L-020

Implementing Message Authentication to Real-Time Streaming Applications

Takeshi Ogino Shin-ichiro Kaneko Shintaro Ueda Nobutaka Kawaguchi
Hiroshi Shigeno Ken-ichi Okada

1. Introduction

Applications such as real-time streaming services via the Internet are attracting attention. But these types of application still carry risks of data tampering and spoofing.

Message authentication using digital signatures are used to solve these issues. By applying digital signatures to the packets to be transmitted, integrity of the digital data and authenticity of the user can be verified. However appending digital signatures to each packet is not practical in the means of computation, and issues such as performance degradation arise.

Many applications via the Internet still do not use message authentication. Therefore in this paper we will show how to efficiently implement message authentication to commonly used applications.

2. Authentication Technology

In this section, we will explain the basic authentication technologies necessary for our implementation.

2.1 Digital Signature

On the sender side, the hash value of the data is encrypted using the sender's private-key of the public-key cryptography. This encrypted value is used as the digital signature and appended to the data. On the receiver side, the signature is decrypted using the public key of the public-key cryptography. Then the hash value of the received data and the hash value contained in the decrypted data are compared and the integrity of the data is verified. By using the sender's public key, verification of the sender's authenticity is also made possible.

2.2 Gennaro's Hash-Chain scheme

Signing and verifying digital signatures require a considerable amount of computation. Several efficient schemes[1][2] are proposed to authenticate real-time streaming. Of these schemes, in this paper we will explain Gennaro's Hash-Chain and use it in our implementation.

In this scheme, several packets are handled as one group. By chaining the hash values of the packets in the group, only one signature per group is needed to authenticate all the packets in the group. Therefore computation is reduced.

The basic concept of the Hash-Chain is shown in Fig1, and the algorithm is described below.

Process on the Sender Side

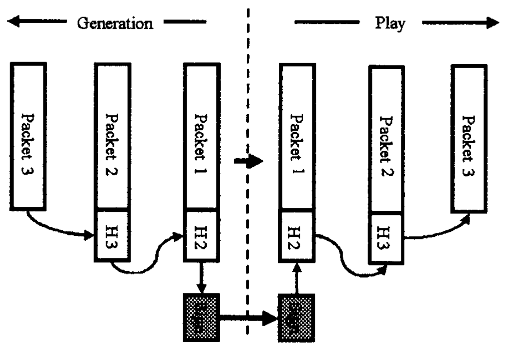

Figure 1: Gennaro's Hash-Chain

1. Buffer until all packets of one group is generated.
2. Append the hash value of the last packet in the group to the previous packet.
3. Calculate the hash value of the packet with the appended hash value.
4. Append the hash value calculated in step 3 to the previous packet.
5. Repeat step 3 and 4 until the first packet in the group is reached.
6. Generate the signature of the first packet and the hash value appended to it.

Process on the Receiver Side

1. Verify the first packet using the signature.
2. Verify the next packet and the appended hash value.
3. Repeat step 2 for the rest of the packets in the group.

There is delay on the sender side to buffer the packets of one group, but no delay on the receiver side. Therefore packets can be sequentially authenticated and played on arrival. Since fewer signatures are needed to authenticate the streamed packets, efficient authentication of streams is made possible.

3. Audio Conference Application

In this section we will explain an application used for audio conferencing.

†Faculty of Science and Technology, Keio University

3.1 RAT

RAT (Robust Audio Tool) is an open-source application that is actually used via the Internet, such as in the case of audio conferencing. RAT supports various types of codec and encryption, but message authentication is not applied. Therefore RAT is vulnerable to data tampering and spoofing.

3.2 Format of a RAT packet

To maintain real-time transmission, only small pieces of data are loaded into a packet. Fig2 shows the format of a IP packet when RAT is used.

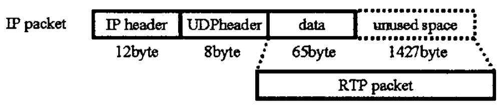

Figure 2: Format of IP packet

Though the maximum size of a UDP packet is 1500 bytes, RAT uses only 73 bytes including the header. Therefore there is 1427 bytes of unused space per packet.

This is a common method to keep the delay due to generating packets short, in applications that require real-time interaction. Generally, to maintain real-time interaction, packets must be generated every 20ms∼80ms. In RAT a packet is generated approximately every 40ms.

4. Inserting Digital Signature

Digital signatures are appended as extra header data. The header is processed to authenticate the packet. However by featuring on the packet format of RAT and other real-time streaming applications, a more efficient way to append authentication information can be realized.

As described above, a large unused space exists in RAT packets. We will use this space to insert the signature. Fig3 shows the format of a RTP packet when a signature is inserted.

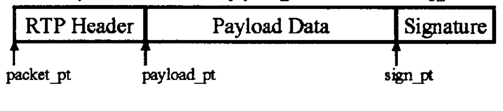

Figure 3: Format of RTP packet

Since RAT uses CBR (Constant Bit Rate), `payload_len`, the length of the payload data is constant. Therefore the end pointer of the payload is constant. Also, `sig_len`, the length of the signature is kept constant.

The signature is inserted into the payload instead of appended as an additional header information. However, `payload_pt` and `sign_pt`, the pointer set at the beginning of the payload and the signature respectively is fixed. Therefore the receiver can determine the beginning of the payload and the signature, since the length are fixed and are common between the sender side and receiver side. No additional header process is needed to separate the signature for verification.

5. Evaluation

Due to the algorithm of the Hash-Chain, there is a need to buffer a certain amount of packets. Therefore there is delay between the sender and receiver side. To check the delay time of our implementation method, we evaluated the end to end delay time between the sender and receiver. The results are shown in Table1.

Table 1: End to end delay

Theoretical value	Actual value
270ms	**306ms**

The theoretical value is calculated as *packet generation time* × *number of packets in a group* + *network delay* + *codec delay*.

It is stated in "The Report of IP Network Technology"[3] that the end to end delay of the IP telephony should be kept under 400ms. Therefore it is possible to say that our method is within the restrictions of real-time interaction.

6. Conclusion

In this paper we show how to efficiently implement message authentication into an application used in the real world. Therefore verification of data integrity and user authenticity in real-time streaming services are made possible.

References

[1] Shintaro Ueda, Shuichi Eto, Nobutaka Kawaguchi, Ryuya Uda, Hiroshi Shigeno and Ken'ichi Okada, "Real-time Stream Authentication Scheme for IP Telephony", In *IPSJ Journal*, Vol.45, No.2, pp.605-613, February 2004.

[2] Rosario Gennaro and Pankaj Rohatgi, "How to sign Digital Streams", In *Proceedings of the Conference on Advances in Cryptology*, pp.180-197, 1997.

[3] Ministry of Internal Affairs and Communications of Japan, "The Report of IP Network Technology", 12/26/2001.