[KVMW98] *T. Kunkelmann, H. Vogler, M.-L. Moschgath, L. Wolf;* **Scalable Security Mechanisms in Transport Systems for Enhanced Multimedia Services;** ECMAST'98, Berlin, 26.-28.05.1998

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*, 352x240 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 px64 Kbit/s. Somewhat usable results for QCIF (quarter common interface format, 176x144 pixels) b/w images can already be achieved with 128 Kbit/s (p=2), the standard supports CIF images (352x288) 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.

Table 2: CPU utilization of different hardware systems for DES encryption in software. The MPEG and M-JPEG cases represent e.g. Pay-TV scenarios (16 - 25 fps), while the H.261 scenario describes an ISDN video conference with three video channels open (12 - 15 fps).

DES CPU usage	*1.5 Mbit MPEG*	*2 Mbit M-JPEG*	*3×128 Kbit H.261*
Intel Pentium-100	86.70 %	115.62 %	21.67 %
DEC Alpha 1000/ 266	65.63 %	87.50 %	16.41 %
Sparc 20 (Solaris)	76.01 %	101.34 %	19.00 %
Sparc 4c (SunOs)	312.77 %	417.03 %	78.19 %

4.2 Performance Aspects for Encrypted Video

As pointed out in [18], modern high-performance workstations and servers are capable of playing MPEG or M-JPEG video in SIF frame format, leaving about 20 to 60 percent CPU time for other jobs when using hardware JPEG support. Without accelerator hardware this idle time decreases to 0 - 40 percent of CPU time. On most desktop workstations such a computing power is not available. Here the frame rate or the pixel resolution has to be reduced to meet the limited CPU capacity. Performance measurements on a PC (100 MHz Pentium) showed that such a system can playback about three H.261 QCIF video streams with frame rates sufficient for video conferencing.

Table 2 shows the hypothetical performance of several hardware platforms needed for decrypting video streams in software. The encryption algorithm used here is DES in an improved version of P. Karn's implementation, as it is used in the SECUDE toolkit [19] for secure multimedia communications.

For the MPEG and M-JPEG scenarios, the need for reducing the encryption effort is obvious, the slower workstations are hopelessly overloaded already with the DES decryption. For the H.261 scenario an encryption CPU usage of 20 percent implies a frame reduction from e.g. 11 to 8.8, violating the lower bounds for human image perception. Therefore partial encryption is a suitable solution also for this case.

The effort necessary for partial encryption grows linear with the encryption rate, the protocol overhead is only marginal compared to the time complexity of the cryptographic algorithm. This is confirmed by the experiments in Table 3, where some results with playback of different TV video clips and simultaneous decryption are presented. The experiments also show the gain in playback performance when using partial encryption, which is expressed by an increased frame rate of about 2 to 5 fps.

4.3 Encrypted Video Frame Examples

The frames presented in Figure 1 are examples from a movie with only 1 percent of the data encrypted, as it will be played by a software player protected against segmentation faults. An unmodified player (e.g. the Berkeley MPEG player [20]) will normally terminate with a core dump for such a protected MPEG stream. Although the video stream will usually look like the left example, in some rare frames a portion of

Table 3: Frame rates (fps) with full and partial encryption (25% encrypted data, UltraSparc)

Video clip	*Kbit/s*	*No encryption*	*Full encryption*	*Partial encryption*
Skating	845.1	20.82	15.14	19.42
Talk	628.8	17.86	12.70	16.56
Soccer	1259.9	16.94	10.88	15.43
Skating (I-Frames)	1922.9	25.50	22.52	24.81

the unencrypted video might be visible, as demonstrated in the right example.

The following example shows that for smaller image frames (e.g. QCIF format), it is not sufficient to leave large parts of the video stream unencrypted, as in the example above where at least 99x64 = 6336 consecutive bits (with DES block encryption) stem from a plain MPEG data stream. The results of 5 percent (10 percent) encrypted data are sufficient to prevent a native playback for QCIF video streams, as can be seen in the example frames of Figure 2.

4.4 Reconstruction of Protected Data

With methods used in cryptanalysis, e.g. statistical and entropy evaluations [21], it may always be possible to detect those portions of a data stream which have been encrypted. However, this will be a difficult job for partially encrypted (MPEG or similar encoded) video streams due to the nearly redundancy-free Huffman encoding. Even if an eavesdropper knows which parts of a stream are encrypted or inaccessible, he needs a substantial effort to extract information where new frames and blocks start. Someone who succeeded in analyzing a partially encrypted video stream might probably reconstruct a video frame as in the examples of Figure 3. Here the non-reconstructible protected information is set to zero (black color for this printed version, a value of zero results in a medium green screen image for the YUV color model).

These examples motivate to protect about 25 to 30 percent of confidential video information. In some rare cases this effort might still be insufficient for the application's confidentiality demands (e.g. virtual management meetings in a worldwide oper-

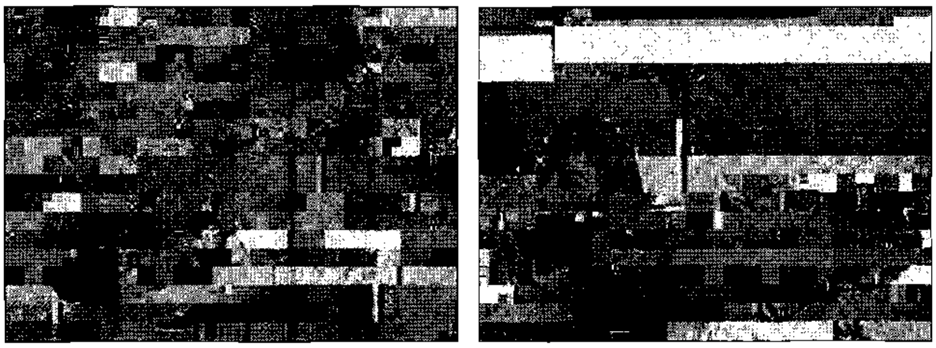

Figure 1: Playback of an MPEG stream with one percent encrypted data (SIF)

ating company, which may use public networks for transport). Here it will make sense to protect the video data by application-specific encryption methods, like those presented in [22].

In other scenarios, where encryption is merely used to aggravate the access for the public, e.g. video-on-demand systems, the expense for reconstructing parts of a video is out of all proportion to the fee for joining the movie broadcast legally. In these scenarios a rate of some few percent encrypted data is sufficient to fulfil all confidentiality requirements.

4.5 Audio Data in Video Streams

Tests with partially encrypted raw audio information show that in a range of 10 to 25 percent encryption, the data becomes indistinct to a human listener. Especially for music this rate lies only around 10 percent. However, with noise filters it is easy to remove the blocks of white noise produced by the encryption. So, for highly confidential video conferences it might be no good choice to encrypt only small portions of the audio data. In these systems the audio data is usually transmitted via a separate communications channel (the MPEG format with interlaced audio is seldom used for symmetric communication), so it is simple to treat audio data separately from video information by the transport system. Due to the much smaller bandwidth (3 to 32 Kbit for MPEG compressed [23], 64 Kbit for raw data in speech quality) the full encryption of audio channels in real-time should always be feasible in those systems.

5 Scalable Security in Transport Systems

As stated in the introduction, within this paper we only consider security aspects related to transport systems i.e. network and transport layer. This approach stands in contrast to some content-based partial encryption schemes for video streams, which have been proposed in the last few years, e.g. the scalable video encryption approach of [24] for video streams based on the JPEG compression method. In [1] a security method is presented, which uses DES as its encryption algorithm. The drawback of these encryption schemes is the interweaving with the content of the data (the video codec), so they cannot be used in a content- or application-independent way, as neces-

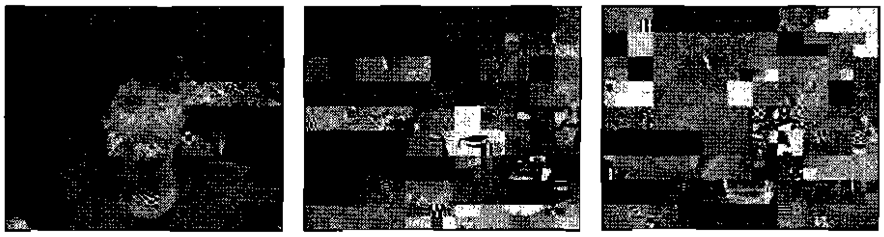
Figure 2: QCIF frame with 1, 5 and 10 percent encryption (left to right)

sary for their usage in the transport system. Some methods, e.g. [25], need a manipulation of the video encoding/ decoding algorithm to work efficiently. A survey of partial encryption methods specialized for video data content and their domain of application can be found in [22].

The concept of providing security independently from applications offers the possibility of using several existing video conferencing tools without changes. In this section we show possibilities for providing scalable security in transport systems and general aspects related to them. Our design goal is not focused on developing yet another transport protocol with inherent security facilities, rather on integrating methods in existing protocols which allow the scaling of the effort spent for encryption.

The first part of this section deals with security enhancements in the transport layer using partial encryption. In the second part we investigate mechanisms provided by the network layer. The idea is to use distinct routes to prevent attackers from having full access to a data-stream. General aspects related to both approaches are discussed in the third subsection.

5.1 Partial Encryption in the Transport Layer

Using encryption in transport protocols is not a new idea. Several protocols [26] [9] provide enhancements for security mechanisms. These protocols have in common that they use full encryption of data streams. As we have motivated, there exist scenarios where full encryption is too expensive, which leads to the idea of partial encryption.
In this subsection only special issues regarding partial encryption are addressed. General aspects and security in transport protocols such as key exchange, authorization, and negotiation of encryption methods [2] are beyond the scope of this paper.

Various possibilities for partial encryption. Partial encryption within a transport protocol can either be applied by encrypting the whole *protocol data unit* (PDU), or by encrypting only parts of a PDU. Considering PDUs we can distinguish to encrypt

- user data,
- header information or
- control information.

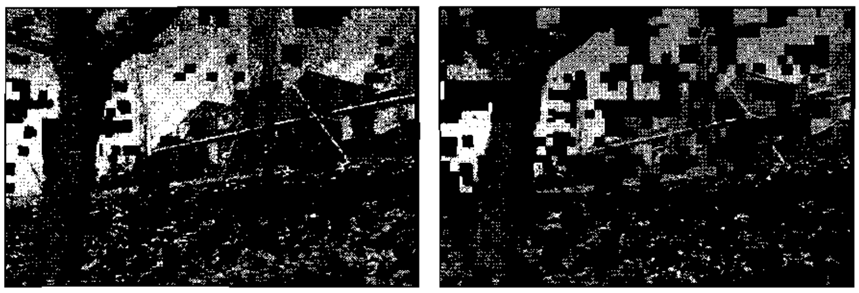

Figure 3: Maximal possible reconstruction for 10 (left) and 30 percent encryption

Encrypting header or control information obscures the context of a data stream, so the content of the stream also gets useless. This method gives a good protection for the data with marginal encryption effort. However, the control and header information can be derived by implicit knowledge, e.g. the knowledge about an MBone conference at a certain time using RTP and the necessary header fields of the PDUs. With this implicit knowledge it is quite simple to reconstruct that kind of information. Even though encrypting header and control information might produce a lot of work for an attacker, it does not protect user data at all and therefore it only makes sense if used in conjunction with user data encryption.

Identification of encrypted data. The identification of encrypted data at the receiver's site is the mandatory step for the interpretation of these data. Generally speaking, two different methods for identifying encrypted user data can be distinguished:

- implicit knowledge by always using the same mechanisms (e.g. alternately 64 bits encrypted/ 128 bits plain) or
- marking of encrypted data.

Marking can be achieved by using flags in PDUs if a whole PDU is encrypted, or by using header extensions if provided by the protocol to specify which data are encrypted.

In both cases a negotiation between participants before data transmission with partial encryption has to take place. Additionally, this offers the possibility to negotiate the rate of encryption as a scalable quality of service parameter.

Marking the encrypted PDUs is the more flexible method due to the possibility of adapting the encryption rate during transmission. On the other hand, a fixed rate with implicit knowledge of the encrypted PDUs is easier to control and produces less computational overhead.

5.2 Distinct Routes in the Network Layer

As the experiments described in Section 4 illustrate, a video stream may not be decoded and viewed successfully by users if less than 70 percent of the plain data is available. Based on these results, a certain degree of security can be gained if a stream of video data is split into several substreams, which are transmitted via disjoined routes. This way, a potential attacker can access only parts of the overall information, and hence not interpret the stream.

In order to use such a scheme, several aspects must be solved, which will be discussed in this section.

Distinct Routes. A scheme based on separate routes only makes sense if several independent routes exist between source and destination. This need not necessarily be the case for the whole route but at least for the parts which are most critical with respect to vulnerability. Thus, the suitability of such a scenario depends on network connectivity, e.g., for a stream between users belonging to different companies which have corporate networks (Intranets) connected by the Internet and where the corporate networks have several distinct connections to the Internet (as illustrated in Figure 4; host B is connected to the networks X and Y). The internal network part can be considered as secure, while for the external - critical - part, different routes exist.

Furthermore, it is necessary that the routes of substreams are disjoined. Hence, the routing methods must know about the substreams and their relations. For this purpose, group handling mechanisms [27] and (inverse) results from multicast routing can be applied.

Stream Splitting. The location where the splitting occurs has to weigh security vs. QoS provisioning. Assume we have networks, protocols, and routing mechanisms which allow for QoS-provided data transmission. The routing mechanisms check that the route chosen can provide the necessary resources. For security purposes the split should be done as soon as possible, yet, this limits the possible choices for routing decisions.

The splitting leads to various substreams which are related to each other. Therefore, we need mechanisms to express this relationship, i.e., the single flows used to transmit a partition must be set into relation, e.g., via group parameters within a flow specification. We believe that this makes only sense if the flows are described for other purposes anyway, e.g., resource reservation.

Before the video data can be presented to the user, the substreams must be received, reassembled, and synchronized. Especially if the substreams encounter largely different QoS, i.e., delay and jitter, the receiver must take appropriate measures, e.g., provide sufficient buffer space.

5.3 Common aspects of partial encryption and distinct routes

Forward Error Correction. Generally, *forward error correction* (FEC) mechanisms should not be applied to the application layer data of streams which are transmitted via partial encryption or a distinct-route approach. The reason is that an FEC coder increases the redundancy of the transmitted data so that an attacker gets access to more information, and hence the security of the stream is reduced. Applying FEC to data which already has been partially encrypted does no harm with respect to the data security.

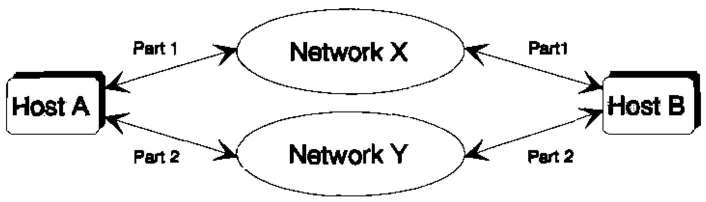

Figure 4: Distinct Routes between two hosts

Data Partitioning. Both proposed methods have in common, that a decision about partitioning the data for encryption (resp. distinct routes) has to be made. Since we provide independence from the application layer, this decision has to be made in the transport layer. This might lead to a weaker security compared with application-level encryption. For particular video streams and application scenarios, default rates can be given. The split also may be performed by a network node based on application-provided information about the stream's structure, e.g. as an extended flow specification. A more practical approach is that the application splits the stream itself into parts and specifies the relation among them, i.e., using a stream group concept as, for instance, described in [27]. However, this requires support of the application.

Block Size. The examples in Section 4 are based on an encryption with 64 bit block size, as used in the most common secret key algorithms. The granularity of encryption is essential for data confidentiality, as the QCIF example frames demonstrate. In case of encryption for whole PDUs or when using distinct routes, this granularity may get too large. In the distinct routes approach an eavesdropper only has access to a maximum of 50 percent of the data. This guarantees a sufficient level of security, although the "encryption block size" is equal to the PDU size. Nevertheless, in this approach a whole frame may be visible at once for an eavesdropper.

To achieve the same level of security when protecting whole PDUs, the encryption rate has to be increased.

Multicast. Another question concerns the applicability for multicast transmission. The difficulty arises due to the idea of multicast is inverse to the hereby discussed distinct routes approach, since multicast should use the same route for data transmission towards various targets as long as possible. The partitioning approach can only be used for the distinct routes concept if various routes through the backbone exist such that distinct routes are used in a multicast fashion.

Using partial encryption along with multicast transmission, the same problems as with full encryption and multicast arise. Since no distinct treatment for the different participants of the multicast session is possible, only one shared key can be used within the multicast group. This implies that all encrypted packets of a data stream have to be encrypted with a session key. Using a session key within a multicast group has to be negotiated during the key exchange phase before the transmission, and coordinated when using the encryption protocol [28]. An alternative concept provides IOLUS [29], which limits the attendance of a multicast session to groups of known users, but for which the partial encryption approach also makes sense due to a frequently decryption and re-encryption at different network nodes in the IOLUS system.

6 Concluding Remarks

In this paper we showed that partial encryption can guarantee an overall sufficient level of data confidentiality for most video transmissions. Even with an encryption rate of 10 percent a video stream gets useless without the matching decryption key.

Based on these observations we discussed different methods for integrating the partial encryption approach into existing transport systems. The advantage of such an approach over complete encryption is the reduced processing load necessary for real-time video applications.

The approach to split a video stream into substreams and transmit them via distinct routes can be applied in some network and application scenarios and can yield improved security with no additional processing load. Yet, it leads to various questions and difficulties of network design which need further studies and its suitability has to be investigated in detail.

Raw audio information should be treated different from compressed video information of the partial encryption mechanism due to the high redundancy inherent to this kind of information. An opportunity for partial encryption methods may give MPEG audio compression [23] which tries to minimize the redundancy of audio information in the data stream, so partial encryption can be a choice for it, too.

To achieve a scalable QoS parameter for security we propose to integrate partial encryption mechanisms in the transport layer. This approach can be integrated without major changes to the transport system. With partial encryption we offer a new method for a QoS-driven scalable security mechanism. This QoS parameter is application-independent and can be adapted to the security needs during the data transmission.

References

[1] J. Meyer, F. Gadegast: *Securitymechanisms for Multimedia Data with the Example MPEG-1-Video.* http://www.powerweb.de/mpeg/doc/secmeng.ps.gz, 1995

[2] B. Schneier: *Applied Cryptography.* 2nd Edition, ISBN 0-471-11709-9, John Wiley, New York, 1996

[3] National Bureau of Standards: *Data Encryption Standard.* FIPS 46, Government Printing Service, 1977

[4] X. Lai: *On the Design and Security of Block Ciphers.* ETH Series in Information Processing, 1, H. Gorre Verlag, Konstanz, 1992

[5] W. Diffie, M.E. Hellmann: *New Directions in Cryptography.* IEEE Transactions on Information Theory, 6, pp. 644-654, 1976

[6] R. Rivest, A. Shamir, L. Adleman: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems.* Communications of the ACM, 21(2), pp. 120-126, 1978

[7] T. Ylönen: *The SSH (Secure Shell) Remote Login Protocol.* http://www.cs.hut.fi/ssh/RFC, 1995

[8] A.O. Freier, P. Karlton, P.C. Kocher: *The SSL Protocol Version 3.0.* ftp://ietf.org/internet-drafts/draft-ietf-tls-ssl-version3-00.txt, 1996

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

[10] T. Aalto: *IPv6 Authentication Header and Encapsulated Security Payload.* http://www.tcm.hut.fi/Opinnot/Tik-110.551/1996/ahesp.html, 1996

[11] ATM Forum: *Phase I ATM Security Specification (3rd Draft).* ATM Forum BTD-SEC-01.03, 1997

[12] J. Gray, A. Kshemkalyani, M. Matyas et al.: *ATM Cell Encryption and Key Update Synchronization.* Telecommunication Systems Journal, Vol. 7(4), pp. 391-408, 1997

[13] ISO/ IEC International Standard 10918: *Digital Compression and Coding of Continuous-Tone Still Images*. 1993
[14] W. Namgoong, N. Chaddha, T.H.Y. Meng: *Low-Power Video Encoder/Decoder Using Wavelet/TSVQ With Conditional Replenishment*. Proc.ICASSP'96, Atlanta, GA, 1996
[15] ISO/ IEC International Standard 11172: *Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s*. 1993
[16] ITU-T Recommendation H.263: *Video coding for low bit rate communication*. 1996
[17] Ron Frederic: *Experiences with real-time software video compression*. XEROX Parc, ftp://ftp.parc.xerox.com/pub/net-research/nv-paper.ps, 1994
[18] P. Bahl, P.S. Gauthier, R.A. Ulichney: *Software-only Compression, Rendering, and Playback of Digital Video*. Digital Technical Journal Vol. 7(4), 1995
[19] German National Research Center for Information Technology (GMD): *SECUDE - A General Purpose Security Toolkit*. http://www.secude.com/, 1996
[20] K. Patel, B.C. Smith, L.A. Rowe: *Performance of a Software MPEG Video Decoder*. Proc. ACM Multimedia, Anaheim, CA, 1993
[21] C.E. Shannon: *Communication Theory of Secret Systems*. Bell System Technical Journal, Vol 28(4), pp. 656-715, 1948
[22] T. Kunkelmann, R. Reinema, R. Steinmetz, T. Blecher: *Evaluation of Different Video Encryption Methods for a Secure Multimedia Gateway*. Proc. 4th COST 237 Workshop, Lisboa, Portugal, Springer Verlag, LNCS 1356, December 1997
[23] D. Y. Pan: *Digital Audio compression*. Digital Technical Journal Vol. 5(2), 1993
[24] T. Kunkelmann, R. Reinema: *A Scalable Security Architecture for Multimedia Communication Standards*. Proc. 4th IEEE Int'l Conference on Multimedia Computing and Systems, Ottawa, Canada, 1997
[25] L. Tang: *Methods for Encrypting and Decrypting MPEG Video Data Efficiently*. Proc. 4th ACM International Multimedia Conference, Boston, MA, 1996
[26] T. Dierks, C. Allen: *The TLS Protocol Version 1.0*. ftp://ietf.org./internet-drafts/draft-ietf-tls-protocol-01.txt, 1996
[27] L. Delgrossi, S. Schaller, L. Wolf: *Relationships among Dependent Real-Time Streams*. 12th Int'l Conference On Computer Communication, Seoul, Korea, 1995
[28] H.Harney, C. Muckenhirn: *Group Key Management Protocol (GKMP) Architecture*. ftp://ietf.org/internet-drafts/draft-harney-gkmp-arch-01.txt, 1996
[29] S. Mittra: *Iolus: A Framework for Scalable Secure Multicasting*. Proc. ACM SIGCOMM, Cannes, France, 1997