Using PGP/GnuPG and S/MIME with Email

Tilman Linneweh `<tilman@arved.de>`

Abstract

PGP/GnuPG and S/MIME are the two most common Encryption Standards used in Email. Both do have various different advantages and disadvantages. This article focuses on the technical and social Implementations and tries to give an Overview over differences and similarieties.

Contents

1 Introduction

1.1 The Problem

Electronic Mail (Email) is transmitted via the ”Simple Mail Transfer Protocol” (SMTP). This protocol was designed in the early days of the Internet and has two important design problems:

- Like all protocols of the early days, the data transmission is done in *cleartext*. The message is not protected from eavesdroppers.
- There is no method of Sender-authentication implemented. Everyone can send an Email with your Email address as sender address.

Suffering from this two problems, Electronic Mail can’t be compared with a normal letter. Email is more like a postcard, which everyone can read and everyone can fake.

Electronic-Mail can not be neither used to transmit private data nor for authentication of transactions in E-commerce. Because of the lack of these features, there have been various attempts to establish a general Encryption standard for Electronic Mail.

This standard has to be based on asymmetric encryption since Electronic Mail is many-to-many communication can’t be handled by symmetric encryption algorithms, since it would require a key for every possible combination between two parties.

1.2 PEM

The first attempt (1985 - 1993) was PEM (Privacy Enhancement for Internet Electronic Mail) defined in [RFC1421], [RFC1422], [RFC1423], [RFC1424] for ASCII text and the extension MOSS (MIME Object Security Services) defined in [RFC1848] for everything else. PEM is like S/MIME, which we will describe later, based on X.509.

Since no major software application implemented the PEM standard, PEM is only used in closed user groups.

Todays most common standards PGP and S/MIME are described in the following sections.

2 PGP/GnuPG

2.1 History

The program ”Pretty Good Privacy” (PGP) was written in 1991 by Phil Zimmermann, while the US Senate debated about a Law Proposal, which required providers of communication systems to ensure that the government is able to obtain the plain text content of their customers communication.

The first versions of PGP used the patented RSA encryption for asymmetric encryption. The encrypted binary data was transmitted uuencoded with an MD4-Hash. Version 2.0 used the IDEA Algorithm as symmetric Encryption and an MD5-Hash.

Since RSA was patented in the USA, it was not legal to use or distribute PGP. In Europe, for commercial use a license for the IDEA algorithm was needed. [Stalli95]

With version 2.5 PGP uses the RSAREF-library from RSA Data Security Inc., which was free to use inside the USA and PGP was now official distributed by the MIT and VIACrypt. Stale Schuhmacher ported PGP 2.5 to the MPILIB, which was free to use in Europe instead of the RSALIB. This version was distributed as the ”International” version of PGP.

After the Lawsuit accusing Phil Zimmermann of violating the export restrictions was turned down, he founded PGP Inc. and started programming of the next version, which was released 1997 as PGP 5.0.

Version 5.0 had support for many cryptographic algorithms. The symmetric part of the encryption could be done with IDEA and additionally Triple-DES and CAST. In Addition to RSA version 5.0 could use Diffie-Hellman (El Gamal) for encryption and DSS/DSA for signing. The Hash code can be generated with MD5 or SHA-1.

At the End of 1997 PGP Inc. was bought by Network Associates (NAI). 1998 NAI released PGP 6.0 which contains a plug-in for MS Outlook and support for Photo-IDs [Gerling01]

1999 6.5 was released. Main Addition was a VPN protocol based on X.509. Also in 1999 the first version of GnuPG was released. It was developed in Europe to prevent the US regulations and patented algorithms were avoided

2000 7.0 supported MS Windows 2000 and contains a plugin for the ICQ Instant Messenger.

In 2001 Security flaws were found in the DSA Algorithm, which under some circumstances made it possible to an Attacker, that has access to the private key stored on the disk and a signed message, to compose messages signed with the victims key. But of course if an attacker has access to the private key, it is more likely that such an Attacker will successfully crack the weak passphrase. [KliRos01]

In 2002 PGP Corporation has bought back the PGP business unit from NAI.

This section was mainly compiled from [Stalli98] and the PGP Website.

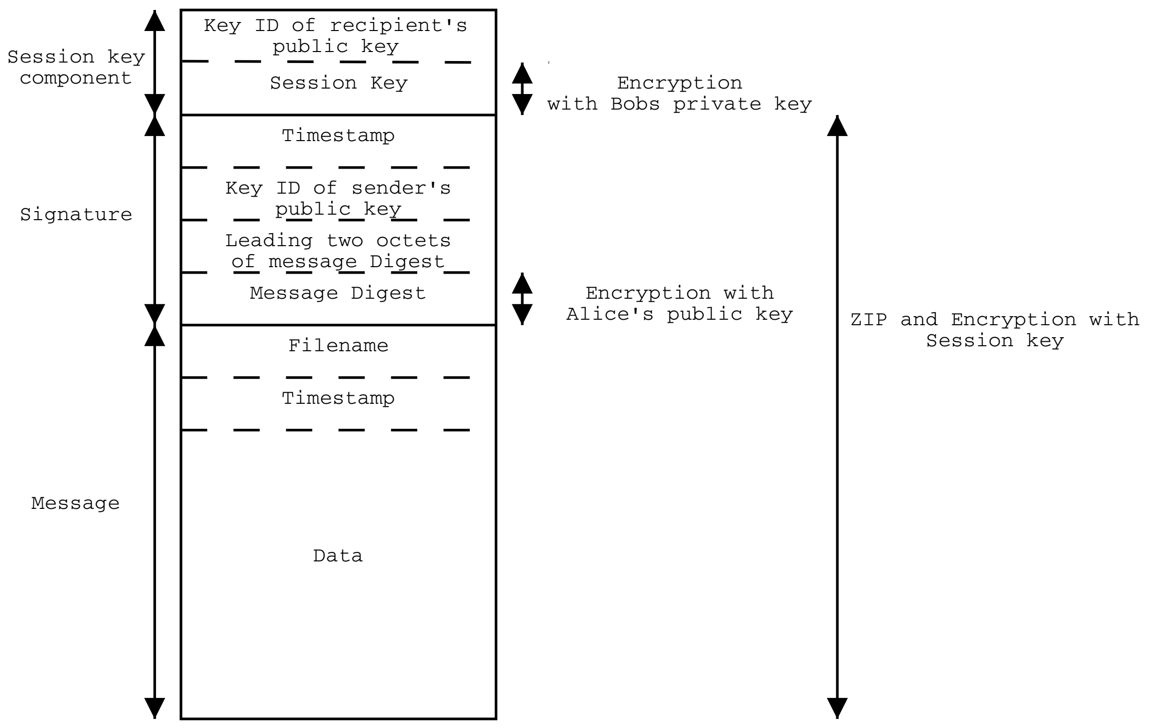
Session key component
Signature
Message
Key ID of recipient's public key
Session Key
Timestamp
Key ID of sender's public key
Leading two octets of message Digest
Message Digest
Filename
Timestamp
Data
Encryption with Bobs private key
Encryption with Alice's public key
ZIP and Encryption with Session key

Figure 1: General Format of a PGP Message (from Alice to Bob)

2.2 Operational Description

2.2.1 Using PGP to sign Messages

Creating a signed message is done in the following steps:

1. The sender creates a message
2. A hash code is taken with SHA-1 or MD5
3. The hash code is encrypted with RSA or DSA using the senders private key and appended to the message.

Verifying a signed message is done in the following steps:

1. The receiver uses RSA with the sender's public key to decrypt and recover the hashcode.
2. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

2.2.2 Using PGP to encrypt Messages

Creating an encrypted message is done in the following steps:

1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.

2. The message is encrypted using CAST-128, IDEA or 3DES with the session key
3. The session key is encrypted with RSA using the recipient's public key, and is prepended to the message.

Decrypting an encrypted message is done in the following steps:

1. The receiver uses RSA with its private key to decrypt and recover the session key.
2. The session key is used to decrypt the message.

2.2.3 Using PGP to sign and encrypt Messages

The methods described above can be applied to the same message. First the message is signed and then the message with the signature is encrypted. PGP compresses messages per default.

2.2.4 The PGP Message Format

The structure of a PGP Message is shown in Figure 1. This Figure shows some aspects, we have not mentioned yet:

- A compression is applied after the signature, to save diskspace and to remove redundancy from the message before encrypting it. The Compression is applied after the signing, because the ZIP Algorithm is non-deterministic.
- Since the encrypted Message is binary, it has to be converted to ASCII before it is delivered. This "armoring" is done with a radix-64 conversion.
- Because a User can use more than one Key pair the PGP Message can also contains Key IDs of sender and receiver, so the right Key for the decryption can be determined. The Key ID consists of the least significant 64bit.

2.3 The Web of Trust

2.3.1 Exchanging keys

If Alice wants to send Bob an encrypted message, she needs to receive the the public key from Bob. This Key exchange has to occur on a different media than email. There are the following possibilities:

- Physical. When Alice and Bob meet, they exchange their keys. This is the most secure method, but it is not always possible
- Telephone, Paper. Alice and Bob can exchange their Keys through letters or telephone in the radix64 format. This is very timeconsuming and can easy result in errors. More practical is the exchange of the fingerprint, which is a hash digest of the key via telephone/paper and then transfer the keys via Electronic Mail and verify the fingerprints. A good example for this method is the Key of the c't magazine, which is printed in the imprint of each magazine.
- Exchange the keys through a third person or authority, which both Alice and Bob trust. This third person Charlie generates a signed certificate. This certificate includes Bob's public key, the time of creation of the key, and a validity period for the key. This certificate is signed with Charlie's public key. The certificate can then directly be sent to Alice. This method requires that Alice has already verified Charlie's key.

Since method three is the most practical method, large chains of people trusting each other exist. These chains are connected to a "Web of Trust". A sample graphic is shown in Figure 2

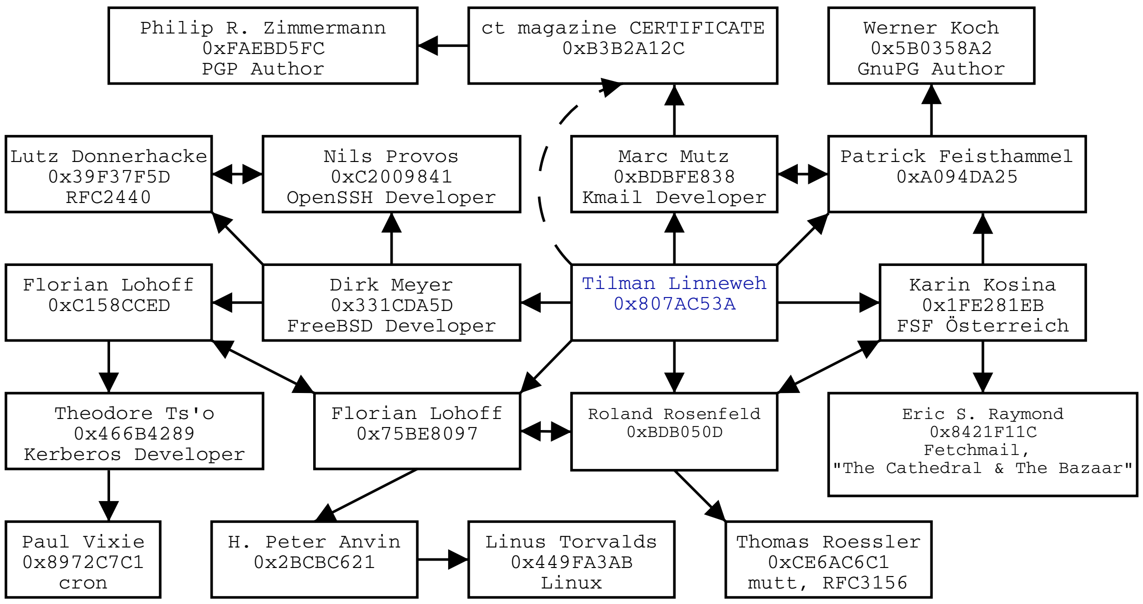

Figure 2: A Web of Trust between myself and some famous hackers

2.3.2 Keyservers

A Keyserver stores Keys and the signed certificates for this keys. Everyone who gets a signed message or who wants to deliver an encrypted message, can contact the keyserver to get the public Key of a person he doesn't know. PGP/GnuPG can be configured to automatically fetch unknown Key IDs from a Keyserver.

There are various public Keyservers all around the world which synchronize each other more or less frequent. They are usual reachable via http, vie Electronic Mail and sometimes via ftp. If you want to make your key available to other people, you can upload your key. Public Keyservers are write-only. It is not possible for anyone to delete a key from a Public Keyserver, so a key that is uploaded to a keyserver is archived and can be requested even if the Keyowner lost his corresponding private key or has even died.

The fact, that there is a key of Person X available on a Keyserver does not mean anything, since the Keyserver-software does not check the validity of the key.

If a Key gets compromised, it is possible to attach a key revocation certificate to the key, to make sure that nobody uses this key. It is also possible to generate a subkey which states how long a key is valid.

Some statistics about the Keys on a public keyservers taken from www.us.pgp.net in April 2002 [Streib02], which also gives an impression about how many people may use PGP:

Size of binary keyring	ca. 1900 MB
Number of Keys	1,649,308
Non-revoked keys with at least one non-self signed signature	158,707
Total non-self signed signatures	332,894

2.3.3 Key Analysis

Like the Figure 2, the global Keyring can be seen as a huge graph. Every Key is a point and every signature is an edge. Only 10% of the Points do have an edge.

There are thousands of small sets between groups of friends. As people from different sets exchange keys, these sets grow together to one larger set. The larger a set is, the higher is the possibility, that other sets get connected to this one. So in the end there will be one large set that sucks all the smaller sets. At the Moment (December 1) there are ca. 16.000 keys connected to the strong set [Harris02]. Over the last year the strong set has grown with a rate of ca. 114 Keys/week.

Here is a table, how many hops are need to reach my key from every other key in the strong set:

0 hops:	1
1 hops:	40
2 hops:	806
3 hops:	3254
4 hops:	5128
5 hops:	3644
6 hops:	1887
7 hops:	722
8 hops:	304
9 hops:	97
10 hops:	37
11 hops:	21
12 hops:	6
13 hops:	2

The mean distance to my key from all keys in the strong set is:	4.4220
The Average mean distance of all keys in the strong set is :	6.4797

Another nice keyring statistic site can be found at [Langasek02], where the global keyring is modeled as a sphere and volume and density are compared over the last year.

2.4 Implementation in existing MUAs

People who want to use, PGP for Electronic Mail are facing various Problems:

1. Some older versions are incompatible with newer version. Older versions don't understand the Algorithms of the newer PGP versions. Users of GnuPG e.g. may not have an IDEA license and the IDEA plugin to read the IDEA encrypted messages.

2. Some broken MTAs are converting mails to different encodings (8bit, 7bit, quoted printable etc.). This may damage the signature.

3. Some MUAs don't understand PGP, e.g. the MS Outlook Express program which is one of the most used MUAs displays PGP Mails as empty Mail with attachment, and because of the Email-viruses, most users don't open Attachments from empty mails anymore.

But the main problem people using PGP are facing is, that there currently two common Mailformats which are used for sending PGP Mail. The ”right” one is defined in [RFC3156]. It defines MIME Content-Types for PGP messages. The other common format declares the PGP body as plain/text. The second is easier to implement for Plugins, because there is no need to implement MIME.

Using MIME makes it easier for MUAs to display the PGP message. The user doesn’t have to read ”BEGIN PGP MESSAGE” or the unreadable characters of the PGP signature, while reading his email. And when he replies, the MUA can automatically remove the parts which are not intended to be read by humans.

MUAs that support OpenPGP native or with Plugins are among others:

- Qualcomm Eudora
- Mozilla
- mutt (The author of mutt has written the RFC)
- Apple Mail
- Sylpheed
- Ximian Evolution

A detailed list can be found in the GnuPG FAQ.

```
From mihi@nacl.ch3cooh.org  Wed Dec  4 13:12:13 2002
Return-Path: <mihi@nacl.ch3cooh.org>
Received: from nacl.ch3cooh.org (chello213047125199.14.univie.teleweb.at [213.47.125.199])
by postler.viennaweb.at (8.11.6/8.11.6) with ESMTP id gB4CCCC00718
for <tilman@arved.de>; Wed, 4 Dec 2002 13:12:12 +0100
Received: by nacl.ch3cooh.org (Postfix, from userid 1000)
id 8D9D356D56; Wed,  4 Dec 2002 13:16:44 +0100 (CET)
Date: Wed, 4 Dec 2002 13:16:44 +0100
From: Michael Bauer <mihi@ch3cooh.org>
To: Tilman Linneweh <tilman@arved.de>
Subject: Re: cing
Message-ID: <20021204121644.GA17121@nacl.ch3cooh.org>
References: <20021202145433.GA11816@nacl.ch3cooh.org> <20021202181031.5fd0dce5.tilman@arved.de>
Mime-Version: 1.0
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted";
boundary="fUYQa+Pmc3FrFX/N"
Content-Disposition: inline
In-Reply-To: <20021204125736.7da89c1c.tilman@arved.de>
User-Agent: Mutt/1.4i


--fUYQa+Pmc3FrFX/N
Content-Type: application/pgp-encrypted
Content-Disposition: attachment

Version: 1

--fUYQa+Pmc3FrFX/N
Content-Type: application/octet-stream
Content-Disposition: inline; filename="msg.asc"

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.6 (OpenBSD)
Comment: For info see http://www.gnupg.org

hQEOAzOkQ8D6NRmGEAP/WKuLwbQqesgxiJGVwG8nAHr+QOXhqxl8vSwwuERdVonQ
xVq1xIdJCjbBbyj3IRelEO8XC4zysZtPmOVwEV+iqen9BkbRoe8YR2GdY3t/RaIv
17QJkrnH42esbScRP0kKZFP772lWTMyTLQlOAUllwxVb5vrBI1Jw8QxS10q5QIgD
+gMgpWfozXgfI1uAW/XXgCWRomDidNJ7hhP9N4t8kGOeX9MWV4LNRZKmLUVxNMKt
SKjtHTQJodwxcVrGARdr6sx4HwN1rOvUn4a37Yj6MkBIUThOQd7QaJ5UrTwh7jQX
HdONW3ubgqEi90YzCn/nxj4e3O0xP+NuWzMx9yRMyWip0ukBynO72ab4xTulLvUl
PFiZWJZ2wiwFBt9eHb2IVJn1ivfwqzWdUChln11gqBknxWpqeFpHBrAyU4wEFfdm
BFvTZG+5m9StVhppKLyanm3p8m1Wv/o0xKRGetnveJsYglRQ2Bq75FKe6di/lAfe
r5K5gcR/OSUwkF30KPJ42hPdWESS3KaE52q+U5BCvRIym9ABobat/h0RFZtDfWwV
qk8jyUcpWg+XMTnS0yWCAlp3xEv5HY7NpdA/0e22+V6YPDAM71yTU7qxhHWm/wxE
kICYMC6r849F/w2rAkMHnXbUieCGmIw50yQ8UmKqaQ28ff5Ag7vq3O5FZDKlREVn
Qityrhp02OEAR0N08/tz2//JFt4rQvZdUb8lpH2OME5WkXFB5stB3jpqlKa4VDK0
Fslt8/C5K290ZMFUE89vB+mSetoXwQ1Z1JUOnSeGT9Lk4NgE1GnYxSWagLgi8mjI
MXV0UkpzdETIxUmsDu5bX/fWXWSAFHLJn2U1wvo90ZCOl0Pc4qfqhRH9sgLn5rwb
7Pievl5rjq/U+7DAJWiK4w7XIfAfbuGyd7XcOUKE19uPTPurK8PrdhtxeP+CPNNf
6+4MOYORsI+k6Vu4srekG8nR79OHhysWMmTooa7eys+BkScL6upS6D4JiQm7s4/1
0vtmn2xHzdVicwpHZ6V53seAgW5Ssvhee0+3gPkOxwhWEhfUt3lwaeyviN7708wr
aBFPDhHj4uu6vSj2ddwhDmSaovu8dUJBxvJlDTLt8pBiSELYSfTXg39t4s8Mz/5S
N/ZlkfMUvyiUdMDoo54StuzilhcyVZU4RTpslKR9XyOr4g==
=+vde
-----END PGP MESSAGE-----

--fUYQa+Pmc3FrFX/N--
```

The other common format declares the PGP body as plain/text and uses the default inline format generated by PGP. This format is standardized in [RFC2440] This is easier to implement for 3rd party Plugins, because there is no need to implement the MIME, the mailbody can be directly passed to a PGP-wrapper. But it may cause more problems with signatures, because signatures can only validated, if the message only contains ASCII characters, and attachments can't be signed.

The "bad" MUAs that only speak the inline format include among others:

- MS Outlook
- pine
- elm
- Lotus Notes
- Novell Groupwise

```
Received: by postler.viennaweb.at (mbox arvedde)
 (with Cubic Circle’s cucipop (v1.31 1998/05/13) Wed Jun 12 18:21:06 2002)
X-From_: kyrah@gnu.org  Wed Jun 12 18:11:35 2002
Return-Path: <kyrah@gnu.org>
Received: from eris.sim.no (N823P008.adsl.highway.telekom.at [62.47.46.200])
by postler.viennaweb.at (8.11.6/8.11.6) with ESMTP id g5CGBYm07690
for <tilman@arved.de>; Wed, 12 Jun 2002 18:11:34 +0200
Received: (from kyrah@localhost)
by eris.sim.no (8.10.2/8.10.2) id g5CGFcl10012
for tilman@arved.de; Wed, 12 Jun 2002 18:15:38 +0200 (CEST)
X-Authentication-Warning: eris.sim.no: kyrah set sender to kyrah@gnu.org using -f
Date: Wed, 12 Jun 2002 18:15:38 +0200
From: Karin Kosina <kyrah@gnu.org>
To: Tilman Linneweh <tilman@arved.de>
Subject: Re: Your signed key!
Message-ID: <20020612181538.D6721@sim.no>
References: <200206121807.50868.tilman@arved.de>
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
User-Agent: Mutt/1.2.5i
In-Reply-To: <200206121807.50868.tilman@arved.de>; from tilman@arved.de on Wed, Jun 12, 2002 at 06:07:41PM +0200
X-Home-Page: http://kyrah.net
X-PGP-Public-Key: 1024D/1FE281EB
X-PGP-Fingerprint: DD56 D01C FAEC C59A CA81 5951 EEFB DD3A 1FE2 81EB

-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.6 (Darwin)
Comment: For info see http://www.gnupg.org

hQEOAzOkQ8D6NRmGEAP+NWb8U4w/M0MAGy21JijZ9qmk9UZzsoaFl6DfaRrbKlGP
ulu8SEI9zGuOmnHsRAYTPVNrEyYkxu3ZwjDZKkDWDrJHCL6cHfpDiQnqa4k6UVwn
vQmpOqS4+G4nkpeJI23CPMQmEarSFgzSYZqAmeKThtTA0dRhp2EwV+FfAforr2UD
/jr954ZQaZ/0inw51Tp1CORe/PHyEtTwd+EGhjkzs9ZLXFEtIvxvyhh7KiTtUAHG
nyNu86nNCCO39RCOooCV+26j8s/N62xX9tN8qCgiXQqh3DkykGnK+o2OMWJiz49H
PDXphFT8lbpfjwhWVewdAMAFmZAEtyuCJ/GSxKEO43fp0sDfAUa2rwpUxm2hudKB
3UUPOwujiqQ0uU5ONV1Jr3860X5+giLO58w4HOGBFqJBEei9A3nW+TkG2/yJqcb5
wnutdCNXXYQbT0FR/7mV0rewCC07EF697OXKn+PvbwpRgiKNq4Bx9Bbu8fTBQYJB
uzI6DC+2w6QI0JnwiyjMbu+CSXMDf2ZZFIP49Vd4reCnOnZFZBGV4xWneCkuxgl/
U3aFY8/s3yy43/ER8nm9CAg9/RSOSwvBPKNpoz1dVQxVG9xuMgZ20jmNsA4qAEgY
GwUe7Bpb3dFF8G/FAA1+5oCICMV7/9fKR6W6F8jL2mBTe6met/E4qi0UjXgU0wkX
+r0ngu34tirw0x216RYToA+qhmMxljzU+AhX7SUpkESbQTIUEeF0OotA6TGSiHSD
d3nAZuw96GQ774az85w+xRYUO9MXWDlSAh1GU3hmxWsm2rFcgShqOgO3a1WBsLlc
58GpUBeRn4WcPN/xxAikhRgVxvU1Bo1xxK5/iGTtfaHTgKkqsrylI0Ig9ZBpoDrO
jbVWFRO1KxKotTjZVSOfnjL5rw==
=0gg6
-----END PGP MESSAGE-----
```

3 S/MIME

3.1 History

In Contrast to OpenPGP, which is a standardized version of the Data Format used by the program PGP, S/MIME was defined as a Standard and later implemented by different programs.

The Development of S/MIME started in 1995 at RSA Data Security Inc. together with major Electronic Mail vendors (Microsoft, Lotus, Qualcomm). S/MIME is based on CMS (Cryptographic Message Syntax) from the PKCS#7 Standard [RFC2315], which was defined by a consortium including RSA, MIT, MS, Apple, Digital, Lotus and Sun and X.509v3. PKCS is also used in SET (Secure Electronic Transaction). PKCS #7 in turn is based on ASN.1 DER (Distinguished Encoding Rules)

In 1998 S/MIME version 2 was published in [RFC2311] and [RFC2312]

In 1999 work continued in the IETF S/MIME Working group resulting in S/Mime version 3 defined in [RFC2632] and [RFC2633]

Currently S/MIME version 3.1 is discussed as an Internet-draft.

3.2 Technical Approach

S/MIME is based on MIME (Multipurpose Internet Mail Extensions) which are defined in [RFC2045] and the following. The procedures of Signing and encrypting are very similar to PGP:

3.2.1 Using S/MIME to sign a Message

Creating a signed message is done in the following steps:

1. The sender creates a message
2. A hash code is taken with SHA-1 or MD5
3. The hash code is encrypted with DSA or RSA using the senders private key and appended to the message.

Verifying a signed message is done in the following steps:

1. The receiver uses a asymmetric algorithm with the sender's public key to decrypt and recover the hashcode. S/MIME Agents must support DSA/DSS and should support RSA
2. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic. S/MIME Agents must support SHA-1 and MD5 and should use SHA-1.

3.2.2 Using S/MIME to encrypt Messages

In contrast to PGP, S/MIME was designed to not violate the US export restrictions for cryptography, so it is possible to restrict the key length.

Creating an encrypted message is done in the following steps:

1. The sender generates a message and a random session key.

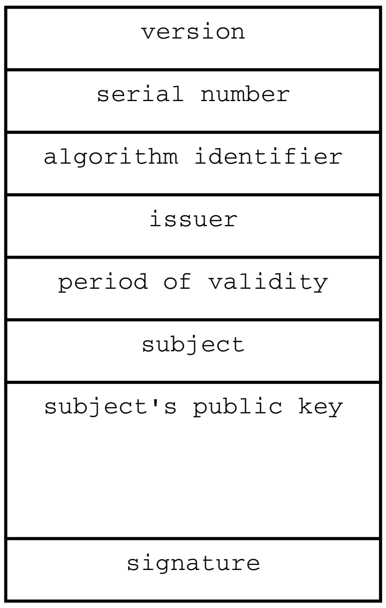

Figure 3: An X.509 certificate (from [Schneier96]

2. The message is encrypted using a symmetric algorithm. S/MIME agents must support 3DES and should support RC2
3. The session key is encrypted with RSA or El-Gamal and the Public Key of the receiver. It is also possible to encrypt the session key with a symmetric algorithm if the Connection is secure. S/MIME agents must support El-Gamal and should support RSA.

Decrypting an encrypted message is done in the following steps:

1. The receiver uses RSA with its private key to decrypt and recover the session key.
2. The session key is used to decrypt the message.

3.3 The S/MIME Certificate Format

The structure of an X.509 Message is shown in 3. The version field contains the protocol version of the certificate. the serial number is a unique number for the CA. The algorithm identifier defines the used algorithms and the needed parameters. The issuer field contains information about the CA. Period of validity contains two dates. subject contains information about the user of the certificate. The ”subject’s Public Key” field contains name and parameters of the algorithms and his public key. The last field contains the signature of the CA.

3.4 Certificate Handling

The user agent generates a key which is registered at a CA. The CA sends an X.509 certificate to the user. The user agent also has a keyring of the most well known CAs, which the user has to trust to verify incoming Mail. In difference to PGP an S/MIME key usually only contains one signature of a CA. A sample tree can bee seen in 4

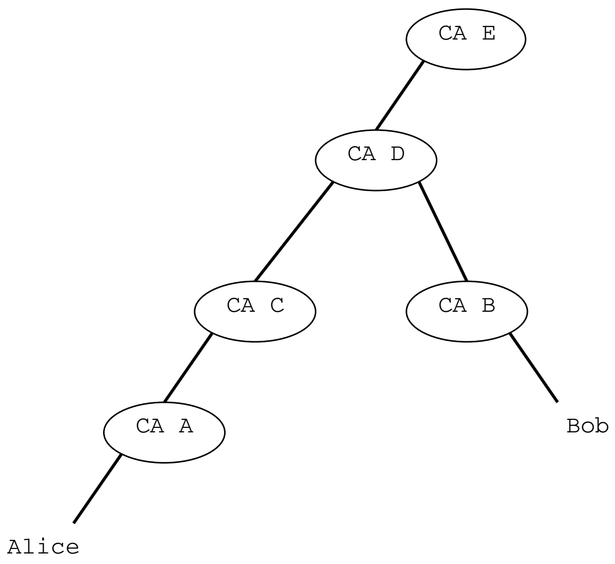

Figure 4: Example for a certification hierarchy (from [Schneier96]

The information required to get a certificate from a CA is not standardized. Often there are several Classes of certificates. Class 1 usually means only the Email-address has been verified. Class 3 usually means personal presence plus ID document and business records.

3.4.1 Exchanging keys

In Contrast to PGP, An S/MIME Signature always contains the Certificate with the Public key of the Originator. The reason is, that there is no central Keyserver like in PGP. But there are Implementations of an S/MIME Keyserver based on the LDAP protocol. With S/MIME it is easier to verify a signature of an unknown person, but it is overhead if you already have the keys of the people you exchange mail with.

Detailed descriptions can be found in [Oppliger00] and [Spiegel99].

3.5 Implementation in existing MUAs

The following common Mail Clients should support S/MIME (according to [Kossel02].

- Eudora
- Mozilla
- Outlook

Below is a sample from an S/MIME signed message.

```
Return-Path: <tilman.linneweh@web.de>
Received: from mailgate5.cinetic.de (mailgate5.cinetic.de [217.72.192.165])
by postler.viennaweb.at (8.11.6/8.11.6) with ESMTP id gBBCcrJ02399
for <tilman@arved.de>; Wed, 11 Dec 2002 13:38:53 +0100
Received: from web.de (fmomail01.dlan.cinetic.de [172.20.1.45])
by mailgate5.cinetic.de (8.11.2/8.11.2/SuSE Linux 8.11.0-0.4) with SMTP id gBBCcsX28088
for tilman@arved.de; Wed, 11 Dec 2002 13:38:54 +0100
Date: Wed, 11 Dec 2002 13:38:54 +0100
Message-Id: <200212111238.gBBCcsX28088@mailgate5.cinetic.de>
MIME-Version: 1.0
Organization: http://freemail.web.de/
From: Tilman Linneweh <tilman.linneweh@web.de>
To: tilman@arved.de
Subject: Digital Signierte Mail
Precedence: fm-user
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1; boundary="---SKER1039610333--";

This is MIME

-----SKER1039610333--
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

Hallo! Dies ist eine Digital signierte Mail
______________________________________________________________________________
Die Gruppen-SMS von WEB.DE FreeMail - mit einem Klick mehrere Freunde
gleichzeitig erreichen! http://freemail.web.de/features/?mc=021181
-----SKER1039610333--
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIIFzAIBATEJMAcGBSsOAwIaMIAGCSqGSIb3DQEHAQAA
oIID+DCCA/QwggLcoAMCAQICBAELA18wDQYJKoZIhvcNAQEEBQAwgaAxCzAJBgNV
BAYTAkRFMRIwEAYDVQQKEwlXRUIuREUgQUcxFTATBgNVBAsTDFRydXN0IENlbnRl
cjEaMBgGA1UEBxMRRC03NjIyNyBLYXJsc3J1aGUxLTArBgNVBAMTJFdFQi5ERSBU
cnVzdENlbnRlciBFTWFpbC1aZXJ0aWZpa2F0ZTEbMBkGCSqGSIb3DQEJARYMdHJ1
c3RAd2ViLmRlMB4XDTAyMTIxMTExMzUyM1oXDTAzMTIxMTExMzUyM1oweDELMAkG
A1UEBhMCQVQxFDASBgNVBAgTC09lc3RlcnJlaWNoMRIwEAYDVQQHEwkxMTcwIFdp
ZW4xGDAWBgNVBAMTD1RpbG1hbiBMaW5uZXdlaDElMCMGCSqGSIb3DQEJARYWdGls
bWFuLmxpbm5ld2VoQHdlYi5kZTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA
6cpJNbYpRzZYhbKpfFB8jIDTFK7iL6uZ3sjoBbVFpkd37mVxg9SjmaxrxRbsUfbQ
BEBOf/RAGPEoa2k1lusorrSWX3V/6syIDy6srqFzLiPWOYEh4c7y61meINFDIGc/
kFjoiBTJB/k8C+YmtN7LQa1CBJaZPAjzrpOA7W5qQrsCAwEAAaOB4DCB3TAsBglg
hkgBhvhCAQQEHxYdaHR0cHM6Ly90cnVzdC53ZWIuZGUvcnZDQS8/cz0wIwYJYIZI
AYb4QgECBBYWFGh0dHBzOi8vdHJ1c3Qud2ViLmRlMBYGCWCGSAGG+EIBAwQJFgcv
cnYvP3M9MBYGCWCGSAGG+EIBBwQJFgcvcm4vP3M9MBoGCWCGSAGG+EIBCAQNFgsv
SGlsZmUvQUdCLzApBglghkgBhvhCAQ0EHBYaRnJlZW1haWwgRW1haWwgY2VydGlm
aWNhdGUwEQYJYIZIAYb4QgEBBAQDAgCwMA0GCSqGSIb3DQEBBAUAA4IBAQCb57Di
tAkfX4em6edsUcglkcuKPJdVf9VGPrmjRF4Efyg/rOcAfjIsliSd6YMWA85kvOA8
ZBYyWkvguDb8kvXi589asBOnULFpGbE/i1KjVFq4qQ6ek/izd8VgVqbFrrPHqwZj
1qPNQXVZKGq3RhBMkDN/yf0eQjmQE8pD+dBOJb7ahfGLl+1MHUZex1YEHZGdVuVv
ZXdo9iJqnxgEbXGXu6L4AnSrklhHS9sS8VCuZ2hSa7/rHNoXOLA0hRvTDtEAp9Yv
JOLwdiZZ54oLAUz3mS8eOueCZGQUVSjNrVCPi1ZHlrncwwFmKwBjnjU/Jj67aVTH
34tSaI1Q+n/LjyP1MYIBrzCCAasCAQEwgakwgaAxCzAJBgNVBAYTAkRFMRIwEAYD
VQQKEwlXRUIuREUgQUcxFTATBgNVBAsTDFRydXN0IENlbnRlcjEaMBgGA1UEBxMR
RC03NjIyNyBLYXJsc3J1aGUxLTArBgNVBAMTJFdFQi5ERSBUcnVzdENlbnRlciBF
TWFpbC1aZXJ0aWZpa2F0ZTEbMBkGCSqGSIb3DQEJARYMdHJ1c3RAd2ViLmRlAgQB
CwNfMAkGBSsOAwIaBQCgXTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqG
SIb3DQEJBTEPFw0wMjEyMTExMjM4NTNaMCMGCSqGSIb3DQEJBDEWBBTWsrIJtD3d
IAIKf7jdCQ6S1VR5vTANBgkqhkiG9w0BAQEFAASBgJVvMHVhUXqlbiT+GChtt1PH
Nx50L4YDRMNKHK+7z/eMIjmOjmC61tWQKLLQK2T2R9AnpQ/UxOWRsWF/ccZpxSJQ
q8rwj1vSLI/CzIlvEsovm++duvhUngrXGQxjbexUPTmNIqX4rrak0CEA5QxH/vsw
TgAHErSBexlfLuP7T5ncAAAAAA==
-----SKER1039610333----
```

3.5.1 Project Ägypten

The German BSI has launched this project to implement a Free Software S/MIME reference Implementation. The Free-Software Companies Intevation, g10 Code and Klarälvdalens Datakonsult AB are contracted to produce S/MIME plugins for KMail and Mutt.

The result of this project has been recently merged into the upcoming KDE 3.1 and the mutt 1.5 Development branch.

3.5.2 Problems

The German BSI (Bundesamt für Sicherheit in der Informationstechnik) makes interoperability tests every quarter. Although everything is standardized, it is quite difficult to find a working combination. In the last tests (results at `http://www.bsi.de/` only 5 out of 12 products were able to communicate with the others.

How this could happen although everything was standardized?

Anyone who has had to work with X.509 has probably experienced what can best be described as ISO water torture, which involves ploughing through all sorts of ISO, ANSI, ITU, and IETF standards, amendments, meeting notes, draft standards, committee drafts, working drafts, and other work-in-progress documents, some of which are best understood when held upside-down in front of a mirror. This has lead to people trading hard-to-find object identifiers and ASN.1 definitions like baseball cards - "I'll swap you the OID for triple DES in exchange for the latest CRL extensions". - Peter Gutmann in his "X.509 Style Guide" `http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt`
In this document Peter Gutmann also tries to collect information about various faults in X.509 Implementations and common broken certificates.

Apart from the difficulties implementing the right standard, there is the problem of distributing the CA certificates. Usually S/MIME User Agents come with some default CAs. If a certificate is signed by a CA not in the User-Agent, the Receiver has to get the Certificate from the CA (e.g. the Website) and verify it. In Contrast to PGP the trust path is narrow and each hop(CA) becomes a Single point of failure. The verifier of a certificate has to trust all CAs on the trust path. If a CA certificate is compromised, there is no easy way to tell the users that do have the CA Certificate in their keyring. . More risks about this PKI system are described in [EllSch00].

Another Problem for S/MIME users are the costs of a certificate:

- A "TC Certificate" from `http://www.trustcenter.de` costs 62 EUR/year.
- A certificate from T-TeleSec `http://www.telekom.de/t-telesec/` is available for 50 EUR/year

4 Summary

4.1 Similarities

Both PGP and S/MIME are based on nearly the same Algorithms: RSA, DSA/DSS, 3DES etc. Both are using asymmetric encryption for exchanging a session key and both use symmetric encryption for encrypting the data.

4.2 Differences

Although having the same goal, PGP and S/MIME started from different directions.The following table shows the main differences:

PGP	*S/MIME*
Large Userbase	Small Userbase.
Supported by the community	Supported by big Companies and Government
Nearly all MUAs understand PGP	Some MUAs understand S/MIME
Easy to start, Many tutorials on the net	before starting a certificate has to be bought
Big Web of Trust	Small "Weak Tree of trusted CAs"
Two incompatible standards	Nearly everything is standardized, but there are lot of different implementations.

In my personal opinion the PGP approach is superior, since it obeys more to the KISS principle (Keep it simple, stupid). Also I trust a thick Web of Trust of various individuals more than a CA-Certificate which was signed by a random dotcom-CA.

4.3 The Future

The use of encrypted Email will probably grow, since currently most countries are increasing their efforts on intercepting communication. Everyone should use encryption as often as possible, to make encrypted email popular and so difficult to forbid.

Both standards will get more users. Currently there is no killer application for encrypted email, but there are various possibilities for the future, e.g. e-government. More MUAs will support S/MIME and PGP/MIME. This will hopefully solve the problems of interoperability.

In the near future none of the both standards will become obsolete, since both have their different purpose and different users.

5 Appendix

5.1 Software

`http://www.pgp.com`	PGP
`http://www.gnupg.org`	GnuPG
`http://sourceforge.net/projects/pks/`	PKS Keyserver

5.2 URLs

- `http://ietf.org/html.charters/smime-charter.html` Charter of the IETF S/MIME working group
- `http://www.ietf.org/html.charters/openpgp-charter.html` Charter of the IETF OpenPGP working group
- `http://www.philzimmermann.com/` Phil Zimmermann's Homepage
- `http://www.pgpi.com/` The international PGP Home Page, Sourcecode, FAQs
- `http://the.earth.li/˜noodles/pathfind.html` Jonathan McDowells Keyring Trust Pathfinder

5.3 Abbreviations

ASN.1	Abstract Syntax Notation One
CMS	Cryptographic Message Syntax
DSA/DSS	Digital Signature Algorithm, Digital Signature Standard
GPG/GnuPG	GNU Privacy Guard
IDEA	International Data Encryption Algorithm
MIME	Multipurpose Internet Mail Extensions
MOSS	MIME Object Security Services
MTA	Mail Transfer agent
MUA	Mail User Agent
PCA	Policy Certification Authority
PEM	Privacy Enhancement for Internet Electronic Mail
PGP	Pretty Good Privacy
PKCS	Public-Key Cryptography Standards
RC2	Rivest Cipher or Ron's Code after its Inventor Ronald Rivest.
RSA	Encryption Algorithm named after the inventors Rivest, Shamir, Adleman
SHA-1	Secure Hash Algorithm 1
SET	Secure Electronc Transaction
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer

References

[3] [RFC1423] David M. Balenson. Privacy enhancement for internet electronic mail: Part III: Algorithms, modes and identifiers. Internet Request for Comment RFC 1423, Internet Engineering Task Force, February 1993.

[14] [RFC2440] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. RFC 2440: OpenPGP message format, November 1998. Status: PROPOSED STANDARD.

[5] [RFC1848] S. Crocker, N. Freed, J. Galvin, and S. Murphy. RFC 1848: MIME object security services, October 1995. Status: PROPOSED STANDARD.

[16] [RFC2311] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. RFC 2311: S/MIME version 2 message specification, March 1998. Status: INFORMATIONAL.

[17] [RFC2312] S. Dusse, P. Hoffman, B. Ramsdell, and J. Weinstein. RFC 2312: S/MIME version 2 certificate handling:, March 1998. Status: INFORMATIONAL.

[25] [EllSch00] Carl Ellison and Bruce Schneier. Ten risks of PKI: What you're not being told about Public Key Infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[13] [RFC3156] M. Elkins, D. Del Torto, R. Levien, and T. Roessler. RFC 3156: MIME security with OpenPGP, August 2001. PROPOSED STANDARD.

[20] [RFC2045] N. Freed and N. Borenstein. RFC 2045: Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies, November 1996.

[7] [Gerling01] Rainer W. Gerling and Stefan Kelm. PGP, quo vadis? - die Zukunft von PGP, GnuPG und OpenPGP. *Datenschutz und Datensicherheit Vieweg Verlag*, 25, 2001.

[11] [Harris02] Jason Harris. Intermediate keyanalyze reports, December 2002. `http://jharris.cjb.net/ka/`.

[4] [RFC1424] Burton S. Kaliski, Jr. Privacy enhancement for internet electronic mail: Part IV: Key certification and related services. Internet Request for Comment RFC 1424, Internet Engineering Task Force, February 1993.

[15] [RFC2315] 5. B. Kaliski. RFC 2315: PKCS #7: Cryptographic message syntax version 1, March 1998. Status: INFORMATIONAL.

[2] [RFC1422] Steve Kent. Privacy enhancement for internet electronic mail: Part II: Certificate-based key management. Internet Request for Comment RFC 1422, Internet Engineering Task Force, February 1993.

[24] [Kossel02] Axel Kossel. Schlüsselfrage: Sicheres e-mail. *c't Heise Verlag*, , 23:142–144, 2002.

[8] [KliRos01] Vlastimil Klima and Tomas Rosa. Attack on private signature keys of the OpenPGP format, PGP^{TM}programs and other applications compatible with OpenPGP. Technical report, Group ICZ a.s., Prague, March 2001.

[12] [Langasek02] Steve Langasek. PGP web of trust: volume and density of the SCC, October 2002. `http://people.debian.org/~vorlon/keyanalysis.html`.

[1] [RFC1421] John Linn. Privacy enhancement for internet electronic mail: Part I: Message encryption and authentication procedures. Internet Request for Comment RFC 1421, Internet Engineering Task Force, February 1993.

[22] [Oppliger00] Rolf Oppliger. *Secure Messaging with PGP and S/MIME*. Artech House Inc., Norwood, MA, USA, second edition, 2000.

[18] [RFC2632] B. Ramsdell. RFC 2632: S/MIME version 3 certificate handling:, June 1999. PROPOSED STANDARD.

[19] [RFC2633] B. Ramsdell. RFC 2633: S/MIME version 3 message specification, June 1999. PROPOSED STANDARD.

[21] [Schneier96] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., New York, NY, USA, second edition, 1996.

[23] [Spiegel99] Gerald Spiegel. Hinter Schloß und Siegel: Sichere email durch S/MIME. *c't Heise Verlag*, ,8, 1999.

[6] [Stalli95] William Stallings. *Protect your privacy: the PGP user's guide*. Prentice Hall PTR, Upper Saddle River, NJ 07458, USA, 1995. With a foreword by Phil Zimmermann.

[9] [Stalli98] William Stallings. *Cryptography & Network Security: Principles & Practice*. Prentice Hall International, 2nd edition edition, 1998.

[10] [Streib02] M. Drew Streib. keyanalyze, April 2002. `http://www.dtype.org/keyanalyze/`.