A New Cryptographic Algorithm for the Real Time Applications

AHMED H. OMARI AND BASIL M. AL-KASASBEH
Faculty of Information Technology
Applied Science University
Amman 11931
JORDAN
a.omari@asu.edu.jo, b_kasasbeh@asu.edu.jo

RAFA E. AL-QUTAISH AND MOHAMMAD I. MUHAIRAT
Department of Software Engineering
Al-Zaytoonah University of Jordan
Amman 11377
JORDAN
rafa@ieee.org, mohmuhaba14@yahoo.com

Abstract: - Recently, many companies have been deploying Real Time Applications (RTA) over the internet like VoIP, Video Conferencing and other Multimedia services. The need to protect user's data and infrastructures becomes more crucial than ever. Encryption is used to provide the security needed for real time applications. Since RTA contain high volume of data, classical encryption techniques are not appropriate, because most of RTA are served via Internet, the encryption and decryption techniques has to take minimal time to achieve acceptable end-to-end delay. In this paper, a new cryptographic algorithm is developed to improve the time for encryption and decryption of data of end-to-end delay and provide higher level of security.

Key-Words: - Real Time Applications security, RTA, Cryptography, Security, Encryption, Decryption.

1 Introduction

The cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication [1], also it means hidden writing, and it refers to the practice of using encryption to conceal text [2]. So changing the original data to a secret message is called Encryption, while Decryption is the reverse. The process of encryption and decryption of the data is based on a mathematical procedure called the Algorithm.

The major security goals that are of concern to the Cryptography are [3]:

1. **Confidentiality:** it is the function of allowing only authorized users to access the information.
2. **Authentication:** it is the function of the receiver verifying the sender and trust that the sender is actually who he claims to be.
3. **Integrity:** it means that the receiver should be able to trust that the message has not been altered during transmission.
4. **Non-repudiation:** it is the inability of the sender or receiver to deny that the message has been sent or received.
5. **Access control:** it restricts the availability of encrypted information.

These five elements has been the focus of the well-configured cryptography systems.

Furthermore, the cryptography is divided into two main categories, the first and the most common category is called *classical cryptosystems encryption algorithms* (also called *single-key* or *symmetric*) which uses a single shared key to encrypt and decrypt a message [3].

The purpose of the *symmetric algorithms* is to decrypt the cipher-text, compared to hashing algorithms where they never intended to decrypt the information that is why this is also called *Private Key Cryptography*. The most common Algorithm within this category is called *Data Encryption Standard (DES)*, it uses a key length of *56-bit* and it can be implemented in hardware and software by executing the algorithm *16* times, but the key is considered un-secure because of the length [4]. The *3-DES* is an improvement over the *DES* because it employs the executing of the algorithm *3* times and it uses a key of *192-bit* long (the effective security it provides is only *112-bit* long), which makes the computation time too long and not suitable for RTA [5].

Advanced Encryption Standard (AES) is another algorithm of the first category which provides much higher security level than *DES* and perform it in *3* to *10* less computational power than *3-DES* [6].

The *AES* algorithm in every round performs four steps (bytes substitution, shift rows, mix columns, and add round key) on each block of *128-bit* plain text, if the *128-bit* key is used it performs *9* rounds, if *192-bit* key is used it performs 11 rounds and if a *256-bit* key used it performs *13* rounds, which makes this protocol ideal for *RTA* encryption/decryption like voice and signaling [7] [8] [9].

The primary weaknesses in the *symmetric encryption algorithms* are keeping the single shared key secure, and key distribution which yields another approach to cryptography called Asymmetric Encryption or Public Key cryptosystem, which represents the second category of cryptosystems [6].

The second category is called *asymmetric cryptosystem algorithms* which uses two keys instead of one [10]. One of the keys is used to encrypt the message and is *called public* key and the second is used to decrypt the message and called *private key*. The two keys are mathematically related. Some of the most commonly *asymmetric algorithms* used are the *RSA* and *Diffie-Hellman*. The *RSA* is an exponentiation cipher which is very popular in business applications. Moreover, the *RSA* is built in operating systems by *Microsoft, Apple, Sun* and *Novell*. It is also found in secure telephones, Ethernet Network cards and on smart cards [11].

In *Diffie-Hellman* algorithm which is based on the discrete logarithm problem, the users share a common secret key and it is an example of asymmetric key exchange protocol [3]; it allows two users to share a secret key securely over the public networks (Internet). Once the key is being shared then both parties can use it to encrypt and decrypt messages using symmetric cryptography. This algorithm is used in IPSec and Secure Shell (SSH) protocols.

Cryptography is a deep mathematical subject, so each encryption scheme has it is roots in mathematics, where RSA based on exponentiations, Diffie-Hellman based on discrete logarithm, some symmetric cryptosystems based on the set theory [3] and some others based on Elliptic Curve which has not been fully tested because it is still new concept [12].

There are many applications of cryptographic algorithms; one of them is called Hashing. Many algorithms have been proposed to implement Hashing, for example One-way Hash is one of them which is widely used in the ATM [1].

Message digest is another application which has different versions and schemes like *MD2*, *MD5* and *SHA-1*, the basic idea is to take a plain text of any length and create a hash of various lengths depends on the version [4]. Hash functions have proven to have weaknesses and should be replaced by more secure methods.

The Internet has worked so far with a best effort traffic model, every packet is treated (forwarded or discarded) equally. This is a very simple and efficient model. Recently many interactive or real-time services have been introduced and the economical importance of the Internet has grown. The IP phones and services based on that technology is threatening the traditional circuit-switched telephone services, especially on long-distance services. Transmitting interactive real-time media is the greatest challenge in packet based networks. The end-to-end delay, the delay variations (jitter), and the packet loss must not exceed some time limits; otherwise, usability of the service degrades badly.

Many companies have been deploying RTA over the internet like VoIP, Video Conferencing and other Multimedia services in recent years. The need to protect users, data and infrastructures becomes more crucial than ever. Encryption is used to provide the security needed for RTA. Since RTA contain high volume of data, classical encryption techniques are not appropriate, because most of RTA are implemented on the Internet, the encryption and decryption techniques has to take minimal time to achieve acceptable end-to-end delays. In this paper, a new cryptographic algorithm is developed to improve the encryption/decryption time end-to-end delay.

The paper is structured as follows: Section 2 discusses the RTA. In Section 3, the new algorithm and an example have been illustrated. Section 4 presents and discusses the results of the example. Finally, Section 5 concludes the paper.

2 Real Time Applications (RTA): An Overview

As voice service providers roll out *Voice-over-IP (VoIP)*, *Instant Messaging (IM)*, and *video conferencing*, the need to protect user's privacy against eavesdropping is becoming a more critical issue.

Except for AES protocol (AES has been chosen to be used in Universal Mobile Telecommunications System UMTS, third generation networks 3G, and mobile networks [13]) all other protocols mentioned above are not suitable for Real Time Applications because of the long delay they impose on packets. For this reason, we need to design an algorithm that have a better delay time even than AES and a better level of security.

One of the major factors that affects delay time and security level is the key length [3], a security key that is *56-bit* long takes less delay but can be easily broken, a key of *192-bit* long needs more processing time and power but provides higher level of security, so they are not suitable for RTA. Another factor that can impact the encryption and decryption delay is complexity of the algorithm [14].

In our algorithm, we claim that the algorithm has a minimum message delay which should be kept to less than *400ms* in RTA [14] and better security level than the standard and known algorithms.

This algorithm has been designed with two major factors in mind, first, time needed for encryption/decryption; where symmetric and asymmetric encryption algorithms like 3-DES, *AES-Rijndael*, and *RSA* [12] takes longer time and are not appropriate for real time applications. Second, the level of security should be high enough so attackers cannot obtain the encryption/decryption key easily.

In our algorithm the key plays the major role in providing higher level of security where; the key is *1024-bit* long, the key is randomly chosen which makes it harder to intercept, a new key is delivered in each packet which makes it very difficult to guess.

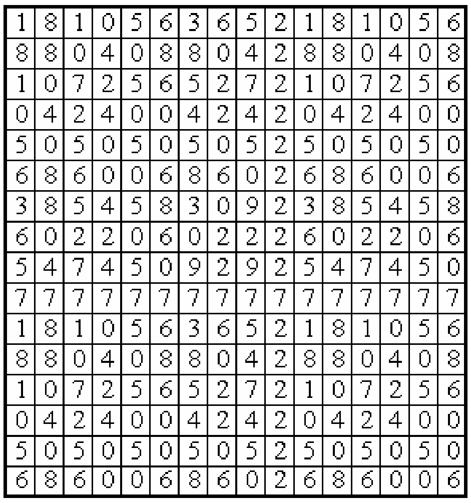
The key is used to randomly generate the indexes of the index table at the sender and the receiver sides. Indexes are not send over the network or in any mean, rather they should be created based on a shared mathematical formula, so the attacker need to guess the *128* entries of the index table in a very short time; which make it very difficult to obtain the key and the attacker will hear noise in best the cases.

3 The Proposed Algorithm and Examples

3.1 The Proposed Algorithm

The proposed algorithm consists of the following steps:

1. **Initiate a Connection:** Connection is initiated by sending an initial packet to the other communicating party, the initial packet depends on the operations performed at the sender side according to a shared mathematical formula, the operations are as follow: constructing an initial random generator table based on a mathematical formula (see Figure 1); the mathematical formula is shared and known previously, the formula is used to initialize the same tables values at the sender and receiver sides. The table size is (*16*16*) rows and columns as shown below, the entries values in the table are ranging from *0-9* only, note that the table could be previously built and shared.
2. **The Key Generation Process:** the sender randomly choose the *1024-bit* key (*128-byte*) of his/her choice
3. **The XoR Encryption Process:** the sender performs an XoR operation over the *1024-bit* key and the data.
4. **Index Generator Table Shifting:** the sender shifts the initial table columns circular right followed by a rows circular down according to a shared values between the sender and the receiver, (these values are exchanged previously by using any known encryption technique for example *RSA algorithm*).



1	8	1	0	5	6	3	6	5	2	1	8	1	0	5	6
8	8	0	4	0	8	8	0	4	2	8	8	0	4	0	8
1	0	7	2	5	6	5	2	7	2	1	0	7	2	5	6
0	4	2	4	0	0	4	2	4	2	0	4	2	4	0	0
5	0	5	0	5	0	5	0	5	2	5	0	5	0	5	0
6	8	6	0	0	6	8	6	0	2	6	8	6	0	0	6
3	8	5	4	5	8	3	0	9	2	3	8	5	4	5	8
6	0	2	2	0	6	0	2	2	2	6	0	2	2	0	6
5	4	7	4	5	0	9	2	9	2	5	4	7	4	5	0
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
1	8	1	0	5	6	3	6	5	2	1	8	1	0	5	6
8	8	0	4	0	8	8	0	4	2	8	8	0	4	0	8
1	0	7	2	5	6	5	2	7	2	1	0	7	2	5	6
0	4	2	4	0	0	4	2	4	2	0	4	2	4	0	0
5	0	5	0	5	0	5	0	5	2	5	0	5	0	5	0
6	8	6	0	0	6	8	6	0	2	6	8	6	0	0	6

Fig.1: An Initial Random Generator Table.

5. **The Index Generation Process:** The sender chooses a large number that represents the (*128* indexes) out of the shifted initial table starting from the (leftmost and top-corner) and continue row wise until the last element of the table (rightmost, down corner); where the first two digits represent the first index, the next two digits represent the second index and so on, for example, from Figure 3 (*67* is the *1st* index, *28* is the *2nd* index, *54* is the *3rd* index, …, and *34* is the *128th* index).
6. **The Key Insertion Process:** the sender inserts and shuffles the key into the XoRed data according to the generated indexes, the resulted output is a collection of padding and mixing of the key and the XoRed data together, this process provides high level of confusion
7. **Sending Packets:** the data is ready to send to the receiver as any other ordinary network packet
8. **The Receivers Process:** Upon receiving the packet, he/she extracts the key back from the shuffled data, the extraction process involves finding the relative position of each single key bit in the data and put it back in it is right position
9. **The Key Extracting Process:** the receiver use the same initial table and same shared values to perform the shifting steps as the sender did, the

resulting output are the key and the XoRed sender data

10. **The Decryption Process:** after getting the key, the received packet is XoRed with the extracted key to get back the original data (decryption)

Now, we can summarize the proposed algorithm to be consisting of the following:

A. At the sender side:

1. Randomly choose the key of *1024-bit* long.
2. Encrypt data by XoRing with the key.
3. Insert the key in the XoRed data by using the indexes generated earlier from the shifted generated table.
4. Send the packet.

B. At the receiver side:

1. Extract the key out of the packet, based on the shifted generated table
2. Decrypt data by XoR with the key.

3.2 Examples

3.2.1 Shifting Process Example

Assume using a small shared value like (*2*, *2*, *0*, *1*, *4*, and *3*), and a small table size (*5*6*); the following two steps are required:

1. Take the first entry (*digit 2*) and perform a table circular right shift.
2. Take the second entry (*digit 2*) and perform a table circular down shift.
3. If the value is *zero* perform no shift.
4. Repeat steps 1 and 2 until completing all entries (input numbers), see Figure 2.

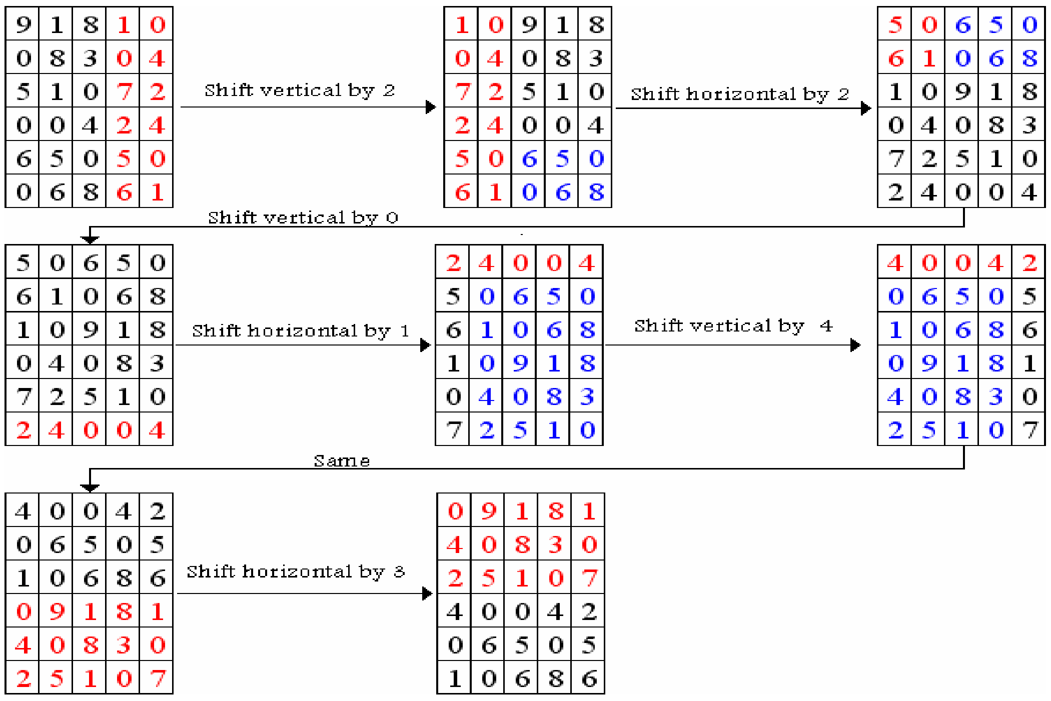
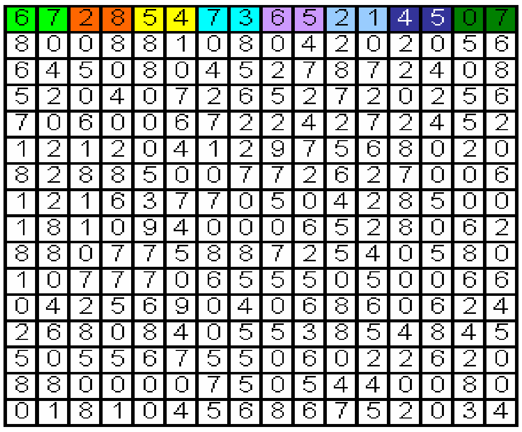


Fig.2: Shifting the Table Right Circular and Down Circular by 2, 2, 0, 1, 4, and 3.

3.2.2 Key Insertion Example

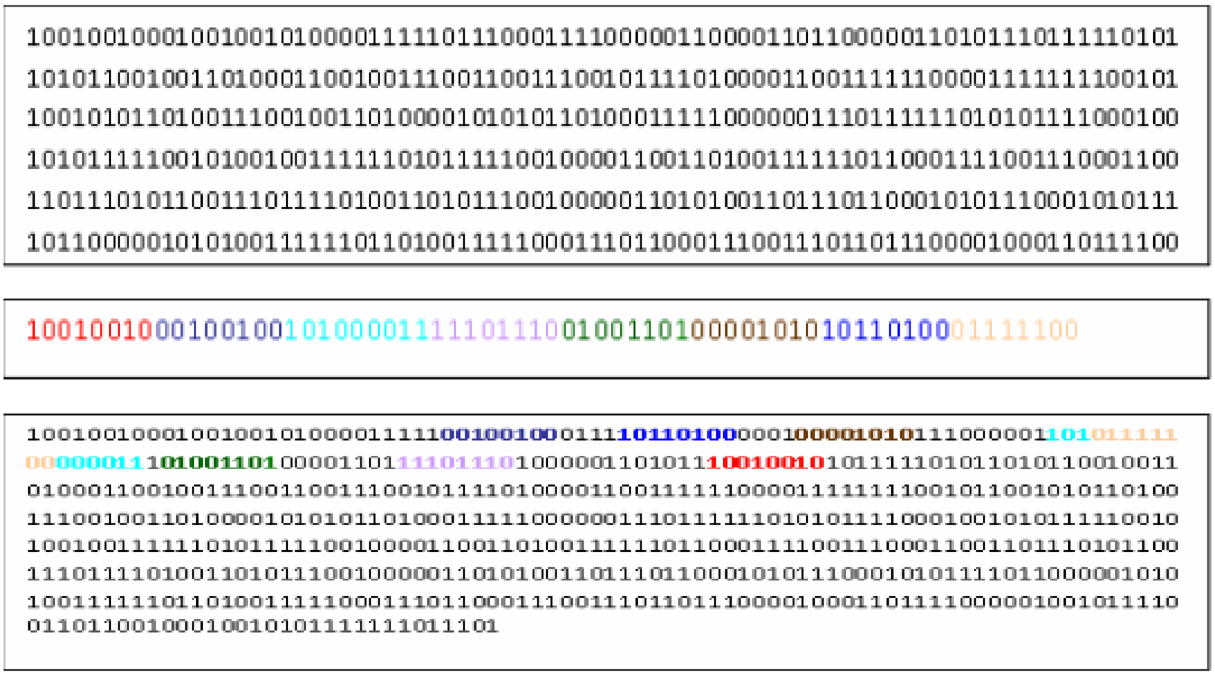
From Figure 3, we notice that the indexes are: *67*, *28*, *54*, *73*, *65*, *21*, *45*, *07*, *80*, …, and *34* (which is the last index of the table), these *128* indexes determine the exact insertion positions of the key bytes. The key insertion process (step 3) depends on the shared values between the sender and the receiver, the algorithm inserts each *8-bit* of the key into the specific position in the table based on the index value (see Figure 4).

Figure 4 shows the insertion of the key in its places within the data. To illustrate the proposed algorithm in this example, we took only the first *8* indexes, that is, *67*, *28*, *54*, *73*, *65*, *21*, *45*, and *07*.



6	7	2	8	5	4	7	3	6	5	2	1	4	5	0	7
8	0	0	8	8	1	0	8	0	4	2	0	2	0	5	6
6	4	5	0	8	0	4	5	2	7	8	7	2	4	0	8
5	2	0	4	0	7	2	6	5	2	7	2	0	2	5	6
7	0	6	0	0	6	7	2	2	4	2	7	2	4	5	2
1	2	1	2	0	4	1	2	9	7	5	6	8	0	2	0
8	2	8	8	5	0	0	7	7	2	6	2	7	0	0	6
1	2	1	6	3	7	7	0	5	0	4	2	8	5	0	0
1	8	1	0	9	4	0	0	0	6	5	2	8	0	6	2
8	8	0	7	7	5	8	8	7	2	5	4	0	5	8	0
1	0	7	7	7	0	6	5	5	5	0	5	0	0	6	6
0	4	2	5	6	9	0	4	0	6	8	6	0	6	2	4
2	6	8	0	8	4	0	5	5	3	8	5	4	8	4	5
5	0	5	5	6	7	5	5	0	6	0	2	2	6	2	0
8	8	0	0	0	0	7	5	0	5	4	4	0	0	8	0
0	1	8	1	0	4	5	6	8	6	7	5	2	0	3	4

Fig.3: The Shifted Table.



```
100100100010010010100001111101110001111000001100001101100000110101110111110101
101011001001101000110010011100110011100101111010000110011111100001111111100101
100101011010011100100110100001010101101000111110000001110111111010101111000100
101011111001010010011111101011111001000011001101001111110110001111001110001100
110111010110011101111010011010111001000001101010011011101100010101110001010111
101100000101010011111101101001111100011101100011100111011011100001000110111100
```

```
1001001000100100101000011110111001001101000010101011010001111100
```

```
100100100010010010100000111100100100011110110100000100001010111000000110101111 1
000000011101001101000011011110111010000001101011100100101011111010110101100100 11
010001100100111001100111001011110100001100111111000011111111001011001010110100
111001001101000010101011010001111100000011101111110101011110001001010111110010
100100111111010111111001000011001101001111110110001111001110001100110111010110 0
111011110100110101110010000011010100110111011000101011100010101111011000001010
100111111011010011111000111011000111001110110111000010001101111000000100101110
011011001000100101011111111011101
```

Fig.4: The Key Insertion Table.

4 Results and Discussion

According to the output results of our algorithm as shown in Table 1, the proposed encryption / decryption algorithm is much better than all other known techniques and provides a better security level for the Real Time Applications (RTA). It shows our experimental results. It shows that all of the encryption processing time is much better and it is less than 1-ms in the worst case, in the decryption processing time we note that at some cases it takes more than *9-ms*, this time is considered relatively high with respect to the encryption time, still this time is acceptable with comparison to *AES-Rijndael algorithm*.

Table 1: The Encryption and Decryption Times.

	Data Size (Byte)	Key (Bit)	Time (MS)
Encryption	1500	1024	0.88125
	1024	1024	0.79075
	1024	512	0.74075
	1024	256	0.71575
Decryption	1500	1024	9.355
	1024	1024	9.183125
	1024	512	3.7725
	1024	256	1.958125

A comparison between AES-Rijndael and our proposed algorithm is illustrated in Table 2. However, this comparison uses the following information:
Data Size = 1024-byte, and
Key = 256-bit.

From Table 2, we can note that our new algorithm achieves best results, where it is *15* times faster than *AES* encryption and *6* times faster than *AES* decryption.

The proposed algorithm is resistant against brute force attacks, where the key is mixed and shuffled strongly inside the XoRed data; it will be very difficult to guess the key.

Table 2: A Comparison between AES-Rijndael and the New Algorithm.

Security Algorithm	AES-Rijndael	Our
Encryption (MS)	10.884	0.71575
Decryption (MS)	10.718	1.958125

5 Conclusion and Future Work

Most of the available encryption/decryption techniques are not perfect for RTA over the Internet since they were originally built for text data, and due to their extensive computations which result in an unacceptable delay and processing time.

The work in this paper attempts to develop a new encryption/decryption approach which adds a minimum delay time that makes it appropriate for RTA. In addition, it provides high level of security by choosing a key length of *1024-bit* long, another interesting property of the algorithm is the ability of using a new different key for each packet. The distribution of the encryption keys is usually carried out through a trusted agency; this results in a significant delay before the real time application starts. Furthermore, this work attempts to provide a new method of key exchange without an intermediate party.

Furthermore, the following points are recommended in order to enhance the proposed algorithm:

1. Maximizing the table size for more than *16*16*.
2. Maximizing the table indexes from *128* digits to a larger size.
3. Enhance the security level by using more than one decimal digit to rotate and shift the table.
4. Use RSA to share the initial shared value.

Acknowledgments
This paper was supported by the Applied Science Private University, Amman, Jordan.

References:

[1] E. Cole, R. Krutz and J. W. Conley, *Network Security Bible*, Wiley Publishing Inc, 2005.

[2] A. Menezes, V. Oosrschot and A. Vanstone, *Handbook on Applied Cryptography*, CRC Press Inc., NY, USA, 2000.

[3] D. Stinson, *Cryptography Theory and Practice*, CRC Press Inc., NY, USA, 1995.

[4] G. Blelloch, Introduction to Cryptography, online: *http://www-2.cs.cmu.edu/afs/cs/project/pscicoguyb/realworld/crypto.ps*, 2000, accessed on Sept. 1, 2008.

[5] G. Carter, E. Dawsony and L. Nielseny, Key Schedule Classification of the AES Candidates, in *Proceedings of the end AES Conference*, Rome, Italy, 1999.

[6] J. Dray, Report on the NIST Java AES Candidate Algorithm Analysis, online: *http://csrc.nist.gov/encryption/aes/round/r1-java.pdf*, 1999, accessed on Sept. 1, 2008.

[7] J. Dray, NIST Performance Analysis of the Field Round Java AES Candidates, online: *http://csrc.nist.gov/encryption/aes/roubd2/conf2/papers/8-jdray.pdf*, 2000, accessed on Sept. 1, 2008.

[8] J. Nakahara, B. Preneel and J. Vandewalle, *Square Attack on Extended Rijndael Block Copher*, COSIC Technology Report, 2002.

[9] D. Baudran, H. Gilbert, L. Granboulan, H. Handschun, A. Joux, P. Nguyae, F. Noilhan, O. Poincheva, T. Pornin, G. Poupard, J. Stern and S. Vaudenay, Report on the AES Candidates, in *Proceedings of the 2nd ASE Conference*, Rome, Italy, 1999.

[10] T. Verhoeff, Encryptography, online: *http://www pa.win.tue.nl/wstomv/software/AESRijndael/rijndael-test.pas*, 2001, accessed on Sept. 1, 2008.

[11] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*, 3rd ed., Prentice-Hall, 2003.

[12] W. Stallings, *Cryptography and Network Security*, 4th ed., Prentice-Hall, 2005

[13] B. A. Forouzan, *Data Communications and Networking*, 4th ed., McGraw-Hill, , 2007

[14] Wikipedia Website, online: *http://en.wikipedia.org*, accessed on Nov., 13, 2008.