Security — Wed, June 4th

Matt Bishop Lecture
transcribed by J. Fritz Barnes

June 4, 1997

1 Overview

- Network Security
  - Electronic Mail Protocols (PEM/PGP)
  - Network Time Protocol (NTPv2)

2 Electronic Mail Protocols (PEM/PGP)

2.1 Introduction

The PEM and PGP protocols are two separate protocols that were designed to enable the *security* for email. PEM was the original protocol but is largely defunct today. PGP is the defacto standard currently in use. The description of these protocols is interspersed since both protocols are rather similar.

Both classical and public key versions of PEM exist. PGP utilizes only public key encryption. In the following description we discuss the public key version of PEM.

The goal is to send messages securely. This is provided by:

1. **Confidentiality**: an optional issue in PEM/PGP
2. **Authenticity of Origin**: means that the person who signed the message sent the message.
3. **Integrity:** message was not changed since it was signed.

4. **Non-repudiation of Origin**: we look at the technical issues. NOTE: the techniques outlined here do not deal with someone claiming that their private key may have been stolen.

When discussing integrity of a message it is important to notice that this does not necessarily mean one should have confidence in the contents of a message. What it means is that the message received by you was what was intended by the user who sent it. One must consider whether "Sender X" say Matt Bishop is trustworthy and assign their level of trust on this basis.

On the Internet we can look at ActiveX and Java. Netscape and Java use the sandbox model of security where they *attempt* to place all foreign code in the sandbox. On the other hand Microsoft lets you decide by telling you where the ActiveXlet came from (integrity maintains that what you received is what you got.). NOTE: this does not stop a malicious organization from sending you an ActiveXlet, but gives you a choice whether to trust them. Deciding trust can be a difficult issue.

2.2 Design

When designing the system the authors tried to carry the following four goals into the implementation:

- Create a solution independent of platform
- Didn't want to redesign mail handlers
- Didn't have to rewrite user mail programs
- Ability to optionally provide services

Many of these goals were because it was believed that people would be unlikely to forsake a mailer that they have grown to like and become accustomed to just for an extra feature of security.

Independence of platform was achieved by creating the solution separate for existing mailers. One can create the body of the message separate from the mailer, it is used to actually mail the encrypted/authenticated message but this can be performed in the same manner as it has always been done to move insecure messages. Optional application is also provided since the same mailers are being used, so you can decide when creating your message whether or not to apply confidentiality.

In the design it was decided to use RSA for encryption. The advantages of using RSA is that it is a public-key encryption system that does not lead to large expansion of messages. At first there were some questions about whether RSA was the correct way to go primarily these were due to the fact that RSA is licensed and patented (which lead to legal questions), and that public key encryption is typically slower than secret key ciphers.

The solution to the speed problem is to use a secret key cipher to encipher the message and use RSA to encipher the key. In PEM this is performed by randomly creating a key to use with DES-CBC (which is called the DEK for data exchange key).

For confidentiality we encipher the message with the DEK. The DEK is then enciphered with the public key of the receiver before sending. (PGP uses IDEA for a secret key encryption of the message body.)

For authentication and integrity we use the following scheme. A Interchange key (IK) is used to encode the Message Integrity Check (MIC). The IK used is the sender's private key. The MIC is a cryptographic checksum constructed from the body of the message that will be sent out. There are several different algorithms for calculating the MIC.

The Body of the message consists of the optionally enciphered text. PEM headers are calculated which define confidentiality technique used, what MIC was chosen, algorithms, etc. These two items combined together are the overall body of the mail message. ALSO for the authentication and integrity the $IK\{MIK\}$ is used as one of the headers.

Some problems with the solution as specified above:

- Different computers use different character sets. Therefore if a message is encoded on one machine it may be deciphered incorrectly on another machine. I.e. going from an ASCII to an EBCDIC machine.

- The second problem involved the fact that some mailers append newlines at the end of a message if it is not quite the right length. This appending occurs after the MIC is generated

In order to solve the above problems a canonical form was specified for mail messages. This form specified the format that mail messages would be in for transmission. Therefore one can create the message in their favored character set. It will be mapped into a character set for sending over electronic mail and then will be mapped into the character set at the far end.

Secondly a printable encoding for the inner encrypted message for confidentiality has to be performed since in many mail protocols the parity bit is not guaranteed to be transferable and a good cipher will leave this bit on as likely as it would off. The printable encoding

- Breaks characters into six bits
- Maps onto printable characters (IA-1)
- Breaks into lines of 64 characters

For users who are not using PEM software MIC-CLEAR might be specified that allows the user to still send the email with an integrity check, yet does not perform the encoding described above. (Otherwise the non-PEM user would not be able to read it.)

The transmission of public key certificates used to certify who a user is, was not discussed in this lecture but was discussed previously. PEM was one of the driving forces in the public key certification strategy.

3 Network Time Protocol - NTPv2

The Network Time Protocol looks at how to synchronize time. Time becomes a problem when dealing with computers. We often need accurate times across systems, yet the Quartz clock of a computer is not very reliable. NTP is a technique used to provide synchronization between clocks in a system.

We can separate machines into the following categories:

- **Primary Servers**: attached to clocks that are believably correct. This is considered Stratum 0 since the server is zero hops away from the correct time. Atomic clocks fit into this category. Note they are assumed to have an accuracy of 2.32 picoseconds.
- **Secondary Servers**: these servers access the primary servers to determine what time it is. This is considered Stratum 1.
- **Servers**: We are at Stratum 2.
- **Clients**: We are at Stratum 3.

Servers synchronize upwards. For the most part we do not want a high stratum because lag due to delay over the networks will reduce the accuracy of our time if we get too many hops away from a server with the true time.

Servers or Clients at a given stratum can be arranged into clouds. Elements that share the same cloud are cohorts. Synchronization between hosts in separate clouds is prohibited, but can be performed within the same cloud.

There are five types of association:

1. **Client**
2. **Server**
3. **Broadcast**: where the server sends the time to all clients
4. **Symmetric Active**: this is used since we have servers in middle stratum. This defines a broadcast where the time of the server is never modified.
5. **Symmetric Passive**: receipt of the messages from the symmetric active server.

In the above deliberation we have not discussed how a source is picked. We can predefine the source. These can fall into categories such as trusted (we would ask them the time), friendly (we are willing to tell them the time, but we don't trust their time), and every one else (we ignore them.) A second technique is to use statistical measures. These measures look at clock offset, round-trip delay and clock dispersion to determine whether they should trust a given host or not.

Resynchronizing the time on a given machine can be a complex operation. The problem is that it is not wise to just jump through time since many dependencies on ordered system time exist. Therefore if we are fast/slow it is a better idea to slow the clock frequency then to change the time. NTP provides techniques to change the frequency that clocks are getting updated.

Lastly we looked at what happens when a plot is received. There are two types of variables that exist in the system. Variables associated with the packet, and those associated with a given peer (which are therefore held in a peer table). The following pseudo-code describes what happens when a packet arrives:

```
if time packet sent == time last packet received
      then sanity = T <--- i.e. apply sanity check later
if time peer got last packet <> time last packet sent
      then sanity = T

update peer variables: stratum, poll, precision, reference id,
                       delay from primary

if peer clock not synchronized or peer not updated
      then sanity = T
```

This is as far as we got. Tune in next time to see the rest of what happens when a packet arrives.