this research is MATLAB Version 7.5.0.342 (R2007b).

A. Abbreviations and Acronyms

The proposed cryptographic algorithm begins with a dynamic binary array [23]. A dynamic array is an elastic array. In other words, it takes variable dimensions rather than a static one even at run time. The choice of a dynamic array is to ensure that the system is able to accommodate any unexpected growth in the size of the input. This attribute is necessary, based on the fact that the tags to be encrypted / decrypted are of variable length and in turn require variable arrays for implementation. The dimension of the dynamic array is [N by 8] where N is the number of dynamic rows. The initial cryptographic array begins with an '*All-Ones*' binary array. This is a binary array with all its elements as '1'. An instance of an '*All-Ones'* binary array of dimension [7 by 8] is shown in Fig. 3.

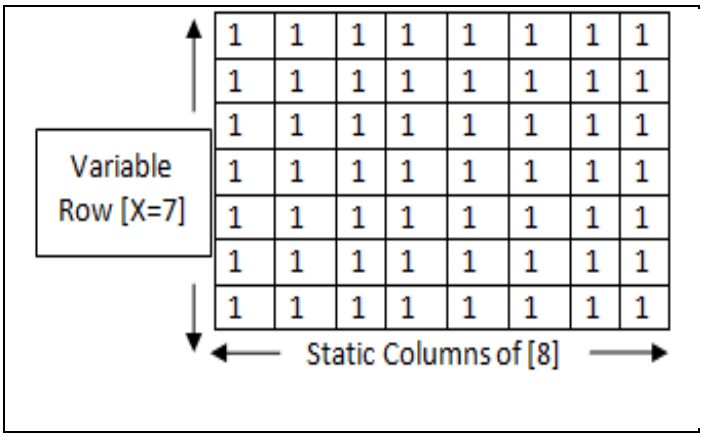

Fig. 3: Sample Cryptographic All-One Array

B. The CryptoKey Digitization

The concept of digitization in this work refers to the conversion of a chunk of data into a digital format. Since the proposed cryptographic system uses binary data, it implies that digitization is synonymous to the conversion of a given data into binary format. The process of digitization follows a set of standard rules. The two levels of digitization applied here are *cryptokey* digitization and *plain tag* (plain text tag) digitization. The former will be discussed in this section, while the later will be examined in another section.

The input to the digitization algorithm is an alphanumeric *cryptokey* (short for cryptographic key). The standard *cryptokey* design format is shown in Table I. Also related to the said format is a special semantic rule. One of the rules is that the first two numeric characters of the *cryptokey* specify the actual locations of the crypto array where *cryptokey* insertion should take place.

Table I: Crypto Key format

	Key Format			
Code	Int1	Int2	Alph1	Alph2
Interpretation	1st Integer	2nd Integer	1st Alphabet	2nd Alphabet
Example	3	4	B	C

The process of *cryptokey* digitization involves the conversion of the last two characters of the *cryptokey* into binary format using a standard coding format. The ASCII code [24] was used to implement the key digitization in this research. For instance, the character 'B' is first converted to ASCII code '66', and then to binary format as '1000010'. Since this is less than the standard row size of 8 characters used for every cryptographic binary array in this research, this system pads requisite number of preceding zeros, giving rise to '01000010'. The digitization operation is done for the next character in the *cryptokey* format, and these results are preserved for insertion into the '*All-One'* cryptographic array earlier described.

C. CryptoKey Insertion

The *cryptokey* insertion takes as input the digitized format of the cryptographic key, which is slotted into the '*All-One'* array to form the *ECR array*. The term *ECR* is an acronym for encryption ready. The *ECR array* is an 'All-One' crypto array which has received all the necessary insertions, and is thus ready to be passed as input to the encryption engine. The *ECR array* formulation will be explained in further details at the plain tag digitization section of this work. Given a cryptographic key '34BC', the algorithmic system interprets this key as made up of two characters 'B' and 'C' which are passed through the key digitization process.

The result of this process is an 8-bits character block for the first character, and another 8-bits character block for the second character. The first two numeric values in the pre-digitized key act as directives to the system on how it should perform the post-digitization insertion. For instance given the pre-digitized key '34BC', system will insert the first digitized block in row '3' and the second one in row '4' of the '*All-One*' array in order to form the *ECR array*.

D. PlainTag Digitization

The input to the plain tag digitization algorithm is an alphanumeric plain tag. The plain tag is an N- block alpha numeric character that is usually combined with the *cryptokey*, to synthesis the *ECR* version of the cryptographic array. The transformation rule is shown in equation (1).

$$\{DC.Key\} + \{DP.Tag\} + \{AllOne.Array\} \rightarrow \{ECR.Array\} \quad (1)$$

where DC.Key = digitized cryptographic key,

DP.Tag=digitized plain tag,

ECR.Array = encryption ready cryptographic array.

The experimental dataset for this system consists of data modeled after the patients' registration information of a private hospital. In order to preserve the identity and confidentiality of information, the actual names of patients were not used in implementation. The choice of a hospital-related dataset is based on the fact that future research will aim at using the proposed e-tag algorithm to build cryptographic protection into the digital images of patients in a typical hospital database. Given a sample patients' registration record in Table II, where the fields *PName, PSex, PYear, PSerialNum* represent the patient's name, sex, year of registration, and serial number respectively.

Table II: Sample Patient Registration Record

PName	PSex	PYear	PSerialNum
DAN GERALD	M	2014	211

The registration numbers *RegNum* and the plain tag *PlainTag* are formed using concatenation operation *CONCAT* shown in equations (2) and (3) respectively.

$$RegNum = CONCAT(PSex, PYear, PSerialNum) \quad (2)$$

$$PlainTag = CONCAT(PName, RegNum) \quad (3)$$

Thus, the patient registration number formed for a patient called 'DAN GERALD' is 'M/2014/211', while the resulting plain tag is 'DAN GERALD/M/2014/211'. Having explained the synthesis of plain tags, it is important to mention that the essence of plain tag digitization is to convert the resulting plain tag into binary form, and to incorporate it into the *ECR* cryptographic array. This process will be explained at this point.

The system picks a block of N-character plain tag, where N is a positive integer, thus forming an N-sized column vector with the elements as the plain tags, arranged serially in a FIFO (first come first served) basis. Each row of the column vector is converted into a binary value via a standard digital coding format such as ASCII code. The resulting binary value is then examined to ensure that it is up to 8 bits, otherwise, it is padded with leading zeros to make up this requirement. The transformation process for a plain tag 'DAN GERALD/M/2014/211' is summarized in Fig. 4.

The resulting binary values are used for insertion into the ECR array to be used for cryptographic operations.

E. PlainTag Insertion

The plain tag insertion algorithm takes as input the digitized plain tags on a row by row basis, and inserts them into the '*All-One'* array to form the *ECR array*. The entire algorithm takes necessary precautions to ensure that there is no conflict arising from the digitized tag and digitized cryptokey insertions.

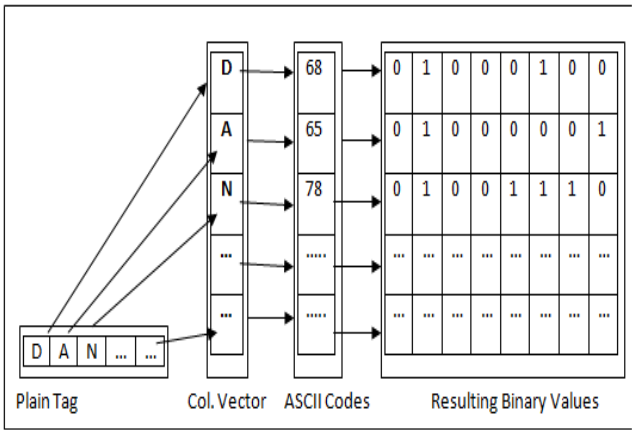

Fig. 4: A Typical Plain Tag Transformation

In other words, the digitized tag insertion should not be mistakenly done in the rows designated for the crypto key insertion and vice versa. One way to achieve this is to ensure that digitized tag insertion takes place first, and that during this process, the rows allocated for key insertions are marked with special codes '99'. This will be further explained in the algorithmic development section.

F. The T-Exchange Operation

The *T-exchange* involves the exchange of the content of the binary terminals. This is illustrated in Fig. 5. The term terminal refers to the column vectors that make up the two extreme ends of the cryptographic array. The two vectors are referred to as the *FR-terminal* and the *FL-terminal* respectively. The acronyms *FR* and *FL* stand for 'far right' and 'far left' terminals respectively.

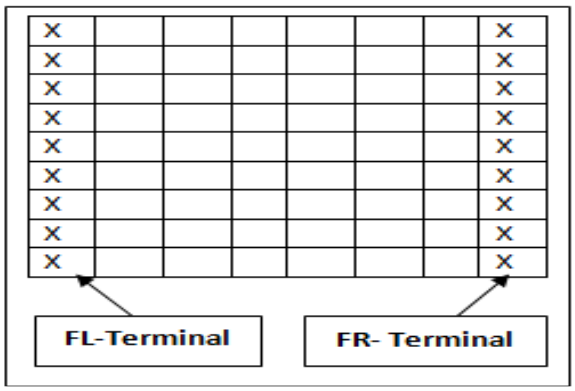

Fig.5: The FR and FL Terminals

The *T-exchange* operation is one which switches the contents of *FR-Terminal* vector with the corresponding contents of the *FL-Terminal* vector. In order to achieve this swap operation, an intermediary vector *InterX* of same size as both *FR* and *FL* terminals provide the requisite redundancy. The mathematical transfer operation is summarized in equations (4).

$$\left.\begin{array}{l} InterX \leftarrow \text{FLTerminal} \\ \text{FLTerminal} \leftarrow \text{FRTerminal} \\ \text{FRTerminal} \leftarrow InterX \end{array}\right\} \quad (4)$$

G. The P-Exchange Operation

A cryptographic array can be demarcated into four quadrants or cardinal regions in line with the cardinal points shown in Fig. 6. These are the NE (North East), SE (South East), SW (South West), and NW (North West) regions respectively.

As shown in the diagram, there are two lines that pass through any two diagonally adjoining regions. The NE-SW and NW-SE diagonal lines are referred to as poles. The pole exchange (*p-exchange*) algorithm swaps the contents of the SW region with that of the NE region. The *p-exchange* is a block swap, while the *t-exchange* is a vector swap operation. Since the NW and SE blocks are not swapped, the NW-SE line is termed a *stationary pole*. On the other hands, since the NE-SW blocks undergo swapping, the NE-SW line is called a *transition pole*.

H. The Cypher Tag Generation Operation

This algorithmic operation generates the *e-tag* in an encrypted format. This is achieved by transforming each of the rows of the final version of the cryptographic array from binary to alphanumeric values. The resulting alphanumeric column vector is then transformed into a single row message block, which is the final cipher tag that could be inserted into a host media for system resource identification purposes.

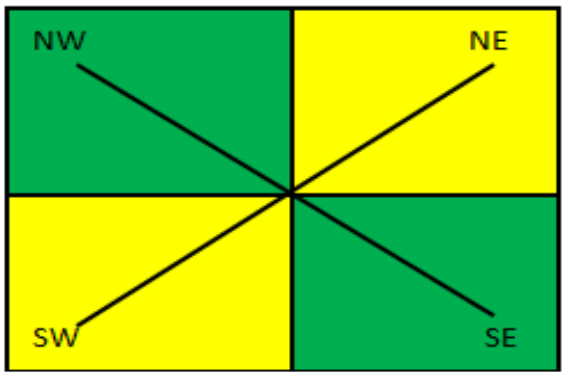

Fig. 6: Illustration of the P-Exchange Regions

IV. SYSTEM MODULES/IMPLEMENTATION

The essence of this section is to present the algorithmic modules in more details. The system is broken into seven

major modules. These are *hibGenCryptoArray.m, hibDigitizeKey.m, hibInsertKey.m, hibDigitizePtext.m, hibTerminalExchange.m, hibPoleExchange.m* and *hibCryptoResult.m*. Each of the program modules have a "dot.m" file extension since system implementation took place in MATLAB.

A. The hibGenCryptoArray.m Module

The major function of this program module is to generate an initial '*All-One'* cryptographic array. As earlier described, this is a binary array made of 1's in its entire content. This module also accepts as input the plain tag as well as pre-digitized cryptographic key. Since the cryptographic array is dynamic and can grow in size, it is important that system should have a standard rule for determining the array size to be used in generating the cryptographic array.

The following special calculation is used to determine the dynamic array dimension. System first scans the input to determine *psize*, which is the number of characters in the plain tag. It also scans through the key to determine the value of *ksize*, which is the number of characters for the actual digitized key insertion. Since the cryptographic array has a constant column of value 8, it follows that the dimension is given by equation (5).

$$[Row, col] = [psize + ksize, 8]\ Array\ of\ Ones \qquad (5)$$

Thus, given a plain tag *Ptag* = 'YARO,D#BSc/10/0801' and a pre-digitized key *Pkey*='34BC', then *psize* = 18 and *ksize*=2. Thus, the dimension of the cryptographic array will be [20 by 8].

B. The hibDigitizeKey.m Module

This program module digitizes the encryption key, and makes it ready for insertion into the cryptographic array. The major program variables in this module are *xbin* and *ybin*. They keep the results of direct conversion of the first and second cryptographic key characters from alphanumeric to binary format respectively. The two other program variables *zpad1* and *zpad2* store the number of leading zeroes used to pad *xbin* and *ybin* in case their sizes are less than 8 bits.

The final outputs of this module are thus *binkey1* and *binkey2* which are 8-bits blocks each. They are ultimately used as building blocks for the *ECR* cryptographic array. The program code that performs the digitization operation for any generic *cryptokey* character *kChar* is listed in Fig. 7.

```
xbin=dec2bin(str2num(int2str('kChar')))
zpad1=num2str(zeros(8-max(size(xbin))))
zpad1= strrep(zpad1,' ', '');
binkey1= [zpad1, xbin]
```

Fig. 7: Program Segment for Single-Char Key Digitization

C. The hibInsertKey.m Module

This program module performs cryptographic insertion process. The digitized key insertion takes place at two designated positions in line with the standard key structure. Thus the digitized keys stored by the variables *binkey1* and *binkey2* are inserted into the cryptographic array rows. Fig. 8 shows the code segment for cryptokey insertion into rows 3 and 4 of the cryptographic array.

```
for k = 1:8
    CryptoArray (3,k)= str2num(binkey1(k))
end
for k = 1:8
    CryptoArray (4,k)= str2num(binkey2(k))
end
```

Fig. 8: CryptoKey Insertion Code Segment

D. The hibDigitInsertPtext.m Module

This program module performs a combined function of both plain tag digitization and insertion. The major program parameter is called *PlainTag*. The system digitizes each of the plain tag characters into binary formats, and stores each of the resulting 8-bits binary block into a program variable called *bintx*. The actual insertion takes place into an '*All-One*' cryptographic array of dimension [x +2 by 8], where x is the size of the plaint tag. As a precautionary measure against errors, system usually inserts a special code '99' into the rows designated for key insertion as shown in the code segment in Fig. 9.

```
% Now exclude the positions of the two keys
  if (k==3)
        for vv=1:8
           % Populate Crypto Key Row with '99'!
            CryptoArray(k,vv) = str2num ('99')
        end
      y=k+1
      if (k==4) || (y==4)
        for vv=1:8
           % Populate Crypto Key Row with '99'
           CryptoArray(y,vv) = str2num ('99')
        E
nd
        y=y+1

      end

   end
```

Fig. 9: Special Code 99 Insertion into CryptoKey Rows

E. The hibTerminalExchange.m Module

As earlier illustrated in Fig. 5, the main essence of this program module is to exchange the contents of the *far-right* and the *far-left* terminals of the cryptographic array. The code segment for the terminal operation is shown in Fig. 10.


```
% Module Name:   hibTerminalExchange.m
% Module Input:   CryptoArray
farlf=CryptoArray(:,1)
farrt=CryptoArray(:,8)
CryptoArray(:,1)=farrt
CryptoArray(:,8)=farlf
```

Fig. 10: Terminal Exchange Code Segment

F. The hibPoleExchange.m Module

This module performs *p-exchange* operation earlier explained. The actual source code is listed in Appendix A. Fig. 11 shows the two arrays used for system testing. The first array is populated in the 'NE' and 'SE' regions with numeric values '9's and '7's respectively. This was used for the experimental testing of the pole exchange operation. The second one is the expected outcome that should result if the pole exchange operation was executed perfectly well. Apart from these two test-data arrays, the pole exchange algorithm was also carefully tested with other varied sizes of arrays. The main aim is to confirm that the module ran as expected.

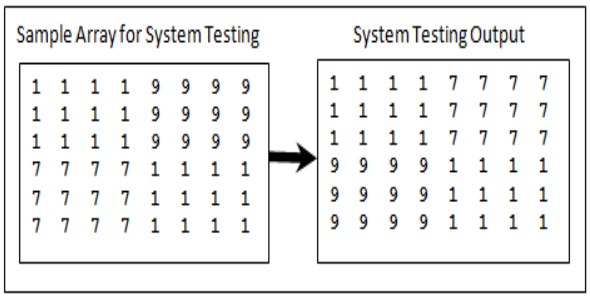

Fig. 11: Sample Data for Pole Exchange Testing

G. The hibCryptoResult.m Module

This program module is responsible for the final output of the cryptographic operation. A sample screen capture of the program output at run time is shown in Fig. 12. The picture demonstrates the encryption of a plain tag "HARUNA MOHAM # M/1999/076" and the resulting cipher tag BÊBÌZÛKË+ÊÃKÃÌ,.Îà. The cryptographic key used for encryption is shown as '34BC' and the size of the pre-encryption tag as 27.

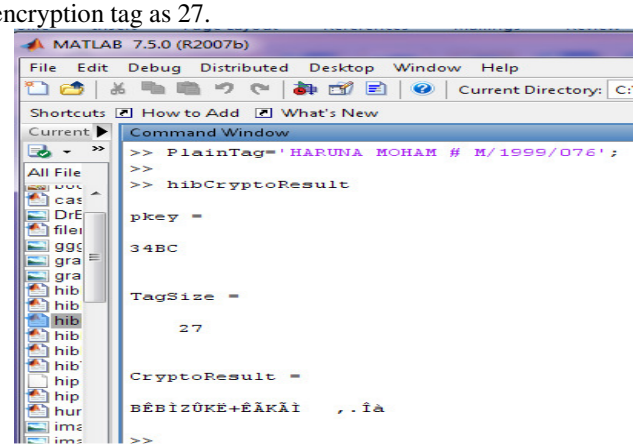

Fig. 12: Screen Capture During Program Execution

V. EXPERIMENTAL RUNS/ OUTPUT

The cryptographic runs on the dataset were successful. Table III shows the encryption output for the first 15 cryptographic tags in the experimental dataset. The left hand side of the table shows the original (plain text) tags, while the right hand side gives the listing of the corresponding encrypted versions.

Table III: Result of Experimental Runs for the first 15 tags.

PLAINTAGS	CIPGER TAGS
CHINARA EWAGU# M/1994/100	ÍJBÌÊKË[Ã*ËÓÃÆ$,(@
OLUMIDE YEMI# F/2000/299	ÌJBÄÚÃÃCÃ*ÓËËÈÂ &NÂ@@HÈ
YAHAYA GULAK # M/1990/317	ÒÊBÌJËÛË#ÊÛKËÊ,NBÀ
LUCAS JAJA # M/2005/116	JÒLÊÃÃÓ+JËKÃ Â@,." ® À
ROSLINR TAMBO # F/1996/153	\ÂJÂÔJËK[#ZËËK.À" &ÎÈ(N Ä"
TUTURA YAWEH # M/1992/917	RÚBÌZÛ[Ë#ÚËÛËH@B ,N@`F"
ONYINYE DIMGBA # F/2008/114	ÌBJÂÔÊCÓÃ#JËËÃBÀÀ" ÂNÀD
FINEFACE ONUKA # M/2015/954	LÂJÂLÊCÃËË*ËKÓÊ`" ÌNB DÈÄD
ODOGWU IJERE # M/2014/805	ÂJBÌÊÃÓÛ#ÊCÃ[ÄÀB ,ÎB`@¨@$ÄD
MEREDI TAKORA # M/2015/955	ÌÂJÂ\ÊCÃ+[ÊËË[À@" ,NBD¨Ä$D
ZUBITA KANGARO # M/2012/567	\ÒJÂLÊSÃ+ÂÊKÃËR®@" ,B ¢ÄF
BLESSING GUYIP # F/2001/451	EBJÂÄÚÓÃCË*ÃÛÛÈ À" F."À`DD
FELIX FUJISHI # M/1999/077	LÂJÂLÊ[++KÚCËÛHH@" ÌHN &
MARIA PUPILA # F/1999/099	ÂÊBÄZËË+[ÚSËKÀÀ N È&
HARUNA MOHAM # M/1999/076	BÊBÌZÛKË+ÊÃKÃÌ,.ÎÀ&

VI. CONCLUSION/ FUTURE RESEARCH

This research is an attempt to implement an algorithm that constructs cryptographic tags using a series of chronological binary operations. The relevance of this research is that the algorithm can be applied to secure database objects of diverse kinds. This work used a dataset of hospital registration numbers. Thus it can be used to tag digital images, digital signatures, video files, and other forms of multimedia files. It is no doubt a scientific contribution to the emerging researches on building hybrid technologies that combine steganographic and cryptographic techniques for system resource protection [25]. A number of areas for future research have been identified. One of such areas is the performance of cryptographic strength analysis to determine the system's level of resistance to cryptanalysis.

REFERENCES

[1] O.C. Abikoye, A.G.Akintola, and S. Ibrahim, "Design and Implementation of a Yoruba Language Cryptosystem", Ilorin Journal of Sc. Vol 1, No. 1, **2014, pp50-60**

[2] www.oxforddictionaries.com, Accessed **Jul. 10, 2015**.

[3] G. Durfee, "Cryptanalysis of RSA Using Algebraic and Lattice Methods", A PhD Dissertation submitted to the Dept. of Computer Science Stanford University, **June 2002**..

[4] M. Ahmed, T. M. Shahriar Sazzad and M. E. Mollah, "Cryptography and State-of-the-art Techniques ", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, **March 2012, pp583-586**

[5] E. Mansoor, S. Khan, and U.B. Khalid. "Symmetric Algorithm Survey: A Comparative Analysis", International Journal of Computer Applications, Vol. 61, No.20, **Jan. 2013 pp12-19**

[6] K.T. Huang, Y.N. Lin, and J.H. Chiu. Real-time Mode Hopping of Block Cipher Algorithms for Mobile Streaming. International Journal of Wireless & Mobile Networks (IJWMN) Vol. 5, No. 2, **April 2013, pp127-142**

[7] R. Masram, V. Shahare, J. Abraham and R. Moona. Analysis and Comparison of Symmetric Key Cryptographic Algorithms Based on Various File Features, International Journal of Network Security & Its Applications (IJNSA), Vol.6, No.4, **July 2014 , pp43-51**

[8] P.T. Kenekayoro. One way functions and public key cryptography, African Journal of Mathematics and Computer Science Research, Vol. 4, No 6, **June 2011, pp 213-216**.

[9] T. Cormen, C. Leiserson, R. Rivest and C. Stein. Introduction to Algorithms (3rd Ed), The MIT Press, Massachusetts**, 2009**

[10] T. Hidema and K. Toshinobu. "An Expansion Algorithm for Higher Order Differential Cryptanalysis of Secret Key Ciphers ", Journal of the National Institute of Information and Communications Technology Vol.52 Nos.1/2, **2005, pp119-128**

[11] F. Massacci and L. Marraro,"Logical Cryptanalysis as a SAT Problem: Encoding and Analysis of the U.S. Data Encryption Standard",Journal of Automated Reasoning Vol 24, Issue 1-2, **2000, pp165–203**.

[12] H. Zodpe, P. Wani and R. Mehta. "Hardware Implementation of Algorithm for Cryptanalysis", International Journal on Cryptography and Information Security (IJCIS), Vol.3, No.1, **March 2013, pp7-15**.

[13] K.R. Shah and B. Gambhava, "New Approach of Data Encryption Standard Algorithm", International Journal of Soft Computing and Engineering (IJSCE) Vol. 2, Issue-1, **March 2012, pp322-325.**

[14] N. Nithin, A. Bongale and G. P. Hegde."Image Encryption based on FEAL Algorithm", International Journal of Advances in Computer Science and Technology, Vol. 2, No.3, **March 2013, pp14-20.**

[15] K.Mohan, S.E.Neelakandan. "Secured Robust Video Data Hiding Using Symmetric Encryption Algorithms", International Journal of Innovative Research in Engineering & Science, Vol 6, Issue 1, **Dec 2012, pp17-29.**

[16] P. Garg, S. Varshney, and M. Bhardwaj. "Cryptanalysis of Simplified Data Encryption Standard Using Genetic Algorithm", American Journal of Networks and Communications. Vol. 4, No. 3, **2015, pp. 32-36**.

[17] K. Alallayah, M. Amin, W. Abdelwahed and A. Alaa. "Applying Neural Network for Simplified Data Encryption Standard (SDES) Cipher System Cryptanalysis". The International Arab Journal of Infor. Tech. Vol. 9, No.2, **March 2012, pp163-169**.

[18] G. Poonam, "Cryptanalysis of SDES via Evolutionary Computation Techniques", International Journal of Computer Science and Information Security, Vol. 1, No. 1, **May 2009, pp1-7**

[19] D. Zhu, "Security Control in Inter-Bank Fund Transfer", Journal of Electronic Commerce Research, VOL. 3, NO. 1, **2002, pp15-22**.

[20] S. Murdoch and R. Anderson."Security Protocols and Evidence: Where Many Payment Systems Fail", In Proceeding of Financial Cryptography and Data Security, Barbados, **March 3-7, 2014, pp1-12**

[21] B. Joshi and A. Khandelwal., "Clustering with Data Encryption in Wireless Communication: A Critical Survey", International Journal of Advanced Computer Research, Vol. 4 Number-3 Issue-16, **Sep.2014, pp813-818**

[22] J. Vagle, "Furtive Encryption: Power, Trust, and the Constitutional Cost of Collective Surveillance", Indiana Law Journal, Vol. 90:101, **2015, pp101-150**.

[23] J. Voldman, M. L. Gray, M. Toner, and M. A. Schmidt., "A Microfabrication-Based Dynamic Array Cytometer", Analytical Chem., 2002, 74 (16), **July 10, 2002, pp 3984–3990.**

[24] A. Singh and U. Jauhari, "Data Security by Preprocessing the Text with Secret Hiding", Advanced Computing: An International Journal (ACIJ), Vol.3, No.3, **May 2012, pp 63-74**.

[25] K. I. Rahmani, A. Arora and P. Naina. "A Crypto-Steganography: A Survey", International Journal of Advanced Computer Science and Applications(IJACSA), Vol. 5, No. 7, **2014, pp149-155**.

APPENDIX A: Source Code for Pole Exchange Module

```
%Module Name:    hibPoleExchange.m
tmp=CryptoArray; %Backup Original Plain Text Array
%aa(1,5)=tmp(4,1),  aa(1,6)=tmp(4,2),  aa(1,7)=tmp(4,3),
aa(1,8)=tmp(4,4)
[row,col]= size(CryptoArray);
hafro=fix(row/2)  %Keep record of Half the Row Dimension of the
CryptoArray.
for k=5:8
   for w=1:4
      for dd=1:fix(row/2)
         if (k-w)==4
            %Check for Odd dimension
            if (row - 2*fix(row/2)) == 1
              CryptoArray(dd,k)=tmp(dd+hafro+1,w); %was 4
            else
               %For Even Row dimension
               CryptoArray(dd,k)=tmp(dd+hafro,w);
            end
         end
      end
   end
end
for k=1:4
   for w=5:8
```

```
    for dd=1:fix(row/2)
      if (w-k)==4
        %Check for Odd dimension
        if (row - 2*fix(row/2)) == 1
          CryptoArray(dd+hafro+1,k)=tmp(dd,w); %was 4
        else
        %For Even Row dimension
          CryptoArray(dd+hafro,k)=tmp(dd,w); %was 3
        end
              end
    end
  end
end
CryptoArray  % Print Out
```

Authors Profile

Dr. Monday O. Eze is a Lecturer and Academic Adviser in the Department of Computer Science, Federal University, Ndufu-Alike, Ikwo (FUNAI), Ebonyi State, Nigeria. He has a Bachelors Degree (BSc) in Computer Science, MBA in Management, MSc in Computer Science, and a PhD in Computer Science. He is a full member of the Nigerian Computer Society (NCS) and the International Institute of Informatics and Systemic (IIIS), Florida USA. His researchinterest includes Algorithms, Computational Modeling, etc. Corr. Author E-mail: eze_monday@yahoo.com.

Chikwube G. Nwankwo is currently a lecturer in the Computer Science Departm-ent of Chukwuemeka Odumegwu Ojukwu University, Nigeria. She has a Higher National Diploma (HND) in Computer Science from Federal Pol
ytechnic Oko. She also has a Post Graduate Diploma (PGD) and MSc in Computer Science from the University of Benin,Nigeria. Her research interests are Software Testing and Software Engineering.

