Computer Security

Standards and Products

Standards and Products

- Kerberos
- Public Key Infrastructure (PKI)
- IPsec
- SSL/TLS
- Electronic Mail Security: PEM, S/MIME, and PGP

Kerberos

- Kerberos is a ***secret key service for providing authentication in a network.***
- Developed as part of ***Project Athena at MIT***.
- Details may be found at:
  - http://web.mit.edu/kerberos/www/
  - http://www.ietf.org/rfc/rfc1510.txt
- The original version of Kerberos was ***Version 4*** (V1-3 were internal development versions).
- ***Version 5*** is a modified version of V4, and corrects a number of security deficiencies
- Both V4 and V5 are in common use.

Kerberos

- A Kerberos service, in a network, acts as a ***trusted arbitrator***.
- It provides ***secure network authentication***, allowing a person to access different machines on the network.
- Based on ***symmetric cryptography*** (DES and others)
- Shares a different secret key with every entity on the network; knowledge of that secret key constitutes proof of identity.
- Kerberos maintains a database of clients and their secret keys.
  - For the human user, the secret key is an encrypted password.
  - Network services requiring authentication, and clients who wish to use these services, register their secret keys with Kerberos.

Kerberos

- Kerberos V4 used a non-standard mode for authentication
  - Fails to detect certain changes to ciphertext
- Kerberos V5 uses CBC mode.

Kerberos

- Kerberos V5 uses a variant of the ***Needham-Shroeder trusted third-party key exchange protocol:***

  Alice and Bob each share keys with the Kerberos KDC.

  1. Alice sends a message to the KDC with her identity and Bob's identity
  2. The KDC generates a message with:
     - a timestamp, T, a lifetime, L, a random session key, K, and Alice's identity

     and encrypts this using the key it shares with Bob (***Ticket to Bob***)

     Then the KDC generates a message with:
     - the timestamp, the lifetime, the session key, and Bob's identity

     and encrypts this using the key it shares with Alice.

     It sends both messages to Alice:

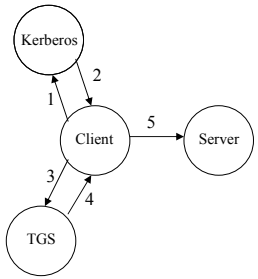
     $E_A(T, L, K, B)$ and $E_B(T, L, K, A)$

Kerberos

3. Alice generates a message with her identity, and the timestamp, T, encrypts it using the session key, and sends it to Bob.
   She also sends Bob the ticket to Bob:
   $E_K(A, T)$ and $E_B(T, L, K, A)$
4. Bob creates a message consisting of the timestamp plus one, encrypts it using the session key, and sends it to Alice:
   $E_K(T + 1)$

Kerberos

- To minimize the use of (plaintext) passwords whenever Alice wishes to use a network resource, Kerberos provides a ***Ticket-Granting Server (TGS)*** which accepts ***Ticket-Granting Tickets (TGTs)*** from users and returns a server ticket.
  1. Client requests TGT from Kerberos
  2. Kerberos returns TGT to client
  3. Client uses TGT to request a server ticket from TGS
  4. TGS returns server ticket to client
  5. Client uses server ticket to request service from server.



Public Key Infrastructure (PKI)

- A ***Public Key Infrastructure (PKI)*** consists of the components necessary to securely distribute public keys.
- Ideally, it consists of:
  - Certificates
  - A repository for retrieving certificates
  - A method for revoking certificates
  - A method of evaluating a chain of certificates from public keys that are known and trusted in advance (***Trust Anchors***).
- More details may be found at:
  - http://www.pki-page.org/
  - http://csrc.nist.gov/pki/
  - http://www.ietf.org/html.charters/pkix-charter.html
  - http://verisign.netscape.com/security/pki/understanding.html
  - http://www.schneier.com/paper-pki.html

Public Key Infrastructure (PKI)

- A certificate is a signed message vouching that a particular name is associated with a particular public key:

  [Alice's public key is 563455]$_{\text{Carol}}$

- How would Bob know to trust what Carol says? If he does not, then he can use a chain of certificates:

  [Carol's public key is 286542]$_{\text{Ted}}$ → [Alice's public key is 563455]$_{\text{Carol}}$

Public Key Infrastructure (PKI)

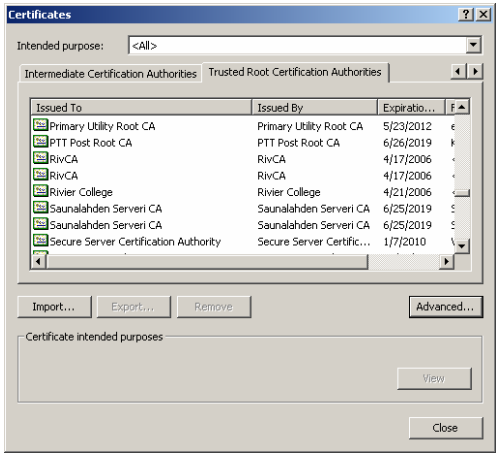
- If Alice signs a certificate vouching for Bob's name and key, then Alice is the ***issuer***, Bob is the ***subject***, of that certificate
- If Alice is trying to find a path to Bob's key, then Bob's name is the ***target***.
- If Alice is evaluating a chain of certificates, then she is the ***verifier*** (a.k.a. ***relying party***)
- Anything that has a public key is called a ***principal***.
- A ***trust anchor*** is a public key that the verifier has decided is trusted to sign certificates.
- In a verifiable chain of certificates, the first certificate will have been signed by a trust anchor.

PKI Trust Models

- Monopoly
  - Single, universally trusted CA *(Who would you vote for?)*
- Monopoly plus Registration Authorities (RAs)
  - Monopoly, plus single CA chooses other organizations (RAs) to check identities and obtain and vouch for public keys
- Delegated Certificate Authorities
  - Trust Anchor CA can issue certificates to other CAs
- Oligarchy[1]
  - Multiple top-level trust anchors (model used by browsers)
- Anarchy
  - Each user is responsible for configuring some trust anchors (used by PGP)
- And others...

[1]**ol·i·gar·chy**
1 : government by the few
2 : a government in which a small group exercises control especially for corrupt and selfish purposes; *also* **:** a group exercising such control
3 : an organization under oligarchic control

Did You Know Your Browser Has Trust Anchors?



Certificate Revocation

- Just as with credit cards, if a certificate gets stolen or compromised, or if someone gets fired from an organization, it is important to be able to ***revoke a certificate***.
- Even though a certificate typically has an ***expiration date***, as with credit cards, the ability to revoke is still important.
- Typically, a CA will periodically issue a signed list of all revoked certificates – a ***Certificate Revocation List (CRL).***
  - A complete CRL, or
  - Delta CRLs
- Verifiers are expected to check for revoked certificates in the CRL

PKIX and X.509

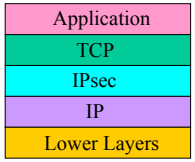
- ***PKIX*** is a profile of X.509; that is, it specifies which X.509 options should be supported.
  - See also http://www.faqs.org/rfcs/rfc3739.html
  - Uses X.500 names (directory names)
    - See http://www.nlc-bnc.ca/9/1/p1-244-e.html
    - C=US, O=Rivier College, OU=CS Dept, CN=Bryan Higgs
    - Our book thinks this choice was inappropriate, especially given the dominance and convenience of domain names (e.g. bhiggs@rivier.edu) – translation between the two is difficult
  - Uses ASN.1
    - See http://www.ncbi.nlm.nih.gov/Sitemap/Summary/asn1.html and http://asn1.elibel.tm.fr/en/index.htm
    - Abstract Syntax Notation One
    - Our book thinks this choice was inefficient.

X.509 Certificates

- Despite its form not being particularly satisfying (at least, to our textbook authors), it seems that X.509 certificates have become pretty much the accepted standard for certificates.
- An X.509 certificate contains:
  - Version number (i.e. which kind of X.509 certificate)
  - Serial number of certificate
  - Signature – the algorithm used to compute the signature on the certificate
  - Issuer – the X.500 name of the issuing CA
  - Validity – the starting and ending times for the certificate to be valid
  - Subject – The X.500 name of the entity whose name is being certified
  - SubjectPublicKeyInfo – an algorithm identifier and optional parameters for the algorithm, and the subject's public key.
  - IssuerUniqueIdentifier (optional)
  - SubjectUniqueIdentifier (optional)
  - AlgorithmIdentifier
  - Encrypted – contains the signature on all but the last of the above fields.
  - Extensions – (only in X.509 V 3 certificates)

IPsec

- ***IPsec*** is an IETF[1] standard for real-time communication security
  - Implemented between the IP and TCP network protocol layers, so usually part of the Operating System:



[1]Internet Engineering Task Force (http://www.ietf.org/)

IPsec

- In 1994, the Internet Architecture Board (IAB) issued a report "Security in the Internet Architecture" (RFC 1636)
  - Included authentication and encryption as necessary security features in the next-generation IP (IPv6)
  - These features were designed to be usable with IPv4, as well as IPv6

IPsec

- Unfortunately, IPsec was developed via a very political committee process, and the result is:
  - Ill-defined – the IPsec documentation does not specify what IPsec is supposed to achieve
  - Poorly documented
  - Very complex – "A camel is a horse designed by committee"
  - Nobody seems happy with it
- See the textbook for some pretty significant criticisms of IPsec, and some sense of the politics that were involved
- Here's another pretty strong critique:
  - http://www.schneier.com/paper-ipsec.html

IPsec

- The principal IPsec documents are:
  - RFC 2401: Security Architecture for the Internet Protocol
    - http://www.faqs.org/rfcs/rfc2401.html
  - RFC 2402: IP Authentication Header
    - Description of a packet authentication extension to IPv4 and IPv6
    - http://www.faqs.org/rfcs/rfc2402.html
  - RFC 2406: IP Encapsulating Security Payload (ESP)
    - Description of a packet encryption extension to IPv4 and IPv6
    - http://www.faqs.org/rfcs/rfc2406.html
  - RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP)
    - Specification of key management capabilities
    - http://www.faqs.org/rfcs/rfc2408.html

IPsec

- Support for the features documented in these RFCs is:
  - Mandatory for IPv6
  - Optional for IPv4
- In addition to these four RFCs, a number of additional papers have been published by the IETF's IP Security Protocol Working Group.
  - The documents are divided into seven groups (are you getting a sense of the complexity?)
    - Architecture
    - Encapsulating Security Payload (ESP)
    - Authentication Header (AH)
    - Encryption Algorithm
    - Authentication Algorithm
    - Key Management
    - Domain of Interpretation (DOI)

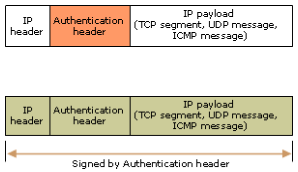
IPsec AH & ESP

- AH (Authentication Header) and ESP (Encapsulating Security Payload) are two types of IPsec headers
  - AH provides integrity protection only
  - ESP provides encryption or integrity protection, or both
  - The textbook makes it pretty clear they think that AH is mostly redundant

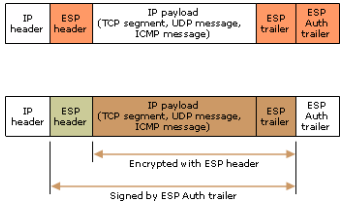
IPsec Transport Mode

- Transport mode simply adds the IPsec information between the IP header and the remainder of the packet.
- It encrypts only the IP payload.

- Authentication Header transport mode



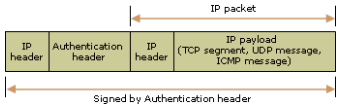
- Encapsulating Security Payload transport mode



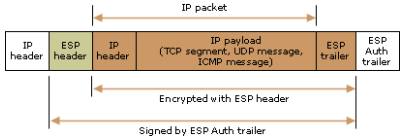
IPsec Tunnel Mode

- Tunnel mode keeps the original IP packet intact, and adds a new IP header and IPsec information outside
- It encrypts both the IP header and the payload

- AH tunnel mode



- ESP tunnel mode



IPsec: Internet Key Exchange (IKE)

- A protocol for performing ***mutual authentication and establishing a shared secret key to produce an IPsec Security Association (SA).***
  - A ***security association*** is a ***cryptographically protected connection***.
- The specification of IKE is in three pieces:
  - RFC 2407: The Internet IP Security Domain of Interpretation for ISAKMP
    - http://www.faqs.org/rfcs/rfc2407.html
  - RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP)
    - Specification of key management capabilities
    - http://www.faqs.org/rfcs/rfc2408.html
  - RFC 2409: The Internet Key Exchange (IKE)
    - A protocol for performing mutual authentication and establishing a shared secret key to produce an IPsec Security Association (SA).
    - http://www.faqs.org/rfcs/rfc2409.html

IPsec: Internet Key Exchange (IKE)

- I particularly like this quote from the book:

"There are two ways to design a system. One is to make it so simple there are obviously no deficiencies. The other is to make it so complex there are no obvious deficiencies." **-- C. A. R. Hoare**

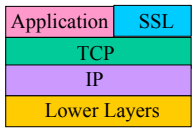
Which do you think applies to IPsec?

IPsec: Internet Key Exchange (IKE)

- For reasons of this over-the-top complexity, I will discuss IPsec: IKE no further.
  - See the textbook for details, plus the RFCs.

SSL/TLS

- ***Secure Sockets Layer (SSL)*** and ***Transport Layer Security (TLS)*** allow two parties to authenticate and establish a session key that is used to cryptographically protect the remainder of the session.
- Designed to run in the application layer:



SSL/TLS

- ***SSL Version 2*** was originally developed by Netscape for Netscape Navigator 1.1 in 1995. (V1 was internal)
- Microsoft improved upon it, fixing some security issues and introduced a similar protocol called ***PCT (Private Communications Technology)***
- Netscape overhauled SSL into ***SSL Version 3***
- The IETF then decided to standardize it, and introduced yet another incompatible protocol, ***TLS (Transport Layer Security).***
- Currently, SSL V3 is the most commonly used.

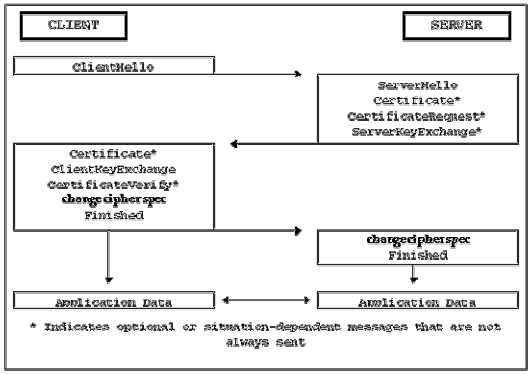
SSL Protocol

- The ***SSL Protocol*** includes two sub-protocols:
  - The ***SSL Record Protocol***
    - Defines the format used to transmit data
  - The ***SSL Handshake Protocol***
    - Sets up the parameters for the SSL record protocol
    - Allows client and server to agree on keys, ciphers, and MAC algorithms
- The SSL V2 protocol is documented at:
  - http://wp.netscape.com/eng/security/SSL_2.html
- The SSL V3 protocol is documented at:
  - http://wp.netscape.com/eng/ssl3/ssl-toc.html

SSL Handshake Protocol

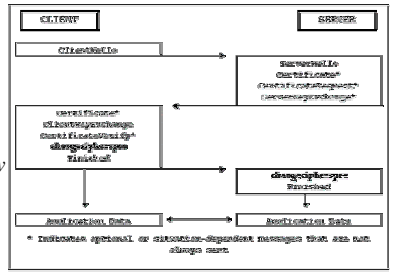
- The purpose of the SSL handshake protocol is to set up parameters for an SSL ***session***.
- Within a session, the client and the server can have multiple simultaneous ***connections***.
  - SSL was designed primarily for HTTP, which typically uses several connections for a particular web site.
  - A simple HTTP connection normally uses ***port 80***
  - An SSL-protected HTTP connection normally uses ***port 443***
  - You can tell that your browser is in an SSL-protected HTTP connection when the associated URL has a protocol of ***https***

SSL Handshake Protocol



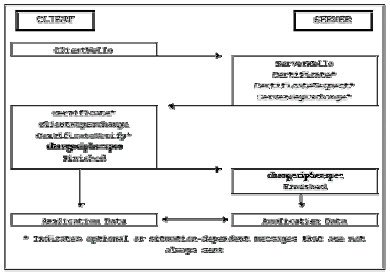
SSL Handshake Protocol

- The *client hello* message includes:
  - The *SSL version number*
  - A *client-generated random number*
  - A *session ID*
  - A *list of cipher suites supported by the client*
  - A *list of compression algorithms supported by the client.*
- The *server hello* message includes:
  - An *SSL version number* (<= client's)
  - A *server-generated random number*
  - The *session ID supplied by the client*
  - The *chosen cipher suite*
  - The *chosen compression algorithm*
- If the server is to be authenticated, it follows the server hello message with its *certificate*.



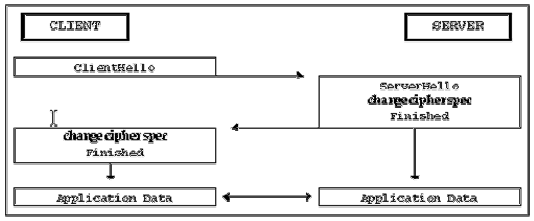
SSL Handshake Protocol

- The client and server then negotiate a symmetric key for the session, using the chosen cipher suite.
- The SSL record protocol is then used to send messages back and forth between the client and server:
  - A message is broken up into fixed-length blocks (if necessary)
  - For each block:
    - The block is compressed.
    - The MAC of the block is calculated
    - The compressed block and its MAC are enciphered
- Before the handshake protocol has agreed upon a cipher suite, the record protocol uses no encipherment, and no MAC.

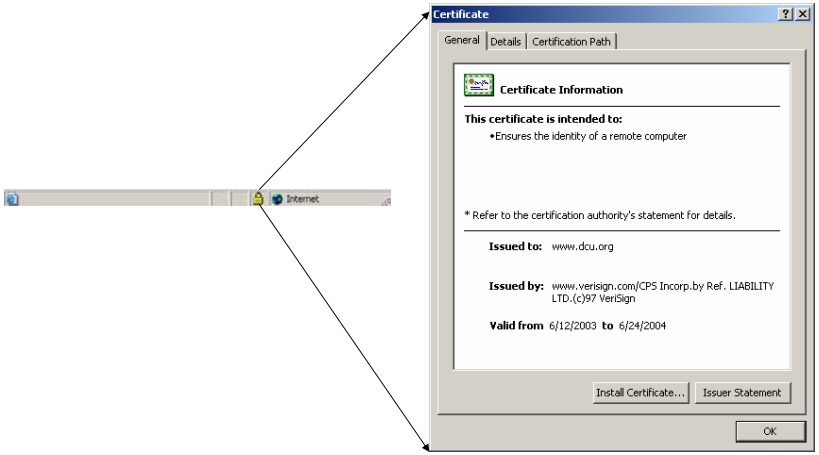


SSL Session Resumption Protocol

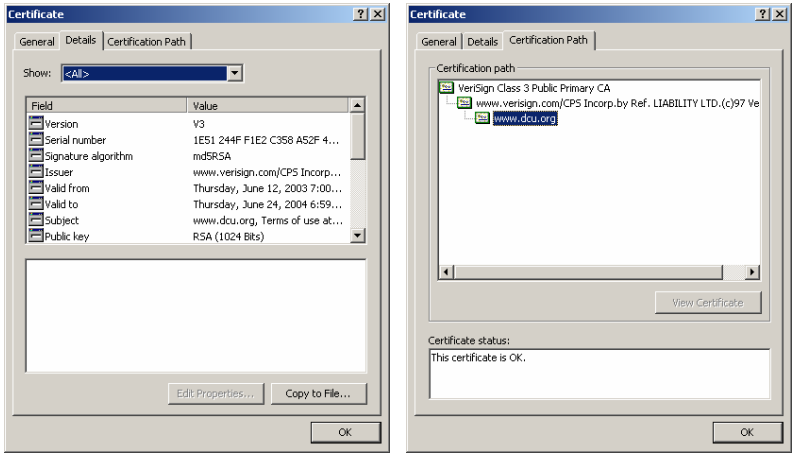
- An SSL session can be resumed, but only if the client and server ***both remember the session and its master secret.***
  - Otherwise, a full SSL handshake must be performed to establish a new session.



https://www.yourbank.com/



Certificate Details



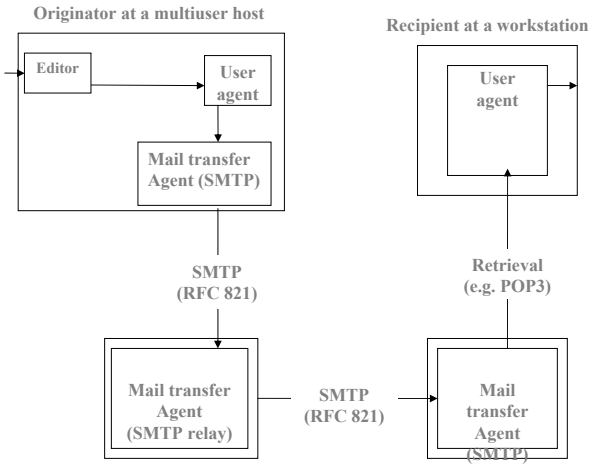
Secure Email

- There have been a number of attempts to come up with a secure email mechanism, to provide:
  - privacy
  - authentication
  - integrity
  - non-repudiation
  - proof of submission
  - proof of delivery
  - message flow confidentiality
  - anonymity
  - containment
  - audit
  - accounting
  - self-destruct
  - message sequence integrity

Secure Email

- The problem of secure email is complicated by:
  - The fact that email is an asynchronous protocol
  - The fact that email store-and-forward hosts often manipulate the email contents in different ways, depending on what platform they are running.
    - Different end-of-line conventions
    - Different maximum line sizes
    - Different character encodings
    - etc...

How Email Works



Multipurpose Internet Mail Extensions (MIME)

- SMTP is not sufficiently rich for multimedia messages
- Include additional headers in the message, which are defined in:
  - RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
  - RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

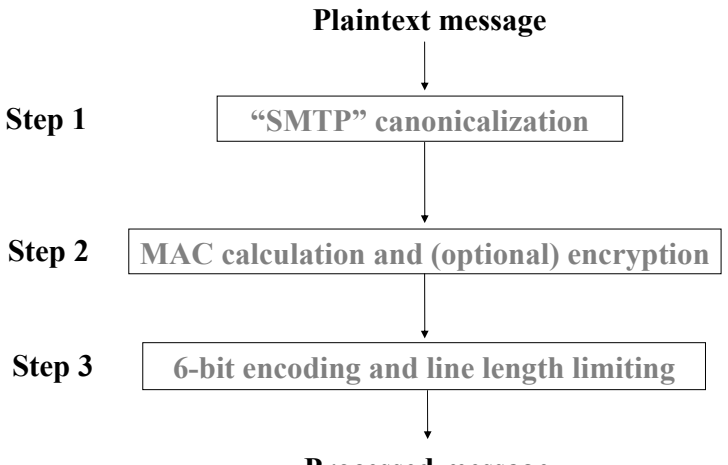
Privacy Enhanced Mail (PEM)

- Primary goal of PEM is to add security services for e-mail users in the internet community
- Began in 1985 as an activity of the Privacy and Security Research Group (PSRG)
- Defined in RFCs 1421/1422/1423/1424
- Consists of extensions to existing message processing software plus a key management infrastructure

PEM

- Compatible with RFC 822 message processing conventions
- Transparent to SMTP mail relays
- Uses symmetric cryptography to provide (optional) encryption of messages
- The RFCs strongly recommend the use of asymmetric cryptography (for digital signatures, certificates and encryption of the symmetric key) because of its ability to support vast distributed community of users

PEM



PEM

- PEM has "pretty much died out" (see textbook), so its major interest is simply that it has influenced later attempts at email security

Pretty Good Privacy (PGP)

- First released in 1991, developed by Phil Zimmerman, provoked export control and patent infringement controversy.
- PGP provides a confidentiality and authentication service
  - can be used for electronic mail and file storage applications.
- Available as plug-in for popular e-mail clients, can also be used as stand-alone software.
  - Microsoft Exchange
  - Microsoft Outlook

Pretty Good Privacy (PGP)

Actual operations of PGP consist of five services:

- Authentication
  - DSS/SHA or RSA/SHA
- Confidentiality
  - CAST or IDEA or RSA or 3DES
- Compression
  - A message may be compressed, for storage or transmission using ZIP
- E-mail compatibility
  - To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII using Radix-64
- Segmentation
  - To accommodate maximum message size limitations.

PGP Signed Message

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
This is simply the text of the message.
It has not been encrypted, simply signed

-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 6.5.3 for non-commercial use
<http://www.pgp.com>

iEYEARECAAYFAj5Ha6AACgkQ99/KQPj2cRNHsQCffKf64LwWQ
MfRIiKUfs6QrokB7twAnR5gDobzGapPgyLKQ0gLklj1WIIp=g
Xad
-----END PGP SIGNATURE-----
```

PGP Key Rings

- The problem:
  - Must maintain a database in order to supports multiple public/private keys.
- The solution：
  - Keys stored locally in a *PGP Key Ring* – essentially a database of keys.

Two rings:

- Private-key ring: stores the public/private key pairs owned by that node
- Public-key ring: stores the public keys of other users known at this node

  - Private keys are stored in encrypted form; decryption key is determined by user-entered passphrase.

S/MIME

- After the development of PEM, an industry working group led by RSA Security, Inc. started to develop another specification for conveying digitally signed and/or encrypted and digitally enveloped data in accordance to the MIME message formats.
- S/MIME (**Secure/Multipurpose Internet Mail Extension**) is a security enhancement to the MIME Internet e-mail format standard.
- S/MIME is ***not restricted to mail***; it can be used with any transport mechanism that transports MIME data, such as HTTP.
- S/MIME is ***likely to emerge as the industry standard*** for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many.