Securing RTP Packets Using Per-Packet Key Exchange for Real-Time Multimedia

Younchan Jung, Enrique Festijo, and J. William Atwood

For secure multimedia communications, existing encryption techniques use an online session key for the key exchange, for which key size is limited to less than 10 digits to accommodate the latency condition caused by user devices only being able to handle low computational loads. This condition results in poor security of recorded encrypted data. In this letter, we propose a packet key scheme that encrypts real-time packets using a different key per packet for multimedia applications. Therefore, a key of a relatively small size can provide after-transmission confidentiality to data of a real-time session.

Keywords: Real-time multimedia, application security, key exchange, packet key, strength of security.

I. Introduction

The evolution of wireless handsets from simple mobile phones to smart devices has enabled the provision of multimedia services across the Internet [1]. However, the trends of wireless media and mobile smart devices have made packet-sniffing and other digital attacks easy, which means that multimedia contents may not be secure across the Internet. Thus, meeting the need to provide end-to-end secure multimedia services at a low cost is an issue of high priority in the Internet era, especially for real-time sessions.

Multimedia Internet KEYing (MIKEY) supports three different methods to set up a session key (S key): pre-shared key, public key, and Diffie-Hellman (DH) key [2]. DH key has the advantage of providing perfect forward security, whereas pre-shared key suffers from scalability problems for larger user groups and public key requires a public key infrastructure to handle the secure distribution of public keys. The Z and Real-Time Transport Protocol (ZRTP), which also defines an S key agreement protocol, relies on a DH key exchange per session to generate the S key [3], [4]. The DH key agreement methods in MIKEY and ZRTP produce a relatively high computational workload because they use a discrete logarithmic algorithm. However, the scenario considered in this letter is that during an on-going session, the participants may decide to change from unsecure mode to secure mode. This requires the generation of an S key. Thus, the S key latency, which is required for the on-demand mode change, corresponds to the latency for the DH key agreement. Such S key latency, which increases as the DH key size increases, should be less than several seconds in real-time multimedia applications.

According to our simulation results, a key size of nine digits requires over 10 seconds to create a DH key under the condition that the DH key computation runs on a user device equipped with an Intel Pentium i7 CPU. This is the reason why such smart devices cannot use an S key for which the DH key size is greater than nine digits even though it is well known that the DH key protocol is very hard to break if one chooses a substantially large number as the value of the key size. Figure 1 shows how an enemy attacks the S key scheme using a brute force approach. One second of speech, such as the phrase, "Good day," can constitute a bundle of 50 packets, wherein each packet contains 20-ms G.711 speech data. A brute force approach will try to decrypt a bundle of 50 packets using every key from 10^9 possible keys. Kwok and Lam [5] implemented a massively parallel RC4 key searching system that breaks a 40-bit RC4 encryption in 28.5 h. A 40-bit key corresponds to

Manuscript received Dec. 7, 2012; revised Jan. 10, 2013; accepted Jan. 21, 2013.

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0015200), the Research Fund, 2012 of The Catholic University of Korea, and by the Discovery Grants Program through the Natural Sciences and Engineering Research Council of Canada.

Younchan Jung (phone: +82 2 2164 4364, ycjung@catholic.ac.kr) and Enrique Festijo (e_festijo@yahoo.com) are with the School of Information, Communications and Electronics Engineering, Catholic University of Korea, Bucheon, Rep. of Korea.

J. William Atwood (william.atwood@concordia.ca) is with the Department of Computer Science and Software Engineering, Concordia University, Montreal, QC, Canada.

http://dx.doi.org/10.4218/etrij.13.0212.0549

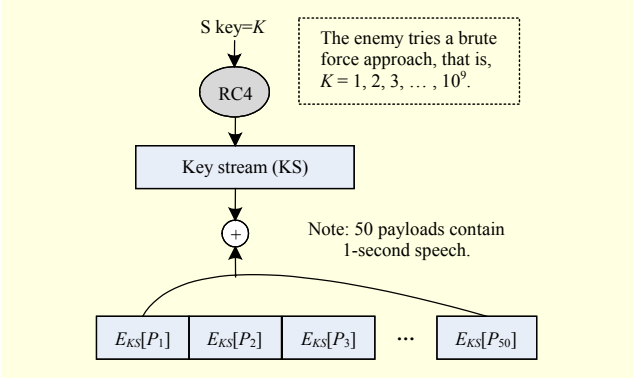

Fig. 1. Vulnerability of after-transmission security for S key scheme with key size = 9 digits.

the key size of 12 in digits. Assuming that the enemy has such a highly parallelized RC4 key searching system capable of 10^5 RC4 decryptions per second every bundle, in the S key scheme with a key size of nine digits, the average time required for an exhaustive key search is around two hours. Then, one must give up either security strength or satisfactory latency condition to establish the S key when one uses the S key scheme in securing real-time applications. This motivates us to propose a packet key (P key) scheme, which guarantees after-transmission confidentiality to be very hard to break.

The basic idea of the P key scheme is to apply the DH key agreement procedure to each RTP packet. Strength of security in the P key scheme can be greatly improved because each P key is different every packet period of the multimedia RTP stream. However, P key raises latency every packet period because per-packet DH key agreement is needed while the S key latency is caused when normal mode changes into secure mode during an on-going session. Let us consider the case in which the same one-second utterance, "Good day," is encrypted by using the P key with a key size of three digits. If one packet is decrypted 10^3 times, the result in each case will be 20 ms of sound, and it might or might not be recognizable. It is likely that some sequences of packets will have to be tried before a recognizable sound will be obtained. These will have to be tried in a correlated way; for a sequence of four, this will result in 80 ms of audio and $10^3 \times 10^3 \times 10^3 \times 10^3$ decryptions. If this is recognizable, then it may be possible to process some sequences of packets using only 10^3 additional decryptions per packet and to carry this for some (short) sequences of packets. If the sequences requiring correlated decryption outnumber the sequences permitting uncorrelated decryption, then the number of decryptions required for a sequence of n packets will be closer to the upper bound of 10^{3n}. Conversely, if more of the packets can be decrypted independently, the number of decryptions will be closer to the lower bound of $n \times 10^3$. It is almost impossible for the after-transmission attacker to understand a one-second utterance such as "Good day," even though the enemy uses a powerful tool capable of 10^5 RC4 decryptions per second.

II. Per-Packet Key Exchange Scheme

The DH algorithm defines q as a prime number and α as a primitive root of the prime number q. Let us assume that Device A (D_A) and Device B (D_B) communicate. We define the secret value for the payload $P_{A,i}$, which is generated by the D_A, as $X_{A,i}$, the blind key of $X_{A,i}$ as $Y_{A,i}$, the P key for the payload $P_{A,i}$ as $K_{A,i}$, and the RC4 key stream for the payload $P_{A,i}$ as $KS_{A,i}$. Then,

$$Y_{A,i} = \alpha^{X_{A,i}} \bmod q, \tag{1}$$

$$K_{A,i} = Y_B^{X_{A,i}} \bmod q = Y_{A,i}^{X_B} \bmod q, \tag{2}$$

$$KS_{A,i} = \text{RC4}\,(K_{A,i}), \tag{3}$$

where Y_B and Y_A are the blind keys of X_B and X_A, respectively. The D_B and the D_ A generate X_B and X_A, which are the session secret values used during their secure session periods, respectively.

As shown in Fig. 2, using the Session Initiation Protocol (SIP), D_A and D_B enter the normal mode of operation for a real-time bidirectional unsecure multimedia session. When D_A initiates a secure voice conversation, D_A and D_B begin to enter the secure mode of operation. First of all, they agree on two global parameters: q and α. Thus, the value of the parameters α and q are made public.

D_A will generate the session secret value X_A and compute the corresponding blind key Y_A, using the same formula shown in (1). Similarly, D_B will generate a session secret value X_B and compute the corresponding blind key Y_B. D_A and D_B securely keep their secret values (X_A and X_B) and exchange session blind keys (Y_A and Y_B) with each other so that each side comes to know the opposite side's session blind key. For per-packet encryption (for example, targeting to encrypt the payload $P_{A,i}$), the sender D_A will generate a packet secret value $X_{A,i}$ and compute the corresponding packet blind key $Y_{A,i}$, using (1). The sender D_A will then compute the P key $K_{A,i}$, using (2). This P key $K_{A,i}$ will be used as an input key for RC4 to generate the key stream $KS_{A,I}$, as described in (3). This key stream $KS_{A,i}$ will be used to encrypt the RTP payload, that is, $E_{KS_{A,i}}(P_{A,i})$. Then, D_A will transmit this encrypted payload along with the packet blind key $Y_{A,i}$. At the receiving end, since the packet blind key $Y_{A,i}$ is transmitted along with the encrypted payload, receiving side D_B can easily obtain the same P key $K_{A,i}$ by using the right part of (2). With the same input key, D_B can generate the same key stream $KS_{A,i}$ by using (3). Using the same key stream $KS_{A,i}$, D_B can decrypt the encrypted payload $E_{KS_{A,i}}(P_{A,i})$, that is, $D_{KS_{A,i}}[E_{KS_{A,i}}(P_{A,i})]$. Finally, the receiving side D_B can obtain the original payload

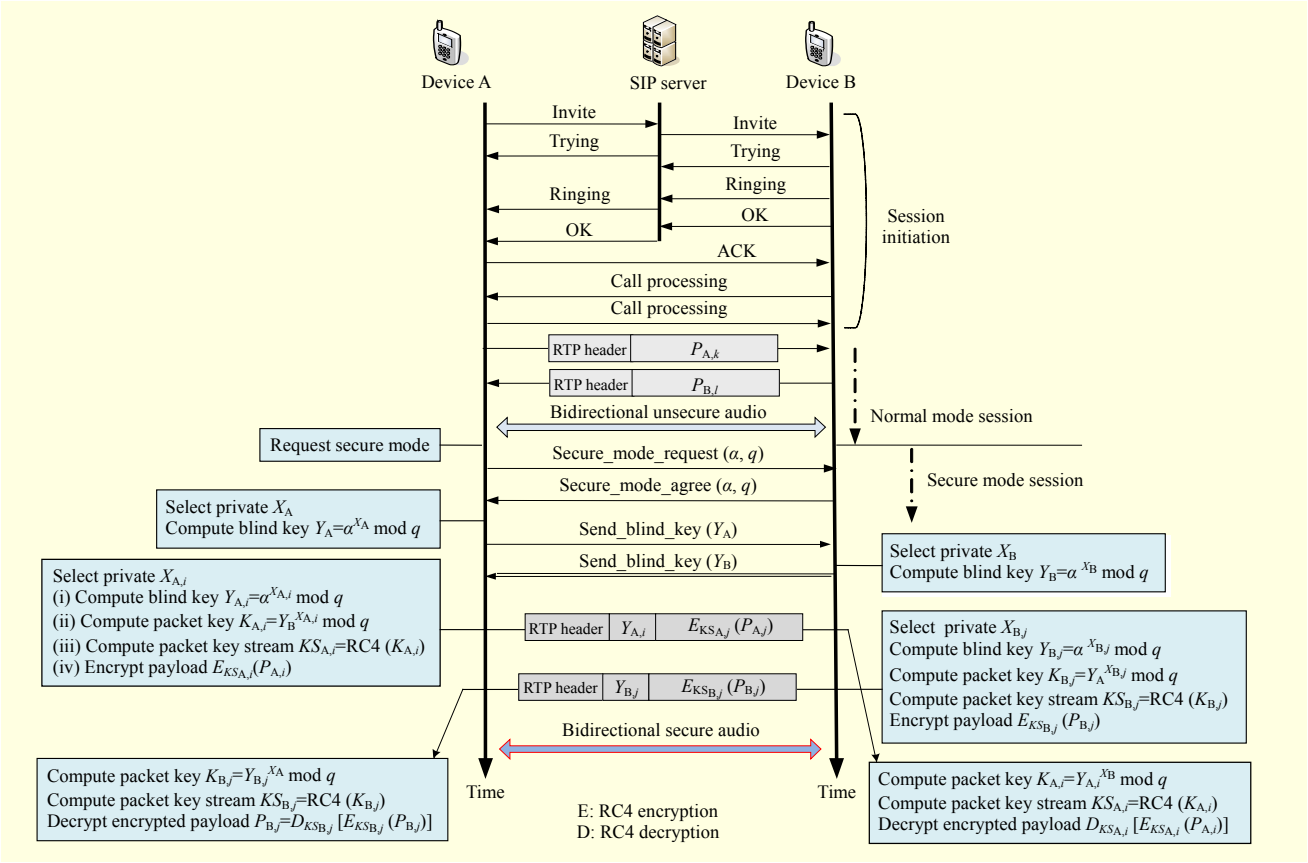

Fig. 2. DH-based per-packet key exchange.

$P_{A,j}$ ($P_{A,j}=D_{KS_{A,j}}[E_{KS_{A,j}}(P_{A,j})]$). Since it is bidirectional audio, as a sender, D_B will also send the encrypted payload $E_{KS_{B,j}}(P_{B,j})$ for the payload $P_{B,j}$ along with the packet blind key $Y_{B,j}$. Then, the receiving side D_A finally obtains the original payload $P_{B,j}$ ($P_{B,j}=D_{KS_{B,j}}[E_{KS_{B,j}}(P_{B,j})]$). Then, consecutive payloads ($P_{B,j+1}$, $P_{B,j+2}$, $P_{B,j+3}$,…) are encrypted using different key streams. This ensures that the same key stream is never reused.

III. Latency for Encryption and Decryption

Effects of encryption and decryption on end-to-end delay need to be investigated, especially for real-time multimedia services. Regarding real-time, it must be proven that the P key techniques cause a negligible level of latency each time the P key is exchanged packet by packet. For the purpose of measuring latencies caused by the P key techniques, we implement a testbed that consists of a video stream server that sends a video packet stream and a client that can receive the video packet stream. The video stream server and client independently run in each Intel Pentium R Dual-Core 3-GHz CPU. The video sample we use is a short movie, that is, "movie.Mjpeg" of 4,170 KB. The Mjpeg-encoded data is grouped into 100-ms packets, which results in a series of 500 RTP packets, each with a length of 1,000 B.

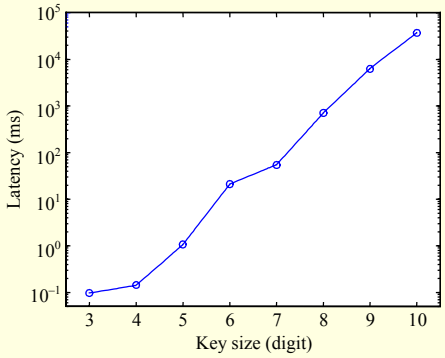

Fig. 3. Latency for DH key scheme.

Using the testbed, for the P key scheme, we measure a latency of [(i)+(ii)+(iii)+(iv)] (delay caused dominantly by discrete logarithmic computation, that is, the DH key agreement), which is indicated in Fig. 2. We run the simulation for different DH key sizes, which range from three digits up to 10 digits. Each point in Fig. 3 represents the average value of 500 measurements. The results can be applied to the DH key agreement for the S key generation. This explains the constraints of the S key scheme, that there is a tradeoff between

the latency time required to establish the S key and security strength for the real-time applications (as the DH key size increases to 10 digits, the S key latency will rise to tens of seconds). This means that the S key scheme cannot be used in secure real-time applications due to the latency constraints in the S key agreements between two parties.

IV. Strength of Security against Brute Force Approaches

Given the prime number q, the enemy may try a brute force approach to search the encryption key depending on the size of q. However, since our scheme is a per-packet encryption with a unique key stream for each packet, the enemy will have a hard time decrypting a series of the encrypted packets. The enemy will check if the decrypted data from the n packets matches the original speech corresponding to the n packets. Because the enemy has no knowledge of the original speech, the enemy can only rely upon whether the concatenated n payloads can be deciphered as noise or not. As discussed in section I, the confidence of the enemy will increase as a longer sequence of packets is decrypted. This explains the weakness of the S key scheme because it allows the brute force enemy to try each possible key to decrypt concatenated payloads from n packets at a time. For more comprehensible comparisons of strength of security, we compute years spent by the enemy to decrypt a given length of speech that corresponds to a bundle of n packets. For the S key scheme, the number of alternative keys can be simply calculated as $10^{\text{“key size”}}$. However, for the P key scheme, the exhaustive key search requires trying every value among $10^{(n \times \text{“key size”})}$ possible keys.

Figure 4 shows the average time required for the exhaustive key search as “Years to Decrypt (YTD).” According to [5], the field-programmable-gate-array-based hardware implementation of a parallel key searching system for the brute force attack on RC4 encryption can achieve a key searching speed of around 10^7 keys per second. Based on this data, we assume the enemy uses a cipher breaking tool capable of 10^5 decryptions per second in RC4. For the S key scheme, given the number of packets in a bundle to decrypt, the graph shows that the S key scheme with a key size of nine digits allows the enemy to understand the encrypted speech within one day’s worth of effort. However, the P key scheme does not allow the enemy to decrypt the encrypted speech investing one year’s worth of effort if the key length exceeds three digits for the case in which n=4 (80-ms speech from G.711 encoder). As a result, we argue that the P key scheme with a key size of four digits is computationally more secure than the S key scheme with a key size of nine digits. The use of a four-digit key requires a low computational workload that corresponds to running a 12-bit discrete logarithmic algorithm.

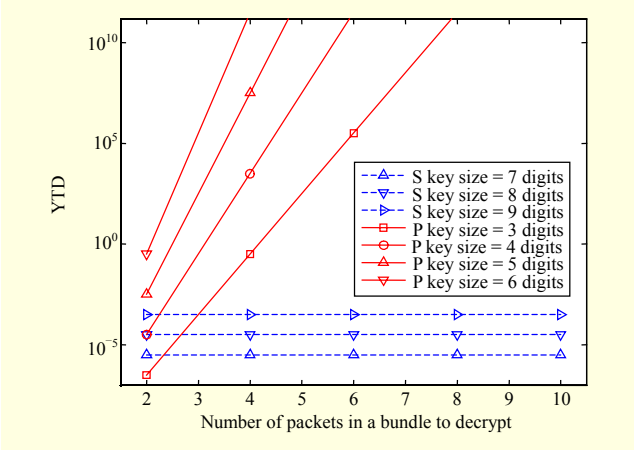

Fig. 4. YTD.

V. Conclusion

This letter proposed a P key scheme wherein DH key exchange procedures repeat every RTP packet. Our P key scheme was designed to eliminate the weaknesses of the S key scheme regarding security strength, as the S key size is limited to less than 10 digits to accommodate the latency condition for the key exchange, which is caused by user devices only being able to handle low computational loads. Comparing the strength of security of the S key scheme with that of the P key scheme, we proved that the P key scheme makes it significantly difficult for the enemy to decrypt the encrypted speech, requiring the investment of at least one year’s worth of effort if the key size exceeds three digits. As a result, we argue that the P key scheme with its key size of four digits will provide stronger after-transmission confidentiality than the S key scheme with its key size of nine digits.

References

[1] F. Almasalha, N. Agarwal, and A. Khokhar, “Secure Multimedia Transmission over RTP,” *10th IEEE Int. Symp. Multimedia*, Dec. 2008, pp. 404-411.

[2] J. Arkko et al., “Multimedia Internet KEYing (MIKEY),” *IETF, RFC 3830*, Aug. 2004.

[3] R. Pecori, “A PKI-Free Key Agreement Protocol for P2P VoIP Applications,” *IEEE Int. Conf. Commun.*, June 2012, pp. 6748-6752.

[4] S. Puangpronpitag, P. Kasabai, and D. Pansa, “An Enhancement of the SDP Security Description (SDES) for Key Protection,” *9th Int. Conf. Elect. Eng./Electron., Comput., Telecommun. Inf. Technol.*, May 2012, pp. 1-4.

[5] S. Kwok and E.Y. Lam, “Effective Uses of FPGAs for Brute-Force Attack on RC4 Ciphers,” *IEEE Trans. Very Large Scale Integr. Syst.*, Aug. 2008, vol. 16, no. 8, pp. 1096-1100.