Secure Transmission of Video on an End System Multicast Using Public Key Cryptography*

Istemi Ekin Akkus, Oznur Ozkasap, and M. Reha Civanlar

Koc University, Department of Computer Engineering,
Istanbul, Turkey
{iakkus, oozkasap, rcivanlar}@ku.edu.tr
http://ndsl.ku.edu.tr

Abstract. An approach for securing video transmission on an end system multicast session is described. Existing solutions use encryption techniques that require the use of a shared key. Although they can achieve efficient encryption/decryption and meet the demands of real-time video, a publicly available service needing only the integrity and non-repudiation of the message is not considered. In this study, we offer such a method using public key cryptography. This method can be used in an end system multicast infrastructure where video originates from one source, but spreads with the help of receiving peers. Two different methods are described and compared: 1) Encryption of the entire packet. 2) Encryption of the unique digest value of the transmitted packet (i.e. digitally signing). The receivers then check the integrity of the received packets using the public key provided by the sender. Also, this way the non-repudiation of the transmitted video is provided.

1 Introduction

End System Multicast (ESM) is one of the most effective ways for distributing data where the network infrastructure does not support native multicasting. An implementation of ESM is given in [1]. The described structure, NARADA, employs the approach in which data is distributed using end systems where each end system forwards the received data to other end systems in a hierarchical scheme. This way, multicasting burdens shift from the routers to the end systems, enabling a scalable multicasting solution. Although with this approach some physical links may have to carry packets more than once and an overhead is introduced, this is acceptable considering the benefit gained. Besides, the end systems are also organized in such a way that the overhead is minimized.

One of the significant areas that ESM can be utilized is video streaming. Current IETF meetings employ this mechanism. These meetings can be watched freely by joining the system and participating with own resources and bandwidth, which is the main reason why the system is scalable. In the peer-to-peer architecture described in [2] and [3] for multipoint video conferencing, the video

* This work is supported in part by TUBITAK (The Scientific and Technical Research Council of Turkey) under CAREER Award Grant 104E064.

of a user is transmitted to a group of users by employing a similar approach in which the users may forward the video they received to other peers. This allows the system to be distributed and scalable with multipoint conferencing capabilities.

Shifting the multicasting burden from the routers to the end systems is quite beneficial in terms of scalability; however, since every end system forwards received data to other hierarchically lower systems, security concerns may be introduced. The intermediate peers may alter the received data and forward them so that the last receivers in the hierarchy may encounter modified data. One way to prevent this would be using encryption. The video would be encrypted by the sender and the receivers would decrypt it upon reception. Since the transmitted video needs to be encrypted and decrypted in a limited time to meet the real-time streaming demands, this is a challenging task. Although there are some efficient encryption/decryption algorithms that can meet the constraints, they all make use of a shared key which can not be used in this context because of the symmetry property of the scheme; the intermediate peers could also use the key to encrypt the video and deceive the next receivers.

In this study, an approach for securing the integrity, authentication and non-repudiation of transmitted video in an ESM system is described. The transmitted video may be seen by everyone. The approach employs public key cryptography, so that the private key is known only to the sender and the public key is freely available. Two approaches are described and compared: 1) Encryption of the entire packet, so that a malicious user could not alter data and deceive the next receiver. 2) Encryption of the unique digest value, so that the receiver could check the integrity and authenticate the source of every data packet. Also, non-repudiation is also provided since only the sender has the private key to encrypt the digest value. The aim of this study is to investigate the feasibility of public-key based asymmetric approaches for secure video transmission among a group of participants in an ESM session. Our ultimate target is to integrate an efficient asymmetric solution for secure video transmission to our peer-to-peer multipoint video conferencing system [2] and [3].

Next section gives a brief literature survey about how encryption techniques are used with video transmission. Section 3 describes the design details and Section 4 presents simulation results. Discussions and conclusions are given in Section 5 and Section 6.

2 Related Work

As a first thought, encryption of the whole data packet by the sender, called nave encryption, seems reasonable. However, this approach brings considerable burden to the systems. To overcome this and enhance performance, Maples and Spanos [4] and Li et al. [5] proposed encrypting only the I-frames of video which are used as reference frames for the others. However, Agi and Gong showed that from only the B and P-frames, which are not encrypted, the video was recoverable and encryption of only I-frames does not bring any performance enhancements [6].

In [7], the first standard compliant system adapted to MPEG-4 is described. The encryption system ARMS enables rich media streaming to a large-scale client population. End-to-end security is established while content is adapted to the network conditions and streamed over untrusted servers. The encryption and decryption scheme used is AES in counter mode. RFC3711 [8] defines the standards of the encryption/decryption schemes that could be used in RTP [9]. There again, for the encryption AES is mentioned because of its low overhead and efficiency.

In [10], Tang proposes that compression and encryption need to be done at the same time to enhance performance and to meet the real-time demands of multimedia. Techniques that first compress the images and then encrypt them bring too much overhead and are not practically usable. They introduce a new cryptographic extension to MPEG, employing random algorithms and image compression methods. Another algorithm for encrypting video in real-time, namely VEA (Video Encryption Algorithm) [11], uses composition of video compression techniques using Discrete Cosine Transform with a secret key minimizing the overhead and meeting the demands of real-time video. Liu and Koenig also presented a novel encryption algorithm, called Puzzle [12], which is independent of the compression algorithm so that it can be used by any encoding algorithm. The encryption is done by using 128-block ciphers.

TESLA [13], Timed Efficient Stream Loss-tolerant Authentication, allows receivers in a multicast group to authenticate the sender. It requires that the sender and receivers are loosely time synchronized meaning that the receivers know an upper bound of the sender's clock. Although encryption is done with symmetric keys, they are not known by the receiver until the sender discloses it. So the receivers need to buffer incoming packets until the key is disclosed. Non-repudiation is not provided.

All these techniques make use of a shared secret key that can work both ways: The encryption and the decryption operations are done using the same key. This is not suitable for an ESM session, since the sender needs to be authenticated which is not possible in a group sharing a symmetric key. Although symmetric key mostly enhances performance, another mechanism for distributing the keys is needed. For a publicly available service, this becomes costly. In [14], chaining techniques for signing and verifying multiple packets (a block) using a single signing and verification operation are proposed. This way, the overhead is reduced. The signature-based technique proposed does not depend on the reliable delivery of packets, but uses caching of the packet digests in order to verify the other packets in the block efficiently. In our scheme, no caching is required and verification is done per packet basis. Simulations show that this can be done fast enough even for large key sizes.

3 Design

The approach introduced is independent of the encoding algorithm of the video since it operates with packets independently. Integrity and authentication can be provided either by encrypting the entire packet using chunks or by creating a unique digest value and encrypting it. On both methods, the real-time video imposes limits on the key size. These limits are the constraints on the allowed times of the encryption/decryption operations which can be very short to manage the bit rate of the video.

The first method describes encrypting/decrypting packets either entirely or after dividing them into chunks. With larger key sizes, these operations take enough time to decrease the performance of the solution. As the simulations will show, operations on the entire packet do not work (because of the packet size) and dividing into chunks would either require many packets to be sent (after each division) or to be buffered to build one packet, which would complicate implementation.

The second method; however, can provide a high security level without increasing the overhead much. The encryption/decryption operations take less time and processing power which make it more suitable for secure video transmission on end system multicast infrastructure. In this method, the sender side generates a unique digest value for each packet. This unique digest value is then encrypted by the private key of the sender creating a digital signature which is appended to the end of the packet. Upon reception of the packet at a receiver, the receiver extracts this signature part from the packet and decrypts it with the public key of the sender. Also, it generates the unique digest value of the packet and compares this with the value received from the sender. If the values are the same, this means that the packet received was not altered along the way. If not, the packet is dropped and appropriate action is taken (e.g. the source may be informed that someone is trying to modify the contents of the packets).

4 Simulation Results

Simulations were performed on a Pentium IV 2.4 GHz processor with 512 MB of RAM running a 2.4.20 kernel Linux. The bit rate of the video was assumed to be 200kbps which makes 25kBps. Packet size was set to 1400 bytes leading an approximate value of 18 packets/sec. The public exponent used was 25-bit and the private key was generated according to the corresponding key size for 50 runs of simulation.

4.1 Encryption/Decryption of the Entire Packet

One approach is to treat the entire packet as a big message and encrypt/decrypt it using the private/public key. This idea has both advantages and disadvantages. First of all, treating the entire packet as a big message makes the implementation easy. Also, encrypting and decrypting it, is a trivial task. However, if the packet is large enough, then the message it represents (1400 bytes = 11200 bits, in our experiments) can overflow the key size, so that the encryption/decryption function would not be one-to-one. This means that an encrypted packet could not be restored by decrypting it. Assuming that the key size is big enough to handle such cases, performance becomes an issue. Since this operation needs to be done in one second for several packets to meet the real-time demands of streaming video, this is clearly not realizable.

Another approach is to divide packets into chunks so that each chunk can be encrypted/decrypted independently: This idea's advantage lies in the part that chunks, a packet is divided into, can be independently encrypted and decrypted. The encrypted/decrypted parts only need to be assembled back to back to form/restore a packet. Since the packet is divided into smaller chunks, preserving the one-to-one mapping of the operations becomes easier. However, this method requires either many packets to be sent after each encryption which would increase communication overhead or to be buffered to build one packet which complicates the implementation.

When considering this method, the issue of determining the chunk size arises. To preserve one-to-one mapping of the encryption and decryption operations, the chunk size should not exceed the key size. Fig. 1 and Fig. 2 show encryption and decryption times with different chunk sizes for different key sizes. In Fig. 1 and Fig. 2, it can be seen that the encryption and decryption times decrease as the chunk size is increasing. Increasing key size increases the encryption and decryption times as shown in Fig. 3. The optimum values for the key and chunk sizes can be determined considering this information. To provide sufficient security, a large key should be picked. To meet the real-time demands of the streaming video with that key, the chunk size should also be large.

As can be seen from the figures, the encryption times are much larger than the decryption times. This is because the public key exponent is a much smaller number than the corresponding private key exponent generated according to it. These values can be further optimized by using similar sized exponent pairs so that the encryption and decryption times are closer to each other.

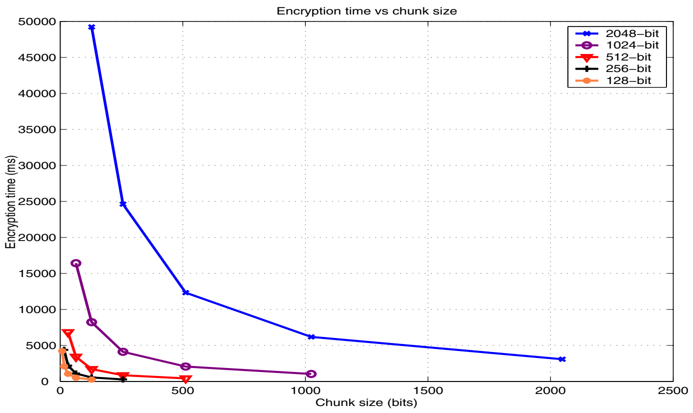

Fig. 1. Encryption times versus the chunk sizes. Corresponding series show the key sizes.

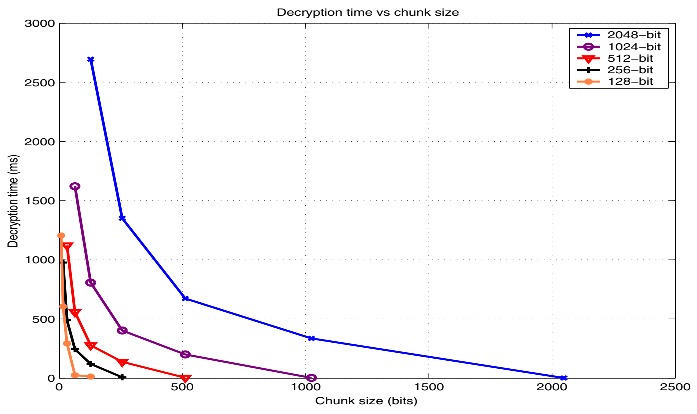

Fig. 2. Decryption times versus the chunk sizes. Corresponding series show the key sizes.

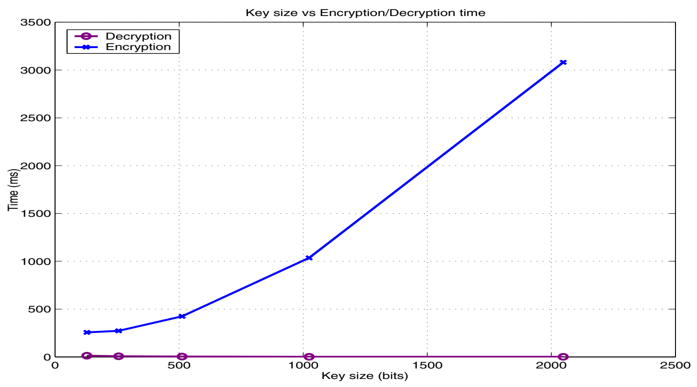

Fig. 3. Encryption/Decryption times versus the key size. Chunk size is equal to the key size.

4.2 Encryption of the Unique Digest Value of Each Packet

Another technique would be to create a unique digest value of the packet and encrypt this value with the private key like done in digital signatures. Here, the entire packet is treated as one large message. The digest operation is a one-way function so that the message can not be restored from the digest value. This digest value is ensured to be unique for each packet so that a malicious user would not be able to create another message that goes with this valid digest value and deceive the next receiver. Instead of the packet, its digest value is encrypted with the private key and appended to the packet. The receiver then checks the validity of the digest value by simply decrypting it with the public key.

Since the digest value is just calculated and encrypted only once for the entire packet, this has surely better performance than trying to encrypt all the chunks. In our simulations, we used SHA-1 for digest generation, and RSA for public-key encryption of the digest. Table 1 gives the encryption and decryption times measured. As can be seen, calculating a unique digest value and encrypting/decrypting it can meet real-time demands without burdening the sender or the receiver, respectively.

Table 1. Total time spent on digest value calculation and encryption/decryption times with corresponding key sizes

RSA with SHA-1					
Key size (bits)	128	256	512	1024	2048
Encryption time (ms)	1.08	6.52	19.98	96.36	592.68
Decryption time (ms)	0.41	1.63	3.61	9.74	32.19

5 Discussions

In order to achieve integrity, authentication and non-repudiation, encrypting the entire video packet seemed to be a valid way; however, the simulations showed that the security that could be achieved using such an idea would be limited. The maximum key size that could be used in this part of simulation was 1024-bit with the given public and corresponding private key exponent sizes. Although with the optimization described above (using similar sized public and private key exponents) in practice, a lower value is expected since the application would consume processing power while dividing the packets into chunks at the sender side and putting them back together at the receiver side.

On the other hand, calculating a unique digest value and encrypting it worked under all key sizes given, even without the optimization. The calculated digest value is not large (20 bytes when SHA-1 is used) that the sender could not handle, and the main advantage is that it is calculated once per packet. Calculating the digest value is a one-way function, so that recovering the data is impossible. However, by encrypting it, the sender can be sure that the receiver can check the integrity of the packet. Also, source authentication and non-repudiation can be ensured without considering confidentiality because the publicly available service does not need it. Since the encryption is possible only by the sender side because of the private key, an intermediate peer could not change the contents of the packet and send it to another receiver, because the last receiver would notice after calculating the digest value and comparing it with the decrypted one.

6 Conclusions

A new approach for securing the integrity of a video transmission is presented. Encrypting video packets would be a first idea; however, all existing techniques to encrypt video require the use of a shared secret key. Although they can meet the real-time demands of streaming video, they can not be used in an ESM system, where the service is free to anyone. By such an application, a shared secret key can not be used, because of the key's symmetry property, meaning that encryption and decryption are done using the same key. We investigated the feasibility of two public-key based approaches for secure video transmission on ESM: 1) Encrypting the entire packet 2) Calculating a unique digest value and encrypting it. Encrypting the entire packet ensures limited security because of the real-time demands and brings too much burden on the sender and receiver side. On the other hand, calculating a unique digest value for every packet and encrypting it, has better performance and can be done faster achieving higher security, source authentication and non-repudiation. As future work, we plan to integrate such an asymmetric solution for secure video transmission to our peer-to-peer multipoint video conferencing system.

References

1. Chu Y., Rao S., Zhang H.: A case for end system multicast, Proceedings of ACM Sigmetrics, (2000).
2. Civanlar M. R., Ozkasap O., Celebi T.: Peer-to-peer multipoint video conferencing on the Internet, Signal Processing: Image Communication 20, pp.743-754, (2005).
3. Akkus I. E., Civanlar, M. R., Ozkasap O.: Peer-to-peer Multipoint Video Conferencing Using Layered Video, to appear on International Conference on Image Processing ICIP 2006.
4. Spanos, G. A., Maples, T. B.: Performance Study of a Selective Encryption Scheme for the Security of Networked Real-time Video. Forth International Conference on Computer Communications and Networks, pp. 2-10, (1995).
5. Li, Y., Chen, Z., Tan, S. M., Campbell R. H.: Security Enhanced MPEG Player. IEEE 1st International Workshop on Multimedia Software, (1996).
6. Agi, I., Gong, L.: An Empirical Study of MPEG Video Transmission. Proceedings of the Internet Society Symposium on Network and Distributed System Security, pp.137-144, (1996).
7. Venkatramani, C., Westerink, P., Verscheure, O., Frossard, P.: Securing Media For Adaptive Streaming, Proceedings of the eleventh ACM international conference on Multimedia, ACM Press, (2003).
8. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: RFC3711 The Secure Real-time Transport Protocol (SRTP) (2004).
9. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications, (2003).
10. Tang, L.: Methods for Encrypting and Decrypting MPEG Video Data Efficiently, Proceedings of the fourth ACM international conference on Multimedia, ACM Press, (1997).
11. Changgui, S., Bhargava, B.: A Fast MPEG Video Encryption Algorithm, Proceedings of the sixth ACM international conference on Multimedia, ACM Press, (1998).
12. Liu, F., Koenig, H.: A Novel Encryption Algorithm for High Resolution Video, Proceedings of the international workshop on Network and operating systems support for digital audio and video NOSSDAV '05, (2005).
13. Perrig A., Song D., Canetti R., Tygar J. D., Briscoe B.: RFC4082 Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction, (2005).
14. Wong, C. K., Lam, S. S.: Digital Signatures for Flows and Multicasts, EEE/ACM Transactions on Networking (TON), Vol 7 Issue 4, IEEE Press, (1999).