Implementation of an authentication protocol for sending audio-video information in Java

Bogdan Groza[1], Dorina Petrica[1], Simona Barbu[2], Mariana Bilanin[2]
Politehnica University of Timisoara[1], Alcatel-Lucent[2], Romania
E-mail: {bogdan.groza, dorina.petrica}@aut.upt.ro, {simona.barbu, mariana.bilanin}@alcatel-lucent.ro

***Abstract* — an application for assuring the authenticity of audio-video information is developed. The application is implemented in Java by using Java Media Framework to send audio and video information over RTP (Real-time Transport Protocol). In order to guarantee that information is not altered during transmission over public networks by malicious adversaries some cryptographic functions and protocols are used for achieving information authenticity. More concrete, a cryptographic protocol which uses Message Authentication Codes and elements of a one-way chain as keys is implemented. The solution proves to be efficient and the computational costs are kept to a minimum.**

I. INTRODUCTION

Sending audio and video information is a common demand in the present days. Information authenticity refers to a guarantee over the source of information; this implies that information was not altered during transmission. However, assuring the authenticity of media information by cryptographic techniques is quite often neglected. Consequently the degree of trust in audio-video information sent over public networks is limited since there is no proof that the received information was not altered by malicious adversaries during transmission. In this context assuring the authenticity of audio and video information is a subject of great interest and for this purpose cryptographic techniques are the only alternative, since cryptography is the only security guarantee when we are working with information.

This paper is concerned with the development of a Java application that can be used to capture audio and video information and then send it to some remote computers from a public network. More concrete, the application captures images from a web-cam connected to a computer and sends the audio-video information through RTP to other computers. A cryptographic authentication protocol is implemented in order to prevent information from being altered during transmission. The Java environment provides good support for both managing multimedia streams and implementing cryptography.

This paper is organized as follows. In section 2 we review some cryptographic primitives that can be used to guarantee the authenticity of information and some cryptographic protocols that can be build upon them. Section 3 holds details about our application and some experimental results while section 4 holds our conclusions.

II. CRYPTOGRAPHIC CONSTRUCTIONS

A. Cryptographic primitives

We proceed by a brief account of some cryptographic primitives that can be used to guarantee the authenticity of a message. There are two constructions that can be used to assure a guarantee over the source of a message:

a) **Message Authentication Codes (MAC)**. A MAC, denoted in this paper as $MAC_k(M)$ where k is the key of the MAC and M is the message which is to be proved authentic, is a symmetric primitive for assuring authenticity which uses a secret shared key between the participants. The great advantage of MAC codes is that they are easy to compute and require only simple operations. The problem that MAC codes are introducing is that as the number of participants increases so does the number of secret shared keys. In general, for a broadcast scenario where a server broadcasts information to n participants, n distinct keys are needed and, more, the server needs to compute n different MAC codes for each broadcasted message, even if the message is the same for all participants. In the case of a communication where each entity needs to send authentic information to any other entity, for n entities the number of keys increases to $n \cdot (n-1)/2$. Fortunately, improvements can be done, some authentication protocols that remove these disadvantages are presented in the next section.

b) **Digital Signatures**. Digital signatures are asymmetric primitives that use a private key to sign a given message and a public key to verify the signature. Digital signatures assure the non-repudiation of information which means that in case of dispute information can be proved to originate from a particular entity to any neutral party. The advantage is that the same public key can be used by any number of entities in order to verify the source of a signed message. Therefore the number of keys does not increase with the number of participants, each entity needs to store only its own private key as a secret and has to be informed in an authentic manner of the public keys that are used by the other entities. Also a digital signature on some message can be verified by any other participant that knows the public key. MAC codes are not a substitute for digital signatures since they do not provide non-

repudiation, however when referring strictly to authenticity the drawback of digital signatures in front of MAC codes is that they are more computationally intensive (digital signatures involve more complex arithmetic operations such as multiplication and exponentiation over groups of large integers). More exactly they are thousands times more expensive than a MAC code. Since they are so computationally intensive, digital signatures are of course inefficient for our application. It should be also underlined that digital signatures can also be constructed on simple one-way functions reducing in this way some of the computational overhead; these digital signatures are called one-time signatures because they can be used only once, by using authentication trees they can be used multiple times. However, even this kind of digital signatures are still more expensive than simple MAC codes.

In order to set a more accurate look on the computational requirements of these cryptographic primitives in section 3.2 some experimental results for the computational time of hash functions, MAC codes and some public key operations are given.

B. *Cryptographic protocols*

As stated in the previous section, using a digital signature is too computational intensive for assuring the authenticity of audio-video information; therefore the best alternative that we have is to use a MAC code. In order to make such a protocol useful we must avoid the use of secret shared keys, since using n distinct secret shared keys on the server side and computing n distinct MAC codes for each message will of course decrease the performance of the protocol. Fortunately, a good solution for this purpose exists: to disclose the key of the MAC only after all the entities that receive information have stored the MAC computed on the particular message $MAC_k(M)$ (in this way the same key and the same MAC on some message can be used for multiple entities). Of course, after key k is disclosed the problem that we have is that this key cannot be used again. However, there is an elegant solution that can be used to remove this problem. The solution is to use as keys elements from a one-way chain. In this way each disclosed key can be used as a commitment for a new key which is used to compute a new MAC and so on. A one-way chain is a recurrent array generated by the successive composition of a one-way function; each element of a one-way chain can be used as a key and is defined as follows:

$$k_i = f^{\eta - i}(x_0), i = \overline{1,\eta} \quad (1)$$

Here k_i is the i^{th} key, η is the length of the one-way chain, x_0 is a random element value and f is a one-way function. Usually in constructing a one-way chain a cryptographic hash function [3] is chosen for implementing f; therefore, one-way chains are usually referred as hash chains. However other cryptographic functions, such as encryption functions, can be also used for this purpose. It is also relevant that using the discrete power function over groups of large integers [5] has the advantage that the length of the chain becomes unbounded since the computational time will depend only logarithmically on the length of the chain.

The first proposal for the use of one-way chains in authentication is in [8] and the first use of such a protocol in a system is in [7], however, the system from [7] is insecure. A more recent proposal of authentication protocols, which is an important step towards the most recent protocols, is in [1]. One-way chains are used by some recent protocols proposed in [2], [4], [5], [9], [10], [11], [12] for assuring information authenticity. These protocols can be divided in two categories according to the construction principles that are used:

a) **Time synchronization.** Disclosing each key at precise time intervals and using a loose time synchronization which lets each client have an upper bound on the time from the sender's side is probably the best solution for a broadcast protocol. This principle is used in the TESLA protocol that was proposed by Perrig et. al. in [9], [10], [11], [12]. Several variants of the TESLA protocol were proposed and it was used even in constrained environments such as sensor networks where computational power and communication abilities are drastically limited. A different protocol where the one-way chains are constructed with the discrete squaring function and allows broadcasting over long time periods (there is almost no limitation on the duration of such a broadcast) is in [5].

The use of a protocol based on time synchronization is useful for broadcasting authentic audio-video information. In this case, in the $i^{th}, i = \overline{1,\eta}$ communication session, the same packet P_i containing authentic information is sent to all our clients:

$$\text{Server} \rightarrow \text{Clients: } P_i = \left\{ i, M_i, MAC_{KD(k_{i+1})}(M_i), k_i \right\}$$

Here i is the session counter, M_i is the message from the session (for example some part of the audio-video content), k_i is the current session key and $KD(k_{i+1})$ is a key derivation process used to derive the key of the MAC from the forthcoming session key.

b) **Authentic confirmation.** Waiting for an authentic confirmation about the arrival of the MAC is the principle used in [2], [4]. Of course, an element of a one-way chain can play the role of such a confirmation.

The use of an authentic confirmation is useful for the case when a response from the other communication participant is expected, this is important for our scenario in the case when the camera is controlled from a remote computer and both commands and responses are sent and received from the camera. The importance of such an authentication protocol for remote control systems is discussed in [6]. For this case the $i^{th}, i = \overline{1,\eta}$ communication session between two entities, denoted as *A* and *B*, is as follows:

$$A \rightarrow B:\ M_{A,i}, MAC_{KD(k_{A,i+1})}\left(M_{A,i}\right), k_{A,i}$$
$$B \rightarrow A:\ M_{B,i}, MAC_{KD(k_{B,i+1})}\left(M_{B,i}\right), k_{B,i}$$

The significance of the notations is the same as previously. Note that in this case the communication takes place between only one sender and receiver. For the case of multiple receivers, each receiver needs to compute its own one-way chain and the sender must wait for a confirmation from all receivers before sending a new packet, this mechanism was proposed in [2]. Such a protocol is not very efficient for a broadcast scenario, but is extremely useful when *B* has also to send information to *A* (note that the previous type of protocol allows only a one-way communication).

III. JAVA IMPLEMENTATION

A. Implementing Cryptography

Java [13] provides support for most cryptographic primitives, such as hash functions and encryption functions. It also provides support for working with large integers with the BigInteger class, these operations are needed to perform public key operations. In order to provide a more accurate look on the computational requirements of cryptographic primitives some experimental results are given in Table 1. Further details on implementing cryptography can be found in Java documentation [13].

TABLE 1. COMPUTATIONAL TIME FOR SOME CRYPTOGRAPHIC PRIMITIVES IN JAVA.

Cryptographic function		CPU		
		Intel Centrino 1.7 GHz	Intel Dual Core 1.6 GHz	Intel Core Duo 6600 2.4 GHz
Modular exponentiation, basic operation for a digital signature (module and exponent size in right column)	512	7.8 x 10^{-3} s	6.1 x 10^{-3} s	3.1 x 10^{-3} s
	1024	48.4 x 10^{-3} s	44.6 x 10^{-3} s	20.3 x 10^{-3} s
	2048	359.4 x 10^{-3} s	323.8 x 10^{-3} s	153.2 x 10^{-3} s
Mac with SHA1	160	0.00859 x 10^{-3} s	0.00812 x 10^{-3} s	0.00406 x 10^{-3} s
Mac with MD5	128	0.00579 x 10^{-3} s	0.00354 x 10^{-3} s	0.00219 x 10^{-3} s
Sha-1	160	0.00281 x 10^{-3} s	0.00212 x 10^{-3} s	0.00109 x 10^{-3} s
Sha-256	256	0.0086 x 10^{-3} s	0.00592 x 10^{-3} s	0.00282 x 10^{-3} s
Sha -384	384	0.01359 x 10^{-3} s	0.01234 x 10^{-3} s	0.00579 x 10^{-3} s
Sha-512	512	0.02625 x 10^{-3} s	0.02324 x 10^{-3} s	0.01141 x 10^{-3} s
MD5	128	0.00156 x 10^{-3} s	9.5E-4 x 10^{-3} s	4.6E-4 x 10^{-3} s

B. Sending media streams

It is commonly known that the TCP/IP (Transmission Control Protocol/Internet Protocol) is beneficial for its reliability, but guaranteeing reliable data transfer slows the overall transmission rate and is unnecessary for most multimedia applications. By contrast, the UDP (User Datagram Protocol) is unreliable but consumes less processing power than TCP/IP. Therefore UDP can be used more efficient for transmitting audiovisual content. Still UDP is not optimized for multimedia transfer and for this purpose RTP (Real-time Transport Protocol) is built on top of UDP and is beneficial for applications used to transmit real-time data, in our particular case audio-video information.

Since RTP is built on an unreliable protocol, it does not guarantee whether RTP packets are being transported successfully. RTP is augmented by a control protocol (RTCP), to allow monitoring the network traffic and track the session's participants. Both RTP and RTCP are independent of the underlying transport and network layers. RTP can be used for both unicast and multicast network; in a unicast network the source sends separate copies of the data to each destination while in a multicast network, the source sends data only once and the network is responsible for sending the data to multiple locations. Multicasting is more efficient for many multimedia applications, such as video conferences. The standard Internet Protocol (IP) supports multicasting.

For dealing with audio-video streams Java Media Framework (JMF) API was used [15]. JMF is an application programming interface for incorporating time-based media into java programs. It provides support for media playback, capturing and storing media data and performing custom processing on media data streams. It supports media data reception and transmission using RTP and RTCP. JMF provides interfaces/classes that handle the construction of Players, Processors, DataSources and DataSinks (Manager), describe the location of media stream (MediaLocator), manage the transfer of media-content (DataSource).

C. The developed application and experimental results

A client-server application was developed in Java; the general setting of our application is suggested in figure 1. The application is based on the solutions for JMF given by Sun Developer Network [14].

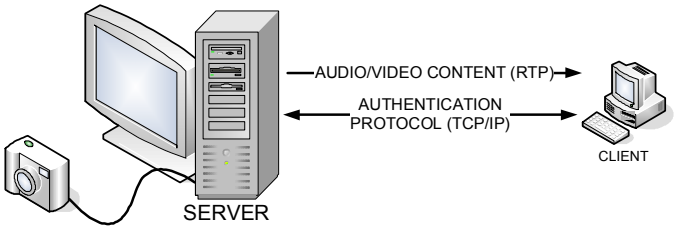

Figure 1. Application setting (media streams are sent over RTP while authentication is done over TCP/IP).

In order to preserve the efficiency, our application sends media streams through RTP while authentication is done through TCP/IP. The authentication protocol that is used, is a variation on the DeMA/DiCA protocol from [4]. We decide to use this protocol in order to let the client request the authenticity of packets at its own choice from the server, making in this way the protocol more flexible. Each session of the protocol is as follows:

Session i

$$A \rightarrow B:\ MAC_{KD(k_{A,i+1})}\left(H\left(M_{i-2}\right)\right), k_{A,i}$$
$$B \rightarrow A:\ H\left(M_i\right), MAC_{KD(k_{B,i+1})}\left(H\left(M_i\right)\right), k_{B,i}$$

Here $H\left(M_i\right)$ is the hash of the message for which the client request to be authenticated by the server, this request is authenticated by the client with $MAC_{KD(k_{B,i+1})}\left(H\left(M_i\right)\right)$ and the server answers by sending a MAC code on this hash (note that

$H(M_i)$ will prove to be authentic only in session *i+1* while the response of the server can be sent only in session *i+2* - that is why A sends to B the value of $MAC_{KD(k_{A,i+1})}\left(H\left(M_{i-2}\right)\right)$ only in session *i*). In order to store the hash of the messages sent to clients the servers stores them in a HashTable, this data structure allows efficient store and retrieve of objects in Java. Also note that this protocol description holds only for one sender *A* and one receiver *B*, but it can be easily be extended for any number of participants.

The server application is responsible for capturing audio-video data, compressing data and encapsulating the data in a format suitable for transmission over the network. The clients are responsible for creating RTP sessions to receive audio/video streams. The video streams are compressed in JPEG format using the software codec. The audio data is compressed using a RTP-specific format. After compression, both audio and video streams are transmitted over the network using RTP over UDP. The system also provides QoS features; the receivers can get the reports about the quality of the audio/video streams which include information like bit rate, packets received and packets loss. Authentication is done with the previously described protocol on a separate channel via TCP/IP.

We tested our application on two Dell Optiplex 745 with Intel Core Duo 6600, at 2.4 GHz connected on a local area network by a MSI RG54SE router and video information was acquired from a Logitech QuickCam Chat webcam (video format was RGB at 320x240 resolution). Some experimental results on sending authentic audio-video information with our application are in tables 2 and 3; these results were determined by using two interfaces:*GlobalTransmissionStats* and *GlobalReceptionStats*, further details about them can be found in JMF [15].

TABLE 2. TRANSMITION STATISTICS

Server	Client	RTP sent	RTCP sent	Local Collisions	Transmit Failed
Dell Optiplex 745, Core Duo 6600 @ 2.4GHz	Dell Optiplex 745, Core Duo 6600 @ 2.4GHz	1746	26	0	0

TABLE 3. RECEPTION STATISTICS

Reception Statistic (for 60 seconds)	Client: Dell Optiplex 745, Core Duo 6600 @ 2.4GHz
Bad RTCP packets	0
Bad RTP packets	2
Packets received	1776
RTCP received	53
SR received	25
Local Collisions	0
Malformed RR	0
Malformed SDES	0
Malformed Bye	0
Transmit failed	0

IV. CONCLUSIONS

A Java application was developed for sending authentic audio-video information that is captured from a remote camera. The Java environment proved to be very useful since both support for managing audio-video content is available in JMF and cryptographic support is present as well. The experimental results show that these protocols are efficient for sending audio and video information and therefore can be used in practice. As future work we are interested in building a complete solution for sending authentic audio-video information which can be efficiently used in many unicast and broadcast scenarios.

Acknowledgements: This work was partially supported by national research grant MEDC-CNCSIS TD-122/2007.

REFERENCES

[1] R. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas, R. Needham, "A New Family of Authentication Protocols", ACM OSR, 1998.

[2] F. Bergadano, D. Cavagnino, B. Crispo, "Individual Authentication in Multiparty Communications". Computer & Security, Elsevier Science, vol. 21 n. 8, 2002, pp.719-735.

[3] FIPS 180-1, National Institute of Standards and Technology (NIST). "Announcing the Secure Hash Standard", U.S. Department of Commerce, 1995.

[4] B. Groza, "Using one-way chains to provide message authentication without shared secrets", Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SecPerU 2006, IEEE Comp. Soc., 2006.

[5] B. Groza, "Broadcast authentication protocol with time synchronization and quadratic residues chains", Second International Conference on Availability, Reliability and Security (ARES'07), pp. 550-557, IEEE Comp. Soc., 2007.

[6] B. Groza, T.L. Dragomir, "On the use of one-way chain based authentication in secure control systems", Second International Conference on Availability, Reliability and Security (ARES'07), pp. 1214-1221, IEEE Comp. Soc., 2007.

[7] N. Haller, C. Metz, P. Nesser, M. Straw, "A One-Time Password System", RFC 2289, Bellcore, Kaman Sciences Corporation, Nesser and Nesser Consulting, 1998.

[8] L. Lamport, "Password Authentication with Insecure Communication", Communication of the ACM, 24, 770-772, 1981.

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, "SPINS: Security Protocols for Sensor Network", Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM, 2001.

[10] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol", Proc. of ACM Conference on Computer and Communications Security, 2001, pp.28-37.

[11] A. Perrig, R. Canetti, J. D. Tygar, D. Song, "The TESLA Broadcast Authentication Protocol", In CryptoBytes, 5:2, Summer/Fall, pp. 2-13, 2002.

[12] A. Perrig, R. Canetti, J. D. Tygar, D. Song, "Efficient Authentication and Signing of Multicast Streams Over Lossy Channels", IEEE Symposium on Security and Privacy, 2000.

[13] Java.sun.com: The Source for Java Developers, http:// java.sun.com/.

[14] JMF 2.1.1 Solutions from Sun Developer Network (SDN), http://java.sun.com/products/java-media/jmf/2.1.1/solutions/.

[15] Java Media Framework at Java.sun.com : Sun Developer Network , http://java.sun.com/products/javamedia/jmf/2.1.1/download.html