An Approach for Ensuring Security and its Verification

Suparna Karmakar
Department of Information Technology
Guru Nanak Institute of Technology
Sodepur, Kolkata, West Bengal, India
E-mail - karmakarsuparna27@gmail.com

Sayani Chandra
Department of Computer Science & Engineering
Guru Nanak Institute of Technology
Sodepur, Kolkata, West Bengal, India
E-mail - sayani49@gmail.com

Abstract- In this paper a new symmetric key cryptographic algorithm has been designed. The algorithm is enriched with various calculations through which the ciphertext becomes unpredictable and breaking the security by eavesdropper becomes harder. In terms of properties of Petri net and the analysis technique of reachability tree, the protocol was proved to be boundedness, deadlock free and liveness.

Keywords- Cryptography, Symmetric key, Petri Net, Reachability Tree, Compression

I. INTRODUCTION

The need to conceal messages has been with us since we moved out of caves, started living in groups and decided to take this civilization idea seriously. The earliest forms of cryptography were found in the cradle of civilization, which comes as no surprise, including the regions currently encompassed by Egypt, Greece and Rome. The art and science of cryptography showed no major changes or advancements until the Middle Ages. As the Internet and other forms of electronic communication become more prevalent, electronic security is becoming increasingly important. So, cryptography is used to protect e-mail messages, credit card information, and corporate data. Cryptography systems can be broadly classified into symmetric-key systems as described in [9], [10], [12] that use a single key that both the sender and recipient have, and public-key systems as described in [9], [10], [12] that use two keys, a public key known to everyone and a private key that only the recipient of messages uses.

In this paper a symmetric key cryptographic algorithm has been designed which ensures better security compared to any other existing encryption algorithms for the following reasons:-

- The prime position shifting technique applied for encryption process is unique and the change in the bit positions is hard to be guessed by the eavesdroppers. Hence it is almost impossible for them to decode the cipher after encryption.
- After applying the expansion method the number of characters in the plain text gets doubled after encryption. Hence the number of characters in the cipher is twice the number of characters in the plain text. Therefore seeing a cipher it is hard to understand the actual number of characters in the plain text for an eavesdropper.
- DES as described in [9], [10], [12] and AES algorithms as described in [9], [10], [12] are very complicated and hard to implement whereas this algorithm includes some basic simple steps like XOR operation, 1's COMPLEMENT operation, 2's COMPLEMENT operation and the unique PRIME POSITION SHIFTING and EXPANSION operations which are very easy to implement. Still this algorithm ensures security.

Using properties of Petri net and mathematical theory relating to it, the algorithm was analyzed and verified to be the correctness. And the algorithm was proved to be boundedness, deadlock free and liveness.

II. BASIC CONCEPT

A. PETRI NET

Petri nets were developed in the early 1960s by Carl Adam Petri [1]. Later on, it was widely generalized and investigated for further improvement in modeling. From [5], [6], [7], [8] it can be found that how achievements in modeling method of Petri Net has been used in various fields such as computer science, automation and computer integration manufacture.. The formalized definition can be found in [2], [3]. Petri nets are a graphical and mathematical modeling tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, parallel, nondeterministic and/or stochastic. As a graphic instrument, it owns a graphically depicting function and simulates the dynamic behaviors of the system through

the flow of tokens. And as a mathematical tool, it depicts system behaviors through building state equation (analysis of dynamic behavior can be found in [1], [4]). Petri nets are defined by a quadruple (S, T, F, W), where

1. S is a finite set of places, represented by circle or ovals.
2. T is a finite set of transitions, represented by squares or rectangles.
3. F $\subseteq$ {S x T} U {T x S} is the flow relation. Arcs connect places and transitions. Inward arcs go from a place to a transition and an outward arc goes from a transition to a place.
4. W: F→N-{0} is the weight function, which associates a nonzero natural value to each element of F. If no weight value is explicitly associated with a flow element, the default value 1 is assumed for the function.

Tokens reside in the places. They are represented by a solid circle or by a dot inside of a place. The execution of Petri net is controlled by the position and movement of tokens. A Petri net is a kind of directed graph with initial marking M0. A marking (state) assigns to each place a positive integer. If a marking assigns a non-negative integer k to a place s, then s is said to be marked with k tokens. A marking is denoted by M, a m-vector. The sth component is denoted by M(s), which represents the number of tokens at place s [1].

B. TRANSITION ENABLING RULE

A transition t is said to be enabled if each input place s of t is marked with at least w(s, t) tokens, where w(s, t) is the weight of the arc from s to t. For a given marking the firing of an enabled transition results in a successor marking M',

Where,

$$M(s) = M(s) – w(s, t) + w(t, s)$$

w(s, t) is the weight of the arc between s and t. w(t, s) is the weight of the arc between t and s.

III. PROPOSED ALGORITHM

A. ENCRYPTION ALGORITHM

- Input a filename where the text to be encrypted i.e. plaint text would be written.
- Convert each character of the plain text to its equivalent binary values from its corresponding ASCII value and store the bits in a n × 8 table, where n is the number of characters and 8 corresponds to the 8 bits to which each character will be converted.
- Input a password from the user via console. Each of the characters of the password is converted to its corresponding binary values and stored in another matrix of the same size as that of the plain text matrix.

  Let the number of characters of the plain text be a and that of the password be b.

  If a=b then

  length of password matrix is same as that of the plain text matrix.

  Else

  If a<b then

  Make the length of password matrix is same as that of the plain text matrix by ignoring the (a-b) characters of the password.

  Else

  The first a-b characters of the password matrix is repeated to make the password matrix of length a.

- **Prime position shifting**

For each character in each row the following is to be done:

The prime positions are 2nd, 3rd, 5th and 7th position among the 8 positions.

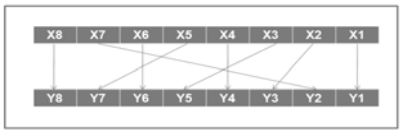
The bit at the 2nd position is shifted to the 3nd position.

The bit at the 3th position is shifted to the 5th position.

The bit at the 5th position is shifted to the 7th position.

The bit at the 7th position is shifted to the 2nd position.

Rest of the bits remains in the same position as shown in the figure below:



- **2's Complement**

For each row of the binary bit table the following is applicable:

First one's complement of each row is found out by xor –ing with a 1-bit row.

Then 1 is added to find out the corresponding two's complement as shown in the figure below:

- **Reverse**

All the bits of each row are reversed as in the diagram below:

- **One's complement**

The resultant of the above process is xor-ed with a table of the same size as that of the original table containing all 1.

10101100
^ 11111111
01010011

- **X-OR with user input password**

The resultant bit table is xor-ed with the password table to generate a new table.

- **Expansion**

For each row of the resultant table following is done:

Make groups of 4 bits each (range of character value is 0 to 15)

Convert them to their equivalent decimal value

Add 32 to each of the value (After adding 32 the range of character value will be 32 to 47, which are all special characters).

Convert it to its ASCII equivalent (each 4 bit character is represented as an eight bit character after adding 32 thus the total number of characters doubles).

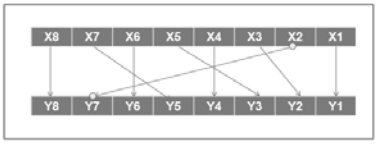
- Display the corresponding characters for each ASCII value and write it to another file for the purpose of decryption.

B. DECRYPTION ALGORITHM

- Input the file name containing the CIPHER text.
- Request the user for a password to decrypt the content of the file (the password should be same as the one used for decryption as the algorithm is a symmetric-key algorithm). The length of the password should half the length of the characters read from the file.
- Read the characters from the file and store them in an array temporarily (Remember the number of characters in the cipher text is double the number of characters in the plain text).
- **Compression**
  Convert the characters to their ASCII equivalent (range 32 to 47)
  Subtract 32 from each value (range 0 to 15)
  Convert each to its binary equivalent each of which will be 4 bits.
  Make groups of 8 bits from every subsequent two 4 bits and store them in a table of length $n \times 8$, where n = number of characters read from the file divided by 2 (This is because the original plain text had n characters and after the expansion at the last step the number of characters doubled).

(The following operations are applied to the table thus obtained after compression. Each of the operations is same as that shown in encryption except prime position shifting which happens in the reverse order).

- **X-OR with user input password**
- **One's complement**
- **Reverse**
- **2's Complement**
- **Prime position shifting** (happens in the reverse order as shown in the figure below)



- For each row the corresponding character is found out by converting the bits to its equivalent ASCII values, thereby getting one character for each row.
- Following characters are written to another file.

IV. PETRI NET REALIZATION OF ALGORITHM

PLACES	
P1	Data given for Encryption
P2	Data is converted to binary
P3	Data of the prime positions are shifted
P4	2's complement of the data is obtained
P5	Reverse of the data is obtained
P6	8 bits of 1's are provided
P7	X-OR of the reversed data & 8 bits of 1's are done
P8	8 bit password is provided
P9	X-OR with data(from the previous step) & 8 bit password is obtained
P10	Encrypted data is obtained
TRANSIIONS	
T1	Binary transformation of the plain text is done
T2	Data of the prime positions are shifted
T3	2's complement of the data is done
T4	Reverse of the data is done
T5	X-OR of the reversed data with 8 bits of 1's is done
T6	X-OR of the data with password is done
T7	Expansion of the x-or ed data is done

TABLE I

PLACES	
P11	Encrypted data is provided
P12	Compression of the data has been done
P13	8 bit password is given
P14	X-OR of the compressed data with 8 bit password is done
P15	8 bits of 1's is provided
P16	X-OR of the data with 8 bits of 1's is done
P17	Reverse of the data is obtained
P18	2's complement of the data is obtained
P19	Data of the prime positions are shifted
P20	Actual data i.e. Plain text is obtained
TRANSIIONS	
T8	Compression of the encrypted data is done
T9	X-OR of the compressed data with the password is done
T10	X-OR of the data from the previous step and 8 bits of 1's is performed
T11	Reverse of the data is taken
T12	2's complement of the reversed data is calculated
T13	Binary bits of the prime positions from the 2's complemented data is shifted
T14	From the binary to actual data is converted.

TABLE II

In the Fig 1 initially when data is required to encrypt by a sender then the data is provided to the place P1 i.e. when a token is present in the place P1 then the transition T1 can fire and the data is transformed to its binary form and stored in place P2. After that the bits from the prime position of binary data are shifted towards right (i.e. bit of 2^{nd} position is shifted to the 3^{rd} position,…..,, bit of 7^{th} position is shifted to the 2^{nd} position), and then 2's complement of this shifted data is obtained after firing of T3. Next the 2's complemented data is reversed (transition T4) and then

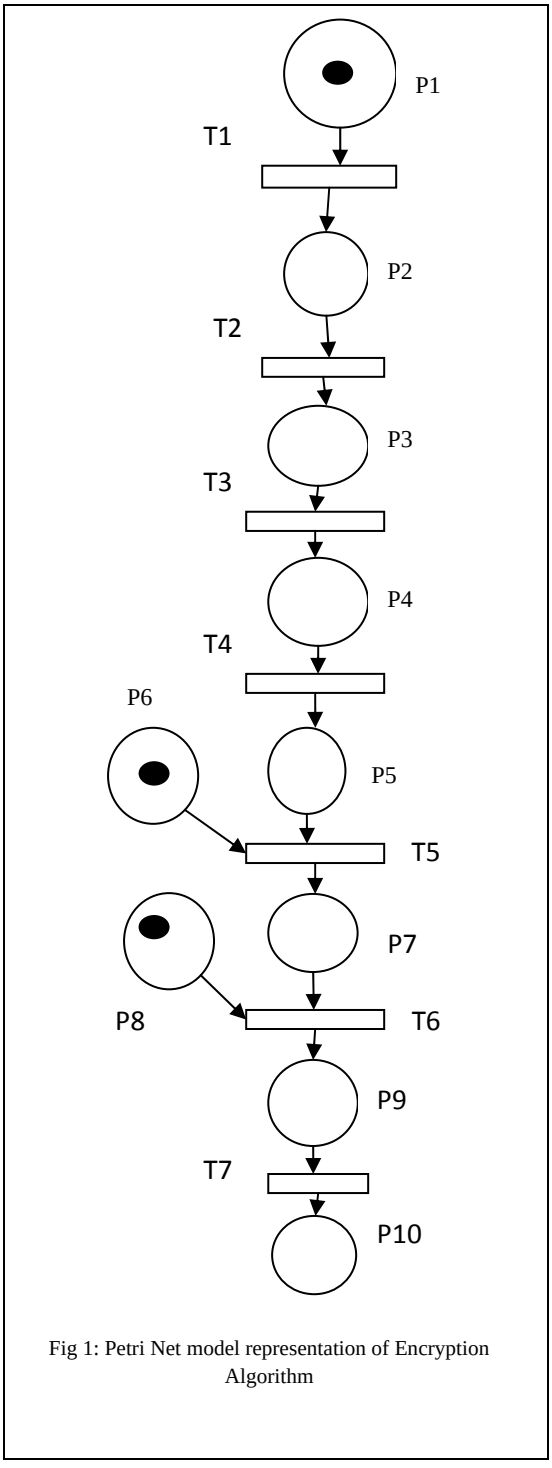


Fig 1: Petri Net model representation of Encryption Algorithm

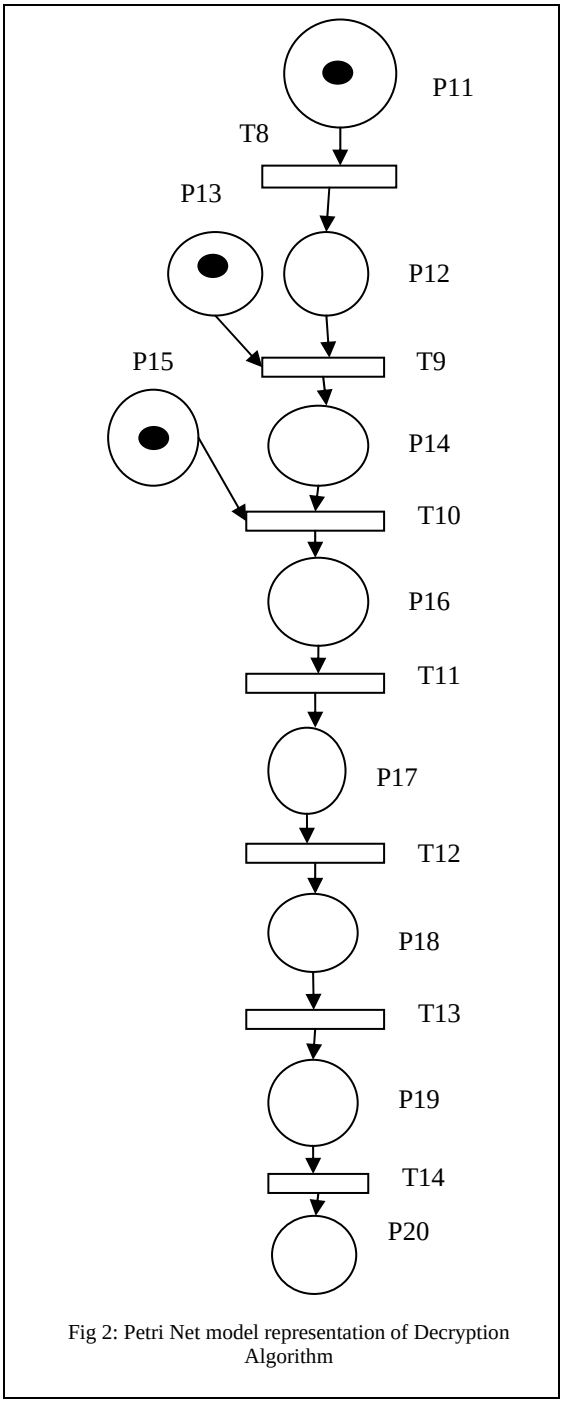


Fig 2: Petri Net model representation of Decryption Algorithm

the reversed data is x-or -ed with 8 bits of 1's. Now when the sender provides the password the data from the previous step is again x-or –ed with the given password (transition T6). Finally the x-or –ed data is expanded following the mechanism as discussed in the proposed algorithm. Thus the encryption process completed and the cipher text is thus obtained in place P10.

Similarly, as in Fig 2 when in the receiver side the data is present and need to decrypt, i.e. token is available in place P11 then first the compression of data (transition T8) is performed according to the mechanism specified in the proposed algorithm as the first step of decryption algorithm. Next the compressed data is x-or –ed with the password provided by the receiver (transition T9). Then the data is again x-or –ed with 8 bits of 1's (transition T10).After that the reverse of the data is obtained and 2's complement of the reversed data is calculated in transition T11 & transition T12 respectively. And then the bits from the prime position of binary data are shifted towards right (transition T13) and finally the actual data i.e., plain text is obtained in place P20.

V. VERIFICATION OF THE ALGORITHM

A. REACHABILITY TREE-BASED ANALYSIS OF ALGORITHM

In a Petri net, the result of firing a transition in a marking M is a new marking M'. So we say that M' is immediately reachable from M if we can fire some enabled transition in the marking M resulting in the new marking M'. So from the initial marking M0, the Petri net can form a reachability tree by any possible transition. The reachability tree for a Petri net is constructed by the following algorithm [4].

Step 1) Label the initial marking M0, make it the root and tag it “new”.

Step 2) While “new” markings exist, do the following;

Step 2.1) Select a new marking M.

Step 2.2) If M is identical to a marking on the path from the root to M, then tag M “old” and go to another new marking.

Step 2.3) If no transitions are enabled at M, tag M “end”.

Step 2.4) While there exist enabled transitions at M, do the following for each enabled transition t at M;

Step 2.4.1) Obtain the marking M' that results from firing t at M.

Step 2.4.2) On the path from the root to M if there exits a marking M"such that M'(p)≥M"(p) for each place p and M'≠ M", i.e., M"is coverable, then replace M'(p) by ω for each p such that M'(p)≥ M"(p). (Note ω can be thought as “infinity”).

According to the reachability tree algorithm, Petri nets model of fig.1 can be transformed into a reachability tree showed in fig.3. As is seen by the graph, from initial marking M0(1,0,0,0,0,1,0,1,0,0).

According to the reachability tree algorithm, Petri nets model of fig.2 can be transformed into a reachability tree showed in fig.4. As is seen by the graph, from initial marking M0(1,0,1,0,1,0,0,0,0,0).

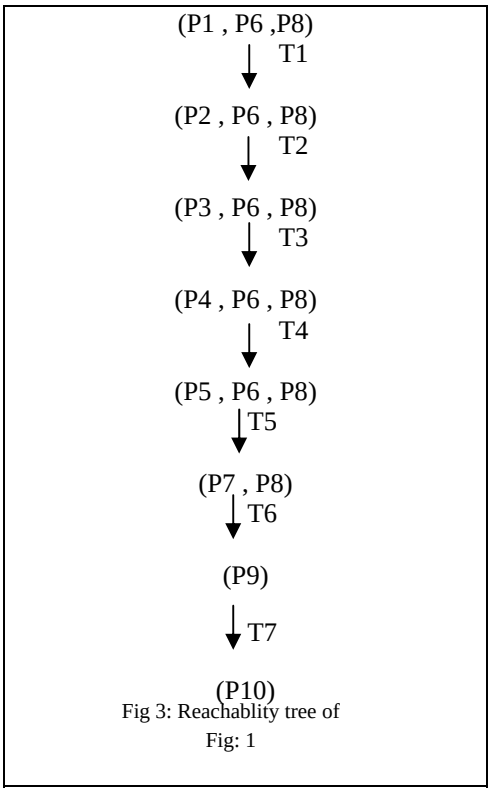


Fig 3: Reachablity tree of Fig: 1

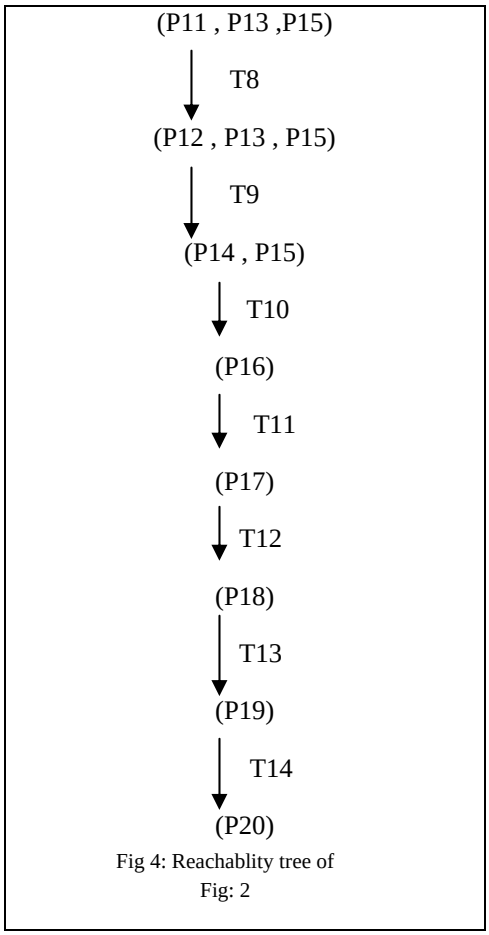


Fig 4: Reachablity tree of Fig: 2

Through the reachability tree, for both of the Petri nets models of Fig 3 & Fig 4 can be studied as follows:

1. The number of Tokens in each node of the reachability tree is always 1, so the net is bounded.
2. Each transition of the system is not fired less than 1 time. There isn't any fired transition in the graph, and each marking of net system has its subsequent state, i.e. no matter what any situation of net is reached, it is possible to fire the transition and reach the final state through some firing sequence. So the net is live and deadlock-free.
3. There is not useless circulation in the graph, and the whole net is reachable.

VI. CONCLUSION

This paper has described a new algorithm (symmetric key cryptographic algorithm). In this algorithm the steps which are used for encrypt the plain text are unique. When any sender encrypts the plain text using the steps of the proposed algorithm it becomes secured and almost unpredictable by eavesdropper. And the working principle of this algorithm has been modeled using a graphical and mathematical modeling tool- Petri Net. By using the reachability analysis technique of Petri Net the algorithm is verified and proved to be boundedness, deadlock free and liveness.

VII. REFERENCES

[1] Tadao Murata ― Petri Nets: Properties, Analysis and Application , Proceedings of the IEEE, Vol.77, No.4, April 1989
[2] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli ― Fundamentals of Software Engineering , Second Edition, Pearson Education, Inc., 2003
[3] Yuan Chongzhi - Principle and Application of Petrinet [M]. Beijing: Publishing House of Electronic Industry, 2005:40-66
[4] Petri Net Model of Session Initiation Protocol and Its Verification - Yang Peng; Yuan Zhanting; Wang Jizeng; Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on Digital Object Identifier: 10.1109/WICOM.2007.466 Publication Year: 2007; Page(s): 1861-1864.
[5] A Petri Net Representation of a Web-Service-Based Emergency Management System in Railway Station- Suparna Karmakar, Dr. Ranjan Dasgupta : International Conference on Computer Science & Information Technology (ICCSIT, 2011), Organiser : World Academy of Science Engineering & Technology, Venue & Date : Italy, 28-30 Nov, 2011, ISBN/ISSN No : pISSN 2010-376x, eISSN 2010-3778, Page(s) : 2284-2290
[6] Jiacun Wang, "Charging Information Collection Modeling and Analysis of GPRS Networks", IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and reviews, vol.37, No.4, July 2007
[7] Alessandro Giua, Carla Seatzu, "Modeling and Supervisory Control of Railway Networks Using Petri Nets", IEEE Transactions on automation science and engineering, Vol.5, NO.3, July 2008
[8] K. Jensen, L.M. Kristensen, L. Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems". International Journal on Software Tools for Technology Transfer, (2007), Springer Verlag, 213-254.
[9] Atul Kahate ― Cryptography and Network Security, Second Edition, Tata McGraw Hill Publishing Company Limited, ISBN – 13 : 978-0-07-064823-4 ISBN – 10 : 0-07-064823-9
[10] William Stallings ― Cryptography and Network Security, Fourth Edition, Pearson Education, ISBN : 978-81-7758-774-6
[11] "Symmetric key cryptography using random key generator" - A.Nath, S.Ghosh, M.A. Mallik, Proceedings of International conference on SAM-2010 held at Las Vegas (USA) 12-15 July, 2010, Vol-2, P-239-244
[12] Behrouz A. Forouzan ― Cryptography and Network Security" Special Indian Edition 2007, ISBN – 13 : 978-0-07-066046-5 ISBN – 10 : 0-07-066046-5