Architectures for Secure Multicast Communication

Rüdiger Weis[1] , Werner Geyer[2*], Christoph Kuhmünch[2]

[1] convergence integrated media GmbH,
Berlin, San Francisco, Amsterdam
ruedi@convergence.de
[2] Praktische Informatik IV
University of Mannheim
{geyer,cjk}@pi4.informatik.uni-mannheim.de

Abstract. Till now security aspects are neglected in most multicast applications. We suggest a protocol stack that provides strong cryptography, realtime streaming, and reliability based on Internet standards/drafts (OpenPGP, RTP) and our own developments (WTP, SMP). We have integrated this stack into our collaborative whiteboard system and tested it in teleseminars and conferences.

1 Introduction and Overview

Especially for multimedia streams multicasting is a very important extension to the standard Internet protocols. Till now most applications neglect important features like secure communications, realtime requirements and reliability.

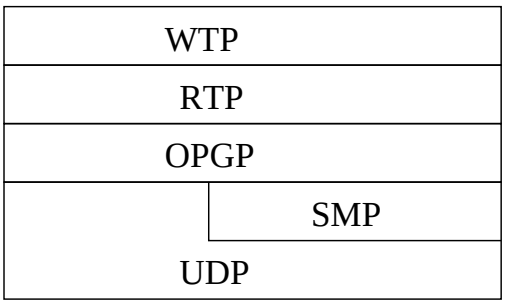

Fig. 1. dlb Stack [GeWe00]

* Since March 15th, 2000: IBM, T.J. Watson Research, USA.

We present a protocol stack that provides

- strong cryptography
- realtime streaming
- reliability

based on Internet standards/drafts OpenPGP (RFC2440 [CDFT98]) and the Real-time Transport Protocol (RTP, RFC1889 [SCFJ96] updated by Draft [SCFJ97]) and our own developments (WTP, SMP). We have integrated this stack into our collaborative whiteboard system and tested it in teleseminars and conferences.

1.1 Whiteboard Transfer Protocol (WTP)

The Whiteboard Transfer Protocol (WTP) is the application protocol of the digital lecture board (dlb) [Geye99]. WTP defines packet formats and the semantics for creating graphical objects or pages, for telepointer data, etc. For a detailed decribtion see [Geye99].

1.2 OpenPGP (OPGP)

The security concept described later uses the OpenPGP (RFC2440) [CDFT98] Internet standard. OpenPGP is an compatible to the de facto standard Pretty Good Privacy (PGP). The OPGP layer realizes the encryption and decryption of the transmitted data, i.e., RTP packets are wrapped into OPGP packets.

In contrast to S/MIME, Open-PGP only provides support for strong cryptography. There is also a possibility to integrate new algorithms. We used this to integrate the two fast, free and secure AES-candidates Rijndael and Twofish.

1.3 Real-time Transport Protocol (RTP)

WTP packets are the payload of RTP packets, a protocol that was chosen for several reasons. As mentioned above, existing MBone recording systems rely on RTP. Furthermore, the timestamps of RTP allow the synchronization with other RTP-compatible data streams (e.g., audio and video). And RTP provides light-weight session control through RTCP.

The Real Time Transport Protocol (RTP) is an application layer transport protocol that has been especially designed to transport data streams with realtime characteristics such as video and to "loosely" control sessions such as video conferences.

RTP has been developed by the Audio-Video-Transport-Group (AVT), a special interest group of the Internet Engineering Task Force (IETF). Its development has been triggered by the joint interest of the group to provide an open interface for exchanging audio and video data over datagram networks such as the Internet.

1.4 UDP and SMP

We use either unreliable UDP connections (e.g., for telepointer data) or reliable SMP connections to transmit the OPGP packets.

The *Scalable Multicast Protocol* (SMP) is a reliable transport service developed in the context of the dlb project [Grum97]. The Scalable Multicast Protocol (SMP) is a new reliable multicast protocol which was developed at the university of Mannheim.

The main features of the SMP are

- high reliability
- good scalability
- different service classes

We have integrated this stack into our collaborative whiteboard system and tested it successfully in teleseminars and conferences.

2 Related Work

Many existing video conferencing systems such as NetMeeting, ProShare, CUSeeMe, or PictureTel provide audio, video, application sharing, and standard whiteboard features but consider neither security issues nor the specific requirements of collaborative types of work, such as reference pointing, raising hands, forming work groups, controlling the course of instruction, etc.

2.1 MBone Tools

The MBone tools vic (video conferencing tool), vat (visual audio tool), and wb (whiteboard) actually support security but only weak DES encryption [MaBr94]. Due to export limitations, the DES encryption cannot be used legally outside the US for a long period.

2.2 MERCI Project

For the platform–independent whiteboard TeleDraw [TeDr98], which is being developed in the context of the MERCI project, it is planned to include MERCI security enhancements; the current version is still insecure [MERCI98]. Since TeleDraw has been designed for video conferencing, it also does not consider requirements of collaborative work.

2.3 Secure Conferencing User Agent (SCUA)

Security within the MERCI project is basically realized by the Secure Conferencing User Agent (SCUA), developed by GMD ([Hiea96], [Baea97], [Hiea97]). SCUA is an email–based approach that allows to initiate conferences securely using PEM (Privacy Enhanced Mail). For the actual transmission of data, SCUA relies on the built–in weak security mechanisms of the MBone tools. After key exchange, either the tools have to be started with the session key as a parameter or the key has to be introduced by hand.

3 Real time transport protocol

Applying MPEG compression techniques to an image sequence results in a bit stream containing the encoded video data. However, to transmit a bit stream of arbitrary length over datagram networks it has to be partitioned into data packets of appropriate size.

In the following we discuss a transport protocol for typical multimedia communication scenarios and applications. Such scenarios are for example audio and video conferencing sessions where several participants are connected via a network which provides unreliable multicast services. Each participant can send real time data and joins and leaves the session dynamically.

The Real Time Transport Protocol (RTP) is an application layer transport protocol which has been especially designed for transporting data streams with real time characteristics such as video and to "loosely" control sessions such as video conferences. RTP has been developed by the Audio-Video-Transport-Group (AVT), a special interest group of the Internet Engineering Task Force (IETF). Its development has been triggered by the joint interest of the group to provide an open interface for exchanging audio and video data over datagram networks such as the Internet. In order to send real time video over the Internet two services have to be provided:

1. As mentioned above the stream has to be divided in small packets which fit in a datagram. This process is called framing [CT90]. RTP provides a standardized packet format which is divided into a header part and a payload part. While the header part provides meta information such as timestamps, sequence numbers and data type identifiers the payload contains the essential data. RTP is open to transport any kind of media and therefore a payload format definition is necessary for each type of media. These payload format definitions are given in additional documents. Section 3.2 explains header and payload formats in more detail.
2. RTP is typically run on top of unreliable protocols like UDP to make use of multicasting services. In order to monitor the quality of service of the underlying network and to give feedback about the participants of a (multicast) session RTP includes a control protocol called Real Time Control Protocol (RTCP). Consequently, a RTP session consists of two streams: The data stream and the control stream. In case that UDP is used as underlying transport protocol applications typically use even port numbers for the data stream and the next higher odd number for the control stream. Section 3.1 summarizes the services provided by RTCP.

RTP is an open protocol which can be used in many applications with different types of data, e.g. live Internet audio/video conferences or Internet TV. The core protocol is defined in Internet draft

[SCFJ97] which revises RFC 1889[Sch96][1]. This document describes protocol specifications which are common in all applications. Additional specifications for a particular application are given in separate documents, which define an application *profile* and one or several *payload format* specifications. The *profile* specifies extensions and modifications of RTP and defines payload type codes in order to identify the payload format. For example a RTP datagram with the payload type value 100 in the RTP header is mapped to MPEG-1/MPEG-2 streams. A profile for audio and video can be found in RFC 1890 [SCFJ96]. The *payload format* specification defines how a particular payload (e.g. MPEG-1/MPEG-2) is to be carried in RTP. There already exist several Internet drafts which define payload format specifications for particular media streams. For example a payload format for MPEG-1/MPEG-2 can be found in [HFGC97].

3.1 RTP control protocol

RTCP defines control packets which are periodically transmitted from each participant to the other participants of the session and performs two mayor tasks:

1. It provides feedback on the quality of service of the underlying network. These informations can be used to allow flow and congestion control functions. E.g. a participant in a video conference can reduce his frame rate if the other participants report high packet loss rates.
2. It allows the transmission of minimal session control information, e.g. the name and the email address of a participant.

3.2 RTP data transport

It is beyond the scope of this paper to discuss all profiles and payload formats in detail. Instead we first describe the RTP-header common to all payloads followed by an overview of the MPEG-1/MPEG-2 payload format as an example for other payload types.

[1] Note that among other changes the draft specify protocol extensions for layered media streams.

The RTP datagram header contains information common to all payload formats. In Table 1 the format of such a RTP datagram header is described.

0 1 2 3 4 5 6 7	8	9 0 1 2 3 4 5 (1 at bit 10)	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 (2 at bit 20, 3 at bit 30)
flags	M	PT	sequence number
timestamp			
synchronization source (SSRC) identifier			
contribution source(CSRC) identifier . . .			

Table 1. Fixed RTP Header Fields

The first eight bits of the RTP header are used as **flags** and contain various informations like the version number and padding bits. The marker bit **M** is interpreted differently in different payload types and is followed by the payload type identifier **PT**. The **sequence number** is used to identify packet loss and to restore the original packet order. The **timestamp** reflects the sampling instant of the data transported within the RTP packet according to the Network Time Protocol. The next header field **SSRC** is intended to used as a unique identifier for a participant of a session which is chosen randomly by each participant. For the rare case that two participants choose the same SSRC the protocol describes algorithms to detect and handle such a collision. The contribution source identifiers **CSRC** are used in order to identify all contributors if data of several participants has been mixed together in the payload. For example in an audio conference one of the participants connects via a low bandwidth connection. In order to reduce the network load a gateway application can be used which “mixes” the data of several other participants in a single packet.

MPEG-1/ MPEG-2 payload format specification Because often unreliable transport protocols are used packet losses may occur frequently. Furthermore participants may dynamically join and leave a session. Internet draft [HFGC97] describes payload formats for MPEG-1 and MPEG-2 video streams which are defined with the

intention to handle these situations gracefully. For example MPEG pictures can become quite large (in the case of I-frames) and a single picture is usually spread over several packets. Hence the payload defines fragmentation rules which guarantee that the MPEG stream is split at crucial points, e.g. at the beginning of a new picture. Furthermore the payload defines a header which contains important meta information about the stream, e.g. the frame number (within the current GOP) and several flags which are set if the packet contains the start of a new picture, a new slice or if MPEG parameters (e.g. frame size) are provided. That way new participants can easily detect packets in the stream which contain important meta information necessary for decoding the pictures by parsing the RTP header.

Table 2 summarizes the MPEG specific RTP header in the payload in order to provide a more practical sense for the abstract description in the previous paragraph.

1																		*2*																*3*	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1				
MBZ					T	TR										A N	N	S	B	E	P				F B V	BFC			F F V	FFC					

Table 2. MPEG specific RTP Header Fields.

The first 4 bits **MBZ** are currently unused. They are reserved for future specifications. The **T** bit specifies the MPEG type. It is set if the RTP packet contains MPEG-2 data and erased if MPEG-1 is transmitted. The next ten bits define the temporal reference **TR** of the current picture relative to the current GOP, followed by several flags: The Active N flag **AN** is only valid for MPEG-2. Together with the new picture header flag **N** it signals changes of the MPEG-2 picture format. The **S** bit is set if the packet contains a new sequence header followed by the **B** bit and the **E** which signal the beginning or respectively the end of a slice. These bits are useful for the decoder if a packet loss occurred. In that case the decoder can easily skip packets until a necessary header is reached. The remaining

bits signal information about the picture type and coding, e.g. if it is an I, B or P frame.

4 OpenPGP

In this section we discuss some aspects of OpenPGP focused on shared secrets scenarios and the dlb seclib implementation.

The program "Pretty Good Privacy" became a "de-facto–standard" for secure E-mail. However some minor weaknesses in the most popular PGP 2.6x version have been found. In version PGP 5.0 some fixes have been implemented:

- New algorithm for the KeyID
- New algorithm for the fingerprints
- Hashfunction: SHA-1 instead of MD5

OpenPGP uses the following core technologies

- Symmetric encryption
- Asymmetric encryption and signatures
- Hash functions
- Compression
- Radix-64 Conversion

This section provides a brief overview focused on shared secret scenarios. The description closely follows [CDFT98].

5 Supported Algorithms in OpenPGP

OpenPGP supports a wide selection of cryptographic basic algorithms.

5.1 Asymmetric Algorithms

In addition to the factorisation–based RSA algorithm OpenPGP offers encryption and signature algorithms based on the Discrete Logarithm Problem (DLP).

ID	Algorithm
1	RSA (Encrypt or Sign)
2	RSA Encrypt-Only
3	RSA Sign-Only
16	ElGamal (Encrypt-Only)
17	DSA (Digital Signature Standard)
18	Elliptic Curve (reserved for)
19	ECDSA (reserved for)
20	ElGamal (Encrypt or Sign)
21	Diffie-Hellman (X9.42)
100 to 110	Private/Experimental algorithm

5.2 Hash Functions

Since several weaknesses were found in the MD5 hash function, some new hash functions are offered.

ID	Algorithm	Text Name
1	MD5	"MD5"
2	SHA-1	"SHA1"
3	RIPE-MD/160	"RIPEMD160"
4	Double-width SHA (experimental)	
5	MD2	"MD2"
6	TIGER/192	"TIGER192"
7	HAVAL (5 pass, 160-bit)	"HAVAL-5-160"
100 to 110	Private/Experimental algorithm	

5.3 Symmetrical Algorithms

Since for commercial applications IDEA is not avaible free of charge new symmetric algorithms can be chosen. Note that there is no weak cryptography or proprietarian ciphers as in S/MIME.

ID	Algorithm
0	Plaintext or unencrypted data
1	IDEA
2	Triple–DES (DES–EDE, 3 Keys)
3	CAST5 (128 bit key, as per RFC2144)
4	Blowfish (128 bit key, 16 rounds)
5	SAFER-SK128 (13 rounds)
6	DES/SK (reserved for)
7	Reserved for AES with 128-bit key
8	Reserved for AES with 192-bit key
9	Reserved for AES with 256-bit key
10	Twofish with 256-bit key
100 to 110	Private/Experimental algorithm.

6 The OpenPGP Message Format

A short description of the OpenPGP Messages Formats used in the dlb security library follows.

6.1 Packet Headers

The packet header consists of one byte that identifies the type of the packet as well as its length.

Bit	7	6	5	4	3	2	1	0

Bit 7 is always set to 1. Bit 6 indicates whether the new (OpenPGP and PGP5) or the old (PGP2.x) packet format is used.

If the packet uses the older format, bits 5 to 2 indicate the content tag, and bits 1 and 0 define which length type is used. Length types for the old packet format are:

Length bits	Length type	Description
00_b	0	The packet has a one-byte length.
01_b	1	The packet has a two-byte length.
10_b	2	The packet has a four-byte length.
11_b	3	The packet is of indeterminate length[2].

[2] The packet length is the spacw remaining of the current packet/file. This packet length shouldn't be used when creating new packets.

If the new packet format is used, the length type is encoded in the next byte (denoted as x_1, the following bytes are named $x_2, x_3, \ldots$):

Condition	Description
(1) $0 \leq x_1 \leq 191$	The length is encoded using one byte.
(2) $192 \leq x_1 \leq 223$	The length is encoded using two bytes.
(2) $224 \leq x_1 \leq 254$	The length is encoded using one byte.
(4) $x_1 = 255$	The length is encoded using five bytes.

1. The length is x_1. Possible values are between 0 and 191 bytes.
2. The length is

$$(x_1 - 192) * 256 + x_2 + 192.$$

   Possible values are between 192 and 8,383.
3. The length is

$$(((((x_2 * 256) + x_3) * 256) + x_4) * 256) + x_5.$$

   Possible values are between 0 and 4,294,967,295.
4. The length is 2^{x_1} AND $1F_h$. This is a partial length, i.e., the next byte describes another length that is added to this length. It is used if the previous length types are not sufficient to define the length. The last length type in a chain of partial length types has to be one of the three previous types, even if it is 0.

6.2 Packet Tags

When the old packet format is used, only tags 0 to 15 are available. When the new packet format is used, tags 0 to 63 are available. The currently defined types are:

Type	Description
0	Reserved - a packet tag must not have this value
1	Public-Key Encrypted Session Key Packet
2	Signature Packet
3	Symmetric-Key Encrypted Session Key Packet
4	One-Pass Signature Packet
5	Secret Key Packet
6	Public Key Packet
7	Secret Subkey Packet
8	Compressed Data Packet
9	Symmetrically Encrypted Data Packet
10	Marker Packet
11	Literal Data Packet
12	Trust Packet
13	User ID Packet
14	Public Subkey Packet
60-63	Private or Experimental Values

7 Symmetrical Encryption

OpenPGP offers extended support for symmetrical encryption. Note that the new *Symmetric-Key Encrypted Session Key* packet type (s. 7.3) is not used by PGP 2.x or PGP 5.0.

7.1 OpenPGP-CFB mode

The OpenPGP-CFB mode is a cryptographically secure variation of the standard Ciphertext Feedback (CFB) mode. An additional feature is a light-weight check if the encryption was succesfull. (Note that this check has a "fuzziness" of 2^{16}.)

Description

Let $\mathcal{B}$ the block size in bytes, $\mathcal{B} = 8$ for Blowfish, etc., and $\mathcal{B} = 16$ for AES candidates OpenPGP CFB mode uses an initialization vector (IV) of all zeros, and prefixes the plaintext with $\mathcal{B}$ bytes of random data

$$\{R_1, \ldots, R_{\mathcal{B}+2}\},$$

such that bytes

$$R_{\mathcal{B}+1} := R_{\mathcal{B}-1} \text{ and } R_{\mathcal{B}+2} := R_{\mathcal{B}}$$

The OPGP CFB Mode "resyncs" after encrypting those $\mathcal{B}+2$ bytes.

1. The feedback register (FR) is set to the IV, which is all zeros.
2. FR is encrypted to produce FRE (FR Encrypted). This is the encryption of an all-zero value.
3. FRE is xored with the first $\mathcal{B}$ bytes of random data prefixed to the plaintext to produce

   $$\{C_1, \ldots, C_{\mathcal{B}}\},$$

   the first $\mathcal{B}$ bytes of ciphertext.
4. FR is loaded with

   $$\{C_1, \ldots, C_{\mathcal{B}}\}.$$

5. FR is encrypted to produce FRE, the encryption of the first $\mathcal{B}$ bytes of ciphertext.
6. The left two bytes of FRE get xored with the next two bytes of data that were prefixed to the plaintext. This produces

   $$\{C_{\mathcal{B}+1}, C_{\mathcal{B}+2}\},$$

   the next **two** bytes of ciphertext.
7. The **resync step** FR is loaded with

   $$\{C_3, \ldots, C_{\mathcal{B}+2}\}.$$

8. FR is encrypted to produce FRE.
9. FRE is xored with the first $\mathcal{B}$ bytes of the given plaintext, now that we have finished encrypting the $\mathcal{B}+2$ bytes of prefixed data. This produces

   $$\{C_{\mathcal{B}+3}, \ldots, C_{\mathcal{B}+3+\mathcal{B}}\},$$

   the next $\mathcal{B}$ bytes of ciphertext.
10. FR is loaded with

    $$\{C_{\mathcal{B}+3}, \ldots, C_{\mathcal{B}+3+\mathcal{B}}\}.$$

11. FR is encrypted to produce FRE.
12. Now the standard CFB mode starts. FRE is xored with the next $\mathcal{B}$ bytes of plaintext to produce the next $\mathcal{B}$ bytes of ciphertext. These are loaded into FR, and the process is repeated until the plaintext is used up.

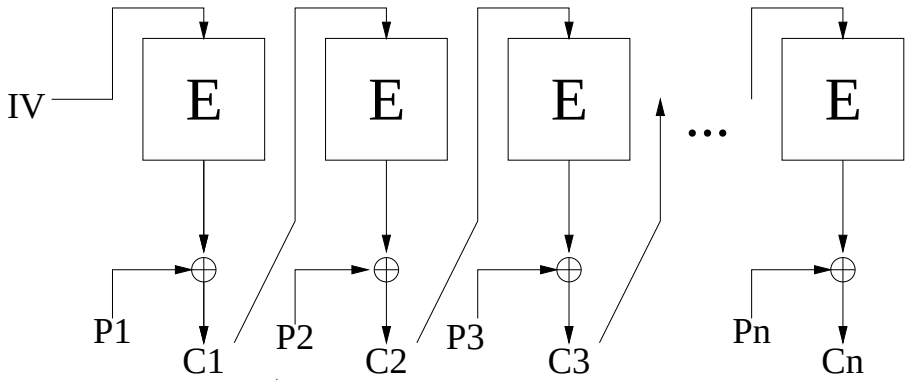

Fig. 2. Standard Ciphertext Feedback Mode (CFB)

7.2 S2K algorithms

In this subsection we discuss the two main String–To–Key algorithms. These algorithms are used to convert the user's passphrase to a symmetrical encryption key.

Simple S2K. The Simple S2K algorithm directly hashes the string to produce key data:

$$\text{KeyData} := \text{HASH}(\text{Passphrase})$$

The manner in which this is done depends on the size of the session key (which will depend on the cipher used) and the size of the hash algorithm's output. If the hash size is greater than or equal to the session key size, the high-order (leftmost) bytes of the hash are used as the key.

Multiple Instances. If the hash size is less than the key size, multiple instances of the hash context are created – enough to produce the required key data. These instances are preloaded with $0, 1, 2, \ldots$ bytes of zeros (that is to say, the first instance is not preloaded, the second is preloaded with 1 byte of zeroes, the third is preloaded with two bytes of zeroes, and so forth).

As the data is hashed, it is given independently to each hash context. Since the contexts have been initialized differently, they will produce different hash outputs. Once the passphrase has been hashed, the output data from the multiple hashes will be concatenated, first hash leftmost, to produce the key data, and any excess bytes on the right will be discarded.

Salted S2K. The *Salted S2K* includes a 64–bit random number ("Salt") in the S2K specifier that gets hashed along with the passphrase string, to help prevent dictionary attacks:

$$\text{KeyData} := \text{HASH}(\text{Salt}||\text{Passphrase})$$

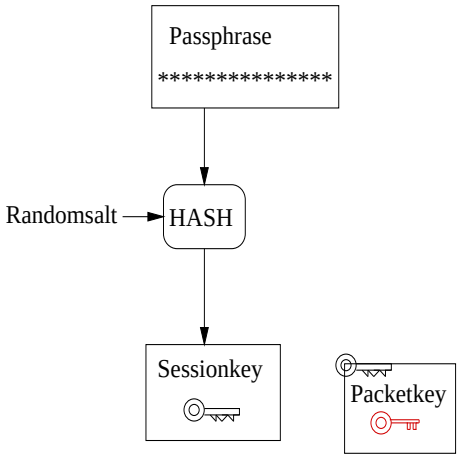

Fig. 3. Passphrase Only.

7.3 Symmetric-Key Encrypted Session Key Packets (Tag 3)

A Symmetric-Key Encrypted Session Key packet holds the symmetric key encryption of a session key used to encrypt a message. Zero or more Symmetric-Key Encrypted Session Key packets may precede a Symmetrically Encrypted Data Packet that holds an encrypted message. The message is encrypted with a session key. The session key is itself encrypted and stored in the *Encrypted Session Key packet* or the *Symmetric-Key Encrypted Session Key* packet.

The body of a Symmetric-Key Encrypted Session-Key packet consists of:

- A one-byte version number (Currently version is 4).
- A one-byte number for the symmetric algorithm used.
- A one-byte number string-to-key (S2K) specifier.
- Optionally, the encrypted session key.

If no encrypted session key is present (which can be detected on the basis of the packet length and the S2K specifier size), then the S2K algorithm applied to the passphrase produces the session key directly. If the encrypted session key is present, the result of applying the S2K algorithm to the passphrase is used to decrypt the encrypted session key field, using the CFB mode with an IV of all zeros.

The decryption result consists of a one-byte algorithm identifier that specifies the symmetric-key encryption algorithm and the session key used to encrypt the following *Symmetrically Encrypted Data Packet*. Because an all-zero IV is used, the S2K specifier must use a salt value.

7.4 Symmetrically Encrypted Data Packet (Tag 9)

The Symmetrically Encrypted Data packet contains data encrypted with a symmetric-key algorithm. When it has been decrypted, it will typically contain other packets (i.e. literal data packets or compressed data packets).

The body of this packet consists of:

- Encrypted data, the output of the selected symmetric-key cipher operating in PGP's variant of CFB mode.

The symmetric cipher used can be specified in an Public-Key or Symmetric-Key Encrypted Session Key packet that precedes the Symmetrically Encrypted Data Packet. In that case, the cipher algorithm byte is prefixed to the session key before it is encrypted. If none of these packet types precedes the encrypted data, the IDEA algorithm is used with the session key calculated as the MD5 hash of the passphrase.

7.5 Literal Data Packet (Tag 11)

A Literal Data packet contains the body of a message; this data is not to be interpreted further.

The body of this packet consists of:

- A one-byte field that describes how the data is formatted.

If it is a 'b' (0x62), then the literal packet contains binary data. If it is a 't' (0x74), then it contains text data, and might need line ends converted to local form, or other text-mode changes. RFC 1991 also defined a value of 'l' as a 'local' mode for machine-local conversions. This use is now deprecated.

- File name as a string (one-byte length, followed by file name), if the encrypted data should be saved as a file.

If the special filename `_CONSOLE` is used, the message is considered to be "for your eyes only". This advises that the message data is unusually sensitive, and that the receiving program should process it more carefully, normally avoiding storing the received data to disk.

- A four-byte number that indicates the modification date of the file, or the creation time of the packet, or a zero that indicates the present time.
- The remainder of the packet is literal data.

Text data is stored with `<CR><LF>` text endings (i.e. network-normal line endings). These should be converted to local line endings by the receiving software.

8 Scalable Multicast Protocol (SMP)

In contrast to the transmission of audio and video streams, interactive, cooperative applications like whiteboards or group editors require reliable data transmission. IP multicast is an unreliable, best-effort service, i.e. data packets can get lost, doubled, or disordered. Since the traditional field of application in the MBone is audio and video broadcasting, there are only few reliable multicast protocols available by now. In addition the complexity of providing multicast reliability in an efficient way hindered the emergence of a multicast transport protocol standard like TCP in the case of unicast. Therefore, reliable multicast protocols are actively researched at present (see [Hof98],[Flo95],[Lev96a],[Bor94]).

Since reliable multicast protocols were not ubiquitous in 1997, we have decided to develop our own protocol called smp (scalable multicast protocol). The background was simply the pragmatic requirement of having a reliable multicast protocol for our whiteboard project dlb (digital lecture board) [Gey98a].

When starting this work we had the following requirements to smp:

- guaranteed reliability
- good scalability
- separation between application level and transport level
- different service classes
- easy handling and light-weight implementation

8.1 Classes of Reliable Multicast Protocols

Depending whether error detection is done at the sender or at the receiver, we distinguish between sender-initiated and receiver-initiated multicast protocols.

In sender-initiated protocols the receivers of data typically send acknowledgements (ack) for each received data packet to the sender. After a certain period of time without having received an ack (time-out) the sender retransmits the data packet. The overhead for the administration of acknowledgements and timers can become rather high in large groups such that an efficient error recovery cannot be ensured due to end system overload. The flooding of senders with

acknowledgements is known as sender implosion [Dan94] or, more specifically, as ack implosion.

In receiver-initiated protocols the single receivers of a communication group are responsible for the error detection. By means of sequence numbers the receiver detects packet loss. The retransmission of lost packets is requested explicitly by the receiver by sending so called negative acknowledgements (nack) to the sender. This approach unburdens the sender from the administration of timers and acks and, since acks are not needed anymore, the required bandwidth is lower in contrast to sender-initiated approaches. But if many receivers - especially in large groups - detect the loss of the same packet, the sender is flooded with nacks (nack implosion). This means a high load at sender's end system and a high amount of unnecessary retransmissions. In 1987 Ramakrishnan et al. [Ram87] have proposed a timer-based scheme to suppress negative acknowledgements in such a case (nack avoidance). An improved version of this scheme has been integrated into the MBone whiteboard wb [Flo95] and is known as SRM (scalable reliable multicast).

Latest approaches avoid the implosion problem by using a hierarchical structure of receivers. Acknowledgements are not send to the sender directly but to the father in a tree structure or to a representative of a local group. Examples for tree-based protocols are Lorax[Lev96a] or Reliable Multicast Transport Protocol (RMTP) [Lin96].

8.2 Selecting a Protocol Class

Analysis of existing protocol classes [Gru97] indicate that sender-initiated as well as simple receiver-initiated protocols without nack-avoidance do not satisfy our requirements for smp. The scalability is limited and, moreover, these protocols consume a high amount of bandwidth. Tree-based protocols have perfect scaling properties but the overhead for the administration of the tree structure is very high especially in dynamic groups (permanent reconstruction). On average, receiver-based protocols possess the most advantageous properties regarding scalability, network load, and end-to-end delay [Gru97]. However, the lack of explicit acknowledgments raises the problem of releasing buffers because the sender will never be sure

whether or not all receivers have received a certain data packet. So pure receiver-initiated protocols can only be implemented in the application itself following the so called ALF paradigm (application level framing). The Mbone whiteboard wb uses this principle to realize reliability based on the SRM protocol [Flo95]. SRM itself can be considered more a frame work than an autonomous protocol. While being very efficient this approach also entails some disadvantages: integrating SRM directly into the application (following ALF) tremendously increases the complexity of application development. The name space of application data and the SRM data need to overlap. Moreover, each application has to tailor or implement its own reliable multicast protocol.

Due to the good performance of receiver-initiated protocols with nack-avoidance we decided to take SRM also as a basis for the development of SMP. To satisfy the requirement of separation between application level and transport level, SMP was designed as an autonomous protocol with its own name and address space. Buffers are released by means of the periodic session messages already known from SRM. SMP uses these messages to acknowledge data. In addition to SRM, SMP offers different levels of scalability, a late join option, source ordering, a simple rate-based flow control and an anycast mechanism.

8.3 Architecture

SMP has been implemented as an autonomous server process that runs on each participant's machine. Applications access this service by means of a client stub (basically a linked library) integrated into the application (see Figure 4). The SMP library realizes the interprocess communication between SMP process and application and it offers a simple socket-like interface. After having established a connection to SMP, the application can open multiple connections (SMP sockets) to different multicast groups. Outgoing data are passed to SMP via interprocess communication. SMP copies and stores these data packets and sends it via IP multicast to the group. The receiving SMP processes also store the data packets and pass them to their local application. Data loss is handles according to the SRM repair scheme. Each SMP process stores the data of each sender as

long as it has received an acknowledgement of all group members via a session message. The replication of data packets sent is required since in SRM each participant may retransmit lost data packets and not only the original sender.

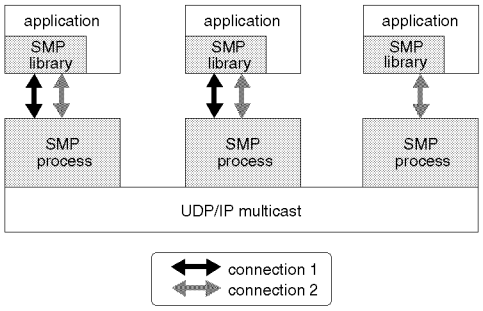

Fig. 4. dlb Stack [Geye99]

8.4 Service Classes

By providing different service classes, SMP can be tailored to application requirements to a certain extent. When establishing a new connection, SMP offers the configuration options depicted in Table 1 The user may opt between different levels of scalability, late join, source ordering, and the maximum rate control limit. Scalability class A supports small groups where scalability is a minor issue and, thus, end-to-end delay can be minimized. While class B offers a good trade-off between scalability and end-to-end delay, class C is thought for large groups requiring a high scalability of the underlying protocol. However, transmission delay suffers from the scalability requirement in this class.

The late join option enables an application to receive all the data from the beginning of a session on. This allows late comers to be initialized with the current application state by replaying all earlier events of the session. The late join option is only efficient for applications with a very small amount of data transmitted because the late join mechanism injects a rather high data volume each time a new participant joins an ongoing session. Note that all the data

Scalability	A	small groups
	B	medium groups
	C	large groups
Late Join	0	no late join
	1	new members get all the data from the beginning on
Source Ordering	0	no source ordering
	1	data are delivered in the sender's original order
Bandwidth	n	sender's constant data rate

Table 1: SMP service classes

from the beginning on is retransmitted. Second, all distributed SMP instances need to store all the data from the beginning of a session on in order to be able to serve as a late join data provider.

The source ordering option provides packet ordering of a single sender's data packets at the receivers. Of course, opting packet ordering introduces an additional delay since gaps in the sequence numbers need to be filled, i.e. repaired by packet retransmission, prior to delivering the data to the application.

SMP also supports a simple rate-based flow control. The specified value defines the continuous data rate at which SMP sends data to the multicast group. Hence, outgoing SMP traffic will never be bursty.

8.5 Application Programming Interface (API)

The SMP service is provided via the application interface of the SMP library (client stub). Access to SMP is implemented by instantiating a SMP object and by calling the corresponding object methods listed in Table 2. Most of the methods are self-explanatory. However, it is interesting to mention that the methods SendDelayedDataRequest and SendDelayedDataResponse provide an anycast mechanism. Anycast allows to select a single participant within a communication group. This basically provides an application-level solution to the sender implosion problem.

9 An Alternative Protocol Stack

In our protocol stack, whole RTP packets are encrypted within OpenPGP packets. This means that neither the RTP header as nor the

Method	Description
Open	open a connection to the local SMP process
JoinGroup	join a multicast group/session
LeaveGroup	leave a multicast group/session
Close	shut down connection to local SMP process
Send	send regular data packets
SendDelayedDataRequest	send delayed data (anycast)
SendDelayedDataResponse	reply to delayed data request (anycast)
ServiceInd	indicate a service event to the application
ServiceAvailable	check whether a running SMP process is available

Table 2: SMP's application programming interface

payload of the RTP packet is accessible for nodes within the network. Only those nodes that have the valid key can read the header informations. In some situations this might cause problems.

Consider a situation where an RTP media stream must be processed/filtered in some way by a media gateway [Kuhm99]. In this case it might be useful if at least the header informations of the RTP packet were accessible to the media gateway. On the other hand the headers contain some informations that should not be readable for outsiders. This holds true especially for some fields in the RTCP packets (e.g., the CNAME, email addresses, etc.).

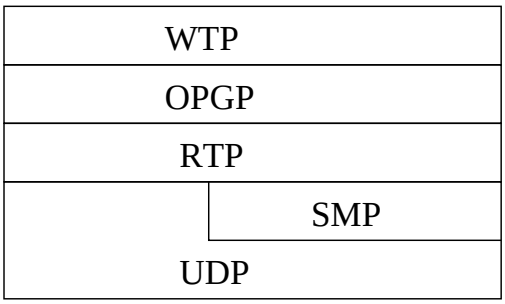

Fig. 5. Alternative dlb Stack [Weis00]

Separate Encryption of Header and Payload

We thus propose that the RTP payload and the RTP header should be encrypted with two different keys. Media gateways should receive the key that allows them to access the RTP header in order to filter the media stream. The payload itself should not be decryptable by the media gateways. Note that in this scenario outsiders are not able to access the header informations.

10 Outlook

Till now we have focused on symmetrical cryptography. Integration of public key techniques especially a direct using of GNU Privacy Guard (GPG) key seems to be a very promising project. Additionally we are working on JAVA card integration to provide highly secure conferences.

References

[Adam97a] Adams, C., "RFC2144, The CAST-128 Encryption Algorithm", May 1997.

[Adam97b] Adams, C., "Constructing Symmetric Ciphers Using the CAST Design Procedure", in: Designs, Codes and Cryptography, v.12, n. 3, Nov 1997, pp. 71–104.

[Adam98] Adams, C.,"CAST-256", AES submission, 1998.

[Baea97] Bahr, K., Hinsch, E., Jaegemann, A., Wang, L., "Handling Confidential Internet Conferences by E–mail", `ftp://ftp.darmstadt.gmd.de/pub/mice/JENC8.ps.gz`

[Blaz96] Blaze, M., "High-Bandwidth Encryption with Low-Bandwidth Smartcards", Fast Software Encryption , Springer LNCS 1039 1996.

[Blea96] Blaze, M., Diffie, W., Rivest, R., Schneier, B., Shimomura, T., Thompson, E., Wiener, M., "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security", a report by an ad hoc group of cryptographers and computer scientists, January 1996.

[BFN98] Blaze, M., Feigenbaum, J., and Naor, M., "A Formal Treatment of Remotely Keyed Encryption (Extended Abstract) ", Eurocrypt '98, Springer LNCS 1403, 1998.

[Bor94] Bormann, C., Ott, J., Gehrcke, C., Kerschat, T., Seifert, N.: "MTP-2: Towards Achieving the S.E.R.O Properties for Multicast Transport". In: Proc. ICCCN, San Francisco, September 1994.

[CDFT98] Callas, J., Donnerhacke, L., Finnley, H., Thayer, R., "OP Formats - OpenPGP Message Format", RFC 2440, Nov 1998.

[CT90] Clark, D., Tennenhouse. D., "Architectural considerations for a new generation of protocols", SIGCOMM Symposium on Communications Architectures and Protocols, Philadelphia, IEEE, September 1990, pp 200 – 208.

[DaRi98] Daemen, J., Rijmen, V., "Rijndael", AES submission, 1998. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

[Dall99] Dallas Semiconductors, iButton Hompage: http://www.ibutton.com/

[Dan94] Danzig, P.B.: "Flow Control for Limited Buffer Multicast", IEEE Transactions on Software Engineering, Vol. 20, No. 1, January 1994, pp. 1-12.

[Dee91] Deering, S., Multicast Routing in a Datagram Internetwork, PhD thesis, Stanford Univerity, California, USA, 1991.

[Ecea97] Eckert, A., Geyer, W., Effelsberg, W., "A Distance Learning System for Higher Education Based on Telecommunications and Multimedia – A Compound Organizational, Pedagogical, and Technical Approach", Proc. ED–MEDIA'97, Calgary, June 1997.

[DBP96] Dobbertin, H., Bosselaers, A., Preneel, B., "RIPEMD–160, a strengthened version of RIPEMD", Proc. Fast Software Encryption (ed. D. Gollmann), LNCS 1039, Springer, 1996, pp. 71–82.

[Dobb96] Dobbertin, H., "The Status of MD5 after a Recent Attack", CryptoBytes, 2(2):1–6, 1996.

[EfGe98] Effelsberg, W., Geyer, W., "Tools for Digital Lecturing - What We Have and What We Need", Proc. BITE '98, Bringing Information Technology to Education, Integrating Information & Communication Technology in Higher Education, Maastricht, Netherlands, March 1998.

[EFF98] Electronic Frontier Foundation, "EFF press release (July 17, 1998): EFF Builds DES Cracker that proves that Data Encryption Standard is insecure", http://www.eff.org/descracker/

[FIPS46] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard", January 1977.

[Flo95] Floyd, S., Jacobson, V., McCanne, S., Liu, C., Zhang, L.: "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", IEEE Transactions on Networking, 1995.

[Gey98a] Geyer, W.: "The Digital Lecture Board - A Teaching and Learning Tool for Remote Instruction in Higher Education", Proc. of ED-MEDIA'98, Freiburg, June 1998.

[Geye98b] Geyer, W., "The digital lecture board (dlb)" http://www.informatik.uni-mannheim.de/ geyer/dlb/dlb.eng.html

[Geye99] Geyer, W., "Das digital lecture board - Konzeption, Design und Entwicklung eines Whiteboards für synchrones Teleteaching" (in German), Reihe DISDBIS, Bd. 58, ISBN 3- 89601-458-7, Infix-Verlag, St. Augustin, 1999.

[GeEf98] Geyer, W., Effelsberg, W., "The Digital Lecture Board – A Teaching and Learning Tool for Remote Instruction in Higher Education", ED–MEDIA '98, Freiburg, Germany, June 1998.

[GeWe98] Geyer, W., Weis, R., "A Secure, Accountable, and Collaborative Whiteboard", Workshop on Interactive Distributed Multimedia Systems and Services, IDMS '98, Oslo, September 1998.

[GeWe00] Geyer, W., Weis, R., "The Design and the Security Concept of a Collaborative Whiteboard", Computer Communications 23, Elsevier, 2000.

[Grum97] Grumann, M., "Entwurf und Implementierung eines zuverlässigen Multicast–Protokolls zur Unterstützung sicherer Gruppenkommunikation in einer TeleTeaching–Umgebung", Master's Thesis (in German), Lehrstuhl für Praktische Informatik IV, University of Mannheim, 1997.

[HiGe97] Hilt, V., Geyer, W., "A Model for Collaborative Services in Distributed Learning Environments", IDMS '97, Darmstadt, LNCS 1309, 1997.

[Hiea96] Hinsch, E., Jaegemann, A., Wang, L., "The Secure Conferencing User Agent – A Tool to Provide Secure Conferencing with MBONE Multimedia Conferencing Applications"m Proc. IDMS '96, Berlin, LNCS 1045, 1996, pp. 131–142.

[Hiea97] Hinsch, E., Jaegemann, A., Wang, L., "Experiences with the Secure Conferencing User Agent – a Tool to Provide Secure Conferencing with MBONE Multimedia Conferencing Applications". `ftp://ftp.darmstadt.gmd.de/pub/mice/JENC7.ps.gz`

[Holf97] Holfelder, W., "Interactive Remote Recording and Playback of Multicast Videoconferences", Proc. IDMS '97, Darmstadt, LNCS 1309, 1997.

[Hof98] Hofmann, M.: "Skalierbare Multicast-Kommunikation in Weitverkehrsnetzen. Dis- sertation, Universit"at Karlsruhe, Infix Verlag, Sankt Augustin, 1998.

[HFGC97] Hoffmann, D., Fernando, G., Goyal, V., Civanlar, M., "Rtp payload format for mpeg1/mpeg2 video", Internet draft, IETF, Audio/Video Transport Working Group, draft-ietf-avt-rtp-new-00, November 1997.

[Hol97] Holfelder, W.: "Interactive Remote Recording and Playback of Multicast Videoconferences" ,in: Proc. IDMS'97, Springer, LNCS 1309, 1997, p. 450-463.

[Kett00] Kettler, S., "Encrypted Whiteboard Communication", Studienarbeit, Universität Mannheim, Feb 2000.

[KiRo96] Kilian, J., Rogaway, P., "How to protect DES against exhaustive key search", Proc. Advances in Cryptology–Crypto '96, Springer, 1996.

[Koch98] Koch, Werner, "The GNU Privacy Guard", 1998. http://www.gnupg.org/

[Kuhm99] Christoph Kuhmünch. A multicast gateway for dial-in lines. In *Proc. European Conference on Multimedia Applications, Systems and Tools EC-MAST*, 1999.

[Lin96] Lin, J. C., Paul, S.: "RMTP: A Reliable Multicast Transport Protocol". In: Proc. of IEEE INFOCOM'96, M"arz 1996, S.1414-1425.

[Lev96a] Levine, B. N., Lavo, D., Garcia-Luna-Aceves, J.: "The Case for Concurrent Relia- ble Multicast Using Shared Ack Trees". URL: http://www.cse.ucsc.edu/research/ ccrg/publications.html, Mai 1996.

[Lai91] Lai, X., "Markov ciphers and Differential Cryptoanalyis", Proc. of EUROCRYPT '91, Advances in Cryptology, Springer, 1991.

[Luck97] Lucks, S., "On the Security of Remotely Keyed Encryption", Fast Software Encryption, Springer LNCS, 1997.

[Luck98] Lucks, S., "On the Power of Whitening", Manuscript, Universtität Mannheim, Fakultät für Mathematik und Informatik, 1998.

[LuWe99a] Lucks, S., Weis, R., "Remotely Keyed Encryption Using Non-Encrypting Smart Cards". USENIX Workshop on Smartcard Technology, Chicago, May 10-11, 1999

[LuWe00] Lucks, S., Weis, R., "How to Make DES-based Smartcards fit for the 21-st Century", Technical Report, Universität Mannheim, 2000.

[MaBr94] Macedonia, M.R., Brutzmann, D.P., "MBone Provides Audio and Video Across the Internet", IEEE Computer. 27(4), 1994.

[MERCI98] Multimedia European Research Conferencing Integration, Telematics for Research Project 1007, 1996–1998. `http://www-mice.cs.ucl.ac.uk/mice/merci/`

[MJ95] McCanne, S., Jacobson, V., "vic: A flexible framework for packet video", *MultiMedia '95 (San Francisco)*, New York, November 1995. ACM, ACM Press.

[Ram87] Ramakrishnan, S., Jain, B. N.: "A Negative Acknoledgement with Periodic Polling Protocol for Multicast over LANs". In: Proc. IEEE INFOCOM'87, M"arz 1987, S. 502-511.

[SCFJ96] Schulzrinne, H., Casner, S., Frederick, R. Jacobson, V., "Rtp: A transport protocol for real-time applications, Internet Request For Comments, IETF, RFC-1889, January 1996.

[SCFJ97] Schulzrinne, H., Casner, S., Frederick, R. Jacobson, V., "Rtp: A transport protocol for real-time applications, Internet draft, IETF, Audio/Video Transport Working Group, draft-ietf-avt-rtp-new-00, December, expiring date June 5^{th}, 1998. 1997.

[Sch96] Schulzrinne, H., "Rtp profile for audio and video conferences with minimal control", Internet Request For Comments, IETF, RFC-1890, Januar 1996.

[Schn96] Schneier, B., "Applied Cryptography Second Edition", John Wiley & Sons, New York, NY, 1996.

[Scea98] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N., "Twofish", AES submision, 1998.

[TeDr98] Part of the Telematics for Research Project 1007 MERCI, 1996–1998. http://www.uni-stuttgart.de/Rus/Projects/MERCI/MERCI/TeleDraw/Info.html

[Weis97] Weis, R., "Combined Cryptoanalytic Attacks against Signature and Encryption schemes", (in German), A la Card aktuell 23/97, S.279, 1997.

[Weis99a] Weis, R., "Crypto Hacking Export Restrictions", Chaos Communication Camp, Berlin, 1999.

[Weis99b] Weis, R., "A Protocol Improvement for High-Bandwidth Encryption Using Non-Encrypting Smart Cards", IFIP TC-11, Working Groups 11.1 and 11.2, 7 th Annual Working Conference on Information Security Management & Small Systems Security, Amsterdam, 1999.

[Weis00] Weis, R., "A Trivial Host Card Encryption Protocol", Technical Report, Universität Mannheim, Feb. 2000.

[WKT97] Weis, R., Kuhn, M., Tron, "Hacking Chipcards", Workshop CCC '97, Hamburg 1997.

[WeGe99] Weis, R., Geyer W., "Cryptographic Concepts for Online-Feedback in Teleteaching Applications", Proc. NLT '99, Bern, Switzerland, 1999.

[Weis98] Weis, R., "Moderne Blockchiffrierer" (in German), in: "Kryptographie", Weka–Fachzeitschriften–Verlag, Poing, 1998.

[WBL00] Weis, R., Bakker, B., Lucks, S., "How to Ring a S/WAN. Adding tamper resistant authentication to Linux IPSec", SANE2000 - 2nd International System Administration and Networking Conference, Maastricht, 2000.

[WeBo00] Weis, R., Bogk, A., "Videoencryption with the JAVA i-button", CEBIT2000, Hannover, 2000. `http://www.informatik.uni-mannheim.de/~rweis/cebit2000/`

[Weis00] Weis, R., "Cryptographic Protocols and Algorithms for Distributed Multimedia Systems", PhD thesis, Universität Mannheim, 2000.