INTERNATIONAL JOURNAL OF COMPUTERS
Issue 1, Volume 3, 2009

DEA-RTA: A Dynamic Encryption Algorithm for the Real-Time Applications

Ahmad H. Omari, Basil M. Al-Kasasbeh, Rafa E. Al-Qutaish and Mohammad I. Muhairat

Abstract**—** The Internet and it is applications are hungry for high level of Quality of Service (QoS), and most of the Internet applications seek to minimize packets delay, especially, the Real-Time Applications (RTA). QoS is considered as a major issue in the Internet, where RTA services like IPTelephony and XoIP become a successful business in the world, call distribution may result in big money loss, for this reason researchers put their efforts to build applications that can deal with different levels of QoS. In addition to the basic QoS some customers ask to preserve confidentiality which makes it more complicated and may result in higher delay time.

Delay is very complex issue specially in RTA, and it consists of many types of delays, such as, Packetization delay (sampling, coder-decoder (codec),compression and encryption), and end-to-end delay (processing, queuing, serialization and propagation delays), our research try to achieve better encryption delay at the user machine CPU level while maintain confidentiality. The proposed algorithm is a new symmetric encryption technique that allows users to choose using new different key for each single packet if they wish. The encryption key is flexible in length, the plain text is flexible in size, the encryption process is very simple, the transposition table is simple too, the shifted transposition table is easy to initiate and complex to regenerate. These properties results in better encryption delay while maintaining confidentiality, the algorithm is 15 times faster and 10 times faster than AES algorithm.

Keywords**—** Cryptography, DEA-RTA, decryption, encryption, QoS, real-time applications, security.

I. Introduction

The Real-Time system is defined in Wikipedia as "*the system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed*" [14]. According to the definition, the time is the main key factor in real-time systems, any application that complete its work after the deadline is considered useless. Real-Time Streaming Applications (RTSA) like XoIP, VoIP, IPTelephony, IPTV, Video on Demand, Video Conferencing and others have been grown dramatically over the last years. There are several organizations, business, workgroups, research bodies, institutions and researchers working on different protocols to support and enhance such applications, IETF, IEEE, ITU-T and others have been identify specialized categories of protocols like RTP, RTCP, SCTP, SRTP and SRTCP to help managing and providing RTSA services..

Security is one of the most important RTSA properties, the underlying real-time protocol like RTP which is often used to transferring RTA data requires minimal delay and jitter, and with huge transmission rates even small timing overhead easily amounts to huge loss of bandwidth [16].

Encryption, especially using DES-CBC causes minor overhead in real-time applications compared to CPU requirements of sampling, digitization and compression [16]. Although encryption overhead is minor, fast encryption / decryption algorithm may result in less delay, which in turns enhances the overall Quality of Service QoS [19, 20, 21].

The RTA delay is generally defined as the time it takes from the instant a data stream is formulated (i.e. you speak into the phone) until the instant that stream is regenerated on the other end.

In time-critical applications, it is important to account for the delay components in the network. Overall RTA quality is a function of many factors that include the compression algorithm, errors and frame loss, echo cancellation, and delay [17]. Delay issues in RTA account for encoding, packetization, and internetworking delays. Delay is part of the QoS from the network perspective where Reliability, Delay, Jitter and Bandwidth are the internetworking QoS components [13]. Packetization is an important contributor to delay, in particular in RTA such as Voice over IP in order to keep packetization delay manageable, packet voice systems must use a minimum sampling rate, modern encoding techniques have been employed and fast encryption/decryption techniques is highly required.

The paper is organized as follows: Section 2 presents an overview of the real-time applications (RTA). Section 3 discusses the security in the real-time applications. In Section 4, Dynamic Encryption Algorithm for Real-Time Applications (DEA-RTA) has been explained with an example. Section 5 presents and discusses the results of the example. Finally,

A. H. Omari is with the Computer Networks Systems Department, Applied Science University, Amman 11931, Jordan (e-mail: a.omari@asu.edu.jo).

B. M. Al-Kasasbeh is with the Computer Networks Systems Department, Applied Science University, Amman 11931, Jordan (e-mail: b_kasasbeh@asu.edu.jo).

R. E. Al-Qutaish is with the Software Engineering Department, Alzaytoonah University of Jordan, P.O. Box: 130, Amman, 11733, Jordan (corresponding author: phone: +962-77-981-1321; e-mail: alqutaish@alzaytoonah.edu.jo, rafa@ieee.org).

M. I. Muhairatis with the Software Engineering Department, Alzaytoonah University of Jordan, P.O. Box: 130, Amman, 11733, Jordan (e-mail: mohmuhaba14@yahoo.com).

Section 6 concludes the paper.

II. 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 [15].

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.

III. The Security in Real-Time Applications

A. General Security considerations

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 secrete writing, and it refers to the practice of using encryption to hide 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.

According to the ITU-T X.800 Security goals (of OSI), and that are of concern to the Cryptography are [12]:

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.

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.

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.

B. *RTA and the Security*

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 Internet provides little assurance of privacy or confidentiality. The use of firewalls and encryption can help mitigate the risks. Security and privacy become mandatory requirements for most of the RTSA applications; for instance IP telephony requires security services such as confidentiality, integrity, authentication, anti-replay and non-repudiation. The available solutions are generic and do not respect voice specificities and constraints [18].

RTSA depends on the underlying IP protocols to provide security services, where the SRTP (Secure Real-time Transport Protocol) is a security of RTP (Real-Time Protocol) standard [18]

SRTP uses a key-stream approach to encipher the flow of information. Key-stream generation is accomplished by the AES [18]. In addition, SRTP use AES-f8 mode or AES-Counter mode as a standard encryption technique.

IV. The DEA-RTA Algorithm

A. *DEA-RTA Description and Components*

The Dynamic Encryption Algorithm for Real-Time Applications DEA-RTA consists of the following components:

1) The Index Generation Process

1.1. Initiate Table: The initial table size is (16*16) rows and columns as shown in figure-, the table entries values are ranging from 0-F (16). Both, the sender and the receiver should have the same copy of the initial table [15], the initial table is not secret it could be announced to the public, see Fig. 1.

9	3	E	4	1	C	6	E	2	0	9	1	1	B	9	B
3	6	B	C	7	C	E	9	4	E	D	C	6	7	7	E
3	2	D	8	3	B	B	6	F	3	A	D	9	8	5	F
D	4	B	C	6	5	5	B	3	D	9	A	C	B	E	2
0	A	0	E	0	8	6	D	2	9	2	6	E	D	9	C
D	F	4	2	4	0	6	0	B	4	0	7	2	B	1	2
6	2	1	D	1	B	F	A	0	A	9	2	9	2	D	9
C	0	E	5	9	5	9	0	2	A	9	C	3	F	7	6
F	4	6	E	D	C	E	6	3	F	7	9	1	F	9	F
1	D	1	E	D	C	6	2	2	6	D	0	4	F	A	7
2	7	8	9	C	3	B	9	2	E	3	4	E	C	4	5
4	F	4	4	9	E	5	F	2	2	4	C	F	3	0	D
8	3	F	5	6	9	D	A	5	0	7	B	8	1	7	4
E	F	0	0	4	1	D	E	B	5	5	7	B	3	2	F
8	0	2	A	4	4	E	1	5	E	9	5	8	6	A	4
6	3	E	9	6	F	7	8	4	7	A	A	0	7	8	6

Fig. 1 the initial table

1.2. Shared value: The shared value is randomly generated or selected by the user (see Fig. 2), the size of this value is not specified, it is recommended use a value that is not less than 10 digits size. The shared value is the most important component of the system, so it must be kept and exchanged securely; for this reason the Diffie-Hellman or any other key exchange scheme is proposed [15]. The shared value is not fixed during the communication session, the sender or the receiver may choose to change this value at any time during the communication, accordingly the other communicating party must be informed. The system could be configured to change the shared value either manual (i.e. by the user) or periodically according to some statistical information (i.e. size of the sent/received data, time-slice, randomly or any other criteria).

1.3. Circular Shifts: The circular left shift and circular down shift is shown in Fig. 7 later.

4	B	2	8	A	9	8	6

Fig. 2 shared Value

1.4. Table of Indexes: The table of indexes generated after performing left and down circular shifts respectively according to the values the extracted from the shared value (known as Shdv). Shdv is a two digits length extracted out from the shared value, then modulo operation is used to perform the shift operations (shifted rows/columns = Shdv Mod 16). The output table is called the Table of Indexes, as in Fig. 3.

F	3	A	D	9	8	5	F	3	2	D	8	3	B	B	6
3	D	9	A	C	B	E	2	D	4	B	C	6	5	5	B
2	9	2	6	E	D	9	C	0	A	0	E	0	8	6	D
B	4	0	7	2	B	1	2	D	F	4	2	4	0	6	0
0	A	9	2	9	2	D	9	6	2	1	D	1	B	F	A
2	A	9	C	3	F	7	6	C	0	E	5	9	5	9	0
3	F	7	9	1	F	9	F	F	4	6	E	D	C	E	6
2	6	D	0	4	F	A	7	1	D	1	E	D	C	6	2
2	E	3	4	E	C	4	5	2	7	8	9	C	3	B	9
2	2	4	C	F	3	0	D	4	F	4	4	9	E	5	F
5	0	7	B	8	1	7	4	8	3	F	5	6	9	D	A
B	5	5	7	B	3	2	F	E	F	0	0	4	1	D	E
5	E	9	5	8	6	A	4	8	0	2	A	4	4	E	1
4	7	A	A	0	7	8	6	6	3	E	9	6	F	7	8
2	0	9	1	1	B	9	B	9	3	E	4	1	C	6	E
4	E	D	C	6	7	7	E	3	6	B	C	7	C	E	9

Fig. 3 table of indexes

1.5. The Extracted Indexes: The table of extracted indexed is generated from the Table of Indexes (Fig. 4), where it is possible to choose the first octet as the first index and the second octet as the second index and so on, in this case we will have 128 indexes. In case we use an initial table of more than (16*16) we could pick the indexes in consistent way with the table size (i.e. we could use the first two octets, octet number one and octet number two as the first index and the octets three and four as the second index and so on).

F 3A D 9 8 5 F 3 2 D 8 . . . B C 7 C E 9

Fig. 4 the extracted indexes

2) The Key Insertion Process

2.1. The Plain Text Data: The plain text data is the data to be sent; this data could be of any data type and format (i.e. text, audio, video … etc.) The plain text data has no restriction on size, but it is recommended to fit on one packet size minus the key size of the Maximum Transfer Unit (MTU); (Plain-Text-Size = (MTU)-(Key-size)).

2.2. The Key Generation: The system key is randomly generated or could be selected by the user, the key could be of variable length size, initially it is 1024-bit size used, the key size can be expanded to 2048 bits or even longer than that. Since the key is dynamic the system can be configured to different keys periodically during the communication session, the users can change the key very frequently and the encryption/decryption speed and operations will not affected, where the key is sent in each single packet.

2.3. XoR (Encryption) Process: The plain-text-data and the key are XoRed, the first 1024 bits of the data is XoRed with the key, the second 1024 bits of the data is XoRed with the same key and so on until no more plain-text-data found. If the plain-text-data is longer or smaller than the key the only matching lengths will be XoRed, this is the reason why we choose a key of variable length size (greater than or less than the plain-text-size).

2.4. The Key Insertion: The key is inserted in the XoRed table generated from step 2.3 above, the insertion process is performed according to the Extracted Indexes (as explained in step 1.4), the first octet of the key will be inserted in the XoRed table according to the value of the first Index value and the second key octet will be inserted according to the second Index value and so on until the end of the key(Figure-6 explain the process), where the insertion is performed at the appropriate location as indicted on the index value (i.e. (F3)16 → (243)10 the first index value in Hex. Converted to decimal value). For more complexity we could insert the first key octet as depending on the first three index octets or the first four index octets. This process adds more complexity and flexibility to the encryption process with no operation cost and makes it harder to the cryptanalysis.

B. The DEA-RTA Architecture

The DEA-RTA components together composed the algorithm architecture, when building the algorithm simplicity was one of the main design goals, maintaining simplicity while preserving high degree of secrecy seems to be hard to achieve [15]. Fig. 5 below shows the detailed architecture of the encryption process. It shows the simplicity integration of different algorithm components. The algorithm is very simple to use and at the same time it is very complex to attack. The architecture is divided into two main components glued together to formulate the detailed algorithm architecture:

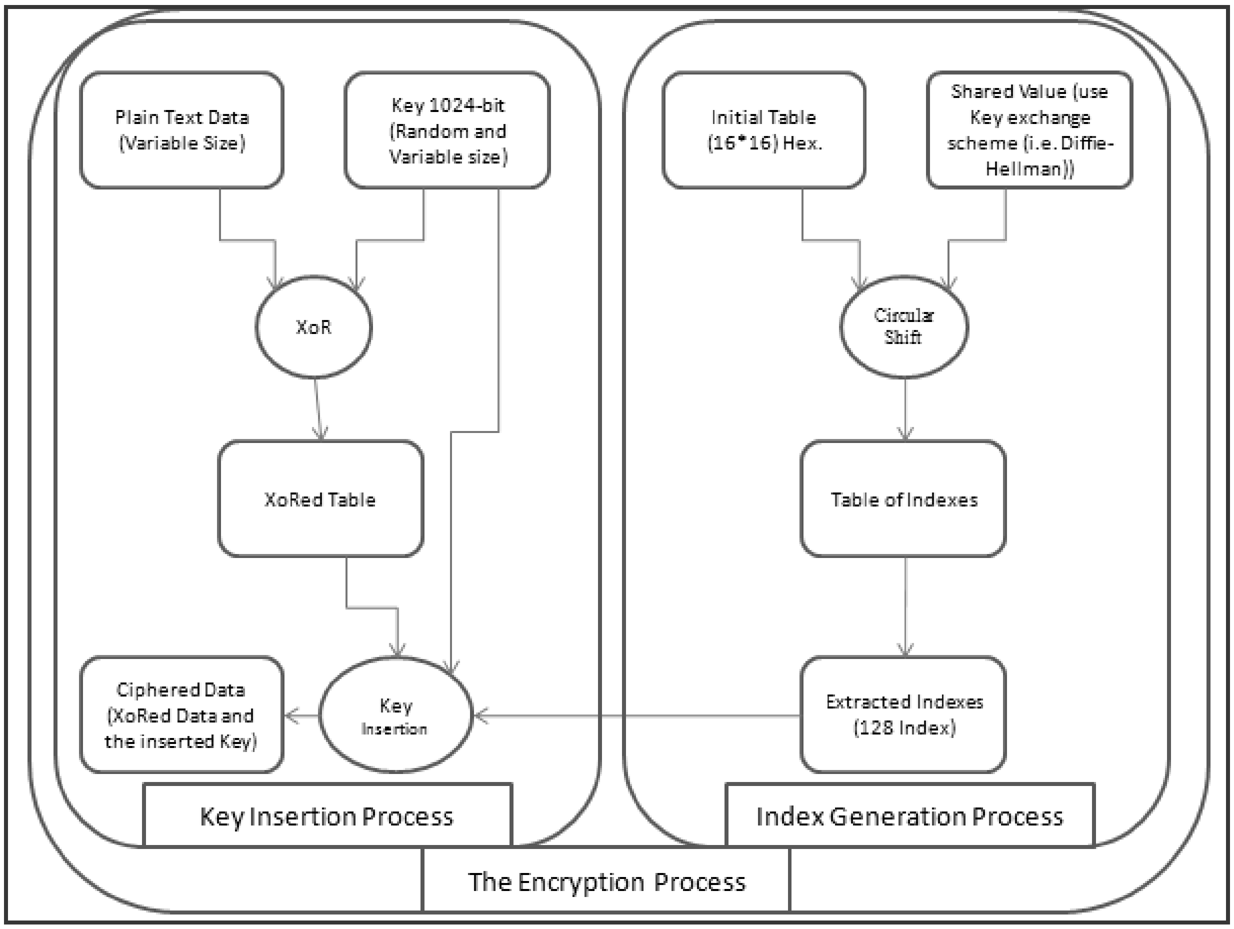
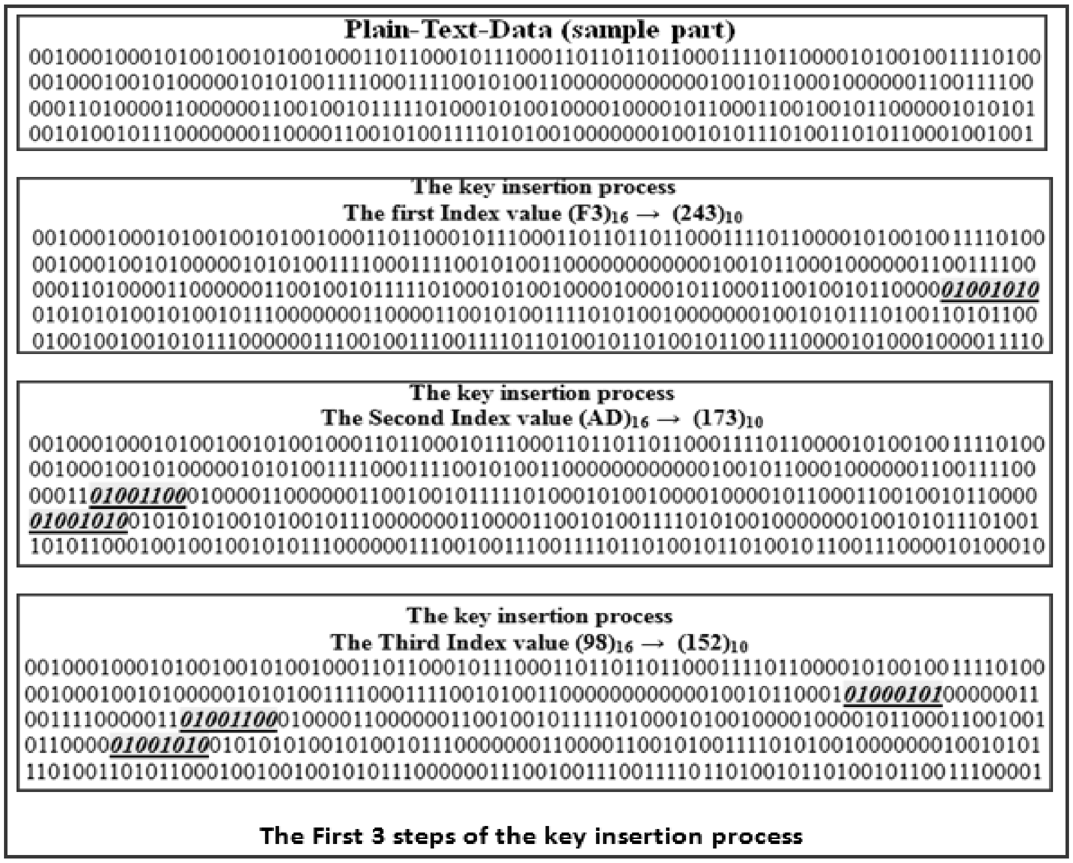

Plain Text Data (Variable Size)
Key 1024-bit (Random and Variable size)
XoR
XoRed Table
Ciphered Data (XoRed Data and the inserted Key)
Key Insertion
Key Insertion Process
Initial Table (16*16) Hex.
Shared Value (use Key exchange scheme (i.e. Diffie-Hellman))
Circular Shift
Table of Indexes
Extracted Indexes (128 Index)
Index Generation Process
The Encryption Process

Fig. 5 the encryption process

1) The Encryption process: this process consists of two main parts:
   1.1. The Index Generation part: The index generation process is performed at the beginning of the encryption/decryption process and repeated when there is a need to change the shared value, this means that step is less operated than the other step. The output of the Index Generator step is the Extracted Index. The Extracted Index resulted from performing a regular left-circular column shift followed by a down-circular column shift according to the Index values, Fig. 6 shows the first three left/down circular shifts, and Fig. 7 shows the circular left/down shift process.
   The Extracted Indexes are taken out of the final result after performing the left/down circular shifts; the Indexes are 128 different indexes, these indexes will remain unchanged until the shared value is changed, changing the shared value depends on many factors for example to change the shared value very frequent will add more complexity over the plain-text and makes it harder to the attacker to guess the key, at the same time this may affect the performance by adding more delay to the packet.
   1.2. The Key Insertion Part: The key insertion process is the most used step in this algorithm, the flexibility of changing the key and use a variable key length size is the corner stone in the algorithm strength, the user may stick to a single key during the communication process and change the shared value or stick to the same shared value and change the key, the result is the same, but it is recommended to change both of the shared value and the key, this will make it more difficult to cryptanalysis either to guess or extract the key. The XoR operations used to generate the XoRed table doesn't consume the machine resources and takes less operations, the XoR operation is very fast to perform and more enhancements like S-Boxes may be used to add more complexity and gives confusion and diffusion to the algorithm. Fig. 6 illustrates this process.



Plain-Text-Data (sample part)

0010001000101001001010010001101100010111000110110110110001111011000010100100111110100
001000100101000001010100111100011110010100110000000000000100101100010000001100111100
000110100001100000011001001011111010001010010000100001011000110010010110000 01010101
00101001011100000000110000110010100111101010010000000100101011101001101011000100 1001

The key insertion process
The first Index value $(F3)_{16} \rightarrow (243)_{10}$

0010001000101001001010010001101100010111000110110110110001111011000010100100111110100
001000100101000001010100111100011110010100110000000000000100101100010000001100111100
000110100001100000011001001011111010001010010000100001011000110010010110000***01001010***
0101010100101001011100000000110000110010100111101010010000000100101011101001101011 00
01001001001010111000000111001001110011110110100101101001011001110000101000100001111 0

The key insertion process
The Second Index value $(AD)_{16} \rightarrow (173)_{10}$

0010001000101001001010010001101100010111000110110110110001111011000010100100111110100
001000100101000001010100111100011110010100110000000000000100101100010000001100111100
00011***01001100***0100001100000011001001011111010001010010000100001011000110010010110000
01001010010101010010100101110000000011000011001010011110101001000000010010101110100 1
10101100010010010010101110000001110010011100111101101001011010010110011100001010001 0

The key insertion process
The Third Index value $(98)_{16} \rightarrow (152)_{10}$

0010001000101001001010010001101100010111000110110110110001111011000010100100111110100
00100010010100000101010011110001111001010011000000000000010010110001***01000101***00000011
001111000001 1***01001100***0100001100000011001001011111010001010010000100001011000110 01001
0110000***01001010***0101010100101001011100000000110000110010100111101010010000000100 10101
110100110101100010010010010101110000001110010011100111101101001011010010110011100001

The First 3 steps of the key insertion process

Fig. 6 the key insertion process

2) The Decryption Process: This process consists of two main parts:
   2.1. The Index Generation part: The index generation process is performed exactly as in the encryption part explained above (see Fig. 6). Since the process is identical at the sender and the receiver sides the explanation above is enough.
   2.2. The Key Extraction Process: The key extraction is the

reverse of the key insertion process; the extraction starts from the last index value (the 128th index) and ends with the first index value (the 1st index) in reverse order. When pointing to the key position the algorithm will extract the next one octet, the first extraction part represents the last key octet and so on until the whole key recovered back from the XoRed table.

2.3. The Decryption Process: The decryption process is easy and straight forward, just perform an XoR operation over the received data and the recovered key, the output is exactly the original plain-text-data after get decrypted. See Fig. 7.

9	3	E	4	1	C	6	E	2	0	9	1	1	B	9	B
3	6	B	C	7	C	E	9	4	E	D	C	6	7	7	E
3	2	D	8	3	B	B	6	F	3	A	D	9	8	5	F
D	4	B	C	6	5	5	B	3	D	9	A	C	B	E	2
0	A	0	E	0	8	6	D	2	9	2	6	E	D	9	C
D	F	4	2	4	0	6	0	B	4	0	7	2	B	1	2
6	2	1	D	1	B	F	A	0	A	9	2	9	2	D	9
C	0	E	5	9	5	9	0	2	A	9	C	3	F	7	6
F	4	6	E	D	C	E	6	3	F	7	9	1	F	9	F
1	D	1	E	D	C	6	2	2	6	D	0	4	F	A	7
2	7	8	9	C	3	B	9	2	E	3	4	E	C	4	5
4	F	4	4	9	E	5	F	2	2	4	C	F	3	0	D
8	3	F	5	6	9	D	A	5	0	7	B	8	1	7	4
E	F	0	0	4	1	D	E	B	5	5	7	B	3	2	F
8	0	2	A	4	4	E	1	5	E	9	5	8	6	A	4
6	3	E	9	6	F	7	8	4	7	A	A	0	7	8	6

(4 columns circular left shift) 11 2 8 10 9 8 6

1	C	6	E	2	0	9	1	1	B	9	B	**9**	3	E	4
7	C	E	9	4	E	D	C	6	7	7	E	**3**	6	B	C
3	B	B	6	F	3	A	D	9	8	5	F	**3**	2	D	8
6	5	5	B	3	D	9	A	C	B	E	2	**D**	4	B	C
0	8	6	D	2	9	2	6	E	D	9	C	**0**	A	0	E
4	0	6	0	B	4	0	7	2	B	1	2	**D**	F	4	2
1	B	F	A	0	A	9	2	9	2	D	9	**6**	2	1	D
9	5	9	0	2	A	9	C	3	F	7	6	**C**	0	E	5
D	C	E	6	3	F	7	9	1	F	9	F	**F**	4	6	E
D	C	6	2	2	6	D	0	4	F	A	7	**1**	D	1	E
C	3	B	9	2	E	3	4	E	C	4	5	**2**	7	8	9
9	E	5	F	2	2	4	C	F	3	0	D	**4**	F	4	4
6	9	D	A	5	0	7	B	8	1	7	4	**8**	3	F	5
4	1	D	E	B	5	5	7	B	3	2	F	**E**	F	0	0
4	4	E	1	5	E	9	5	8	6	A	4	**8**	0	2	A
6	**F**	**7**	**8**	**4**	**7**	**A**	**A**	**0**	**7**	**8**	**6**	**6**	**3**	**E**	**9**

4 (11 rows down shift) 2 8 10 9 8 6

4	0	6	0	B	4	0	7	2	B	1	2	**D**	F	4	2
1	B	F	A	0	A	9	2	9	2	D	9	**6**	2	1	D
9	5	9	0	2	A	9	C	3	F	7	6	**C**	0	E	5
D	C	E	6	3	F	7	9	1	F	9	F	**F**	4	6	E
D	C	6	2	2	6	D	0	4	F	A	7	**1**	D	1	E
C	3	B	9	2	E	3	4	E	C	4	5	**2**	7	8	9
9	E	5	F	2	2	4	C	F	3	0	D	**4**	F	4	4
6	9	D	A	5	0	7	B	8	1	7	4	**8**	3	F	5
4	1	D	E	B	5	5	7	B	3	2	F	**E**	F	0	0
4	4	E	1	5	E	9	5	8	6	A	4	**8**	0	2	A
6	**F**	**7**	**8**	**4**	**7**	**A**	**A**	**0**	**7**	**8**	**6**	**6**	**3**	**E**	**9**
1	C	6	E	2	0	9	1	1	B	9	B	**9**	3	E	4
7	C	E	9	4	E	D	C	6	7	7	E	**3**	6	B	C
3	B	B	6	F	3	A	D	9	8	5	F	**3**	2	D	8
6	5	5	B	3	D	9	A	C	B	E	2	**D**	4	B	C
0	8	6	D	2	9	2	6	E	D	9	C	**0**	A	0	E

4 11 (2 columns left shift) 8 10 9 8 6

6	0	B	4	0	7	2	B	1	2	**D**	F	4	2	**4**	0
F	A	0	A	9	2	9	2	D	9	**6**	2	1	D	**1**	B
9	0	2	A	9	C	3	F	7	6	**C**	0	E	5	**9**	5
E	6	3	F	7	9	1	F	9	F	**F**	4	6	E	**D**	C
6	2	2	6	D	0	4	F	A	7	**1**	D	1	E	**D**	C
B	9	2	E	3	4	E	C	4	5	**2**	7	8	9	**C**	3
5	F	2	2	4	C	F	3	0	D	**4**	F	4	4	**9**	E
D	A	5	0	7	B	8	1	7	4	**8**	3	F	5	**6**	9
D	E	B	5	5	7	B	3	2	F	**E**	F	0	0	**4**	1
E	1	5	E	9	5	8	6	A	4	**8**	0	2	A	**4**	4
7	8	4	7	A	A	0	7	8	6	6	3	E	9	**6**	F
6	E	2	0	9	1	1	B	9	B	**9**	3	E	4	**1**	C
E	9	4	E	D	C	6	7	7	E	**3**	6	B	C	**7**	C
B	6	F	3	A	D	9	8	5	F	**3**	2	D	8	**3**	B
5	B	3	D	9	A	C	B	E	2	**D**	4	B	C	**6**	5
6	D	2	9	2	6	E	D	9	C	**0**	A	0	E	**0**	8

Fig. 7 the circular left/down shift

V. 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	DEA-RTA
Encryption (MS)	10.884	0.71575
Decryption (MS)	10.718	1.958125

VI. 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.

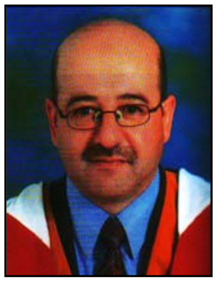
Furthermore, the following points are recommended in order to enhance the proposed algorithm:

1) Maximizing the table size to be larger than 16*16.
2) Maximizing the table indexes from 128 digits to a larger size.

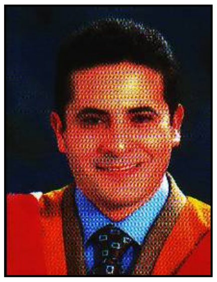
3) Enhance the security level by using more than one decimal digit to rotate and shift the table.
4) Use RSA to share the initial shared value.

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. 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. 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. 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. 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 Sept., 2008.
[15] A. H. Omari, B. M. Al-Kasasbeh, R. E. Al-Qutaish and M. I. Al-Muhairat, A New Cryptographic Algorithm for the Real-Time Applications, in Proceedings of the 7th International Conference on Information Security and Privacy - (ISP'08), Cairo, Egypt, from Dec. 29 Dec. 31, 2008.
[16] V. Hallivuori, Real-Time Transport Protocol (RTP)security, Telecommunications Software and Multimedia Laboratory, T-110.501 Seminar on Network Security, 2004.
[17] Cisco, Voice Quality Understanding Delay in Packet Voice Networks, Document ID: 5125, Cisco Systems, 2009.
[18] C. Bassil, N. Rouhana and A. Serhrouchni, Critical voice network security analysis and new approach for securing Voice over IP Communications, SETIT 2005, 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications March 27-31, 2005, Tunisia.
[19] C. Slav, T. Balan, E. Franti, and M. Dascalu, Exploring the Cellular Automata Phenomenology for Cryptographic Applications, WSEAS Transactions on Communications, Vol. 4, No. 4, 2005, pp. 186-191.
[20] T. Datri, T. Askerov, S. Lebedev and A. Askerov, R-Conversion Method as a New Generation of Cryptography Systems, WSEAS Transactions on Systems, Vol. 2, No. 1, 2003, pp. 103-106.
[21] B. Ontiveros, I. Soto, R. Carrasco, A New Cryptography Algorithm using Cab Curves an LDPC for Wireless Communication Systems, WSEAS Transactions on Mathematics, Vol. 6, No. 1, 2007, pp. 422-425.



Ahmad H. Omari received the B.S. degree in Computer Science from Yarmouk University, Jordan in 1985, the MSc in Computer Science from the University of Jordan, 2000, and PhD in Computer Information Systems in Message Authentication from the Arab Academy for Financial and Banking Sciences AABFS, Jordan, 2004. During 1985-1989, he worked as software developer, during 1989-1995, he was systems analyst and systems engineer, during 1995-2000, he assigned as DBA and project manager, during 2000-2003, he worked as e-Government project manager and Vice President of the Communication and Computer Department in the Public Security Directorate, Jordan.



Basil M. Al-Kasasbeh received the M.Sc. and Ph.D. degrees in Networks, Systems and Communication Devices from Siberian State University of Telecommunications and Informatics, Novosibirsk in 1994 and 2002, respectively. During 2002-2003, he was an Assistant Professor at the Faculty of Information Technology, University of Jordan. Since 2003, he is an Assistant Professor at the Faculty of Information technology in the Applied Science University in Jordan. His research interests include: Mobile and Wireless Systems, Mobile IP, and IPV6. Dr. Al-Kasasbeh authored/co-authored more than 20 research papers in international Journals and conferences.

Rafa E. Al-Qutaish received the B.Sc. degree in Computer Science from Yarmouk University, Jordan in 1993, the M.Sc. degree in Software Engineering from University of Putra, Malaysia in 1998, and Ph.D. degree in Software Engineering from the School of Higher Technology (ÉTS), University of Quebec, Canada in 2007. From Sept. 2007 to Sept. 2008, he was an assistant professor at the department of Software Engineering in the Applied Science University. Since Sept. 2008, he is an Assistant Professor at the department of Software Engineering in Alzaytoonah University of Jordan. His research interests include: Communication Software Engineering, Software Measurements, Software Engineering Standardizations, Software Quality Engineering, and Applied Artificial Intelligence. He is a senior member (SM'07) of IEEE-CS and acting member (M'04) of ACM. Dr. Al-Qutaish authored/co-authored more than 24 research papers in international journals and conferences.

Mohammad I. Muhairat received the MS degree in computer engineering from kharkov state technical university of radio electronics, Kharkov, Ukraine in 1997, and the Ph.D. degree in computer engineering, from kharkov national university of radio electronics, Kharkov, Ukraine in 2002. During 2002-2005, he was an assistant professor at the computer science department at faculty of science and information technology, Al- Zaytoonah University, Amman, Jordan. Since 2005, he is the head of the software engineering department at faculty of science and information technology, Al-Zaytoonah University, Amman, Jordan. His research interests are in software engineering fields like: requirements specification, design and testing, UML notations, formal methods for requirements specification and design.