Securing Internet Electronic Mail

Mark Vandenwauver[1] and Frank Jorissen[2]

[1] Katholieke Universiteit Leuven, ESAT/COSIC
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
mark.vandenwauver@esat.kuleuven.ac.be
[2] Uti-maco Belgium
De Vunt 9, B-3220 Holsbeek, Belgium
frank.jorissen@utimaco.de

Abstract. Thanks to the widespread success of the Internet, the use of e-mail has become common practice. More and more people are even using it as a primary means of communication. Unfortunately, regular users are not aware of the risks they are facing. If you send a regular letter, you can rely on the confidentiality of its content but not so with plain e-mail. Each message can be intercepted by a trained computer user connected to the net. Indeed in this paper we will show how easy it is to read other people's e-mail, and even change it without being caught. Thanks to an extensive use of cryptography, we can limit these risks. We will present and analyze an overview of the latest available standards and tools.

1 Introduction

Even in the previous era of central mainframes, there was the possibility of monitoring or active attacks on the serial lines which connected terminals to the central mainframe. However, the move to LANs in general, and Ethernet in particular, has made matters much worse. The Internet is a prime example of this. The attacks are very much easier to carry out on an Ethernet than they were on serial lines. Indeed, these attacks can often be carried out using just software, with no hardware-level interception or modification required.

Tools that are used to check the functionality of the network (LAN analyzers, also known as network sniffers) can also be used to listen in on any traffic on the Internet. Using this software (Esniff is an example of such a public domain tool), it is very easy to read all e-mail messages that pass by your computer. Services such as anonymous remailers make it impossible to track down a message received through one of them. Important e-mail messages can be altered with the use of some hardware. The need for an electronic equivalent of registered mail where you can prove to an arbitrator that the letter was sent and delivered, has become imminent due to the emergence of services as Electronic Commerce.

The rest of the paper is organized as follows. In section 2 we will describe the basic cryptographic protocol used by most of the available tools for securing Internet e-mail. Section 3 will elaborate on some of the vocabulary used in the

B. Preneel, V. Rijmen (Eds.): COSIC'97 Course, LNCS 1528, pp. 209–223, 1998.

Internet security sector. An overview of the available systems and standards will be given in section 4. A commercially available product (CryptMail) that implements most of these standards is detailed in section 5. We will finish with our conclusion.

2 Basic Cryptographic Protocol

The properties of secret key cryptography and public key cryptography are well known [1]. In this real world situation we will combine both in a hybrid system to take advantage of their respective benefits. The basic e-mail scenario is the one illustrated in figure 1. We have two entities taking part in the protocol. We have the sender (A) of the message and the receiver (B). An important aspect is that it is a one step protocol. There is usually no prior interaction between sender and receiver to setup cryptographic keys or to authenticate each other. Basically the services that we want to offer are :

- confidentiality of the message
- authenticity of the receiver/sender of the message
- integrity of the message
- non-repudiation of origin and delivery of the message

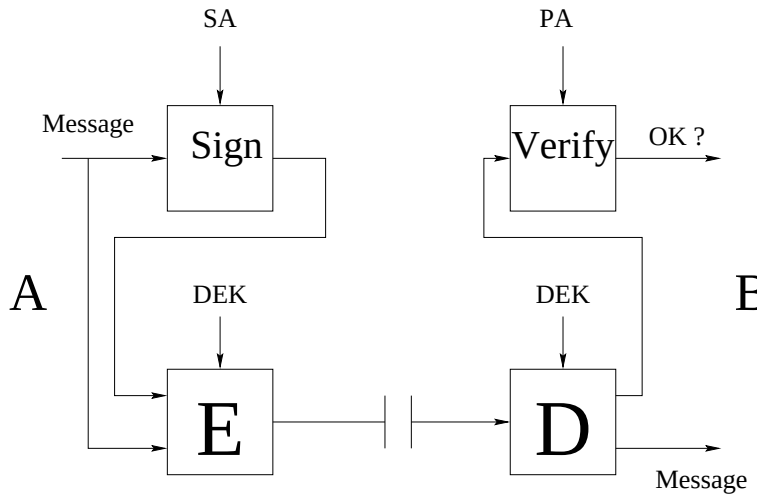

Fig. 1. Secure e-mail scheme

To protect against an intruder listening in on the electronic mail, the message will be enciphered using a symmetric key algorithm. The reason for using symmetric technology is obvious : performance. The key that is used in this way

is called the *Data Encrypting Key* (DEK). To avoid that all of these DEK's have to be distributed in an off-line way, we will use a public key system to send the DEK along with the message. When A sends a secure message to B, he uses B's public key P_B to encipher the DEK. When B receives the message he will be able to decipher the encrypted DEK because he knows the corresponding private key (S_B). Also he is the only one to have this key. This means that nobody but B can read the message (authenticity of the receiver).

A digital signature [2] will also be applied by the sender using his private key (S_A). The receiver of the message can then verify this signature using A's public key (P_A). At this moment he will be sure that :

- nobody has changed the message (integrity);
- sender A is the real sender (authenticity of the sender);
- he can prove to an outsider (e.g. a judge) that A really did send this message (non-repudiation of origin).

Non-repudiation of delivery is usually not covered by the basic secure e-mail system. The whole procedure is illustrated in figures 1 and 2. Figure 1 details the signing and encrypting of the message and figure 2 shows how the DEK is transmitted.

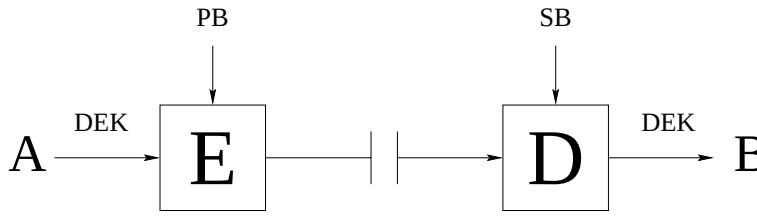

Fig. 2. Transmission of the DEK

3 Internet

3.1 Internet Standardization

Not unlike the rest of the world, the Internet community found a need for standardizing its protocols [3]. The procedure is however very different from the one used by regular standardization bodies as ISO or ITU-T.

The top organization is called the Internet Architecture Board (IAB). The IAB has 2 sub-groups called IRTF (Internet Research Task Force) and IETF (Internet Engineering Task Force). The IRTF focuses more on the long-term research, while the IETF tries to solve the short-term engineering problems. The IETF is again divided in several workgroups. To understand the rest of this

paper, it is necessary to know the different stages an idea has to go through in order to become an Internet standard. In all there are four :

1. Internet Draft
2. Proposed Standard
3. Draft Standard
4. Standard

The first stage is an Internet Draft. Anyone on the Internet can publish it. When there is sufficient interest in the community the idea will be explained in an RFC (Request For Comments) by one of the IETF working groups and it becomes a Proposed Standard. These RFCs form a set of rules with which product developers can build their products. To advance to the Draft Standard stage, there must be working implementations that have been thoroughly tested for 4 months. Ultimately the IAB can decide to have the RFC declared as an Internet Standard. Internet Standards are e.g. IP and TCP/IP. All of these standards are available from various Internet sources.

3.2 Certificates (X.509)

While public key cryptography simplifies key management a lot, it does not solve all problems at once. It remains essential to find a way to distribute each party's public key in a safe and reliable way. This problem is usually tackled by introducing Trusted Third Parties (TTPs) called Certification Authorities (CAs), that issue certificates.

Basically, a certificate contains a digital signature of the CA on a person's name and his public key, thus binding them together. In this way the CA guarantees that a particular public key belongs to a particular user. If the user can now prove that he knows the corresponding private key (e.g. by signing a challenge), he will be authenticated to his correspondent. The certificate can be seen as a token to prove one's identity just as a driver's license or a passport. Other well known Internet security tools such as the Secure Sockets Layer (SSL) also use these certificates. There are now a number of commercial CAs on the market : Verisign, AT&T, Thawte, ... In Belgium, Belsign and Isabel offer this functionality to their customers.

The ITU-T has issued a guideline (X.509) concerning certificates [4] and what they need to contain. The naming directives from X.500 are used. Up to now there have been three major releases. Version 1 originated in 1988, version 2 was published in 1993 and version 3 in 1996.

A version 1 certificate looks like this :

- *Version*: version number.
- *Serial number*: A unique number attributed by the CA. Using this serial number and the name of the issuing CA, the certificate can be located.
- *Signature algorithm*: Information about the algorithm used to sign the certificate.

- *Issuer name*: The unique name (also known as distinguished name) of the CA.
- *Validity period*: Begin time and end time of the validity of the certificate. This is based on Universal Time to avoid confusion about time zones. (This will cause problems in the year 2000, ...)
- *Subject name*: All details about the owner of the public key that is being certified.
- *Subject public key information*: All data needed to verify the certificate.
- SK_{CA}: This is the signature of the CA. The public key of the CA is needed to verify it.

Version 2 added two fields (Issuer unique identifier and Subject unique identifier) to make implementation of directory access control possible. The latest release (V3) has made the biggest change by introducing the Extensions field. They have been defined to improve the functionality and to answer some specific requests of people using X.509 certificates. These extensions contain a criticality flag indicating whether a non-compliance will mean that a certificate will be rejected or not. Specific extensions can be defined in future standards or by user communities. Some of the possible extensions include:

- Alternative naming so that the certificates can be used on Internet.
- More information about the key : e.g. whether the key can be used for key management, verification of a digital signature, encryption of data, ...
- Other identification data such as the e-mail address of the person concerned.
- Where the Certificate Revocation List (CRL) can be obtained. This is the list of the certificates that are not valid anymore. Reasons for an invalid certificate could be : the user has increased the bit-length of its modulus and needs a new public key, the user has left the organization thus rendering the certificate futile, or in the worst case scenario, the private key of a user has been broken or publicized.

3.3 Public Key Cryptographic Standards (PKCS)

This 'standard' is an initiative of RSADSI. It was written back in 1991 when the need for a standard, specifying how public key cryptography should be applied, became obvious. Other big corporations (Microsoft, DEC, ...) also support it. The standard is due for an update in 1997 and people are invited to submit their comments. In the case of electronic mail the PKCS#7 standard is very important.

PKCS#7 [5] is a list of specifications of how to apply public key cryptography to offer all the requested services (confidentiality, authentication, integrity, non-repudiation). Messages that have been encrypted using PKCS#7 can be viewed and verified, independent of the platform or the specific mail client used. PKCS#7 contains both algorithm specific as algorithm independent encryption standards. This means that some algorithms are supported implicitly while others need to abide certain syntax rules to obtain interoperability. E.g. :

DES, RSA and Diffie-Hellman are supported completely. Furthermore PKCS#7 also defines algorithm independent standards for digital signatures, digital envelopes and certificates.

PKCS#7 supports 2 kinds of certificates : X.509 and the scheme defined in PKCS#6. Basically there are 6 types of PKCS#7 messages :

- Data
- EnvelopedData: encrypted data
- SignedData: data + digital signature
- SignedandEnvelopedData
- DigestedData: data + hash
- EncryptedData: encrypted data without the DEK.

3.4 Multi-purpose Integrated Mail Extensions (MIME)

An Internet electronic mail consists of two parts : the header and the body. The header is a record, structured according to RFC 822 [6]. Until recently, the body could only contain messages that were written in ASCII. As a consequence all non-text needed to be transformed. MIME (originally defined in RFC 1521, now superseded by RFC 2045) installs a new format. This allows for other types than basic ASCII and also non textual documents. In MIME [7] several objects can be sent along in one mail message.

The format of each object is called Content-Type and these are some that are defined :

- Text Content-Type : to visualize textual information in different character sets (e.g. é, è, à, ...)
- Multipart Content-Type : to send several objects in one mail message
- Message Content-Type : contains a whole message.
- Application Content-Type : to send binary data to a specific application (e.g. a .doc file to MS Word)
- Image Content-Type : to send still images
- Audio Content Type : to send voice and/or sound files
- Video Content-Type : to transfer moving images.

A new Content-Type can always be defined. Herefore one needs to apply to IANA (Internet Assigned Numbers Authority). IANA is the organization that makes sure this is done in a clear, well documented, and public way.

The biggest problem about plain MIME is the lack of security. The solution was to define new Content-Types in RFC 1847 [8] for security (the Security Multi-parts). They are called multipart/signed and multipart/encrypted. When a mail client receives an encrypted mail message, the multi-parts make sure the message is transferred to the correct decryption engine.

4 Protocols

4.1 The Oldies

Privacy Enhanced Mail (PEM) Internet Privacy Enhanced Mail is the oldest protocol to secure electronic mail. The first version dates from 1985 and the definitive version originated in February 1993. PEM thus constitutes one of the standards for secure Internet e-mail. It is specified in RFCs 1421-1424 [9,10,11,12]. One short-coming of PEM : it only supports mail bodies according to RFC 822, thus only ASCII text.

PEM offers the four basic cryptographic services. PEM can be used with both symmetric key technology and public key technology. The number of algorithms that is supported is very limited. To obtain confidentiality, it uses the DES in CBC mode. To exchange the DEK and to sign the messages, RSA is applied together with the MD5 hashing algorithm. Unfortunately, the name they chose for the signature is the MIC (Message Integrity Check) and this is a term that is usually defined in other terms. The key management is strictly hierarchical and is specified in detail in RFC 1422. The key management protocol is based on X.509 certificates and uses the X.500 directory services. To solve the problem of defining the CAs, PEM has defined the two top levels of CAs : the Policy Certification Authorities (PCA) and the ultimate arbiter, the Internet Policy Registration Authority (IPRA). Each PCA must list his official policy and depose this with IPRA. PEM is very strict with certificates and CRLs. This results in a great confidence in the reliability of a certificate.

Structure of a PEM message

Messages sent using PEM are first converted to a canonical form so they all follow the same conventions about white space, carriage returns and line feeds. This transformation is done to eliminate the effects of all the message transfer agents that modify the messages from the sender to the receiver. Without canonicalization, these changes would affect the results of comparing hashes, etc.

A PEM message consists of a header and the encrypted data (body), separated by a blank line.

- The first field within the header is the Proc-Type. It contains the version number and the services that are offered by this PEM message.
  - MIC-CLEAR : text is sent in the clear. A non-PEM compliant mail reader will also be able to read this message.
  - MIC-ONLY : the message has been signed and then encoded to represent the messages independent of the mail platform.
  - ENCRYPTED : the message has been signed, encoded and then encrypted with the DEK.
  - CRL : the message contains a certificate revocation list.
- The second field describes the Content-Domain and this is usually RFC 822 compliant text. In case of an encrypted message we then have a first DEK-Info field. It specifies the symmetric algorithm used and also what initial value (IV) was used in the CBC mode.

- With the Originator-Certificate field we can obtain the public key or the certificate of the sender of the message.
- The Key-Info field is optional but is used most of the times when encryption is applied to the whole message. It contains the algorithm used to transfer the DEK and the DEK enciphered with the public key of the sender. The reason for this is that when the message is filed, the sender would be unable to decipher the message since the DEK is normally only enciphered with the public key of the recipient of the message. This would mean that only the recipient could read the message.
- Next is the Issuer-Certificate field that contains the certificate of the issuer of the certificate of the sender. It is clear that when the the issuer's CA is low in the CA hierarchy, we will also find all of the other issuers' certificates here so that the recipient of the mail message can verify the whole certificate chain.
- The MIC-Info field consists of three sub-fields. The first one gives the hash algorithm used, the second one the digital signature algorithm and the third the actual value of the signature. When the message is ENCRYPTED, the MIC is also enciphered with the DEK so that no one but the intended receiver can verify the MIC. These sub-fields are described in detail in RFC 1423.
- One of the last fields in the header of the message is the Recipient-ID-Asymmetric. Analogue to the Originator-Certificate it contains the address of the issuer of the recipient's certificate and its validity period.
- The ultimate Key-Info, with the DEK enciphered with the public key of the recipient, completes the information needed by the recipient to reconstruct the whole message.

Pretty Good Privacy (PGP) PGP is essentially the result of one man's work : Phil Zimmerman. It is a complete e-mail security package that provides all the requested services and even compression. Even better, the complete package (even the source code) can be obtained FREE from numerous points on the Internet. There also exists a commercial version in the US, sold by VIACRYPT, but this is not available outside the US because of the US arms export restrictions. Because of the price (nil) and the quality of the available implementations, PGP has become very popular on the Internet and has become a de facto standard.

It has been involved in several controversies, mainly because the way it first was exported out of the US (this is an illegal act) has never been very clear. Phil Zimmerman has been indicted on several accounts but has not been found guilty. His life has been made difficult by the official US agencies because of this. Last year (in 1996) he founded PGP Incorporated to commercialize his work and to further improve PGP. Since the first releases, there have been independent implementations outside of the US so that the export problem has been resolved.

The algorithms used by PGP are : RSA and MD5 for the digital signature, IDEA (CBC) for the bulk encryption and RSA for the key management (as in PEM). The latest international version (2.6.3i) supports RSA keys of up to 2048

bits long. IDEA is a block cipher invented by Massey and Lai [13] that uses a 128 bit key (rendering exhaustive key search unlikely). It is patented but PGP has obtained a licence to use it for free in any non-commercial applications. It is no coincidence that neither of these three algorithms has been designed by a government agency and that RSA and IDEA are offered with key-lengths that are much more than the ones usually allowed by the US government for export (512 bits for RSA and 40-56 bits for a symmetric key).

The key management scheme implemented by PGP is pretty unique. In contrary to the strictly hierarchical system of X.509, it is based on a web of confidence (trust of peers). Each PGP user locally has two files containing its keyrings. The public key ring is a list of all the public keys of the people the user usually corresponds with. Now how can a user be convinced that the public key of someone is genuine ? Well, he asks his friends. Each participant in the PGP scheme asks his friends to digitally sign his public key, thus generating some form of 'certificates'. Of course they are not real certificates since they are not issued by a TTP, but they will suffice in closed and/or restricted communities. There are also PGP servers on the Internet where you can obtain public keys and they offer a bit more confidence. Other possibilities to broadcast your public key is to include it in your finger information or to put it on your WWW home-page. A clear advantage of this kind of system is speed. Obtaining a secure X.509 certificate can be a very lengthy procedure and until you have it you can not start securing your e-mail. With PGP, you install the program, generate your key pair and you are on your way. Revoking a key is however very difficult, sometimes even impossible. The value of the non-repudiation service offered is legally very doubtful because of the absence of an official TTP.

PGP message

Here's how one generates a PGP message. The sender first hashes his message and, using his private key, digitally signs the hash result. When the receiver eventually gets the message, he can verify this signature to convince him of the integrity and authenticity of the message. The signature and the original message are now concatenated into one single message. The resulting message is now compressed using the ZIP program (that uses the Liv-Zempel algorithm). This message is now split into 64 bit blocks and they are encrypted using IDEA in CBC mode with the DEK (PGP calls this the session key). The DEK is also encrypted with the public key of the recipient using the RSA algorithm. The concatenation of the encrypted blocks and the encrypted DEK, is then encoded using the Base-64 algorithm. The output of this is ASCII, which means that it can be incorporated into an RFC 822 message body. At the receiver's end all of these transformations can be inverted and the recipient will be able to read the message.

4.2 The Newbies

S/MIME In early 1995, several major e-mail vendors (Microsoft, Lotus, Qualcomm, ...) got together with RSADSI to design a secure, interoperable messaging

standard. The result of their work is S/MIME. It stands for Secure/Multipurpose Internet Mail Extensions and integrates MIME with the PKCS#7 standard.

S/MIME as of the time of writing of this article is not yet published in an RFC. Its current status is that of an Internet Draft [14]. When the current compatibility tests will have been finished successfully, S/MIME will be submitted to the IETF as an RFC. Several public versions have also been made available for testing by the general public.

S/MIME recommends three symmetric encryption algorithms : DES, Triple DES and RC2. The key-size of RC2 can be adjusted to 40 bits, which is the maximum strength allowed for export outside of the US. As recent experiments and literature study have shown, 40 bit does not represent a secure system with the current state of the art in computing. It can be broken in a matter of hours on a medium size university network, or in minutes if one uses the FPGA technology to build a dedicated piece of hardware.

Key management is based on X.509 certificates. Because S/MIME is new, it supports the Version 3 certificates (PEM e.g. does not). Evidently the RSA public key cryptosystem is used to generate digital signatures, exchange keys, ...

In the draft, the new content-type application/x-pkcs7-mime is defined, to specify that a MIME body part has been cryptographically enhanced according to PKCS#7. S/MIME uses the MIME security multiparts. PKCS#7 describes a series of encodings to transfer messages over 7-bit electronic mail systems but MIME solves this with its content transfer encodings. With regular PKCS#7 there are no restrictions on the format of the data that will be encrypted or signed, but in S/MIME this data needs to be a MIME entity on its own. This will allow the result of removing the signature and/or encryption to be passed on directly to the MIME engine.

PGP/MIME PGP/MIME combines PGP with MIME, the Internet standard messaging protocol. It has recently completed its final phase of the IETF's standard process as it was published as RFC 2015 [15] in October 1996. PGP/MIME has also been adopted by the Internet EDI Working Group to be one of the core components of their draft proposal to introduce EDI over the Internet. PGP/MIME has inherited the basic PGP properties : use of strong encryption without any limits on the key-length, a model of trust based on peers, ... It is compatible with previous PGP releases. A first commercial package supporting PGP/MIME was offered in February 1997 with the release of PGPMail 4.5 for Windows 95 and NT.

The integration was done by using the MIME security multi-parts. Three new Content-Types have been defined accordingly : application/pgp-encrypted, application/pgp-signature and application/pgp-keys. The first one is used to send encrypted messages, the second to send digitally signed e-mail and the third to distribute public keys of individuals. The standard allows for messages to be both signed and encrypted.

Before encrypting with PGP, the message should be written in a MIME canonical format The resulting MIME multipart/encrypted message will con-

tain exactly two parts. The first MIME body part must have a content-type of pgp/encrypted, while the second MIME body part will contain the actual encrypted data and is labeled application/octet-stream.

When a PGP signature is applied, the data also needs to be in its canonical form. The digital signature is calculated on both the data to be signed and its set of content headers. The signature will be detached from the actual signed data so that the signing process won't affect the signed data in any way.

MIME Object Security Services (MOSS) MOSS is an integration of PEM and MIME. It is a framework that uses the MIME security multiparts to add digital signature and encryption to MIME objects. It is described in RFC 1848 [16]. MOSS leaves a lot of implementation options (such as the choice of algorithms or what kind of key management to apply) open. This makes it a very general protocol. But this also means that it is not sure that implementations offered by different vendors will be able to communicate in a secure way.

In its current form, MOSS supports two kinds of key management : one based on X.509 certificates and a manual scheme. Messages containing keys and messages containing encrypted (or signed) data are separated into different MIME messages. This allows for an encrypted message to be decrypted before the public key of the correspondent has been verified. This verification can then be done off-line.

The basic structure of a MOSS message is that of a MIME multiparts message using the security multiparts. Before a message can be signed, it is first canonicalized. Then the digital signature and other control information is generated. This control information must then be embodied in a predefined MIME content-type. Eventually the control information and data information body part will be embodied in a multipart/signed type.

Message Security Protocol (MSP) MSP was introduced by the US government's Secure Data Network System (SNDS) [17] program. SNDS was initiated by the National Security Agency (NSA) to look for methods to implement security in distributed computer systems. MSP is a protocol to provide secure message transfer on the X.400 network and is based within the OSI network model. MSP provides a large set of cryptographic services and it is the only one to have some form of non-repudiation of delivery. This is achieved by asking the recipient to send back a digitally signed acknowledgement of receipt.

There were some plans to make a MIME implementation of MSP available. This Internet Draft defined a new content/type called application/msp. However this draft has expired and has not been updated or proposed as an RFC.

5 Implementation in Cryptmail

Commercial security products, as Utimaco's *CryptMail*, need to evolve dynamically with their application domains. Therefore these products should be build

on a highly flexible and extendible architecture. To adapt products to specific security requirements, it is also important that they can easily be customized. Experience learns us that most of the necessary changes are done at the same architectural elements, so it is particularly in these areas that a product architecture needs to be flexible.

Four such architectural elements are:

- the Protocol area
- the Key Management area
- the Algorithm area
- the Hardware area

The Protocol Area

As explained in Section 4, a significant number of security protocols already exists. Evenmore, due to a lack of interoperability testing, and often because of commercial reasons, each protocol also has its different flavors, depending on the company who implemented it. This results in a growing need for interoperability between the different protocols and protocol flavours. This property can be achieved through the adoption of a structure that allows the user to choose which protocol he wants to use, rather than to oblige him to use a hardcoded one.

The Key Management area

Key management is one of the most underestimated aspects when implementing a cryptographic system. As already seen in Section 3.2, certificates are used to a great extent to achieve trust in a public key. In these certificates, the extensions that are introduced in X.509 version 3 confront implementors with a lot of questions. The exact meaning of an extension, although also defined by the ITU-T, is at present mostly determined by the first implementation of it. Likewise, some extensions (and even non-extension fields) are given a deviating meaning in standards such as e.g. SET (Secure Electronic Transactions), compared to the original definition. One might say: if the envisaged application doesn't match the protocol standard, one abuses the standard to make it fit to his application's needs ! Also, anyone is able to include new, proprietary extensions in a certificate. Of course, other applications then need to process those extensions. This may become unfeasible without the prior establishment of a limited set of interoperable profiles.

The Algorithm area

There are several reasons for having an architecture that allows algorithms to be both selectable and replaceable. Different users have different security needs, different local standards, etc.... There usually is a trade-off between the delay resulting from encryption, and the security level of the message. This means that key lengths and algorithms need to be changeable. Cryptographic algorithms also are subject to attacks, and nobody can guarantee that what is at present a secure algorithm, will also remain secure in the future.

The Hardware area

To obtain the high security level required for applications such as financial transactions, products may need to interface to hardware devices to securely

store secret keys. However, many types of hardware devices exist. Some examples: smart cards, SmartDisks (TM), cryptographic boards and servers. Also, different smart cards may have different functionalities and characterisitics. Rather than obliging a customer to use one specific type of smart card, a security product needs to incorporate the hardware device chosen by the customer in a fast and transparent way.

Problems of customization

From the above, one can understand the need for a flexible product (architecture). There is however also a drawback to this. A user may get confused by all the options he can choose from, especially when he isn't familiar with technical details of security. Also, all these options result in a vast amount of functionality that is not used after some start-up decisions are made.

Solution

The customer should only get what he really needs. This involves e.g. that a customer who has decided to only use PKCS#7 will only get the software for this protocol. This is realised by making use of a switching module. This switching module is able to scan all DLL's that are present on the system and generates a table with all the functionality that is present. According to this table, the switching module reroutes the calls to the encryption-decryption layer to the right functions. If he needs to make use of some functionality that is not present on the system, he will first try to remap it to other functions. If this doesn't work, the application will issue a warning. Another functionality offered by the switch table, is the customisability of the GUI. The application decides which options and buttons are present based on the functionality present in the table. All custom libraries are also dynamically loaded. This architecture has some clear advantages.

1. The ability to create a customised product. Each customer only gets the DLLs that he will use.
2. The reduction of the time needed to start the application. Only the supporting framework now needs to be loaded and all the other libraries only are loaded when they are really used.
3. A system can also keep track of which customer has received what part of the software. This enables the support division to focus on a much smaller base.

Cryptmail

PKCS#7 and S/MIME are typical examples of standards in which the above switching architecture can be employed usefully. In the following paragraph, the corresponding requirements in the CryptWare Architecture will be discussed.

1. As said in Section 4.2, S/MIME is based on PKCS#7. Instead of having two libraries, one for S/MIME and one for PKCS#7, that fully implement the protocols, the S/MIME library uses the PKCS#7 library. This means that the S/MIME module also uses the switching module to map its cryptographic functions back to the PKCS#7 DLL. The choice of PKCS#7 in the implementation is done by using so called "set option" commands, that

are able to choose a certain protocol. When sending a certain message, this library also can point to a sort of standard message header. This has the advantage that when the customer wants to change some standard values in the header, only a template file needs to be copied onto the system of each user.

2. As mentioned in Section 4.2, PKCS#7 uses the X.509 certificate layout, without the need for certain extensions. In case of S/MIME however, some standard extensions need to be present concerning the use of the certificates and under which policy they have been issued. This results in an extra library to be loaded, to process these extensions. Again policy decisions can be based on a separate file, to enable the system administrator to quickly change the policies that are accepted to sign a certain message.
3. S/MIME and PKCS#7 specify the use of different algorithms that need to be accepted in each compliant implementation. When some changes to the standard are made, and other algorithms become mandatory, an extra library only needs to be copied onto each system to upgrade the system. This results in a big improvement over the distribution of a new application to each customer.
4. Currently, no support for hardware is specified in the standard. But, as hardware support is frequently required nowadays, and in the context of PKCS#7 it is under investigation (see Section 3.3), it is likely that this will be added in a future version of the standard. Using the CryptWare architecture, this will also result in a new library that needs to be present.

6 Conclusion

Just as we have seen an explosion in the number of Internet users over the last years, there has been an increase in the number of solutions for securing electronic mail. Two years ago, people could choose between PEM and PGP. PGP was the market leader, mainly because of its availability and price (FREE). Now that people are committing themselves to the use of e-mail for business appications, the concern for security has grown. Due to the shift towards multimedia and the importance of user friendliness, three new schemes were proposed recently. PGP/MIME inherits most of the benefits from PGP but keeps on promoting its web of confidence scenario. While this can be used within a company or for EDI (when previous bilateral agreements are signed), it is not a good solution for electronic commerce. It will be interesting to see how PGP will evolve now that Phil Zimmerman has founded PGP Inc.

On the other hand, both S/MIME and MOSS use X.509 certificates. This means that the level of confidence in the other party's public key will be higher for secure (level 3 in the Verisign language) certificates. MOSS suffers from the fact that there are few implementations available. S/MIME is pushed by some of the largest companies (Microsoft, Netscape, ...) and will be incorporated in most of the commercially available secure e-mail tools. The reason for including MSP in this picture is that it provides non-repudiation of delivery. This is essential when using e-mail for executing secure electronic transactions.

References

1. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
2. International Standards Organization, *Digital signature schemes giving message recovery – Part 2: Mechanisms using a hash-function*, Draft International Standard ISO/IEC 9796-2, December 1996.
3. A. S. Tanenbaum, *Computer Networks*, Prentice Hall, 1996.
4. International Telecommunication Union - Telecommunications Standardization Sector, *The Directory - Authentication Framework*, Recommendation X.509, 1996.
5. RSA Laboratories, *Cryptographic Message Syntax Standard*, PKCS #7, Version 1.5, November 1993.
6. D. Crocker, *Standard for the Format of ARPA Internet Text Messages*, RFC 822, University of Delaware, August 1982.
7. N. Freed, N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC 2045, Innosoft and First Virtual, November 1996.
8. J. Galvin, S. Murphy, S. Crocker, and N. Freed, *Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted*, RFC 1847, TIS and Innosoft, September 1995.
9. J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*, RFC 1421, IAB IRTF PSRG, IETF PEM WG, February 1993.
10. S. Kent, *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate Based Key Management*, RFC 1422, BBN Communications, February 1993.
11. D. Balenson, *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers*, RFC 1423, TIS, February 1993.
12. B. Kaliski, *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*, RFC 1424, RSA Laboratories, February 1993.
13. X. Lai, J. L. Massey, and S. Murphy, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology-EUROCRYPT '91, LNCS 547, August 1991.
14. S. Dusse, *S/MIME Message Specification: PKCS Security Services for MIME*, Internet Draft, RSA Laboratories, September 1996.
15. M. Elkins, *MIME Security with Pretty Good Privacy (PGP)*, RFC 2015, Aerospace Corporation, October 1996.
16. S. Crocker, N. Freed, J. Galvin, and S. Murphy, *MIME Object Security Services*, RFC 1848, Cybercash, Innosoft and TIS, October 1995.
17. C. Dinkel (Ed.), *Secure Data Network System (SNDS) Network, Transport and Message Security Protocols*, U.S. Department of Commerce, National Institute of Standards and Technology, Report NISTIR 90-4250, 1990.