4.1: Secure Applications

Secure Shell (SSH)

One of the oldest applications: remote login
SSH is a secure protocol for this purpose:
- Log into a remote machine
- Execute commands on that machine
- Transport files from one machine to another

SSH provides:
- Strong encryption, server authentication, integrity protection
- Compression (optional)
- An own secure transport layer protocol

SSH protocol is subdivided into:
- Connection Protocol
- Authentication Protocol
- Transport Layer Protocol

SSH Transport Layer Protocol

Runs on top of TCP and provides:

Connection Setup
- Over TCP, SSH connections use port 22

Security functions
- Server authentication (no user authentication)
- Data confidentiality
- Integrity protection

Negotiation of the functions during connection setup
- Key exchange method (e.g. Diffie-Hellman)
- Public key algorithm (e.g. RSA)
- Symmetric encryption algorithm (e.g. 3DES or IDEA in CBC mode)
- Message authentication algorithm
- Hash algorithm (e.g. MD5)
- Optionally, also a compression algorithm
SSH Authentication and Connection Protocol

*SSH authentication protocol* for user authentication
- Located on top of the SSH transport protocol
- Supported SSH authentication methods:
  - Public key (required): signature created with a private key of the user. Often: private keys stored encrypted at client side and the user must supply a passphrase before the signature can be generated
  - Password (optional)
  - Host-based (optional): the name of the host on which the user is logged in is used in authentication

*SSH connection protocol* for remote work
- Located on top of the SSH transport protocol
- All channels established by a user are multiplexed onto a single, encrypted tunnel provided by the SSH transport protocol
- Provides interactive login sessions, remote execution of commands, forwarded TCP/IP connections, and forwarded X11 connections

Secure E-Mail

The sending of e-mails raises several security issues:

- **Privacy** - prevent anyone but the intended recipient from reading the message
  - Usually, the sender generates a session key \( S \) and sends it to the receiver, encrypted with the receiver's key
  - Purpose of using a session key \( S \) for an e-mail encryption instead of the receiver's key: only need to encrypt the message ones even if there is a receiver list:

  \[
  m \rightarrow K_{Bob}[S], S[m] \\
  m \rightarrow K_{Carol}[S], S[m] \\
  m \rightarrow K_{Dave}[S], S[m]
  \]

  *Message flow confidentiality* - hide from an eavesdropper the information about sender-recipient relationships
  - Use mix concepts from anonymity

- **Authentication** - guarantee the identity of the sender to the recipient
  - Implementation: sender digitally signs a message digest of the e-mail

- **Integrity** - guarantee to the recipient that the message has not been altered since it was transmitted by the sender by using encryption
  - Usually implemented together with authentication

- **Message sequence integrity** - verify that a sequence of messages arrived in the order it was transmitted, without any loss e.g. by including sequence numbers before encryption

- **Containment** - ability of the network to prevent messages with certain security levels from leaving a particular region (for instance a virtual private network)
  - Need to assign a security level marker to each sent message which would be considered by routers in the forwarding process (e.g. as part of a firewall)

- **Non-repudiation** - the recipient is able to proof that the sender really sent the message; the sender cannot deny having sent the message
  - Simplest way: digitally sign the message

- **Proof of submission** - prove (to the sender) that a message with certain content was sent to the mail delivery system on a particular date
  - Implementation by computing a message digest from the message concatenated with other useful information, as e.g. time of submission; then sign the message.
  - User can later use the signed message digest to prove message and time at once

- **Proof of delivery** - proof that the recipient received a certain message on a particular date
  - Implemented by having either the recipient or the mail delivery system sign a message digest from the message concatenated with the time of receipt (as in proof of submission)
Privacy Enhanced Mail (PEM)

PEM was developed by IETF with the aim of enhancing e-mail with:
- encryption,
- authentication
- integrity protection

The PEM specification consists of four parts:
- Message format (in addition to RFC 2822)
- Certificates (as defined in X.509), a certification authority (CA) hierarchy, and Certificate Revocation Lists (CRLs)
- Cryptographic algorithms to be used (e.g. DES-CBC, RSA, MD5, …)
- Formats of control messages

PEM has been designed to work with a mail infrastructure that only handles plain text→ Enhancement: S/MIME to integrate Multi-purpose Internet Mail Extensions (MIME) into a secure mail standard

PEM Message Format

PEM messages can consist of several parts of different security level:
- **Ordinary**: unsecured data
- **Integrity-protected unmodified data (MIC-CLEAR)**: the original message is included unmodified as part of the PEM message, but an integrity check is added
- **Integrity-protected encoded data (MIC-ONLY)**: the message first is base64-encoded, then an integrity check is added
- **Encoded encrypted integrity-protected data (ENCRYPTED)**: an integrity check on the message is computed, then message and integrity check are encrypted with a randomly selected per-message secret key. The encrypted message/integrity check and the key are each base64-encoded to pass through message transfer agents as ordinary text

Parts are included in a „normal“ mail body (like for MIME) and separated by markers

PEM Message Example

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type: 4,ENCRYPTED
Content-Domain: RFC822
DEK-Info: DES-CBC,C4E711C7F3F33772
Originator-Certificate: MIIBrDCCAVYCAUwwDQYJKoZIhvcNAQECBQAwSjELMAkGA1UEBhMCREUxOzA5BgNVBAoTMkdlc2VsbHNjaGFmdCBmdWVyIE1hdGhlbWF0aWsgdW5kIERhdGVudm ... ZnYW5nIFNjaG5laWRlcjBZMAoGBFUIAQECAgIAA0sAMEgCQQCOj39uwvnutrnDbdAyu12Bioomecmogic5m42nB1VWhrCTgaj3acA7rJRildhlyiuIwbJV0ac7QILEb97YssWdAgMBAAEwDQYJKoZIhvcNAQECBQADQQAzmvCqJqw6PvvsiDFUcbNCc/4CgIg5xko1J6a+RvyDmgBUNgjTsNk90NqxHL62xMB8h2l0Mw5pN5+AIwWTBkDw
MIC-Info: RSA-MD5,RSA,
WQRudJHT8QKNRvSpgQlrnFBpFNOP+Ek6JwFImEXnnEW87/dL/sQe+Z4+JCEXZbnJwxaXlPb3HUEIxmheSQJakFxnA9UYCo+n
Recipient-ID-Asymmetric:
MEoxCzAJBgNVBAYTAkRFMTswOQYDVQQKEzJHZXNlbGxzY2hhZnQgZnVlciBNYXRoZW1hdGlrIHVuZCBEYXRlbnZlcmFyYmVpdHVuZw==,38
Key-Info: RSA,
YIt1/6EYVV2zheVNiTyh0DA3vwX/GhMk2IDYp7MSlFhQ4PIODaXrIYUwTAzEw/oE7cpI1jFxf3hqjjaSkZrkDA==
Recipient-ID-Asymmetric: MF8xCzAJBgNVBAYTAkdCMSIwIAYDVQQKExlVbml2ZXJzaXR5IENvbGxlZ2UgTG9uZG9uMRkwFwYDVQQLExBDb21wdXRlciBTY2llbmNlMREwDwYDVQQLEhwQQVNTV09SRA==,2D6A05AC
Key-Info: RSA,
Hu9/pODM8eMVhXnkKb75YEz4ToWeFWecpOGzoysz4uEGRsgTc/5kBoTz3ovykqXN74JmRaD7VkWJWWKwki/7rA==
qu8HYmqLIHBKfZo5cH3VrVQdp626fbNsSYQnHTcCPyI4ZfCM+WWBz5ZjYqHmaV5RGIJKmNjiGbws=
-----END PRIVACY-ENHANCED MESSAGE-----
```

Message content:
Hi,
this is a PEM example letter.
Regards

When Alice wants to forward Bob a message she received from someone else (say, Fred), she encapsulates it in a new message to keep Fred’s signatures:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Recipient-ID-Asymmetric:
MEoxCzAJBgNVBAYTAkRFMTswOQYDVQQKEzJHZXNlbGxzY2hhZnQgZnVlciBNYXRoZW1hdGlrIHVuZCBEYXRlbnZlcmFyYmVpdHVuZw==,38
Key-Info: RSA,
YIt1/6EYVV2zheVNiTyh0DA3vwX/GhMk2IDYp7MSlFhQ4PIODaXrIYUwTAzEw/oE7cpI1jFxf3hqjjaSkZrkDA==
Recipient-ID-Asymmetric: MF8xCzAJBgNVBAYTAkdCMSIwIAYDVQQKExlVbml2ZXJzaXR5IENvbGxlZ2UgTG9uZG9uMRkwFwYDVQQLExBDb21wdXRlciBTY2llbmNlMREwDwYDVQQLEhwQQVNTV09SRA==,2D6A05AC
Key-Info: RSA,
Hu9/pODM8eMVhXnkKb75YEz4ToWeFWecpOGzoysz4uEGRsgTc/5kBoTz3ovykqXN74JmRaD7VkWJWWKwki/7rA==
qu8HYmqLIHBKfZo5cH3VrVQdp626fbNsSYQnHTcCPyI4ZfCM+WWBz5ZjYqHmaV5RGIJKmNjiGbws=
-----END PRIVACY-ENHANCED MESSAGE-----
```

In case of a message in ENCRYPTED mode, Alice first has to decrypt the message from Fred and re-encrypt it with Bob’s key.
**PEM Certification Authority Hierarchy**

PEM specifies a single root CA called the IPRA (Internet Policy Registration Authority)

- IPRA certifies PCAs (Policy Certification Authorities), organized as a tree:
  - PCAs
    - HACA
    - DACA
    - NACA
    - Various organisations' CAs
    - Individuals or CAs

Each PCA has to enforce a policy from one of three possible security levels:

- **High Assurance (HA)**: super-secure, i.e. implemented on special hardware, tamper resistant, etc.
- **Discretionary Assurance (DA)**: well managed at top level, but does not impose any rules on the organisations to which CA certificates are granted
- **No Assurance (NA)**: only constraint: not allowed to issue two certificates with same name

**S/MIME**

S/MIME introduces security in MIME-encoded mails:

- Similar algorithms to PEM
- Two new MIME content parts are defined to specify algorithm and contents

1. **Multipart/Signed**
   - Consists of a MIME header with all information for checking the signature, and two informative blocks:
     - The signed message
     - Digital signature

2. **Multipart/Encrypted**
   - Information for decryption (e.g. algorithms, keys)
   - Encrypted contents

**S/MIME Example**

```
Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1; boundary=boundary42
--boundary42
Content-Type: text/plain
This is a clear-signed message.--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9H04VQpfyF467GhIGfHYT6
4VQpfyF467GhIGfHYT6jH77n8HHGghyHfHfHfHfHfHj756tbB9H0TRfFvbnj
n8HHGTrfFvhnj7H77n8HHG04VQbnj7567GhIGfHYT6ghyHhHuLjFyF4
7Gh1GfHYT64VQbnj756
--boundary42--
```

- **Block 1**: signed message
- **Block 2**: digital signature

**PGP – Pretty Good Privacy**

PGP specifies encryption and integrity protection on files (not only for mail)

- Most relevant differences to PEM:
  - **Key distribution**
    - PEM assumes a rigid hierarchy of CAs
    - PGP assumes anarchy – certificates/keys are distributed informally, as well as certificate revokations
  - **Certificates**
    - PEM infrastructure decides whom the user should trust to certify people
    - PGP leaves it up to the user whom he trusts
  - **Encoding**
    - PEM expects plain text (by using encodings like base64, expanding the contents by about 33%)
    - PGP allows to specify whether a file is text or binary, additionally zip-compression is possible
### PGP vs. PEM

<table>
<thead>
<tr>
<th>Feature</th>
<th>PGP</th>
<th>PEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports encryption?</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports authentication?</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports non-repudiation?</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports compression?</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Algorithm for data encryption (bits)</td>
<td>IDEA, 3DES</td>
<td>DES, 3DES</td>
</tr>
<tr>
<td>Key length for data encryption (bits)</td>
<td>128, 112</td>
<td>56, 112</td>
</tr>
<tr>
<td>Algorithm for key management</td>
<td>RSA, DSA/ElGamal</td>
<td>RSA</td>
</tr>
<tr>
<td>Key length for key management (bits)</td>
<td>Up to 4096</td>
<td>Variable</td>
</tr>
<tr>
<td>Certificates are X.509 conformant?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Key certification</td>
<td>Ad hoc</td>
<td>IPRA/PCA/CA-hierarchy</td>
</tr>
<tr>
<td>Can eavesdroppers read messages?</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Can eavesdroppers read signatures?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Developed by</td>
<td>Small team</td>
<td>IETF</td>
</tr>
</tbody>
</table>

### PGP Message Format

<table>
<thead>
<tr>
<th>Component</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>session key</td>
<td>key ID of $K_{xcv}$, session key k, timestamp</td>
</tr>
<tr>
<td>signature</td>
<td>ID of $K_{xnd}$, leading two bytes of hash, hash, filename, timestamp</td>
</tr>
<tr>
<td>message</td>
<td>Data</td>
</tr>
</tbody>
</table>

- Encrypted with public key of the receiver $k_{xcv}$
- Encrypted with private key of the receiver $k_{xnd}$
- Compressed and encrypted with $k$

### PGP and Trust

In PGP, each user decides which keys he trusts:
- More secure if the user is careful
- Each user decides how much trust he places on different people (none, partial or complete, resp. more levels in the commercial version) – possible risk for careless users?
- More difficult to find a path of certificates (especially in large environments), e.g. if Alice wants to find out if to trust Bob:
  - If Alice has a disorganized mass of certificates, how can she find a chain that leads from a key she knows to a key of Bob?
  - There might be multiple chains, and some might lead to different keys for Bob
  - If Alice finds a chain, how much can she trust that chain?
- With each key PGP stores a quantity, indicating how much the key should be trusted as being legitimate, and how much the owner of the key should be trusted in certifying other keys

### Trust Management

Check the trust level of a key:
- Computed by the PGP system
- If at least one signature trust is ultimate, then the key legitimacy is 1 (complete)
- Otherwise, a weighted sum of the signature trust values is computed:
  - Always trusted signatures has a weight of $1/X$
  - Usually trusted signatures has a weight of $1/Y$
  - $X, Y$ are user-configurable parameters
- Example: $X=2$, $Y=4$
  - 1 ultimately trusted, or
  - 2 always trusted, or
  - 1 always trusted and 2 usually trusted, or
  - 4 usually trusted signatures are needed to obtain full legitimacy
Example – Check of Trust

X = 1, Y = 2

- untrusted / usually untrusted
- usually trusted
- always trusted
- ultimately trusted (you)

signature

legitimate