A NOVEL SECURE CODE APPROACH USING LATIN SQUARE CRYPTO SYSTEM AGAINST REVERSE ENGINEERING

N.Sasirekha[1], M.Hemalatha[2]

ABSTRACT

Security is of fundamental importance to deliver software to the customers. Reverse engineers can extract the source code from the binary file which is given to the clients. There are several methods to protect software from reverse engineering. This paper proposes the implementation of Latin Square (Quasi group) based permutations that have very good encryption properties. Latin Square scrambles the binary file and makes the job of a reverse engineer a tough task.

Key words : *Reverse Engineering, Latin Square, Encryption, Decryption, Indexed Table, software protection.*

I. INTRODUCTION

Compiled form of the software is sold to the clients instead of its source code. This is compiled form is usually in machine language, which only the computer, and not humans, can understand. The original language in which the software is written, is not given to the clients. The source code language could be C or C++. **Reverse engineering** is the process of discovering the functions of software through the analysis of its machine code by the observation of its actions. Reverse engineering merely studies the software without changing anything. If the software is changed in any way, then that process is known as cracking.

Reverse engineering is done for a number of reasons. The one most familiar to software users is probably piracy, where one gets fully working software without paying for it. Other reasons include discovering various vulnerabilities in the software so as to attack other users of the software, and stealing the algorithms or computations used in the software. It is for preventing reverse engineering that software developers now commonly protect their software before releasing it to the public. It is used as a learning tool; as a way to make new, compatible products that are cheaper than what is currently on the market; for making software interoperate more effectively or to bridge data between different operating systems or databases; and to uncover the undocumented features of commercial products. Reverse engineering can be used for both good and bad.

- Find vulnerabilities
- Analyse patches
- Write exploits and malware
- Develop security patches

[1]Doctoral Research Scholar, Department of Computer Science, Karpagam University, Coimbatore.
E-mail : nsasirekhaudt@gmail.com

[2]Professor, Department of Computer Science, Karpagam University, Coimbatore.
E-mail : hema.bioinf@gmail.com

If people use the reverse engineering for bad things, commercial softwares has to be protected. **Software protection** is the use of special techniques to deter or prevent a malicious user from modifying the software.

Random numbers have applications in lottery games, shuffling cards, online gambling and secure communication. They may be used in generating random element for information dispersal [1] [2] and secret sharing algorithms [3]. Another application may be found in generation of shared secret keys between sensors in a sensor network [4] using factors of randomly generated matrices. A random number generator, with good statistical properties, based on a recursive algorithm is proposed in [5, 6]. Several factors are involved in choosing a pseudorandom number generator i.e. memory constraints, computational power available etc. This paper presents pseudo random number generator based on Quasigroups.

Quasigroups or Latin Squares are n x n matrices similar to Sudoku. The computational cost of the scheme is very low, because it needs only table look up operations to create sequence of numbers and storage requirement for two n x n matrices.

II. Literature Survey

Gosler's software protection survey (1985)[12] examine circa1985 the protection technologies which includes of hardware security tools, ûoppy disc signatures (magnetic and physical) and analysis of refutation approaches and slowing downward the interactive dynamic analysis. The major goal is software copy prevention, but Gosler experiential that the effectiveness of resisting copying ought to be balanced by the potency of resisting software analysis and that of software modification. Useful tampering is generally headed by reverse engineering.

Collberg et al. [7] provided a compact outline of the approaches to protect against these threats. Software watermarking for instance focuses on protecting software reactively against piracy. It usually implants hidden, distinctive data into an application in such a way that it can be guaranteed that a particular software instance belongs to a particular individual or company. When this data is distinctive for each example, one can mark out copied software to the source unless the watermark is smashed. The second group, code obfuscation, protects the software from reverse engineering attacks. This approach comprises of one or more program alterations that alter a program in such a way that its functionality remains identical but analyzing the internals of the program becomes very tough. A third group of approaches focuses to make software "tamper-proof", also called tamper-resistant.

Cappaert et al. [8] presented a partial encryption approach depending on a code encryption approach. In order to utilize the partial encryption approach, binary codes are partitioned into small segments and encrypted. The encrypted binary codes are decrypted at runtime by users. Thus, the partial encryption overcomes the faults of illuminating all of the binary code at once as only the essential segments of the code are decrypted at runtime.

Jung et al. [9] presented a code block encryption approach to protect software using a key chain. Jung's approach uses a unit block, that is, a fixed-size block, rather than a basic block, which is a variable-size block. Basic blocks

refer to the segments of codes that are partitioned by control transformation operations, such as "jump" and "branch" commands, in assembly code. Jung's approach is very similar to Cappaert's scheme. Jung's approach tries to solve the issue of Cappaert's approach. If a block is invoked by more than two preceding blocks, the invoked block is duplicated.

Gutmann [10] put forth an apparent conversation of the security concerns facing cryptographic usage in software under general-purpose operating systems, and analyzes the design difficulties in nullifying these concerns faced by using secure cryptographic co-processors.

However, the above discussed schemes did not meet the security requirements and moreover had an efficiency problem. Moreover, time cost and space cost should also be taken into consideration. Thus, a novel cryptographic technique is proposed in this approach.

III. Reverse Engineering Attacks

Software reverse engineering is the method of getting the original source code from the binary source code. Competitors may use reverse engineering to number out how to implement that cool characteristic. Crackers may use it to see how they can bypass their license policy (Roshen Chandran, 2008). This section presents a few reverse engineering attacks (Matias Madou, 2005; Sebastian Schrittwieser and Stefan Katzenbeisser, 2011; Larry D-Anna et al, 2003). Attacker can utilize these attacks to understand functionality of software.

A. Static Analysis Attack

In this attack, attacker builds Control Flow Graph (CFG), it is a higher level representation of code of software. It contains a set of nodes, indicating instructions in code and a set of edges between nodes representing possible control paths. CFG of program helps attacker to understand the functionality of software.

B. Dynamic Analysis Attack

The dynamic analysis attack in which the attacker must executes software using set of inputs and the output of software by creating execution traces. Attacker can also outline the software to make known of actual paths selected for program execution. Dynamic analysis provides significant influence in locating secret information (Example: cryptographic keys) present in software. Dynamic analysis is harder than stagnant analysis because software needs to be run on different inputs.

C. Code Clone Detection Attack

In this code clone detection attack, the attackers detect and take away the code clones present in program to understand the functionality of software. Attacker can make use of existing code clone detection techniques such as matching Abstract Syntax Tree (AST) (described by Baxter and others (1998)) to detect and remove code clones.

IV. Need For Software Protection

The security level in an application consists of the necessary resistance of the application beside reverse engineering and tampering attacks. By means of some model parameters of this level is discussed below:

- Vulnerability: Open systems are more vulnerable to attacks than closed systems. Desktops, mobile devices are additionally vulnerable to white-box attacks compared to physically protected servers easy to get by means of a network link.

- Value of contented: Type of attacks and the amount of resources invested by an attack depend on the value of the content (code or data) and on the nature of the application.
- Content lifetime: Content or properties with a longer lifetime require a higher level of trust and security.
- Security life cycle: The security of an application can be designed to be at regular intervals. Systems without promote possibilities need a higher security level than systems with systematic upgrades.
- Sensitivity for global attacks: Global attacks are attacks concerning a whole group of software instances. This is feasible when code contains a "global secret". A single key for all legitimate users or an unpatched security flaw allows an attacker to develop an automated attack and spread it through the Internet.

The actual security level is always a exchange between the need for software protection and to implement these software protection techniques.

V. Methodology

A code encryption scheme is proposed with the use of an indexed table to protect the software. The indexed table can solve the problem of multiple paths. Furthermore, it solves such problems as loops, recursions, and multiple calls.

Step 1: Source code compiling process illustrated in Figure 1. After this step, the source code is compiled and outputs a binary image.

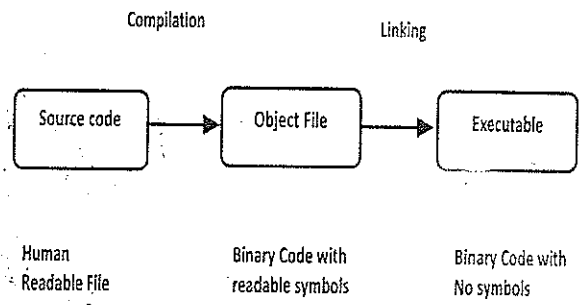

Figure 1: Compilation Process

Step 2: Construction of the indexed table proposed by Sungkyu Cho et al(2011)[11].

Step 3: Input the Indexed table to Quali group encryptor.

Latin square(Quasigroups) distribute a narration with the popular game Sudoku and the long lived Latin squares. The core of a Quasigroup is denied in the same manner as a Latin square. These consist of an n2 set of ordered triples having the form (r_i, c_j, v_{ij}); $r_i, c_j, v_{ij} \in$ "Integers, with the additional stipulation that for each (r_i, v_{ij}) and (c_j, v_{ij}) is unique. This relation can be represented as an $n \times n$ square matrix with r_i and c_j being the row and column indices and v_{ij} is the value in the r_ith row and c_jth column. The difference between a Quasigroup and a Latin square is the deûnition of an operator "·" on a Quasigroup. This operator is somewhat simple in that it performs a table/matrix lookup from the Quasigroup. A Quasigroup equation of the form $x \cdot y = z$ translates directly to the ordered triple (r_x, c_y, v_{xy}), where. $z := v_{xy}$ By these definition, the Quasigroup operation is together closed and invertible, making Quasigroups prime candidates for encoding systems.

Latin squares have been formerly investigated for their application to encryption, where they are known wholly

by their Quasigroup name. Gligoroski et al. (2004) look at tributary cipher and public key implementations with quasigroups. A multi-level quasigroup accomplishment was proposed by Satti and Kak (2009)[15]. Satti and Kak (2009) united the execution with indices and nonce's to look up on the potential of the encryption. On the other hand, their system also focuses on a stream cipher execution. Marnas et al. implement a quasigroup all-or-nothing system. Though, quasigroup encryption used here is to substitute the XOR operation inside the other system, for this reason at the end of the actual encryption the other cryptosystems is used. Quasigroups have also been applied to error correction and in construction of message authentication codes (MAC).

One can outlook the quasigroup transformation as a replacement or substitution and permutation operation. These operations form the basis of a variety of encryption systems mainly in speech encryption (Borujeni, (2000)[14]. Furthermore, the algorithms proposed in this paper do not require any computations to be performed but only table look up operations for encryption and decryption.

The output of the proposed encryptor is reliant in the lead of the index numbers and the orders of the matrices (r, s) which are sent by the trusted influence. The encryption is also dependent on six multiplier elements that are generated by a secret algorithm based on the index numbers, the order of the matrices under contemplation and nonce (random number generated by trusted authority). This key is rationalized by the network on a normal basis (a long time ago in every time interval that is far less that T, the time needed to use brute force to decrypt the key that is sent by the trusted authority).

Figure 2 illustrates the quasigroup encryptor. The component has takes the raw data stream and randomizes it based on the encryption key (the encryption key), and the output data has popular autocorrelation properties.

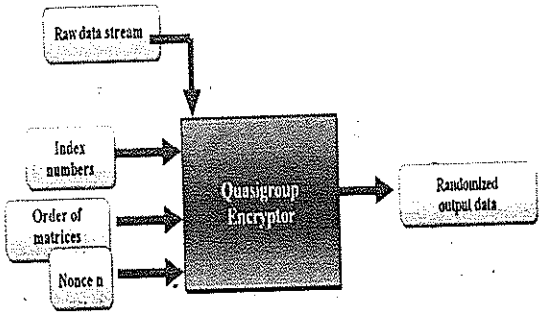

Figure 2: Quasi Group Encryptor

Latin Square definition and Theory

A Latinsquare(quasigroup) is denoted as G can be defined as a group of elements (1, 2, 3... n) all along with a multiplication operator such that for every element x and y there exists a only one of its kind solution such that z such that the subsequent two conditions are satisfied:

$$x * a = z$$
$$y * b = z \qquad (5.1)$$

In equation 5.1, the elements a, b and z belong to the Quasigroup 'G'. Or, a quasigroup is a binary system (Q,*) satisfying the two conditions.

For any a, b belongs to Q there exists a unique x belongs to Q such that a*x=b

For any a, b belongs to Q there exists a unique y belongs to Q such that y*a=b.

Quasi

Input data: d1, d2, d3,... dn

Output data: e1, e2, e3,...en

The two matrices: R, S

Multiplier Elements: q1, q2, q3,... qn

The indices: I1, I2, I3, ... In

The encryptor is defined by QE (stands for Quasi-Encryptor), and the decryptor is defined as QD (stands for Quasi-Decryptor).

Encryption:

It should be that if Q is a quasigroup such that a1,a2, a3,... an belong to it then the encryption operation QE, which is defined over the defined elements, maps those elements to another vector $b_1, b_2, b_3, \ldots b_n$ such that the elements of the resultant vector also belong to the same quasigroup.

The mathematical equation used for encryption (basic level) is defined by:

$$E_a(a_1, a_2, a_3, \ldots, a_n) = b_1, b_2, b_3, \ldots b_n \quad 5.2$$

where the output sequence is defined by:

$$b_1 = a * a_1$$

$$b_i = b_{i-1} * a_i$$

where i increments from 2 to the number of elements that have to be encrypted, and a is the hidden key (leader in Markovski and Dimitrova terminology (Dimitrova and Markovski, 2004) [13]. Equation (5.2) describes a typical single level quasigroup encryptor. It is assumed that the initial input data given by the vector. $a_1, a_2, a_3, a_4, a_5, a_6$. It is mapped to the vector $b_1, b_2, b_3, b_4, b_5, b_6$ by equation (5.1). The following steps are used during the process of encryption:

$$b_1 = a * a_1 = 2 * 2 = 1$$

$$b_2 = b_1 * a_2 = 1 * 4 = 1$$

$$b_3 = b_2 * a_3 = 4 * 1 = 4$$

$$b_4 = b_3 * a_4 = 4 * 2 = 5$$

$$b_5 = b_4 * a_5 = 5 * 3 = 1$$

$$b_6 = b_5 * a_6 = 1 * 3 = 2$$

The series of the sequences obtained is given as an input to another level of the encryptor. This procedure is repeated for several times. Multiple levels of mapping give lesser surety with similarity of the output data to that of the input data. This makes the decryption of the original data very hard.

In certain implementation, the multiplier element is varied. The multipliers are constructed by a special algorithm called "MEG1" that constructs the multiplier elements depending on the index numbers, Nonce, r and s given by the following equations [13](Dimitrova and Markovski, 2004):

$$E_{h1,h2,h3....hn}(a_1, a_2, a_3 \ldots a_n) = e_1, e_2, e_2, \ldots e_n \quad (5.3)$$

where

$$e_1 = a * a_1 \text{ and } e_i = e_i - 1 * a_i$$

In the above equation, the incoming data is first mapped through the first multiplier element h_1 then the resultant data is mapped taking into account the second multiplier element h_2. This process continues till all the multiplier elements are exhausted.

$$b_1 = h_1 * a_1;\ b_2 = b_1 * a_2; \ldots b_n = b_n - 1 * a_n$$

$$c_1 = h_2 * b_1;\ c_2 = c_1 * b_2; \ldots c_n = c_n - 1 * b_n$$

$$\vdots \qquad (5.4)$$

$$e1 = hn * s1;\ e2 = e1 * s_2; \ldots e_n = e_{n-1} * s_n$$

where the vector $(h_1, h_2, h_3, \ldots h_n)$ comprises of all the multiplier elements. In this approach, this encryption key is communicated all along by means of quasigroup encryption. It is to be observed that in the above two techniques another reliable encryption approach is necessary to preserve the secrecy of the encryption. Moreover, it is essential to broadcast the quasigroup that is being used for encryption, which is one of the main precincts of the above technique. Suppose the eavesdropper (a secret listener to private conversations) breaks the encapsulating cipher, it is possible to get access to the quasigroup used for the encryption and all the other needed data to get the data.

This approach uses the index based approach where the given data is encrypted through a number of levels of encryption. Consider that this distorted order encryptor is given the input of all 1s. It is to be experiential that behind the second level encryption, the input vector is mapped to the sequence which has symbols ranging from 1 to the order of the second matrix. Thus, if an index key is present which references the matrices stored in the memory of the reception device, the intruder would not know which matrix is stored at a given index.

In order to further improvement the efficiency of this quasigroup encryptor, an additional function can be built-in that arranges the quasigroups. According to the Nonce and this makes the encryption more time dependent and it can be experiential that at any given point of time the output of the encryptor is different even if the same set of indices are given to the technique.

The Multi Level Indexed encryptor is denoted as

$$QE^{I_r, I_s}_{h_1, h_2, \ldots, h_n}(a_1, a_2, a_3 \ldots, a_n) = e_1, e_2, e_3, \ldots, e_n \qquad (5.5)$$

where $(a_1, a_2, a_3 \ldots, a_n)$ is the input data and $e_1, e_2, e_3, \ldots, e_n$ is the output vector I_r and I_s are called indices that are arrays which have the indices of quasigroups having corresponding order. The vector $(h_1, h_2, \ldots, h_n)$ is the Hidden key or the Secret key. It is the output of the MEG-1 algorithm.

The methodology is implemented. Figure 3 shows the generation of the Indexed Table. Figure 4 shows the generation of the Latin Square and the Quasi group encryption applied on the Indexed table is shown in Figure 5.

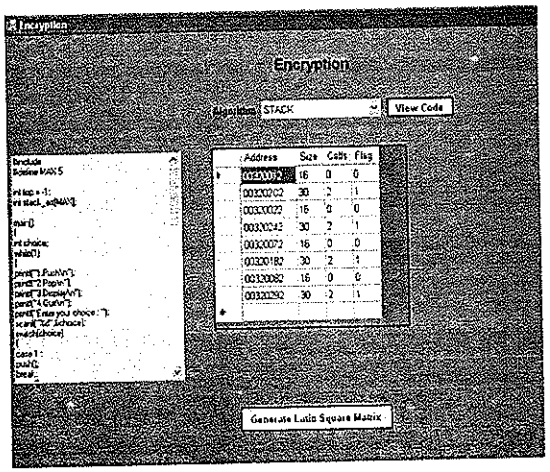

Figure 3: Generation of Indexed Table

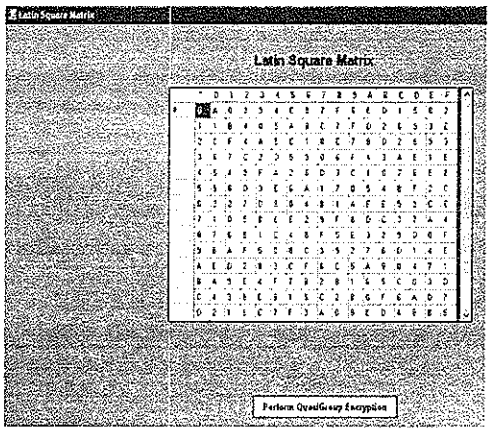

Figure 4: Generate Latin Square Matrix (Quasi Group Matrix)

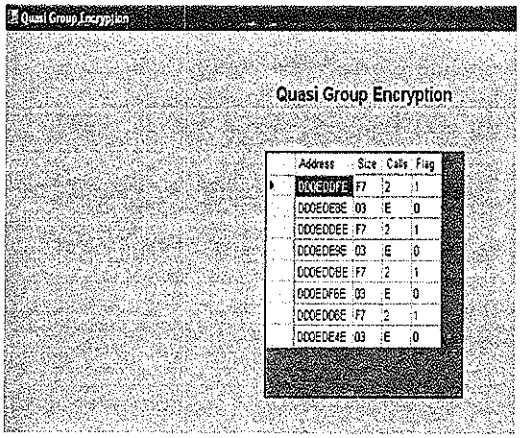

Figure 5: Generate Quasi Group Encryption on the Indexed Table

Decryption:

This process is highly alike the process of encryption which has presently been discussed. The key point to be considered is the construction of the inverse matrix. The left inverse '\' is used for the quasigroup decryption. The fundamental equation for decryption is:

$$D(a_1, a_2, a_3 \ldots, a_n) = e_1, e_2, e_3, \ldots, e_n \qquad (5.6)$$

where $e_1 = \frac{a}{a_1}$ and $e_i = \frac{a_i - 1}{a_i}$

In order to carry out the process of decryption, the inverse matrix of a given quasigroup has to be constructed and execute mapping procedure as described in the previous section, equation (5.7) has to be used instead of (5.6).

The decryptor for a multilevel indexed based algorithm may be defined as follows:

$$QD^{I_r, I_s}_{h_1, h_2, \ldots, h_n}(e_1, e_2, e_3, \ldots, e_n) = a_1, a_2, a_3 \ldots, a_n \qquad (5.7)$$

The elements of the inverse (left-inverse) of quasigroup are labeled as v the indices along the horizontal are labeled w and the indices along the vertical are labeled u^{-1}. The implementation of the decryption process is shown in Figure 6.

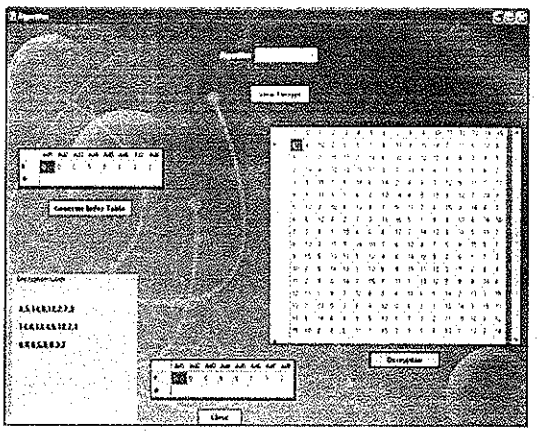

Figure 6: Decryption

VI. Experimental Results

The performance of the proposed approach is compared with the (Sungkyu Cho et al. 2011) code encryption scheme. For instance, if a program P and its modified version P' is available. Then, the time cost C_t and the space cost C_s is defined as

$$C_t(P, P') = \frac{T(P')}{T(P)} \qquad (6.1)$$

$$C_s(P, P') = \frac{S(P')}{S(P)} \qquad (6.2)$$

where T(X) is the execution time of program X, and S(X) is its size.

The encryption time and decryption time of three programs are evaluated. At the moment, external libraries such as ".dll" files are eliminated as they are implemented externally to the executable file. The results are shown in Table 1.

Table 1: Results Comparison

Features	Sungkyu Cho et al., Code Encryption Scheme			Proposed Latin Square Crypto System		
	Pgm1.exe	Pgm2.exe	Pgm3.exe	Pgm1.exe	Pgm2.exe	Pgm3.exe
Original file size (B)	3584	4096	3072	3584	4096	3072
Number of blocks	12	26	18	12	26	18
Decryption and re-encryption time (s)	0.0016	0.0032	0.0024	0.0012	0.0024	0.0018
C_t	6.680	6.119	10.985	5.985	5.845	9.215
C_s	1.027	1.674	1.047	0.925	0.942	0.895

VII. Conclusion

Confidentiality and data authenticity are mainly focused in assuring efficient security. Several techniques are available in the literature for providing security to the software. However, most of the schemes do not meet the security requirements for code encryption schemes, and also had efficiency problems. Recently, encryption has provided the means to hide information. This paper presented and discussed code encryption schemes for protecting software against various attacks like reverse engineering, tampering etc. A new code encryption approach based on an indexed table to guarantee secure key management and efficiency is proposed in this paper. The performance of the proposed approach is evaluated based on the time cost and space cost. It is observed that the proposed Latin Square Crypto System shows 25%, 10.33% and 22.72% decrease in Decryption Re-encryption time, Time cost and Space cost respectively when compared with Sungkyu Cho et al code encryption scheme.

Reference

[1] A. Parakh and S. Kak, *"Online data storage using implicit security," Information Sciences,* vol. 179, no. 19, pp. 3323-3331, 2009.

[2] A. Parakh and S. Kak, *"Space efficient secret sharing for implicit data security," Information Sciences,* vol. 181, no. 2, pp. 335-341, 2011.

[3] A. Parakh and S. Kak, *"A tree based recursive information hiding scheme,"* in *IEEE ICC 2010 - Communication and Information System Security Symposium ('ICC'10 CISS')*, Cape Town, SouthAfrica, 2010.

[4] A. Parakh and S. Kak, *"Matrix based key agreement algorithms for sensor networks,"* in *IEEE ANTS 2011 - 5th International Conference on Advanced Networks and Telecommunication Systems*, Bangalore, India, 2011.

[5] V. Gonzalez-Diaz, G. Setti, P. Fabio and M. Franco, *"A Pseudorandom Number Generator Based on Time-Variant Recursion of Accumulators," IEEE Transactions On Circuits And Systems,* vol. 58, no. 9, pp. 580-584, September 2011.

[6] A. Parakh, *"A d-Sequence based Recursive Random Number Generator,"* in *International Symposiumon System and Information Security; Cryptology ePrint Archive, Report 2006/310*, 2006.

[7] Collberg, C.S., Thomborson, C., 2002. *"Watermarking, tamper-proofing, and obfuscation - tools for software protection"*, IEEE Transactions on Software Engineering, Volume: 28 , Issue: 8, Page(s): 735 – 746,

[8] Jan Cappaert, Nessim Kisserli, Dries Schellekens, and Bart Preneel, 2008. *"Toward Tamper Resistant Code Encryption: Practice and Experience,"* LNCS, vol. 4991, pp. 86-100.

[9] Jung, D.W., Kim, H.S., Park, J.G., 2008. *"A Code Block Cipher Method to Protect Application Programs From Reverse Engineering,"* J. Korea Inst. Inf. Security Cryptology, vol. 18, no. 2, pp. 85-96 (in Korean)

[10] Gutmann, P., 2000. *"An Open-source Cryptographic Co-processor"*, Proc. 2000 USENIX Security Symposium.

[11] Sungkyu Cho, Donghwi Shin, Heasuk Jo, Donghyun Choi, Dongho Won, and Seungjoo Kim, 2011. *"Secure and Efficient Code Encryption Scheme based on Indexed Table"*, ETRI Journal, Volume 33, Number 1.

[12] J. Gosler, *"Software Protection: Myth or Reality?"*, Advances in Cryptology –CRYPTO'85, Springer-Verlag LNCS 218, pp.140–157 (1985).

[13] Dimitrova, V., Markovski J., 2004, On Quasigroup Sequence Random Generator. Proceedings of the 1st Balkan Conference in Informatics, Y. Manolopoulos and E. Spirakis, Eds., 21-23, Thessaloniki, Greece, pp. 393 – 401.

[14] Borujeni, S., 2000. Speech encryption based on fast fourier transform permutation. In Proc. of 7th IEEE International Conference on Electronics, Circuits and Systems, ICECS., 1: 290 –293.

[15] Satti, M. and S. Kak, 2009. Multilevel indexed quasigroup encryption for data and speech. IEEE Trans on Broadcasting., 55: 270-281.

AUTHOR'S BIOGRAPHY

N. Sasirekha, completed MCA, M.Phil. in Computer Science. She is a doctoral research scholar in Computer Science at Karpagam University, Coimbatore, Tamilnadu, India. She is

currently working as Assistant Professor in PG Department of Computer Applications at Vidyasagar College of Arts and Science, Udumalpet, Tamilnadu. She has presented Fifteen papers in various National Conferences, Seminar and Six papers in International Conference. She has published eight research papers in International Journals. She is a reviewer for two International Journals and International conferences. Her area of research is Software Engineering, Security.

Dr. M. Hemalatha, completed M.Sc., M.C.A., M. Phil., Ph.D (Ph.D, Mother Terasa women's University, Kodaikanal). She is Professor & Head and guiding Ph.D Scholars in Department of Computer Science at Karpagam University, Coimbatore. Fourteen years of experience in teaching and published more than hundred papers in International Journals and also presented more than eighty papers in various national and international conferences. She received best researcher award in the year 2012 from Karpagam University. Her research areas include Data Mining, Image Processing, Computer Networks, Cloud Computing, Software Engineering, Bioinformatics and Neural Network. She is a reviewer in several National and International Journals.