An Approach of Using RTP with Best Performance of Confidentiality and Authenticity

Musaddeque Anwar Al-Abedin Syed, M.M. Naushad Ali and Shah Ahsanul Haque

Lecturer, Electrical and Electronic Engineering Department, International Islamic University Chittagong (Dhaka Campus), Bangladesh

***Abstract*—This paper presents an approach of using the Real-time Transport Protocol (RTP) emphasizing key security aspects; confidentiality and authenticity. This approach takes a media file as input, encrypts it, creates a message digest of the encrypted data and then transmits it along with the main input file to the user. On the other side, the receiver calculates digest and compares it with the received one, if match occurs, then decrypts and plays it in real time player. In current specification of RFC1889, only the confidentiality is described and authenticity is left for lower layer protocols. This work experiments both on authenticity and confidentiality. For authenticity MD5, SHA-1, and SHA-2 hash algorithms can be used and for confidentiality AES-128 and Triple DES cryptographic algorithms can be used. In fact, SHA-2 is better than other hash algorithms in terms of security but SHA-1 is better than SHA-2 in terms of time efficiency. In addition, we show the experimental results on SHA-2 algorithm. On the other hand, AES-128 is better than Triple DES in terms of time efficiency and security. Hence, SHA-1 and AES-128 are chosen for authenticity and confidentiality respectively for the security of RTP. The experiments were performed using J2SDK1.5.**

***Keywords*—RTP, Transport, Cryptographic, Algorithm, Hash**

I. Introduction

Real-time Transport Protocol (RTP) is to provide the data required by a particular application and sometimes could be integrated into the application processing rather than being implemented as a separate layer. RTP is a modular protocol. The use of this protocol for a specific purpose requires an application area specific RTP profile. RTP profiles are used for refining the basic RTP protocol to suit for a particular application area. RTP profiles define how and by which data format is encapsulated to RTP packets. RFC1889 [1] defines basic fields for the transportation of real time data. It also defines Real-time Transport Control Protocol (RTCP), purpose of which is to provide feedback on transmission quality, information about participants of RTP session, and to enable minimal session control services. RTP is an application level protocol that is intended for delivery of delay sensitive content, such as audio and video, through different networks. RTP provides end-to-end network transport functions suitable for applications transmitting real-time data. It does not provide QoS (Quality of Service) which means that it has no flow control, no error control, no acknowledgement and no mechanism to request retransmission. It does not do so because if a missing packet is retransmitted, then it might happen that the retransmitted packet reaches to the user too late to use which may hamper real-time use of streams. If some packets are lost during transmission (it is very common for real-time protocols) then the lost packets are generated by interpolation rather than retransmission. However, to improve performance of RTP, another protocol; Real-time Transport Control Protocol is used with RTP. It handles feedback on delay, jitter, bandwidth, congestion, and other network properties. RTCP also handles inter-stream synchronization. In this paper, an analysis of the security of RTP and an approach to modify RTP for authenticity are presented. Side-by-side, its usage scenario and the importance of time consideration to transmit streams using RTP are discussed with experimental results.

After this introduction, Section II discusses RTP time considerations, in section II tells about packet format and confidentiality. Section IV talks about some authentication algorithms, Section V presents the experimental results, and finally Section 6 concludes the paper.

II. Time Consideration in RTP

If a video or audio file over the Internet is intended to access in real-time, the most important parameter is bandwidth of the network. The next important parameters are minimum clip size and its duration as well as processor speed of both server and client. At first, let us assume that the files are being accessed without any security consideration. Then, let us review the following mathematical calculations for audio or video clip to access in real-time.

One second file clip size = oneSecFileSize bits,
Time duration of each clip = cSec seconds,
Upload Transmission rate = uRate bits per second,
Download Transmission rate = dRate bits per second,
*Time to upload, tUpload = oneSecFileSize *cSec/uRate,*
*Time to download, tDownload = oneSecFileSize *cSec/dRate*

If the time to upload or download a clip is more than the time to play a clip, the player will wait and the receiver will see a break, i.e., max (tUpload, tDownload) >cSec. For continuous playing of clips, following condition must be true:

Max (1/uRate, 1/dRate) > 1/ oneSecFileSize
Min (uRate, dRate) > oneSecFileSize

According to the equation, the waiting time between clips at the receiver does not depend on clip size. The only variable that matters for a continuous playback is the size of a one-second file and that the provided upload and download rates meet the above condition. Lag time

between playing and capturing is: (cSec + tupload + tdownload). From this value, the maximum lag with no break in the feed is 3*cSec and the minimum lag is cSec. To get the clip as close to real time as possible, cSec should be reduced. Next, we'll apply the above analysis to the following cases:

A. Both Sender and Receiver Have a Low Bandwidth Connection

Let us assume the uRate = dRate = 20Kbits/sec. In this case, the one-second file size should be less than 20Kbits. If the clip size is 10 seconds, the maximum playback lag will be 30 seconds. It is observed that the minimum file size for transmitting a one-second video (with no audio) is 8Kbits using H263 encoding and 128x96 pixels video size. It is also observed a minimum file size with the video and an 8-bit mono audio with an 8KHz-sampling rate to be 80Kbits.

B. Either the Sender or the Receiver Has a Low Bandwidth Connection

Let us assume that the lower rate is 20Kbits/sec and the other rate is much higher. In this case, the one-second file size should be less than 20Kbits, but the maximum playback lag is about 20 seconds if the clip size is 10 seconds.

C. Both Sender and Receiver Have High Bandwidth

It is noted here that the one-second-clip size may vary from format to format of the file; that is, how the file is encoded. For example, the one-second-clip size of MP3 is less than that of WAV file. But the important point here is that when cryptographic algorithms are applied on the clip, then an extra time is added to the processing of clip with each side. So, if strong encryption algorithms are applied to the clip, extra time is needed on both sides and upload or download time will be affected and time lag between them will also be changed. So real time access of data is also affected. Therefore, to provide security in RTP, the considerable parameters are bandwidth of the network, file format of clips, upload and download of the clip, processor speed, memory and applied cryptographic and hash algorithms [2], [3].

III. RTP PACKET FORMAT AND CONFIDENTIALITY

A. RTP fixed header files

Whenever data is transferred with RTP, it always adds a fixed header with the payload. The RTP header has the format shown in Figure 1.

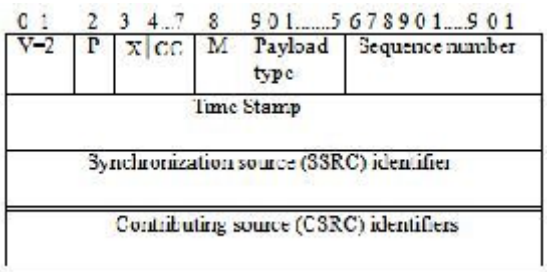

Figure 1. RTP Header [1]

The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer. Version (V) is 2 bits wide. This field identifies the version of RTP. The version defined by this specification is two (2). Padding (P) is 1 bit wide. If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored. Padding may be needed by some encryption algorithms with fixed block sizes or for carrying several RTP packets in a lower-layer protocol data unit. Extension (X) is 1 bit wide. If the extension bit is set, the fixed header is followed by exactly one header extension.

CSRC Count (CC) is 4 bits wide. CC contains the number of CSRC identifiers that follow the fixed header. Marker (M) is 1 bit wide. The interpretation of the marker is defined by a profile. Payload type (PT) is 7 bits wide. This field identifies the format of the RTP payload and determines its interpretation by the application. Additional payload type codes may be defined dynamically through non-RTP means.

B. Confidentiality

RTP is commonly used for broadcasting content trough network. On such use, confidentiality is not necessarily even desired. As RTP is also commonly used for video conferences and for shared white board applications, need for confidentiality should be obvious – for example when RTP based video conference is used for telemedicine, confidentiality is of paramount importance.

IV. AUTHENTICATION ALGORITHMS

To provide authenticity, digital signature is sent to the receiver by the sender. This can be accomplished by sending digest of transferring packets using known private key of sender and receiver. Used hash algorithm is described here:

A. Secure Hash Algorithm-1 (SHA-1)

Secure Hash Algorithm (SHA) was developed by NIST (National Institute of Standards and Technology) and published as a federal information processing standard (FIPS 180) in 1993. SHA-1 (FIPS 180-1) was a revised version of FIPS 180 in 1995. It takes 64-bit block input and produces 160-bit output.

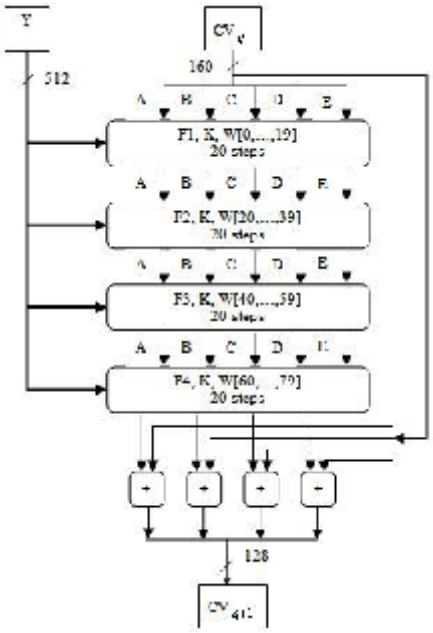

Figure 2. SHA-1 processing of a single 512 bit block

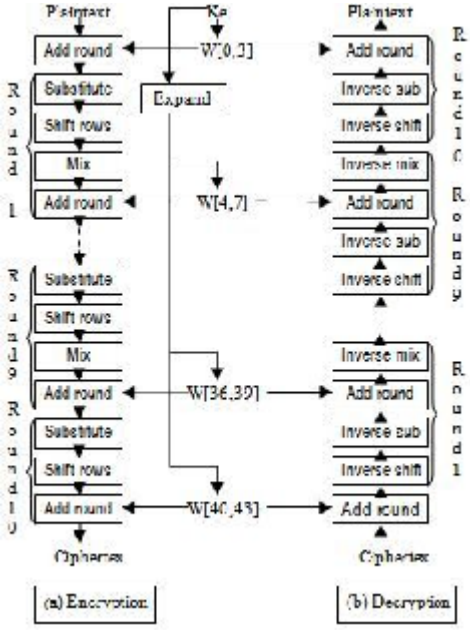

Figure 3. AES encryption and decryption

SHA-1 Logic - This algorithm takes as input a message with a maximum length of less than 2^{64} and produces as output a 160-bit message digest. To produce a digest, the overall processing consists of the following steps:

Step 1: Append padding bits - The input is processed in 512 bit blocks. The message is padded so that its length in bits is congruent to 448 modulo 512 that is, the length of the padded message is 64 bit less than an integer multiple of 512 bits. Padding is always added, even if the message is already of the desired length. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

Step 2: Append length - A 64-bit representation of the length in bit of the original message (before the padding) is appended to the result of step 1.

Step 3: Initialize MD buffer - A 160-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as five 32-bit registers (A, B, C, D, E). These registers are initialized to the following 32-bit integers (hexadecimal values) with big-endian format are shown below -

Word A = 67 45 23 01
Word B = EF CD AB 89
Word C = 98 BA DC EF
Word D = 10 32 54 76
Word E = C3 D2 E1 F0

Step 4: Process message in 512-bit (16-word) blocks - This step consists of four rounds of processing of similar structure but with 20 steps each using a different primitive logical function referred to as f1, f2, f3 and f4. The processing of a single 512 bit block of SHA-1 is given in Figure 2 (SHA-1 compression function).

Step 5: Output - In this algorithm, L 512-bit blocks have been processed to produce 160-bit message digest.

The behavior of SHA-1 can be summarized as follows:

$$CV_0 = IV$$

$$CV_{q+1} = SUM_{32}(CV\,q, ABCDE_q)$$

$$MD = CV_L$$

Where,

IV = initial value of the ABCDE buffer, defined in step 3

$ABCDE_q$ = the output of the last round of processing

L = no. of blocks in message (with padding & length fields)

CV_q= chaining variable processed with *q*th block of message

MD = final message digest value SUM_{32} = addition modulo 2^{32} performed separately on each word

B. Cryptographic Algorithm (AES-128)

AES is able to give equal to or better performance than 3DES [4] which is enough secured. But 3DES having some drawbacks, now AES is chosen for reasons of both efficiency and security. The Rijndael proposal [5] for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192 or 256 bits. Among these various sizes for design simplicity, speed and code compactness on a wide range of platforms and measuring resistance against all known attacks, 128-bit key is most commonly implemented. In Figure 3, the overall structure of AES is shown. The input to the encryption and decryption algorithm is a single 128-bit block.

AES key expansion - The AES key expansion algorithm takes a 4-word (16-bytes) key and produces a linear array of 44 words (156 bytes). This is sufficient to provide a 4-word round key for the initial Add Round Key stage and each of 10 rounds of the cipher.

V. EXPERIMENTAL RESULTS & ANALYSIS

For our implementation, a server was setup and another computer was connected to the server. The connected computer was referred to as client. In this system, maximum 100 clients can be connected to the server. This can be extended if needed. Though multiple clients are connected to the server, data transfer from server end is possible to only one client at a time. The hash and cryptographic algorithms were coded/implemented using Java. For that Kawa Netbean (Program Editor) was used to edit Java code and J2SDK 1.5 tool was used which can be compared to a compiler. Here, the encryption and decryption methods are implemented at the coding design. At the time of data transfer, the edited file is browsed using a J2SDK 1.5 tool. Here, time built-in-method in java was used for time calculation. It was also used at the server end. Ovarll, for this experiment, we used the following system configurations:

Table 1. System Configuration

Processor	RAM	LAN	Tools
Pentium IV (1.6 GHz)	128 MB	100 Mbps	J2SDK 1.5

Processing of SHA-1: Table 2 shows the time needed for various sizes of file and their processing rate using SHA-1.

Table 2. Processing of SHA-1

Serial No	File size in MB	Time Needed in millisecond	Processing rate KB/sec	Average Processing rate KB/sec
1	2.133	454	4766.63	5198
2	3.87	767	5180.00	
3	5.181	1016	5305.00	
4	6.743	1313	5258.00	
5	8.26	1594	5309.08	
6	9.024	1720	5370.87	

The performance graph of SHA-1 is given in figure 4:

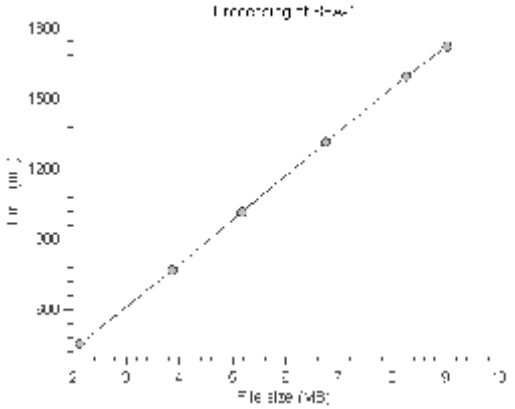

Figure 4. Performance Grapah of SHA-1.

Processing of AES-128: Table 3 shows the experimental results that we got. We have done the experiment using different file sizes and measured encryption time, processing rate, decryption time, etc.

Table 3. Processing of AES-128

Serial no	File size (MB)	Encryption Time(ms)	Processing rate KB/sec	Average Processing rate KB/sec	Decryption Time(ms)	Processing rate KB/sec	Average Processing rate KB/sec
1	0.98	1622	618.7	1143.35	1933	519.15	1022.39
2	2.34	2353	1018.34		2573	931.27	
3	2.71	2503	1108.68		2714	1022.5	
4	3.01	2563	1202.6		2784	1107.13	
5	3.83	2692	1456.87		2914	1345.89	
6	4.17	2935	1454.88		3335	1280.38	

The performance curves of AES-128 are shown in figure 5.

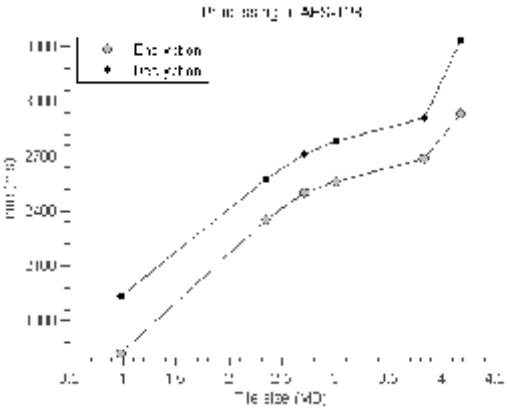

Figure 5: Performance of AES-128

Processing of SHA-2: Table 4 shows time needed for various sizes of file and their processing rates using SHA-2 algorithm.

Table 4. Processing of SHA-2

Serial No	File size in MB	Time Needed in millisec	Processing rate KB/sec	Average Processing rate KB/sec
1	2.133	2562	1082	968
2	3.87	4609	993	
3	5.181	6110	884	
4	6.743	7906	986	
5	8.26	9687	940	
6	9.024	10422	923	

The performance curve of SHA-2 is shown in figure 6.

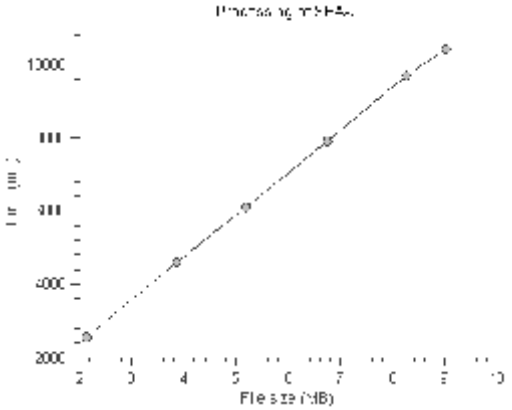

Figure 12. Performance Curve of SHA-2

VI. CONCLUSIONS

RTP is widely used protocol in accessing real-time data over the internet. But it alone is not sufficient to provide quality of service. Some protocols like SIP, SDP, H.323 and SRTP are used as higher level protocols to provide security in RTP. RTP is a profile specific protocol; its design criteria may vary from application to application. From our experiments and analyses, we have found that our proposed model will work very efficiently for audio or video conferencing. One can use this model for some real-time applications.

REFERENCES

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", Internet Draft, IETF, January 1996, available at: http://www.ietf.org/rfc/rfc1889.txt

[2] William Stallings, Data and Computer Communication. 7th Edition, Prentice Hall, 2003.

[3] William Stallings, Cryptography and Network Security. 4th Edition, Prentice Hall, 2005.

[4] Rhee, M.Y., Internet Security: Cryptographic principles, algorithms, and protocols. Wiley Publishers, 2003, pp. 71.

[5] Daemen, J. and Rijmen, V., "AES Proposal: Rijndael", 1999, available at: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf