Scalable Security Mechanisms in Transport Systems for Enhanced Multimedia Services

T. Kunkelmann[1], H. Vogler[1], M.-L. Moschgath[1], L. Wolf[2]

[1] Information Technology Transfer Office, Wilhelminenstr. 7
[2] Institute for Industrial Process and System Communication, Merckstr. 25
Darmstadt University of Technology
D-64283 Darmstadt, Germany
[1] {kunkel,vogler,malu}@ito.tu-darmstadt.de
[2] Lars.Wolf@KOM.tu-darmstadt.de

Abstract: Data confidentiality is a very important issue for communication in open networks. Secure communication usually will be achieved by encryption mechanisms. For distributed multimedia applications the usage of encryption in real-time can cause a performance problem due to the time complexity of the cryptographic algorithms. In these cases partial encryption is a solution to satisfy real-time demands.

In this paper we examine the usage of partial encryption in transport systems for multimedia data. This implies that the partial encryption scheme cannot take advantage of special properties of the multimedia data content. So we first demonstrate that in most cases it is sufficient to encrypt only a small portion of randomly chosen data from a video stream to achieve an adequate level of security.

There are different approaches to integrate partial encryption mechanisms in transport systems. As a first approach, we investigate the integration in the transport layer. This offers several facilities for the integration. An alternative approach is located in the network layer, where alternative routing methods for a multimedia data stream are analyzed. A discussion of the impact of partial encryption to transport system mechanisms concludes this paper.

1 Introduction

In the rapid growing market of Internet communication, the confidentiality of transmitted data in an insecure network becomes a very important issue. Encryption is the most common solution to protect data against unauthorized access. There already exist mechanisms and algorithms for encryption, which guarantee that only authorized receivers are able to decrypt the data. This is a suitable path for many applications to achieve data confidentiality in an open and insecure network, like the Internet.

However, encryption is - depending on the algorithm - very complex and results in time consuming processing. Analyzing a transmission of video and audio data for a live conference, encryption can be too slow to encrypt or decrypt high bandwidth multimedia data for a real-time transmission in software.

Encryption and other security mechanisms in general can be placed at different layers of a communication stack. In a rough classification, the layers can be divided in

network, transport and application layer. In this paper we focus on transport system aspects, since one of our main goals is to provide independence of the underlying network, and also of the application. Therefore, we propose a general approach for the security mechanisms, which, however, has the drawback of having no knowledge of the inherent properties of the data, so no specialized encryption mechanisms (as e.g. in [1]) can be used.

All existing approaches of transport system encryption, enlisted in Section 3, perform encryption for the whole data stream, which results in time consuming computation, too expensive for real-time multimedia data streams, as mentioned above. In future, using more powerful CPUs or special encryption chips can improve this situation, yet, the demand for higher bandwidth of new real-time applications increases as well, e.g. by higher quality of video and audio. Besides high-end computers with perhaps support of encryption hardware, there are many desktop computer systems with low processing power, included in distributed multimedia applications as well. It is a too severe restriction to limit security to high performance computing systems only.

Our investigations in the context of encryption of video data indicate that it is not necessary to encrypt the whole data stream, instead a small amount of the data stream to be protected is sufficient for many applications. If a potential eavesdropper intercepts a transmission of video data and receives only some unencrypted parts, this information will be useless. This is due to the special nature of video encoding. As a prerequisite for our security mechanisms presented in this paper, this aspect is central and will be discussed in detail in Section 4. Based on this fact, we examine two different approaches in this paper. First, we investigate the possibility of partial encryption in transport systems, especially in transport protocols. Secondly, we investigate the mechanism of splitting a data stream into several parts and using distinct routes for each part of the data stream. The pros and cons of these approaches are discussed in Section. 5 Concluding remarks are presented in Section 6.

2 Cryptographic Methods

This section gives a short overview of methods used for cryptography and the terminology used in this domain of computer science [2].

2.1 Symmetric-Key Cryptography

Symmetric-key or secret-key cryptography uses the same key to encrypt and decrypt a message. For example, if the plaintext is denoted by the variable P, the ciphertext by C, the encryption with key x by $E_x(\ )$, and the decryption with key x by $D_x(\ )$, then the symmetric algorithms are functionally described as follows: $P=D_x(C=E_x(P))$. The problem with symmetric-key cryptography is the exchange of the secret key so that nobody can spy them.

Symmetric encryption algorithms may be further divided into stream ciphers and block ciphers. Stream ciphers (e.g. RC4) are generally implemented as the exclusive-or (XOR) of the data stream with the key stream, they decrypt consequently only one

bit of plaintext at a time. The security of a stream cipher is determined by the quality of the key stream. A completely random key stream with same length as the plaintext would effectively implement an unbreakable one-time pad encryption. A deterministic key stream with a short period would provide minor security. In contrast, block ciphers as the most common ciphers (e.g. DES [3], TripleDES, IDEA [4]) simultaneously encrypt a number of bits (typically 64). The security of these algorithms increases with the used key length, while the performance decreases.

2.2 Public-Key Cryptography

Public-key cryptography was invented in 1976 by W. Diffie and M. Hellman [5] to solve the depicted problem of secure key exchange. With public-key cryptography, each person gets a pair of keys, a public and a private key. The sender encrypts the plaintext with the public key of the receiver, who decrypts the ciphertext with his private key. The best known public-key cipher is RSA [6]. Further ciphers are *Digital Signature Standard* (DSS), *ElGamal*, and *Diffie-Hellman*.

The drawback of public key cryptography is the weak performance of the algorithms. Public key methods usually are 100 to 1000 times slower than symmetric key methods [2]. So they are merely used for a secure symmetric key distribution. Such a combination of both cryptographic mechanisms is called *hybrid encryption*, the symmetric key used for data encryption is usually denoted as a *session key*.

3 Encryption Support in Existing Protocols

For secure communication, some existing and proposed protocols support at least basic encryption and authentication techniques. Besides the Secure Shell protocol, which in fact is an application providing security support for other applications layered on top of it, the most common implementations and specifications of security functionality in transport protocols are:

- **Secure Shell (SSH)** [7] is a software package that provides secure login sessions and X server communication in an insecure network environment. It features strong cryptographic authentication, strong encryption, and integrity protection. Authentication in SSH is host-based; it does not perform user authentication.
- **Secure Socket Layer (SSL)** [8] is a protocol for secure WWW connections and was originally developed by Netscape. SSL is application-protocol independent, therefore a higher level protocol can layer on top of it transparently. The protocol provides privacy, authentication and reliability.
- **Real-time Transport Protocol (RTP)** [9] intends the usage of DES as a crypto-algorithm if the underlying protocol has no provision for encryption methods. The sender and receiver have to agree about using encryption. Authentication and integrity checks are not defined in the current RTP specification.
  Packets protected by encryption are marked with a flag in the RTP header. RTP uses DES in CBC mode (*cipher block chaining* [2]). To avoid known-plaintext at-

Table 1: Data planes and security policies defined in the ATM Security Specification

Plane	*end-to-end*	*switch-to-switch*	*end-to-switch*
User	Authentication Confidentiality Integrity	Authentication Confidentiality	
Control	Authentication	Authentication	Authentication
Management	Authentication	Authentication	Authentication

tacks, the RTCP (real-time control protocol) packets are extended with a 32 bit random number, for RTP data packets this problem does not arise due to a different time stamp in each packet. By using the header flag for marking an encrypted RTP packet, this protocol can easily support a selective encryption method, which may have a granularity down to the RTP packet size.

- **IPv6:** Two extension header fields, *Authentication Header* (AH) and *Encapsulated Security Payload* (ESP) are integral parts of IPv6 [10], which have been defined by the IETF IP Security Working Group. AH provides message authentication and integrity. ESP provides message confidentiality and integrity. ESP may optionally provide authentication if an appropriate algorithm is used. There are two modes to incorporate security by an ESP header:
  1. **Transport mode:** The packet data consists only of encrypted payload
  2. **Tunnel mode:** The packet data consists of a whole IP packet (datagram). This mode allows tunneling IPv4 packets via an IPv6 subnetwork.
- **ATM:** The ATM Forum is currently defining a security standard for ATM [11]. Table 1 shows the different planes where security will be provided, and the three security policies.
  The *switch-to-switch* encryption [12] and *end-to-switch* encryption models usually need hardware encryption support in the switches, so our proposed solution in Section 5.1 will mainly target the *end-to-end* encryption model, since this kind of encryption is usually performed in software on a workstation.

All these protocol approaches only support encryption for the whole data stream, which results in time consuming computation, too expensive for real-time multimedia data streams if the protocol stack is implemented in software. This is especially true if the same machine performs the protocol decoding functions and also the decompression and display of the video streams it receives. Therefore, a solution for this problem can be achieved by using *partial encryption* methods.

4 Partially Encrypted Multimedia Data

In this section we present some example snapshots of video frames, where the data stream of the video communication channel has been made partially inaccessible, e.g. by encryption. As we can see from our examples, for most applications it is sufficient to protect 5 to 30 percent of the data stream in order to render the video data useless.

To motivate partial encryption in software solutions, we first give an overview of video applications with typical data rates occurring there.

4.1 Digital Video Formats

In today's video applications, several data formats for digital video are in use. The most common formats used together with the bandwidth they occupy are listed here.

Motion-JPEG (M-JPEG) consists of a sequence of single video frames encoded with the JPEG still image coding standard [13]. M-JPEG is used mainly for video conferencing tools due to a symmetrical expense for encoding and decoding. The drawback of M-JPEG is the high bandwidth needed. To achieve TV quality with M-JPEG video, a bandwidth of about 8 to 15 Mbit/s is needed. In some M-JPEG implementations, a bandwidth reduction is achieved by *conditional replenishment*, i.e. omitting those DCT blocks in the M-JPEG data stream with no changes to the previous frame. This leads to a bandwidth reduction of 2:1, up to 4:1 for video conferencing (talking head) scenes [14].

MPEG [15] supports data rates of about 1.5 Mbit/s (MPEG-1 profile), which meets the possibilities of network and CD-ROM video playback. Audio and video information are multiplexed in an MPEG system data stream, where the video data occupy a bandwidth of 1.15 Mbit/s for a SIF (*source input format*, 352×240 pixels for an NTSC video source) encoded video.

Besides *I-Frames* (intracoded frames), MPEG also provides *P-Frames* (predictive frames) and *B-Frames* (bi-directional prediction), using motion compensation to reduce the amount of data. Here only the difference to a suitable data block in a neighboured frame is transmitted. This reduces the size for B-Frames to about 17 - 28 percent of the corresponding I-Frame size, leading to peaks (bursts) during the transmission of an MPEG video stream.

H.261 and H.263 [16] are standards for transmitting video data streams over an ISDN connection at data rates of p×64 Kbit/s. Somewhat usable results for QCIF (quarter common interface format, 176×144 pixels) b/w images can already be achieved with 128 Kbit/s (p=2), the standard supports CIF images (352×288) at high quality up to 1.92 Mbit/s (p=30) bandwidth. The bandwidth for the video stream is kept constant by adapting the frame rate or the image quality at the encoder if necessary. The encoding schemes used are similar to MPEG, supporting *intraframe* and *interframe* encoding.

Network Video (nv) [17] uses wavelets as its compression technique and also conditional replenishment for data reduction. The bandwidth is 2.5 times of H.261, but the decoding effort is about 20 percent less in time. So this format becomes an alternative for computers with limited CPU performance. Due to the largely increased bandwidth the nv format impedes software decryption of a video stream in real-time.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.