Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2010, Article ID 371513, 9 pages
doi:10.1155/2010/371513

*Research Article*

# A Speed-Adaptive Media Encryption Scheme for Real-Time Recording and Playback System

**Chen Xiao,[1] Shiguo Lian,[2] Lifeng Wang,[3] Shilong Ma,[1] Weifeng Lv,[1] and Ke Xu[1]**

[1] *State Key Laboratory of Software Development Environment, School of Computer Science & Engineering, Beihang University, Haidian District, Beijing 100083, China*
[2] *France Telecom R&D Beijing, Haidian District, Beijing 100080, China*
[3] *Department of Electronic & Information Engineering, Beijing Electronic Science and Technology Institute, Fengtai District, Beijing 100070, China*

Correspondence should be addressed to Shiguo Lian, sglian@gmail.com

Received 29 March 2010; Accepted 2 August 2010

Academic Editor: Liang Zhou



The recording and playback system (RPS) in video conference system needs to store mass of media data real-timely. Considering the security issue, media data should be encrypted before storing. Traditional full encryption and partial encryption algorithms are not applicable to RPS because they could not adjust their speed to meet the throughput variation of media data in real-time RPS. In this paper, a novel lightweight speed-adaptive media-data encryption (SAME) scheme is proposed firstly. Secondly, the SAME is improved to a packet-based algorithm according to the implementation of data storage in RPS system. Thirdly, an RPS oriented queue theory-based autoadaptive speed control mechanism for SAME is designed. Finally, these schemes are integrated into the practical system, that is, AdmireRPS, an RPS of a heterogeneous wireless network-(HWN-) oriented video conference system. Theoretical analysis and experimental results show the SAME is effective enough to support real-time applications. In addition, the proposed schemes also can be used in video surveillance and other video recording systems.

## 1. Introduction

With the development of multimedia and network technologies, video conferences and some other streaming media systems have attracted significant research efforts [1–8]. Recording and playback system (RPS), which stores large-volume media data and plays them back according to requirements of users, plays an important role in a video conference system. Since the media data stored in RPS contain all the crucial information of the meetings, they should be encrypted properly. However, traditional full-encryption algorithms are not applicable due to the high-volume of media data and real-time requirements of multimedia applications [1]. How to maintain the security of media data becomes a challenge task [1, 8, 9]. To solve this problem, a lot of multimedia (e.g., image, video, or audio) content encryption schemes have been proposed during the last decade. In this paper, more attention is paid on video encryption, since bandwidth of image or audio data is comparatively less than that of video. According to different viewpoints, those encryption schemes can be divided into different kinds, respectively. From the way they reduce the computational complexity, these schemes could be divided into three types, that is, partial encryption [10, 11], joint compression encryption [1], and improved full encryption [12].

Partial encryption encrypts a selected crucial subset of media data. One kind of partial encryption focuses on I-frames or intramacroblocks in video frames. The algorithm proposed in [10] encrypts only the I-frames in MPEG2. Although the other data are still clear, and might be eavesdropped by attackers, it is very hard to rebuild a clear video copy without the I-frames. However, the practical analysis given in [13] shows that without the I-frames video contents are still discernible, because the unencrypted I macroblocks in the P-frames can be fully decoded. So, it is not secure enough for some confidential applications [14]. An improved scheme called SECMpeg is proposed in [15].

It has implemented 4 levels of security. One important level is to encrypt all the I-blocks in P- and B-frames. This scheme can provide higher security but suffers a substantial increasing of complexity. Moreover, it needs the encryption process to parse the P- and B-frames, and is not suitable for RPS. Another kind of partial encryption focuses on discrete cosine transform (DCT) coefficients, motion vector, and other sensitive data. For example, Tang [16] encrypted videos by scrambling discrete cosine transform (DCT) coefficients. Shi and Bhargava [17] proposed a video encryption algorithm, where the 64 most significant sign bits of DCT coefficients and motion vectors in each $16 \times 16$ macroblock are encrypted by a symmetric cipher. In some other schemes [18, 19], intraprediction mode, residue data, and motion vector are encrypted partially to keep format compliance. The third kind of algorithms use some lightweight algorithms to encrypt the sensitive data, for example, in image encryption, Lian et al. [20] used chaotic cipher to encrypt JPEG2000 images. With regard to audio encryption, several partial encryption schemes are also proposed [1, 11].

Joint compression encryption is another kind of media content encryption. Wu and Kuo [21] implemented encryption operations in entropy coding. In [22], the scheme combined fixed length code (FLC) and variable length code (VLC) codeword, which is achieved by permuting the codeword and encrypting the index of code table in MPEG-4. In [23], a perceptual video encryption scheme is proposed based on exploiting the special feature of entropy coding in H.264.

Improved full-encryption schemes use some lightweight algorithms to encrypt the whole bit stream. In [1, 20, 24], chaotic stream ciphers are constructed and used to encrypt video data. Besides chaotic stream ciphers, VEA scheme is another improved full-encryption scheme [12]. The main idea of VEA is as follows. Divide the plaintext into two segments: *Even* and *Odd*, encrypt the *Odd* with a standard encryption algorithm $E(Odd)$, and the other half *Even* is exclusive-ORed (Xor) with the plaintext of *Odd*. This mechanism can reduce the complexity to nearly one half and achieve a sufficient security level. This algorithm is extended to encrypt one fourth of the plaintext in [25]. These schemes can encrypt the compressed video data, and fit the RPS. But their encryption rate is changeless, so the encryption throughput of them could not adjust to meet the variation of media data throughput in real-time RPS.

On the other hand, according to the data format or application they oriented, these encryption schemes can be classified into two types, that is, one is all-purpose scheme, and the other focuses on special codec standards or applications. As mentioned above, the schemes in [10, 15] select the I-frames or I-blocks to encrypt, and the ones in [16–19, 26] choose the DCT coefficients and sign bits of the motion vectors to encrypt. These schemes can be used in different codec standards. The other kind of scheme focuses on the special image, audio, and video standards. For example, the schemes in [20, 27] aim at encrypting the image of JPEG2000, the one in [11] aims at G.729 data encryption, and the ones in [18, 19] study the AVC encryption. There are also some schemes focusing on special applications, for example, joint fingerprint embedding and en/decryption algorithms are proposed and used in digital rights management (DRM) [3, 28]. The security of multimedia content in IPTV is studied in [4, 5]. Chen et al. [2] proposed a mathematical model-based dynamic optimal selective control mechanism for optimizing the security of multidatastreams in video conference. These schemes are designed based on the special needs of applications.

Among the mentioned algorithms above, joint compression encryption needs to analyze the compressed data, which is infeasible for practical RPS systems. Partial encryption and improved full-encryption algorithms have also some limitations, because they could not adjust their speed to meet the variation of media data throughput of RPS. Therefore, in this paper, we design a secure real-time recording and playback system, named AdmireRPS, based on a novel Speed-Adaptive Media-Data Encryption (SAME). As of our knowledge, this is the first media encryption scheme considering the adaptive media throughput. Firstly, by combining the block cipher and stream cipher, a lightweight speed-adaptive media-data encryption is proposed. Secondly, a packet-based SAME (PSAME) is proposed, according to the implementation of data storage in a practical RPS system. Thirdly, a queue theory based autoadaptive speed control mechanism for SAME is designed. Finally, those schemes are integrated into AdmireRPS system. Theoretical and experimental analyses show the performance of SAME is effective enough to support real-time applications.

The rest of the paper is organized as follows. Section 2 presents the overview of AdmireRPS system and its security problems. The design and implement of AdmireRPS are given in Section 3. The speed-adaptive media-data encryption (SAME) and autoadaptive speed control mechanism for SAME are presented in Section 4, and Section 5 discusses the performances and presents the theoretical analysis and experimental results. Finally, Section 6 concludes the paper.

## 2. Admire and Encryption Bottleneck

*2.1. An Overview of Admire System.* As is shown in Figures 1 and 2, the Admire (ADvanced Multimedia Interactive Real-time Environment) system [2, 6, 7] is a heterogeneous wireless network-oriented large-scale multimedia collaboration system. It is compatible with multicast, unicast, wired, and wireless clients, and includes not only video conference system but also RPS, immediate message system (IMS), electric white board, collaborate edit, desktop sharing, and so forth. This system has been used in hundreds of universities and institutes. In the biggest session, there are more than 100 users joined in at one time.

*2.2. Encryption Bottleneck in Admire System.* To meet the application demands of some confidential departments, a special edition with security concern is developed, in which data encryption is operated by a specified confidential encryption algorithm which is similar to DES and implemented in a PCI card [2]. Although the card has


Figure 1: A practical video conference in Admire system.

a declaratory encryption throughput of 224 Mbps, we find its stable external throughput is only 12 Mbps in fact [2]. We also find that the software implementation of traditional encryption algorithm on universal processor is no more than 200 Mbps, which is inadequate to the data throughput of MediaGateway [7], RPServer, and other servers. Consequently, the encryption becomes a bottleneck in the system.

The encryption speed can be accelerated according to the growth of the CPU speed, but in the recent years the improvement of CPU speed decelerates because of the limit of manufacturing technology of semiconductor device, so the perspective of the growth of encryption speed is not optimistic. Comparatively, the disk densities improved 100 percent per year, which is faster than Moore's Law, something like 60 percent a year [29]. Moreover, core network bandwidth and image process ability growing even faster than disk densities. Predicatively, the dissymmetry between the throughput of encryption algorithm and the data volume to be encrypted will aggravate. Furthermore, there is a remarkable variation of media data throughput in Admire system, so a speed adaptive encryption scheme is needed.

## 3. The Proposed AdmireRPS

*3.1. AdmireRPS in Admire System.* AdmireRPS is the secure recording and playback system of Admire system. It can encrypt and save the specified streaming media data into media files according to the requirements of the participants. When a user misses a meeting, RPS can playback the audio/video for her/him, and asynchronous collaboration could be achieved.

Figure 2 shows the architecture of AdmireRPS. On the left of the box with broken lines are RPS clients, in the box are the servers, including RPServer which records and playbacks media data, ControlServer [2] which works as the GateKeeper (GK) and ControlUnit (CU) of H.323 system, and AdmireDB which saves the session and access control information. All these components exchange control message using AdmireMBus [2], a lightweight message bus which is fully encrypted. Media data are transmitted in media channel, which is built based on multicast or application layer multicast [7]. The details of the secure recording and playback process will be presented in the following content.

*3.2. Media Data Recording and Encryption.* The media data record process in AdmireRPS is shown in Figure 2, which includes the following 3 steps. Firstly, the client sends the record request to ControlServer via the secure control channel. Then, ControlServer inserts the media information, including session information, multicast address of the session, Synchronization SouRCe (SSRC) identifier, and so forth, into AdmireDB. Finally, ControlServer asks RPServer to record media data in the specified multicast address (the client can also require RPS to record data with specified SSRC in a multicast address).

The data in one session could be saved into a single file or several files according to SSRC. However, we found stable throughput of hard disk is correlate inversely with the number of files accessed at one time, thus single-file scheme could achieve a higher performance. On the other hand, multifile scheme can playback media data with selected SSRCs in one session separately. This scheme can achieve higher flexibility. Client has the right to choose either of single or multifile in record request.

The record process in RPServer saves each received RTP [30] packet from the specified multicast address. Considering the playback process should send those packets according to the received time, the file structure is designed as follow (shown in Figure 3). For each packet, there is a 96-bits header. The first 16-bits records the total block length. The 17th bit EF is an encryption flag, which specifies the encryption mode and will be explained in Section 4. The second 32-bits is time stamp, which is the time offset from the beginning packet to the current. Because the playback module could not decode the encrypted RTP packet, the last 32-bit records the synchronize source identifier of the current packet, which is got from the RTP header.

As is shown in Figure 4, the RTP packet is encrypted using packet-based SAME algorithm. A speed control algorithm is also introduced, which calculates the encryption speed control parameter $l$ according to the input packet rate. Those two algorithms are proposed in Section 4.

*3.3. The Secure Playback Process.* Similar with the record process, playback process in AdmireRPS works as follows.

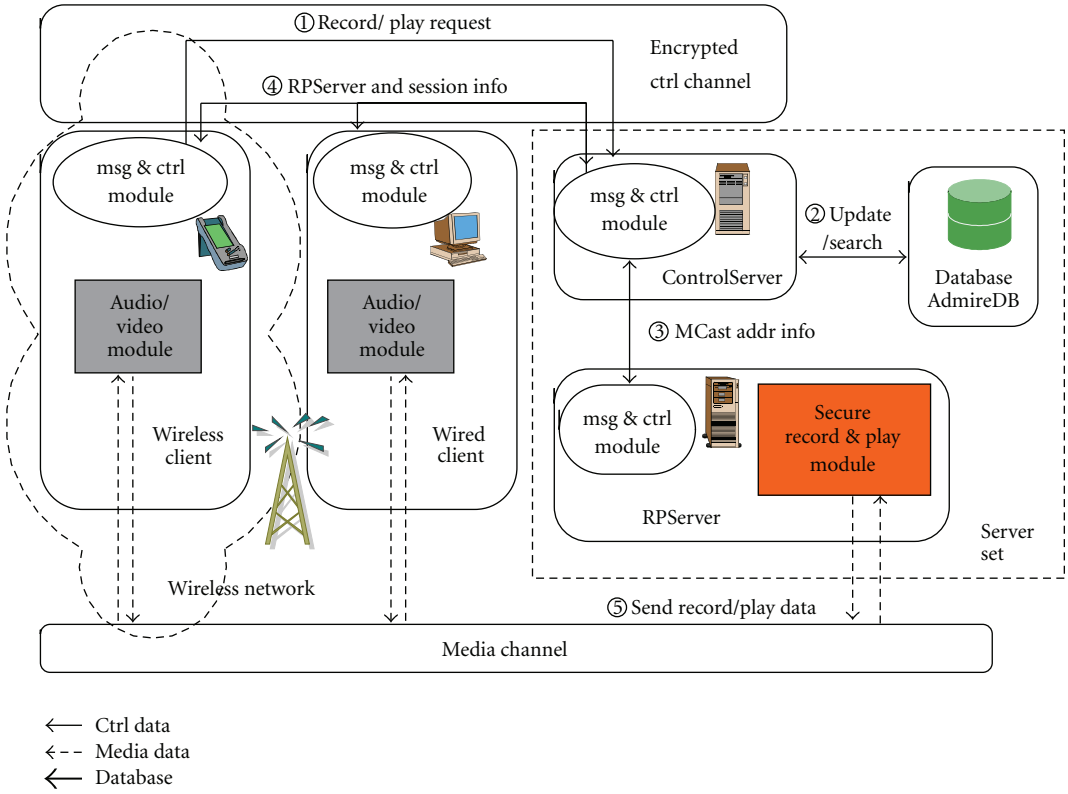


Figure 2: Architecture and control process of AdmireRPS.

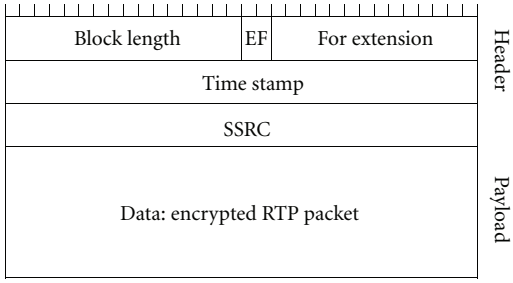


Figure 3: Block structure of encrypted RTP packet in RPS media file.

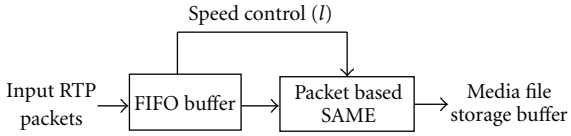


Figure 4: RTP data encryption in RPS system.

Firstly, Client sends media file inquiry request to ControlServer via the encrypted control channel. Secondly, ControlServer inquires the media file information from AdmireDB, and sends it to Client. Thirdly, Client selects the media file her/she wants. Fourthly, ControlServer asks RPServer to playback the specified media file in a negotiated multicast address. Fifthly, RPServer reads packets are from

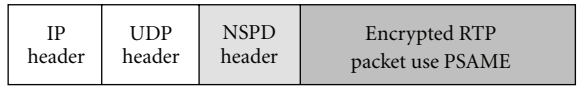


Figure 5: The protocol data unit in playback system.

the media file and sends them to the negotiated multicast address, according to the time stamp. RPServer can send the encrypted packets or plaintext packets. When the packets sent in ciphertext, a session key will be sent via the encrypted control channel simultaneously.

To minimize the refactoring of the other modules in Admire system, an AdmireNSPD (Network Service Provider Daemon of Admire system) module [7] is designed to substitute the Winsock, and accomplish the NAT penetrating task. The PDU in our system is depicted in Figure 5. RPServer packets the encrypted RTP data into the NSPD payload, and uses one bit in NSPD header as the encryption flag. According to this flag, client determines the decryption way (more details are in Section 4).

## 4. Speed-Adaptive Media-Data Encryption (SAME)

Statistical characteristics of compressed audio/video data are dramatically different from the ones of text data, because the variable-length codes and other processes used in compression remove the redundant information from the

original data. Statistical analysis in [12] shows that the coded data have high randomness at the byte level. Based on this statistical characteristic of media data, we extend the idea of VEA algorithm to a new method that uses traditional block cipher to encrypt a part of data (part I), and uses its plaintext as the stream cipher key to encrypt another part of data (part II). By changing the ratio between parts I and II, we can adjust the speed of the encryption algorithm.

*4.1. The Basic SAME Algorithm.* In the basic algorithm, firstly, the plaintext is divided into segments with a same length. Secondly, a selected traditional block cipher algorithm is used to encrypt one segment. Thirdly, for the next $l$-blocks, use the plaintext of the previous segment as its stream cipher key. Assuming media data are saved in a FIFO buffer, the basic algorithm consists of the following steps (also shown in Figure 6).

This algorithm is designed for media file rather than real-time packets in RPS. The improved algorithm for packets is proposed in Section 4.2. To avoid the file header being guessed by the attackers, the first $n$-segments are full encrypted in step (2), where $n$ is calculated from the session key. Although the probability of a segment being got by attackers is very little, the permutation proposed in [12] could be used before the dividing in step (1). The encryption speed control parameter $l$ in step (4) is given in Section 4.3. This important parameter can adjust the speed of the encryption algorithm, and the experimental result is shown in Section 5.2. For file encryption, this parameter should be properly saved either by saving in a separate file, or using 1 bit in the header of each segment as encryption flag EF. The decryption process can determine the decryption way according to EF.

Since most standard encryption algorithms need the length of plain-block divided exactly by a very number $n$ (e.g., 8 bytes), the rear filling method in (1) is used in step (6). Here, the length of filled bytes is also the value to be filled. Figure 7 gives two examples of $n = 8$,

$$\begin{aligned} Filling\ Length &= n - (Length \mod n) \\ Filled\ Value &= n - (Length \mod n) \end{aligned} \tag{1}$$

*4.2. Improved Algorithm for RTP Packets.* The former algorithm, which is designed for byte stream, is suitable for encrypting large-volume media file. Since both recording and playback processes in AdmireRPS work on RTP packets, a packet-based algorithm can achieve higher efficiency. Therefore, we design a packet-based improved algorithm shown as follows.

The one bit EF of block header in Figure 3 can be decided by.

$$\mathrm{EF} = \begin{cases} 1, & \text{if packet is fully encrypted,} \\ 0, & \text{if Xor with the previous packet.} \end{cases} \tag{2}$$

Because adjacent packets could have different length, the Xor operator in step (3) is implemented as (3), where $p_j^i$ is $j$th byte of packet $i$, $c_j^i$ is its ciphertext, and $\mathrm{PL}^i$ is the length of packet $i$. That is to say, if the current packet is longer than the former one, duplicate the former packet at its rear. An example is given in Figure 8, where $\mathrm{PL}^{i-1} = 1000$ and $\mathrm{PL}^i = 1005$,

$$c_j^i = p_j^i \oplus p_{j \bmod \mathrm{PL}^{i-1}}^{i-1}. \tag{3}$$

*4.3. Adaptive-Speed Control Mechanism.* In this subsection a speed control mechanism is designed to determine the encryption speed control parameter $l$ in SAME, while the input throughput and upper limit of the expected queuing delay is given.

A FIFO queue is used in RPServer to buffer the input data (as is shown in Figure 4). The new packets are inserted to its *rear*, while the encryption process gets packets from the *front*. Since the volume of media data in video conference change dramatically, speed control mechanism should ensure that the queuing delay is stable and under control, while take full advantage of encryption recourse.

In order to find the relationship among the input bandwidth, queuing delay and encryption throughput, we make the following assumption: (1) packets arriving follows $\lambda$-Poisson distribution, (2) encryption capacity is $C$, (3) packets length $L_{\mathrm{packet}}$ is a constraint, and (4) memory is much greater than packet length. This is a typical model of an M/M/1/K queuing system [31]. Then, the average queuing delay $d_{\mathrm{queue}}$ is

$$d_{\mathrm{queue}} = \frac{1}{\mu - \lambda} * \frac{1 - (K+1)\rho^k + K\rho^{k+1}}{1 - \rho^{k+1}}, \tag{4}$$

where $\mu = C/L_{\mathrm{packet}}$ is the packet number algorithm can encrypt in a unit time interval, $\rho = \lambda/\mu$ is the load rate, and $\lambda$ is the packets arrive rate.

If we assume that main memory capacity of RPServer is much greater than packet length, so the parameter $k$ is approaching to the infinite, and the following equation can be got:

$$\begin{aligned} \lim_{k \to \infty} d_{\mathrm{queue}} &= \frac{1}{\mu - \lambda}, \qquad \mu = \frac{1}{d_{\mathrm{queue}}} + \lambda, \\ C &= L_{\mathrm{packet}} \left( \frac{1}{d_{\mathrm{queue}}} + \lambda \right). \end{aligned} \tag{5}$$

Therefore, given an upper limit of the expected queuing delay $d'_{\mathrm{queue}}$ ($d'_{\mathrm{queue}} > 1/\mu$), the minimum encryption speed $C'$ should satisfy.

$$C' \geq \left( \left( \frac{1}{d'_{\mathrm{queue}}} \right) + \lambda \right) * L_{\mathrm{packet}}. \tag{6}$$

We can use (6) to calculate the minimum throughput of SAME with a limited queuing delay. For example, using

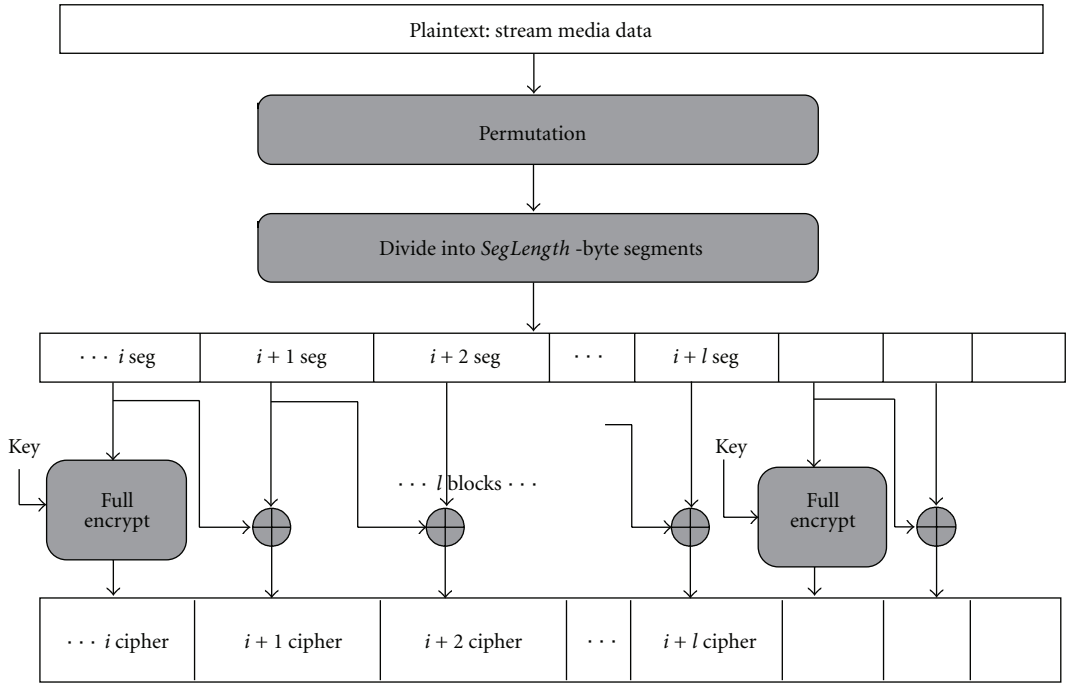


Figure 6: Speed adaptive media-data encryption (SAME) algorithm

***The basic SAME algorithm***
Begin:
(1) Permute and divide the byte stream in the FIFO buffer into segments with a length of *SegLength*.
(2) Use the traditional block cipher algorithm $F$ to encrypt the first $n$-segments
  Do until the last segment:
    (3) Use algorithm $F$ to encrypt the first segment $\text{Seg}_i$ in the buffer.
    (4) For the next $l$ blocks, its ciphertext $\text{CSeg}_j = \text{Seg}_{j-1} \oplus \text{Seg}_j$.
    (5) Repeat the steps (3) and (4).
  End Do
(6) For the last segment, fill it using the filling method shown in Figure 7, then encrypt it using $F$.
End

Algorithm 1

***The decryption process***
If (EF = full encrypt) then
  $\text{PlainText} = F^{-1}(\text{Cipher}_i)$;
Else
  $\text{PlainText}_i = \text{PlainText}_{i-1} \oplus \text{Cipher}_i$.
End

Algorithm 2

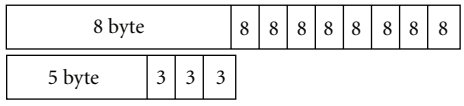


Figure 7: Rear padding in SAME.

encryption algorithm in PCI card, if $L_{\text{packet}} = 8\,\text{Kb}$, $\lambda = 11000$, and $d'_{\text{queue}} = 4\,\text{ms}$, we can find that $C'$ should be bigger than 90 Mbps. Then we look up in Figure 9 and find $l$ should be not less than 64 (the fifth asterisk of the blue line in Figure 9(a)). In addition, in the practical system when the number of queuing packets exceeds a gate valve, the throughput of the SAME can be increased by adding the parameter $l$.

## 5. Performance Analyses

*5.1. Theoretical Security Analysis Based on Shannon Theory.* The following analysis is focused on the basic SAME algorithm, and the result could be deduced to the improved algorithm. Considering $\text{Seg}_i$ to $\text{Seg}_{i+l}$, based on Shannon theory [32], the attack difficulty of ciphertext-only attack is $H(KP \mid C)$, where $K$ is the key of encryption algorithm

**The improved encryption algorithm**
Begin:
(1) Use the traditional block cipher algorithm $F$ to encrypt the first $n$-packets
Do until the end:
(2) Use algorithm $F$ to encrypt the first packet in buffer.
(3) For the next $l$ packets, let its ciphertext $\text{CPacket}_j = \text{Packet}_{j-1} \oplus \text{Packet}_j$.
(4) Repeat steps (3) to (4).
End Do
(6) Full encrypted the last packet,
End

Algorithm 3

$$\begin{array}{lccccccccc} & p_1^{i-1} & p_2^{i-1} & \cdots & p_{1000}^{i-1} & p_1^{i-1} & p_2^{i-1} & p_3^{i-1} & p_4^{i-1} & p_5^{i-1} \\ \text{Xor} & p_1^i & p_2^i & \cdots & p_{1000}^i & p_{1001}^i & p_{1002}^i & p_{1003}^i & p_{1004}^i & p_{1005}^i \\ & c_1^i & c_2^i & \cdots & c_{1000}^i & c_{1001}^i & c_{1002}^i & c_{1003}^i & c_{1004}^i & c_{1005}^i \end{array}$$

Figure 8: Rear padding in packet-based SAME.

$F$, $C$ is the ciphertext, and $H$ is the entropy function. After segmentation, we get

$$H(KP \mid C) = H\left(K\text{Seg}_i\text{Seg}_{i+1} \cdots \text{Seg}_{i+l} \mid \text{Cphr}_i Cphr_{i+1} \cdots C\text{phr}i + l\right). \quad (7)$$

Since SAME uses the previous segment as the stream cipher key of the current segment, the security of the current segment depends on the previous one, finally security of $\text{Seg}_{i+1} \ldots \text{Seg}_{i+l}$ depends directly or indirectly on $\text{Seg}_i$, whose security depends on the theoretical secrecy of algorithm $F$. Thus, the ciphertext-only attack of SAME has an attack difficulty of

$$H(KP \mid C) >= H\left(K\text{Seg}_i \mid C\text{phr}_i\right). \quad (8)$$

Only when adjacent segments are related with each other, that is,

$$H(KP \mid C) = H\left(K\text{Seg}_i \mid C\text{phr}_i\right). \quad (9)$$

However, statistical analysis of compressed video stream shows that the data have high randomness at the byte level, moreover the permutation before data dividing can also reduce the relativity. Thus, the security is often not smaller than the first segment's encryption.

The attack difficulty of known-plaintext attack is

$$H(K \mid PC) = H\left(K \mid \text{Seg}_i\text{Seg}_{i+1} \cdots \text{Seg}_{i+l} C\text{phr}_i C\text{phr}_{i+1} \cdots C\text{phr}_i + l\right). \quad (10)$$

Because only $\text{Seg}_i$ and $\text{Cphr}_i$ relate to Key, we can obtain that the known plaintext attack of SAME has an attack difficulty of

$$H(K \mid PC) = H\left(K \mid \text{Seg}_i C\text{phr}_i\right). \quad (11)$$

Table 1: Throughputs of SAME with different CPU loads (measured in MBps).

| Algorithms | Load 1 | Load 2 | Load 3 | Load 4 |
|---|---|---|---|---|
| Algorithm in PCI | 989.1 | 977.2 | 974.8 | 847.1 |
| Software DES | 2607 | 2532 | 2519 | 2021 |

That is, the security depends only on the first segment's encryption.

*5.2. Throughput Analyses on SAME.* Owing to the limited scale of our Admire system, we study the throughput of SAME in simulation programs. As is shown in Figures 9(a) and 9(b), two tests, that is, Test A and Test B, are designed to evaluate the throughputs of SAME with DES similar algorithm in PCI card and DES implementation in software, respectively. These two simulation programs are implemented in C++, and each of them occupies only one processor each time. Test A runs in a Windows XP system with Intel Core Duo T2300 1.66 GHz and 512 M RAM, and Test B runs in a Windows NT workstation with dual-processor Intel PIV 2.4 G and 512 M RAM. When the parameter $l + 1$ is {1, 4, 16, 64, 256, 1024, 4096}, the throughputs of SAME in Test A and Test B are {1.45, 2.774, 5.796, 23.12, 90.03, 334.1, 989.1, 1973} and {21.91, 41.4, 86.12, 314.9, 959.5, 1935, 2607, 2843}, respectively.

The results of SAME are compared with the original full-encryption algorithm and VEA. When $l = 0$, the SAME algorithm is equal to full encryption, and when $l = 1$, the SAME is equal to VEA. Experimental results also show that when $l$ is small, the speed nearly increases linearly. When $l$ is very large (e.g., $l > 1000$), limited by the speed of stream cipher algorithm, the throughputs are both less than 3 GBps for the two cases.

Table 1 shows the throughputs in Tests A and B when $l + 1 = 1024$ and the computers have different loads. In column 1, only the SAME process is busy, so there is one CPU busy and the other is always idle. In column 2, beside the SAME process, there is another process with heavy load running on the other CPU, thus the two CPUs both have heavy loads. Column 3 gives the results when a process with on average 35 MBps hard disk transmission is added. The last column shows the results when the SAME process contests

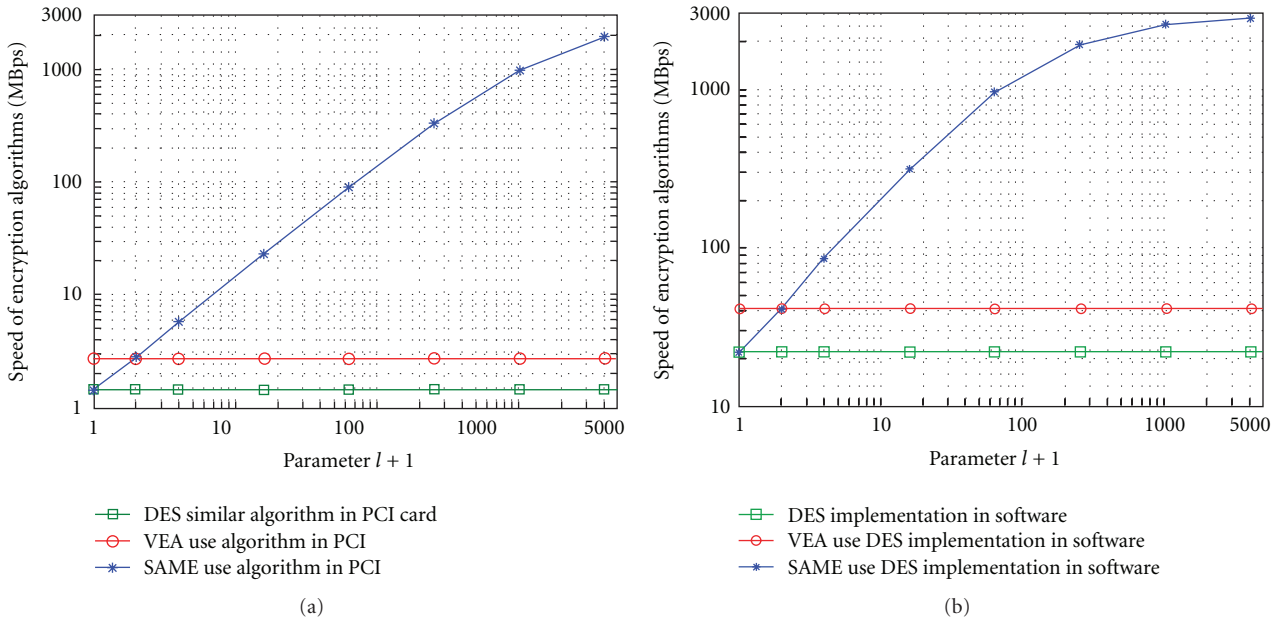


Figure 9: The throughput of SAME.

CPU time with other two CPU intensive process. Comparing column 2 with column 1, we can find that, if only one CPU intensive process contest CPU time with SAME, the throughput of SAME reduces less than 3%. That is because both two process could use one CPU every time, the impact is not notable. But when the number of CPU intensive processes is bigger than the number of CPUs, the other processes would observably impact the throughput of SAME, as is shown in column 4. On the other hand, as in column 3, the I/O intensive process would not impact the SAME a lot.

In Admire RPServer, only the SAME encryption process is CPU intensive, other process like RTP parsing and file saving are either I/O intensive or not CPU intensive. Therefore, the SAME is effective enough for the high-volume real-time data in AdmireRPS.

## 6. Conclusions and Future Work

In this paper, a security scheme for RPS system is designed and implemented, which is based on the speed-adaptive media-data encryption (SAME) algorithm. Firstly, combining the block cipher and stream cipher algorithm, the basic SAME algorithm is proposed. Secondly, a packet-based SAME is proposed according to the implementation of data storage in AdmireRPS system. Thirdly, a queue theory based autoadaptive speed control mechanism for SAME is designed. Finally, the packet-based SAME algorithm and the speed control mechanism are integrated into AdmireRPS system. Theoretical analysis and experimental results show the security and speed of SAME are suitable for real-time applications. Furthermore, the proposed schemes can also be used in video surveillance and other video recording systems.

## Acknowledgment

This work was supported by the Major State Basic Research Development Program of China (973 Program) (Grant no. 2005CB321902) and Project (no. SKLSDE-2010ZX-06) of the State Key Laboratory of Software Development Environment.

## References

[1] S. Lian, *Multimedia Content Encryption: Techniques and Applications*, Auerbach Publication, Taylor & Francis Group, London, UK, 2008.

[2] X. Chen, M. Shilong, X. Ke, W. Lifeng, and L. Weifeng, "A dynamic optimal selective control mechanism for multi-datastream security in video conference system," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '07)*, pp. 871–874, Beijing, China, July 2007.

[3] S. Lian, "Secure video distribution scheme based on partial encryption," *International Journal of Imaging Systems and Technology*, vol. 19, no. 3, pp. 227–235, 2009.

[4] S. Lian and Y. Zhang, "Protecting mobile TV multimedia content in DVB/GPRS heterogeneous wireless networks," *Journal of Universal Computer Science*, vol. 15, no. 5, pp. 1023–1041, 2009.

[5] S. Lian and Z. Liu, "Secure media content distribution based on the improved set-top box in IPTV," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 560–566, 2008.

[6] T. Jin, J. Lu, and X. Sheng, "Admire—a prototype of large scale e-collaboration platform," in *Proceedings of the 2nd International Grid and Cooperative Computing Workshop*, vol. 3033 of *Lecture Notes in Computer Science*, pp. 335–343, Shanghai, China, 2004.

[7] T. Huang and X. Yu, "An adaptive mixing audio gateway in heterogeneous networks for ADMIRE system," in *Proceedings of the Grid and Cooperative Computing (GCC '03)*, vol. 3033

of *Lecture Notes in Computer Science*, pp. 294–302, Shanghai, China, 2004.

[8] F. Liu and H. Koenig, "A novel encryption algorithm for high resolution video," in *Proceedings of the 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '05)*, pp. 69–74, Washington, DC, USA, June 2005.

[9] H. Yin, C. Lin, F. Qiu, J. Liu, G. Min, and B. Li, "CASM: a content-aware protocol for secure video multicast," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 270–277, 2006.

[10] G. A. Spanos and T. B. Maples, "Performance study of a selective encryption scheme for the security of networked, real-time video," in *Proceedings of the 1995 4th International Conference on Computer Communications and Networks (ICCCN '95)*, pp. 2–10, Las Vegas, Nev, USA, September 1995.

[11] A. Servetti and J. C. De Martin, "Perception-based partial encryption of compressed speech," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 8, pp. 637–643, 2002.

[12] L. Qiao and K. Nahrstedt, "A new algorithm for MPEG video encryption," in *Proceedings of the International Conference on Imaging Science, Systems, and Technology*, Las Vegas, Nev, USA, 1997.

[13] L. Agi and L. Gong, "An empirical study of MPEG video transmissions," in *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pp. 137–144, San Diego, Calif, USA, 1996.

[14] T. Lookabaugh, I. Vedula, and D. C. Sicker, "Selective encryption and MPEG-2," in *Proceedings of the ACM Multimedia*, Berkeley, Calif, USA, 2003.

[15] J. Meyer and F. Gadegast, "Security Mechanism for Multimedia Data with the Example mpeg-1 Video," Project Description of SECMPEG, Technical University of Berlin, Germany, 1995.

[16] L. Tang, "Methods for encrypting and decrypting MPEG video data efficiently," in *Proceedings of the 4th ACM International Multimedia Conference & Exhibition (ACM Multimedia '96)*, pp. 219–229, Boston, Mass, USA, 1996.

[17] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm," in *Proceedings of the 6th ACM International Multimedia Conference*, pp. 81–88, Bristol, UK, 1998.

[18] S. Lian, Z. Liu, Z. Ren, and H. Wang, "Secure advanced video coding based on selective encryption algorithms," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 621–629, 2006.

[19] S. Lian, Z. Liu, Z. Ren, and Z. Wang, "Selective video encryption based on advanced video coding," in *Proceedings of Pacific-Rim Conference on Multimedia (PCM '05)*, vol. 3768 of *Lecture Notes in Computer Science*, pp. 281–290, 2005.

[20] S. Lian, G. Chen, A. Cheung, and Z. Wang, "A chaotic-neural-network-based encryption algorithm for JPEG2000 encoded images," in *Proceedings of the International Symposium on Neural Network (ISNN '04)*, vol. 3174 of *Lecture Notes in Computer Science*, pp. 627–632, Springer, Dalian, China, 2004.

[21] C.-P. Wu and C.-C.J. Kuo, "Efficient multimedia encryption via entropy codec design," in *Security and Watermarking of Multimedia Contents III*, vol. 4314 of *Proceedings of SPIE*, pp. 128–138, San Jose, Calif, USA, 2001.

[22] J. Wen, M. Severa, W. Zeng, M. H. Luttrell, and W. Jin, "A format-compliant configurable encryption framework for access control of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 545–557, 2002.

[23] L.-F. Wang, W.-D. Wang, J. Ma, C. Xiao, and K.-Q. Wang, "Perceptual video encryption scheme for mobile application based on H.264," *Journal of China Universities of Posts and Telecommunications*, vol. 15, supplement, pp. 73–78, 2008.

[24] S. Lian, J. Sun, J. Wang, and Z. Wang, "A chaotic stream cipher and the usage in video protection," *Chaos, Solitons and Fractals*, vol. 34, no. 3, pp. 851–859, 2007.

[25] A. S. Tosun and W. C. Feng, "Lightweight security mechanisms for wireless video transmission," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, 2001.

[26] C. Shi, S. Wang, and B. Bhargava, "MPEG video encryption in real-time using secret key cryptography," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '99)*, Las Vegas, Nev, USA, 1999.

[27] S. Lian, J. Sun, D. Zhang, and Z. Wang, "A selective image encryption scheme based on JPEG2000 codec," in *Proceedings of the The 2004 Pacific-Rim Conference on Multimedia (PCM '04)*, vol. 3332 of *Lecture Notes in Computer Science*, pp. 65–72, Springer, 2004.

[28] S. Lian, Z. Liu, Z. Ren, and H. Wang, "Secure distribution scheme for compressed data streams," in *Proceedings of the IEEE Conference on Image Processing (ICIP '06)*, April 2006.

[29] J. Gray and D. Patterson, "A conversation with Jim Gray," *ACM Queue Storage*, vol. 1, no. 4, 2003.

[30] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 1889, 1996.

[31] T. G. Robertazzi, *Computer Networks and System: Queuing Theory and Performance Evaluation*, Springer, Berlin, Germany, 2000.

[32] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, pp. 656–715, 1947.