International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

A Novel Message Encryption Strategy using Random Numbers for Point-To-Point Message Transfer

Surya Ahuja[1], Nishit Angrish[2]

[1]BTech IT, VIT University, Vellore, Tamil Nadu, India

[2]BTech IT, VIT University, Vellore, Tamil Nadu, India

Abstract: ***This paper presents a novel technique for public-key encryption using matrices and complete random number generation. The key is encrypted on the client side and the decryption takes place subsequently on the server side. The key obtained through the method described in the paper uses mathematical operations to take care of pseudo randomness that is a characteristic of common cryptography paradigms but the conversion is efficient for a limited number of users taking part in the encrypted data transfer.***

Keywords: Matrices, random numbers, Pseudo Random number, date and time variables, elliptical equation, public key cryptography

1. Introduction

Asymmetric (or public) key cryptosystems use two different keys that are not feasibly derivable from one another, one for encryption and another for decryption. A person wishing to receive messages, generates a pair of corresponding encryption and decryption keys. The encryption key is made public, while the corresponding decryption key is kept secret. Anyone wishing to communicate with the receiver may encrypt a message using the receiver's public key. Only the receiver can decrypt the message, since only he or she has the private key. Asymmetric key cryptosystems may also be used to provide for digital signatures, in which the sender encrypts a signature message using the sender's private key. In turn, any other party can confirm that the signed message originated from the sender by decrypting the signed message with the corresponding public key. [8].Encryption is the technique of converting information into a form that can be "hidden" in a manner such that no unauthorised third part has access to it.

One of the most important topics that encompasses public-key cryptography is the need for a secure and efficient data encryption and key generation algorithm which can be put to use for exchange of information.

The reason why algorithms like RSA work is because the algorithm used for encoding is efficient to such an extent that it renders decoding by an unauthorized third party infeasible in real time.

Random Number generation is one of the most indispensable part of network security as it plays an important role in key generation. It is mainly applied in cryptology in a manner such that either the key is a random number or is generated using a function by utilizing a series of random numbers.

Our methodology strives on the generation a unique key that can be used for the generation of an intermediate "sending matrix" that can in turn, be sent over a socket generated through java network programming to a server where the corresponding intermediate "sending" matrix is decoded and the plaintext obtained.

The purpose of this paper is to efficiently generate large nonsingular matrix (S, S^{-1}) pairs and permutation matrices over the binary field using short keys. The motivation of this work is to provide a solution to the long-key problem in algebraic-code cryptosystems. A special class of matrices which have exactly two 1's in each row and each column is defined, and their properties are investigated to facilitate the construction of these algorithms. The time complexities of these algorithms are studied and found to have $O(n)$ n-bit word operations. [12]

Pseudo-Random numbers are generally achieved using a Pseudo-random number generator (PRNG), achieved by the linear feedback shift register and the linear convergence algorithm. [1][2][3].The essence of the first way is a linear transformation of the n-input and has a huge relevance even today. The second way has a high generation speed and a longer cycle of the output sequence, but its hardware implementation is very complicated and the safety performance is very poor.

Our method tends to move away from both these methods by utilizing the system date and time parameters (Hours, minutes, second and Day, Month, Year).This method offers a much more accurate randomness behaviour as compared to the generalized ways of producing random numbers which have considerable pseudo random behaviour associated with them.

2. Algorithm Components

Our methodology comprises of 4 main matrices-

- **The chief text matrix**

This matrix represents the original plaintext that may be discretely chosen by the user or selected from a file. The text is converted into 3 x N matrix format by using the ASCII values for each of the characters appearing in the text. Here

N refers to the variable number of columns that may be generated during the conversion. The main reason for selecting ASCII encoding is the sole purpose of ease with which it can be handled in programming languages such as Java and C++.

For example, the text "quod erat demonstrandum."would be represented as follows –

$$\begin{bmatrix} 100 & 32 & 101 \\ 114 & 97 & 116 \\ 32 & 100 & 101 \\ 109 & 111 & 110 \\ 115 & 116 & 114 \\ 97 & 110 & 100 \\ 107 & 109 & 46 \end{bmatrix}$$

Any unoccupied spaces occurring in the matrix would be padded by filling it in through the "space" character (ASCII value: 32) in order to ensure the 3 x N format.

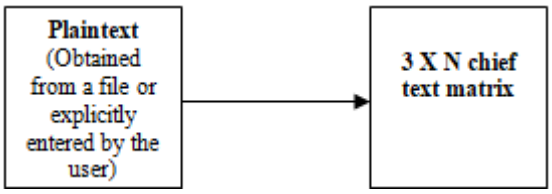

Figure 1: Creating the Chief Text matrix

- **The intermediate key matrix**

This matrix forms the backbone of our method. The matrix is of 3 x 3 format with the first two rows comprising of the current Date and Time variables (each following the European format of dd/mm/yyyy and a 24 hour clock hh/mm/ss format respectively) and the third row being generated by performing an arithmetic operation (addition) on the other two rows.

For example, the key matrix at 29/09/2017 01:53:23 would be represented as

$$\begin{bmatrix} 29 & 09 & 2017 \\ 01 & 53 & 23 \\ 30 & 61 & 2040 \end{bmatrix}$$

The possibility of different time and date variables at different situational instances has been used for generation of randomness in the matrix.

- **Derived key matrix**

This matrix is created through the use of (1).

$$y^2 = x^3 + ax^2 + bx \quad (1)$$

That is similar to the equation of an ellipse being utilized in elliptical key cryptography. Here "x" is substituted as the intermediate key matrix and the constants "a" and "b" are both set to 1.The final result is given by the matrix associated with y^2 in the equation. The matrix, like its parent intermediate key matrix is in the 3 x 3 format.

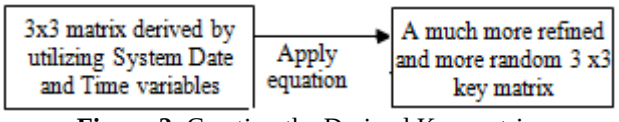

Figure 2: Creating the Derived Key matrix

- **Intermediate sending matrix**

This matrix is created by performing multiplication between the Derived key matrix and the chief text matrix. This is of the 3 x N format where N again represents the variable number of columns.

3. Design and Implementation

The algorithm is implemented as follows -
The chief text matrix is generated by taking ASCII values for each of the characters present in the text and creating a 3 x N matrix.

Similarly the key matrix is created from the intermediate key matrix and the derived key matrix through the methodology of using the Date and Time variables and then substituting the intermediate key matrix in equation (1) (Steps as already described in the section 2).

Now the final key matrix obtained is multiplied with the chief text matrix to obtain an intermediate sending matrix that contains the text message encoded.

The key matrix is now embedded into the intermediate sending matrix in the rows where the original matrix ends. This results in the formation of a 3 x (N+3) matrix which is ready for deployment over a network.

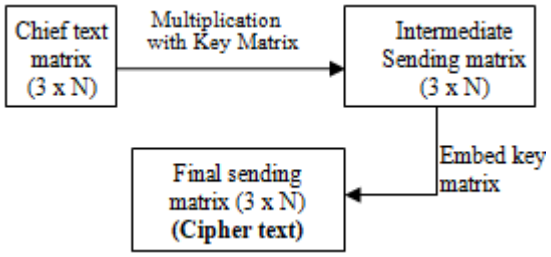

Figure 3: Forming the encrypted message.

On the other side, the key matrix is first extracted from the sent matrix as information regarding its position is present with the receiving client already.

The inverse of the key matrix is then calculated and is further multiplied with the sent matrix (available after extraction of the key matrix) to obtain the decoded matrix.

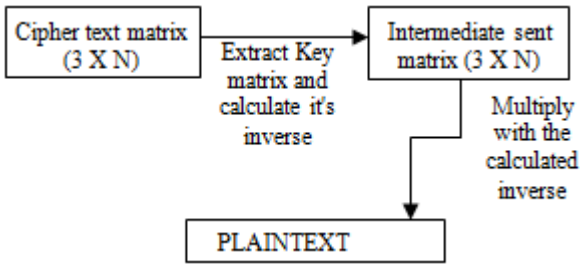

Figure 4: Decrypting the received message.

3.1 System Specifications

The algorithm was implemented on a system with the following specifications-
RAM: 6 GB
PROCESSOR: Intel ® Core ™ i5-3210M CPU @2.50GHz

HARD DISK: 500 GB
OS: 64-bit Windows Operating System

3.2 Equations Utilized

The elliptic curve adaptation of the Diffie-Hellman exchange is known as elliptic curve Diffie-Hellman (ECDH.) exchange. The requirement of the two parties involved in an ECDH exchange is that, they share the same curve parameters and a generating base point prior to the start of communication. [4]

$$y^2 = x^3 + ax^2 + bx \quad (2)$$

This refers to the prime equation being used in elliptical key cryptography. Our method does make use of the equation but not
in the way it has been defined in the traditional sense. Our methodology makes use of a variant of this equation for increasing the size as well as the complexity of our key matrix.

4. Algorithm Analysis

Our algorithm focuses on point-to-point message transfer between two clients joined together by a network. The analysis of the algorithm was performed along with one of the related works, RSA that is one of the major algorithms being used in world today. [5].The comparison with RSA is only limited to the strength of its encryption technique and the processing time.

Table 1: Comparison between our approach and RSA

Features	Our algorithm	RSA
Key	Same key is used for encryption and decryption	Different keys are used for encryption and decryption
Scalability	Not Scalable	Not Scalable
Avalanche Effect	Higher because of high randomness	High
Power Consumption	Low because of O (n^2) space and time complexity.	Very High
Confidentiality	Lower because of the need for the information about positioning of the key in the sent matrix to be already present in with the receiver.	Low

Some of the tests that we performed on RSA by taking different values of the prime numbers p1 and p2(considering only the prime numbers that were large enough for execution) that form an integral part of the algorithm were as follows –

Table 2: Analysis of RSA algorithm

P1	P2	Time taken for algorithm to completely execute
24001	24019	2 min 43 sec
29197	242013	2 min 27 sec
25414	25801	1 min 8 sec
25919	25999	2 min 40 sec
10739	10733	1 min 28 sec

Each of these tests were performed on system with the same configuration as mentioned in one of the earlier sections. Apart from one anomaly the tests were in accordance with the graph given below [5]

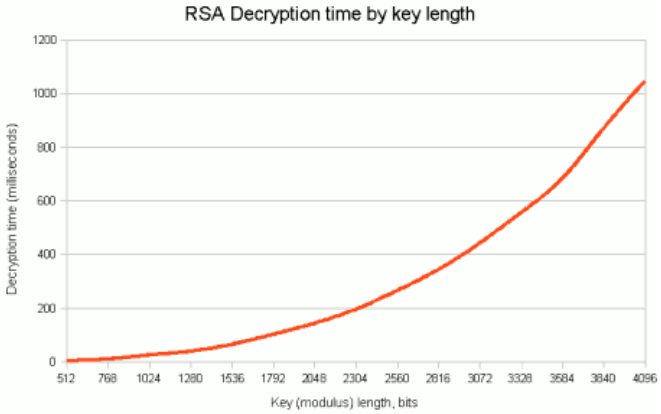

Figure 5: Behavior of RSA decryption with key size

Similar tests conducted on our algorithm yielded the following results with the assumption that that the position of the key matrix is present with the receiver–

Table 3: Analysis of our algorithm

Number of characters taken originally	Time taken in program execution(encryption and decryption)
100	5.02 sec
150	5.12 sec
200	5.20 sec
250	5.28 sec
400	5.40 sec
1000	12 .43 sec
2000	21.23 sec

As is projected by our analysis of the execution time of our algorithm, provided that the key is present with the receiver the execution time of our algorithm is comparatively low, and also offers high security because if the key is not present with the receiver and in case the message is trapped by the third party, it takes high resource consumption and large brute force attack times in order to find the plaintext, that too not completely accurate.

5. Conclusion

Our algorithm offers high usability despite its lack of scalability and low confidentiality. If exploited well, it could easily be used for providing high security over web sessions as well as during multiple client information exchange. The security service offered by the algorithm is high and the encrypted message cannot be easily decrypted in the absence of the key.

6. Future Work

There are several areas where the algorithm can be improved in order to enhance its security, processing time and security. A system that we are currently working on will remove the need for the information about the positioning of the key matrix in the sent matrix to be already present with the receiver. This method involves including randomness in the position of the key matrix within the sent matrix itself so as

to introduce more security. The matrix that will be sent to the user would then include the row and column positions of the key which could then be extracted. The scalability of the matrix also needs to be worked upon so that the size of the key varies with the text that is to be sent.

References

[1] Ding Jun Li Na Guo Yixiong Yang Jun, "A high-performance pseudo-random number generator based on FPGA", Wireless Networks and Information Systems, 2009. WNIS '09. International Conference.
[2] Ma Hua, Zhang Xiaoqing, Zhang Pengge. A pseudo-random number generator based on the linear congruence algorithm[J], Pure and, Applied Mathematics,,Vo.21 No.3, Sep. 2005.
[3] Min Min. "On the Production of Pseudo-random Numbers in Cryptography", JOURNAL OF CHANGZHOU TEACHERS COLLEGE OF TECHNOLOGY, Vo1. 7, No. 4, Dec. 2001.
[4] Abdul Azim, M. " An Efficient Elliptic Curve Cryptography based Authenticated Key Agreement Protocol for Wireless LAN Security" High Performance Switching and Routing, 2005. HPSR. 2005 Workshop.
[5] Dr Sudesh Jakhar, Aman Kmar, Mr Sunil Makkar, "Comparative Analysis between DES and RSA Algorithm".
[6] Younchan Jung ,"Session Key Generation for a Group Call and Device for Security Control",Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference.
[7] MengboHou, QiuliangXu ,"Two-Party Authenticated Key Agreement Protocol from Certificateless Public Key Encryption Scheme".
[8] Mohammad Peyravian, Stephen M Matyas, Allen Roginsky, Nevenko Zunic, Generation of RSA Keys That Are Guaranteed to be Unique for Each User, Computers & Security, Volume 19, Issue 3, 1 March 2000, Pages 282-288, ISSN 0167-4048.
[9] Ananya Gupta, Anindo Mukherjee, Bin Xie, Dharma P. Agrawal, Decentralized key generation scheme for cellular-based heterogeneous wireless ad hoc networks, Journal of Parallel and Distributed Computing, Volume 67, Issue 9, September 2007, Pages 981-991, ISSN 0743-7315.
[10] Horng-Twu Liaw, A dynamic cryptographic key generation and information broadcasting scheme in information systems, Computers & Security, Volume 13, Issue 7, 1994, Pages 601-610, ISSN 0167-4048, 10.1016/0167-4048(94)90011-6.
[11] LeinHarn, Hung-Yu Lin, A cryptographic key generation scheme for multilevel data security, Computers & Security, Volume 9, Issue 6, October 1990, Pages 539-546, ISSN 0167-4048.
[12] Hung-Min Sun, Tzonelih Hwang, Key generation of algebraic-code cryptosystems, Computers & Mathematics with Applications, Volume 27, Issue 2, January 1994, Pages 99-106, ISSN 0898-1221.