ENHANCING COOPERATIVE PLAYBACK SYSTEMS WITH EFFICIENT ENCRYPTED MULTIMEDIA STREAMING

Giancarlo Fortino[1], Wilma Russo[1], Eugenio Zimeo[2]

[1]DEIS, University of Calabria, Via P. Bucci, Rende (CS), Italy
[2]RCOST, Dep. of Engineering, University of Sannio, Corso Garibaldi, Benevento, Italy

ABSTRACT

Distributed platforms for live and on-demand media streaming delivery such as content distribution networks and media on-demand systems, are being diffused mainly due to the widespread availability of IP-based, bandwidth-capable digital networks. Provision of multimedia group services is usually supported by transmitting media streams to subscribers organized in a multicast group. Although multicast streaming saves bandwidth and improves scalability, it is prone to be hacked. This paper proposes an efficient technique centered on the Blowfish symmetric encryption algorithm for securing media streams based on the Real-time Transport Protocol (RTP). The developed technique along with an ad-hoc key distribution mechanism is seamlessly embedded into our Java-based cooperative playback system - ViCRO^C, which allows multicast transmission on-demand of archived multimedia sessions to a cooperative group of clients.

1. INTRODUCTION

Multimedia streaming on the Internet [7] is being fostered by advances in IP-compliant communication infrastructures (last-mile, 3G cellular, and satellite networks), standard multimedia internetworking protocols and architectures, and, notably, commercial interests of public and private companies foreseeing a new, easily reachable entertainment market.

In the last years, the research focus on multimedia systems is shifted from stand-alone and/or distributed mono-thematic applications to high-level networks or platforms providing media services ranging from TV broadcast, multi-party conferencing, to video on-demand. Such high level platforms, often named Content Distribution Networks (CDN) [6], deliver multimedia services on demand to their customers.

Since customers subscribe for a service, they not only expect to receive the service according to a certain degree of quality of service (QoS) but also require that the service be secure. Although, from the service provider point of view, security is convenient to be applied to any service (e.g., pay per view), it is even more necessary, from the customer perspective, for those media services which involve a selected, collaborative group of participants.

Significant examples are private multi-party videoconferencing [7] and cooperative playback sessions [8]. Both are usually based on the transmission of media streams to an IP-based multicast group. IP multicast does make it simple for an eavesdropper host to anonymously join a multicast group and receive traffic destined to that group without the legal members knowledge.

The currently most adopted solution is to encrypt the media streams so fulfilling the confidentiality requirement. Media encryption can occur at several points in the TCP/IP stack. Since the IP security architecture [18] is still to be deployed, often encryption is realized at RTP (Real-time Transport Protocol) level. The basic RTP protocol [22] specifies a standard way to encrypt and decrypt RTP and RTCP packets using symmetric encryption schemes such as DES [5] and also proposes a mechanism for generating a key. Recently a new security framework has been proposed along with a reference implementation, Secure RTP (SRTP) [2].

However, all the proposals suggest that security features are implemented with minimal delay and jitter. It should be evident that with huge transmission rates even a small timing overhead easily amounts to huge loss of bandwidth.

This paper proposes an efficient technique for encrypting multicast RTP-based media streams centered on a symmetric encryption algorithm that improves DES performance. It has been first integrated in the Java Media Framework [16], by implementing custom plug-ins, and then used to enhance the functionality of ViCRO^C, our cooperative playback system (CPS) [8, 9].

ViCRO^C is a media on-demand system based on an adaptation of the RTSP [23] protocol (called MACπ) atop of the lightweight reliable multicast protocol (LRMP) [20] which allows an explicitly formed group of clients to cooperatively share the control of a multimedia session playback. Authentication and key distribution are embedded in a SDP/SIP-based mechanism for group organization. Security is now a distinctive feature of our system compared to related CPSs in literature [14, 25].

The rest of the paper is organized as follows. §2 provides a brief overview of CPSs. §3 introduces the adopted efficient encryption technique. In §4 the integration of the proposed technique into ViCRO^C is elucidated. Finally conclusions are drawn and directions for future work are provided.

2. COOPERATIVE PLAYBACK SYSTEMS

Cooperative playback systems (CPSs) are multicast-based media on-demand systems which further provide cooperative remote control of playbacks to an explicit group of collaborative users [8]. The main functionality of a CPS can be summarized as follows:

- Group organization, which contains group formation and group management. In particular, the latter allows creating a users' group working on and controlling the same playback session.
- Media streaming, which transmits multicast media streams based on RTP to a demanding customers group.
- Control sharing, which enables a group of users to cooperatively control a media streaming session.
- Joint-work, which allows the playback session users to collaborate with each other by questioning on the session contents.

A CPS can be fruitfully exploited to support the collaborative learning on-demand paradigm [9] which enables a virtual class of students to go over an archived lesson and exchange questions so as to cooperatively construct new knowledge. Such an activity demands for security at different levels (media, control and joint-work) as well as initial authentication during group organization.

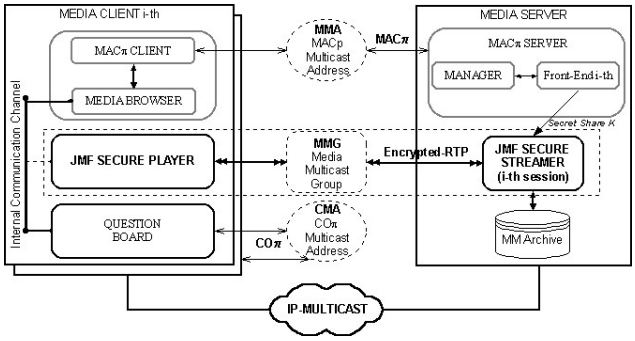

Figure 1. ViCRO^C architecture.

To date, few systems, MASH Rover [25] and ViCRO^C [8, 9], share all the functionality of a CPS. In particular the architecture of ViCRO^C is depicted in fig. 1 which reports the basic components and their protocol-based interaction and highlights the security-enhanced blocks. Media streaming and playing are supported by the Streamer and Player components which are based on JMF in which security is integrated (see §3), and are respectively located at the Media Server (MS) and Media Client (MC) sites. Playback control sharing is enabled by MACπ (Multicast Archive Control Protocol) which is a multicast version of RTSP [23]. Collaboration among users is based on the COπ (COllaborative Protocol) which allows multicast exchanging of questions and annotations. Both MACπ and COπ are based on a scalable reliable multicast transport protocol [20]. The Multimedia Archive keeps stored MPEG and RTP-based media files [10].

3. EFFICIENT ENCRYPTION OF RTP-BASED MEDIA STREAMS

Sending multimedia data, encapsulated in RTP packets, in clear after a process of session initialization and authentication allows RTP data to be recorded or relayed by hackers.

At a large extent, protecting multimedia sessions means to guarantee authentication, confidentiality and integrity. In this paper we focus only on authentication and confidentiality and in particular we propose: (i) a technique for protecting multimedia contents transported over RTP, by using a symmetric encryption algorithm, and (ii) a related schema for exchanging private keys among an MS and all MCs joining a cooperative playback session in our ViCRO^C system.

Although encryption overhead is typically minor compared to CPU requirements of modern compression algorithms for voice and video, real-time constraints imposed by RTP require an efficient encryption algorithm implementation in order to assure a correct (i.e., without packet or frame losses) reproduction of A/V contents at destination. A small encryption overhead is not very noticeable if connection speeds available are much lower than the encryption throughput [13], but it is wasteful if data compression is performed by a dedicated processor and data throughput is high (i.e., 10-30 Mbps for HDTV with MPEG-2) or videos are pre-registered in compressed format on a video server.

A widespread agreement [2, 4] states that an ideal encryption scheme for multimedia streaming:

- must be fast;
- should not expand the message size, in order to efficiently use the available bandwidth;
- should avoid end-to-end encryption of RTP headers to allow for header compression over the air link.

The above requirements suggest: (1) to use encryption for protecting only the data area of RTP packets, whereas RTP headers can be transported in clear; (2) to select encryption schemes that do not expand the message size and (3) do not require much memory. Following these indications, H.323 [12] provides a little support for multimedia data encryption by an extension of H.235 [11], currently implemented only by a few vendors. This extension allows using one of the following symmetric encryption schemes to encrypt only RTP packet payloads: DES [5], RC2 [21], triple-DES [17], AES [1].

Therefore, in order to enhance ViCRO^C with security features, we have analyzed some symmetric encryption techniques: DES, Blowfish [26] and Rijndael (AES) [1], for the purpose of selecting the fastest one. The choice of these techniques is motivated by the following reasons: DES is the most famous and used symmetric cipher book; Rijndael was proposed as an alternative to DES for multimedia streaming with thin clients [4]; Blowfish is flexible and efficient.

We have conduct a simple experiment (whose results are reported in table 1) with Java-based implementations of the cited encryption algorithms in order to extract a characteristic parameter, that we have named "time for encrypting a byte". The experiment has been performed on a PC with a CPU Pentium II, 350 MHz. The adopted DES and Blowfish implementations are provided by Sun [15] whereas the Rijndael implementation is provided by Bouncy Castle [3]. We have used: (1) with Rijndael, blocks of 128 bits and a key of 128 bits (the minimum key length with this algorithm); with DES, blocks of 64 bits and a key of 64 bits; with Blowfish, blocks of 64 bits and a key of 128 bits. DES and Blowfish algorithms have been used in ECB (*Electronic Code Book*) mode, by adopting the PKCS5Padding scheme, which guarantees that the size of input data is a multiple of 64 bits, as ECB mode requires to correctly operate. Although the ECB mode is the simplest and fastest operational mode, it has problems depending on:

- production of independent encrypted blocks that may be reordered on the network without the receiver is able to identify the alteration;
- cryptanalysis attacks in the presence of repeated patterns.

Hence, we have used the ECB mode only for performance measurements, whereas to avoid the cited problems the CBC (*Cipher Block Chaining*) mode (applied at each RTP packet to tolerate losses) has been used in the CPS implementation.

Time for encrypting a byte (ms)			Max throughput (Mbps)		
DES	Blowfish	Rijndael	DES	Blowfish	Rijndael
8.28E-4	4.68E-4	3.51E-3	9.66	17.09	2.27

Table 1: Time for encrypting a byte of data using DES, Blowfish and Rijndael

As table 1 shows, Blowfish [26] is the fastest algorithm. It was designed in the 1993 by Bruce Schneier as an alternative symmetric encryption algorithm to DES (*Data Encryption Standard*) and IDEA [19] (*International Data Encryption Algorithm*). Blowfish is a block cipher, with blocks of 64 bits, based on the *Feistel* network [27]. The length of its keys is variable from 32 to 448 bits.

Blowfish's notoriety is mainly due to the method of key scheduling adopted. The round subkeys and the whole content of S-boxes are created by multiple iterations of the cipher book. This feature improves security by making it difficult a full scanning of the key space even when the key length is small. So, a satisfactory security is guaranteed even by short keys, which allow data to be quickly encrypted. However, to achieve this goal, Blowfish requires an initialization phase to extract both subkeys and S-boxes from the key and to store them for the successive encryption phase. In particular, for storing 18 32 bit subkeys and 4 S-boxes, each with 256 32 bit entries, 4168 bytes are required. So, Blowfish could not be used efficiently in very small-memory equipped devices such as smart cards. Nevertheless we can conclude that Blowfish is ideal to efficiently assure confidentiality to ViCRO^C, since it doesn't use thin clients (such as mobile phones) so far.

4. INTEGRATING SECURITY INTO VICRO^C

ViCRO^C has been enhanced with authentication and secure streaming. The former is performed during the group organization, whereas the latter regards the encryption of RTP packets payload sent by the Streamer to the MCs' Players.

4.1. Secure Group Organization

Each MC wishing to join a cooperative session has to contact the MS in order to authenticate itself and receive the session key. The scheme adopted has been inspired by existing centralized Group Key Management (GKM) schemes [27] and has been integrated with SDP/SIP [24]. In particular, it is based on asymmetric cryptography, a Key Distribution Center (KDC) and a Certification Authority (CA), (see fig. 2). The KDC may coincide with the MS since the small number of group members in a CPS does not introduce scalability issues.

In order to join a secure group, a MC sends a unicast SDP/SIP message to the MS. The MS replies to the MC with the session id (sID) and asks the MC for sending its identity. At this point the RSA [7] algorithm is used both to authenticate the MC and the MS and to exchange the session key. In particular, the scheme consists of several steps. (-1-) The MC sends its ID, the session ID, and the ID encrypted with its RSA private key (D_{MC}) and the result with the public key (E_{MS}) of the MS. To this end, each MC must know the CA and asks it for receiving the public key of the MS encapsulated in a digital certificate. (-2-) Upon reception of the SDP message, the MS decrypts the message by using its private key (D_{MS}) then queries the CA using the MC's ID in order (-3-) to obtain the MC's public key (E_{MC}).

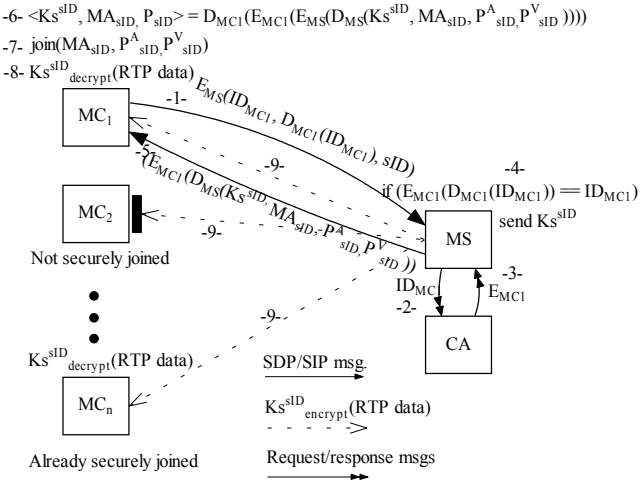

Figure 2. Distributed schema for secure cooperative group organization.

E_{MC} is then used to decrypt the encrypted part of the SDP message for extracting the MC's ID; (-4-) if the ID extracted coincides with the ID transported in clear in the SDP message, the MC's identity is authentic and so (-5-) the MS sends the MC a SDP message containing the session key (Ks^{sID}) and the session information encrypted with D_{MS} and in turn the result with E_{MC}. This way, if the MC knows the public key of the MS, it is sure that the MS' identity and the session key received are valid. By using the MS public key and its private key, the MC (-6-) extracts from the received message the session key, the multicast addresses and ports. At this point, the MC joins (-7-) the multicast A/V session and becomes ready to receive and play encrypted RTP-based streams (-8-) sent from the MS (-9-).

In order to avoid cryptanalysis attacks, the MS periodically generates a new session key and transfers it, encrypted with the old session key, to MCs by using multicast. To guarantee backward confidentiality, the session key is also regenerated whenever a new MC joins the group. Currently, forward confidentiality is not efficiently supported.

4.2 Secure RTP-based streaming

Confidentiality has been easily and seamlessly added to ViCRO^C through the JMF-based implementation of the streamer and the player and by exploiting the customizability of the JMF architecture.

JMF [16] is a Component Framework based on plug-ins, which allows a special software component, called *processor*, to be customized. A processor is composed of smaller components: some of which (*Demultiplexer*, *Multiplexer*) used for all the tracks of the multimedia data and others (*Codec*, *Effecter*, *Renderer*) organized in chains, one for each track. Such components can be added, customized or removed in order to guarantee the desired behavior to the processor. In order to transmit multimedia data over the Internet, each track of the

output data source of the processor is hooked to a specific component, called *Session Manager (SM)*, which is able to establish a unicast or multicast multimedia session and send multimedia data encapsulated in RTP packets.

Due to this software organization, two choices for adding confidentiality to the JMF implementation of ViCRO^C are feasible: (1) modifying the SM, in order to encrypt all RTP packets coming from the processor; (2) adding or customizing a processor component. We have exploited the second approach in order to avoid modifications of JMF source code and to integrate the encryption with the multimedia data processing. In fact, SM is not based on plug-ins and so its modification requires source code availability. On the other hand, each video frame has to be subdivided in fragments in order to fit an RTP packet. Usually, to avoid IP fragmentation, the size of RTP packets is set smaller than the MTU (Maximum Transmission Unit) of the used physical network. In addition, the size can be purposely chosen to assure an effective overlapping of data processing (video compression, frame extraction, fragmentation, encryption) and transmission.

Therefore, in order to seamlessly integrate security in the Streamer and Player, we have implemented a custom *CoDec* component of the processor, both for the transmission and for the reception of data, by integrating the Blowfish algorithm based on a 128 bit key and on the CBC operational mode.

A codec is a generic component whose role is to process incoming data and produce data for the successive codec or for another component. The codecs, named *Encrypter* and *Decrypter*, are positioned after the *Packetizer* and before the *Depacketizer*, respectively. Packetizer and Depacketizer are other codecs used to transport video frames and audio samples over RTP. A different packetizer/depacketizer pair is necessary for audio and video streaming. For audio streaming, several samples are aggregated in a single RTP packet, whereas for video streaming each frame is transported by several RTP packets. Therefore, implementing the encryption before the packetizer implies to encrypt the whole frame before transmitting it in RTP packets, so limiting parallelism, whereas providing encryption after the packetizer increases parallelism and allows the packetizer to chose the size of RTP packets on the basis of the encryption throughput.

Currently, we have successfully tested the system with MPEG-1 compressed files with bit rates ranging from 150 Kbps to 1.5 Mbps.

5. CONCLUSIONS

In this paper we have described the enhancement of our Java-based cooperative playback system - ViCRO^C - with security features. Confidentiality is obtained by integrating the Blowfish symmetric encryption algorithm in a JMF plug-in, in order to efficiently encrypt/decrypt the payload of RTP packet transmitted over IP-multicast. In addition, in order to allow a MC to join a secure multimedia session, we have developed a secure group organization protocol to acquire the cooperative session key which is based on the RSA asymmetric encryption algorithm and a dedicated certification authority.

Future work is geared at: (i) completing the performance evaluation of the security-enabled system; (ii) encrypting the control commands and the collaboration messages.

6. REFERENCES

[1] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] Baugher et al., "The Secure Real-time Transport Protocol", *Internet Draft*, draft-ietf-avt-srtp-05.txt, June 2002.

[3] Bouncy Castle. 2002. http://www.bouncycastle.org.

[4] R. Blom, E. Carrara, K. Normann, and M. Naslund, "RTP Encryption for 3G Networks", *Internet Draft*, draft-blom-rtp-encrypt-00.txt, Nov. 2000.

[5] D. Coppersmith, "The Data Encryption Standard (DES) and its Strength Against Attacks", *IBM Journal of Research and Development*, May 1994.

[6] C.D. Cranor, M. Green, C. Kalmanek, D. Shur, S. Sibal, C.J. Sreenan, and J.E. Van der Merwe, "Enhanced Streaming Services in a Content Distribution Network," *IEEE Internet Computing, 5*(4), 2001, pp 66-75.

[7] J. Crowcroft, M. Handley, and I. Wakeman, "*Internetworking Multimedia*," Morgan Kaufmann Publishers, San Francisco, 1999.

[8] G. Fortino and L. Nigro, "A Cooperative Playback System for on-demand Multimedia Sessions over Internet", *Proc. of IEEE Int'l Conference on Multimedia and Expo*, New York (USA), 2000.

[9] G. Fortino and L. Nigro, "Collaborative Learning on-Demand on the Internet MBone", in *Usability Evaluation of Online Learning Programs*, C. Ghaoui Ed., Idea Publishing Group, USA, 2002, to appear.

[10] G. Fortino, G. Confessore, and A. Mantuano, "Design and Implementation of a Dynamic VRML-browsable, Movie On-Demand System Distributed over Internet," *Proc. of IEEE Int'l Conference on Multimedia and Expo (ICME'02)*, Lausanne (CH), Aug. 2002.

[11] *H.235*. Security and encryption for H-Series multimedia terminals. 2000. http://www.itu.int/publibase/itu-t/.

[12] *H.323*. Audiovisual and Multimedia Systems, Packet-Based Multimedia Communications Systems, 2000. http://www.itu.int/publibase/itu-t/.

[13] V. Hallivuori, "Real-time Transport Protocol (RTP) Security," *Seminar on Network Security,* Helsinki Univ. of Technology (FI), 2000.

[14] W. Holfelder, "Interactive remote recording and playback of multicast videoconferences", *Proc. of IDMS'97*, Darmstadt, Germany, Sept. 1997.

[15] Java Cryptography Extension (JCE), Sun Microsystems, Inc., 2002. http://java.sun.com/products/jce/.

[16] Java Media Framework API. Sun Microsystems. 2002. http://java.sun.com/products/javamedia/.

[17] B. Kaliski and M. Robshaw, "Multiple Encryption: Weighing Security and Performance", *Dr. Dobb's Journal*, Jan. 1996.

[18] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," *IETF RFC 2401*, Nov. 1998.

[19] X. Lai, "On the Design and Security of Block Ciphers", Konstanz, Germany: Hartung-Gorre, 1992.

[20] Light-weight Reliable Multicast Protocol. http://webcanal.inria.fr/lrmp/, 1998.

[21] R. Rivest, "A Description of the RC2 Encryption Algorithm", *Internet Draft,* draft-rivest-rc2desc-00.txt, June 1997.

[22] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *IETF RFC 1889*, Jan. 1996.

[23] H. Schulzrinne A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP) ", *IETF RFC 2326*, Apr. 1998.

[24] H. Schulzrinne, E. Schooler, and J. Rosemberg, "SIP: Session Initiation Protocol", *IETF RFC 2543*, March 1999.

[25] A. Schuett, S. Raman, Y. Chawathe, S. McCanne, and R. Katz, "A Soft State Protocol for Accessing Multimedia Archives," *Proc. of NOSSDAV*, Cambridge, UK, July 1998.

[26] B. Schneier, "Description of a New Variable-length Key, 64-Bit Block Cipher (Blowfish)", *Proc. of Cambridge Security Workshop*, Springer-Verlag, 1994.

[27] W. Stallings, "Cryptography and Network Security: Principles and Practice", 2nd ed., Prentice Hall, 1999.