Trust Model & Key Management
- Chapter 7&5 (Stallings), Chapter 15

- PGP: An Email Security Implementation
- PKI: Public Key Infrastructure

Email Security (Application Layer)

- PGP
- PEM (Privacy Enhanced Mail)
- MIME (Multipurpose Internet Mail Extensions)

Pretty Good Privacy

- Philip R. Zimmerman is the creator of PGP, 1991.
- PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.
  - Encrypt the e-mail…
  - Send E-mail securely to a known recipient
  - Digitally sign E-mail so that the recipient(s) can be sure it is from you

Why Is PGP Popular?

- It is available free on a variety of platforms
  - available on Unix, PC, Macintosh and Amiga systems.
- Based on well known algorithms.
- Wide range of applicability.
- Not developed or controlled by governmental or standards organizations.
Operational Description

- Consist of five services:
  - Authentication
  - Confidentiality
  - Compression
  - Segmentation
  - E-mail compatibility

PGP Cryptographic Functions

PGP Operation – Authentication
1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic

PGP Operation – Confidentiality
1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128/IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message
**PGP Operation – Confidentiality & Authentication**

- Uses both services on same message
  - Create signature & attach to message
  - Encrypt both message & signature
  - Attach RSA encrypted session key

**Segmentation and Reassembly**

- Often restricted to a maximum message length of 50,000 octets.
- Longer messages must be broken up into segments.
- PGP automatically subdivides a message that is too large.
- The receiver strip of all e-mail headers and reassemble the block.

**Summary of PGP Services**

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Signature</td>
<td>DSS/SHA or RSA/SHA</td>
</tr>
<tr>
<td>Message Encryption</td>
<td>CAST or IDEA or three-key triple DES with Diffie - Hellman or RSA</td>
</tr>
<tr>
<td>Compression</td>
<td>ZIP</td>
</tr>
<tr>
<td>E-mail</td>
<td>Radix - 64 conversion</td>
</tr>
<tr>
<td>Compatibility</td>
<td>-</td>
</tr>
<tr>
<td>Segmentation</td>
<td>-</td>
</tr>
</tbody>
</table>
Format of PGP Message

PGP Keys

- **Session Keys**
  - Need a session key for each message
    - Of varying sizes: 56-bit DES, 128-bit CAST or IDEA, 168-bit Triple-DES
  - Uses random numbers
- **PGP Public & Private Keys**
  - Since many public/private keys may be in use, need to identify which is actually used to encrypt session key, to sign the message hash
    - Could send full public-key with every message
    - But this is inefficient
  - Rather use a key identifier based on key
    - Is least significant 64-bits of the key

PGP Key Rings

- Each PGP user has a pair of keyrings:
  - **Public-key ring** contains all the public-keys of other PGP users known to this user, indexed by key ID (or/and user ID)
  - **Private-key ring** contains the public/private key pair(s) for this user, indexed by key ID & encrypted by a hashed passphrase
    - The user selects a passphrase to be used for encrypting private keys
    - Using SHA-1, a 160-bit hash code is generated from the passphrase, which is then discarded
    - Encrypting private key by CAST-128 with the 128 bits of the hash code as the key. The hash is then discarded, and the encrypted private key is stored
Applying Key Rings in Message Transmission

- Signing the message
- Encrypting the message

Applying Key Rings in Message Reception

- Decrypting the message
- Authenticating the message

Signing Keys in Key Rings: Why?

- How do you know that a person’s key is really theirs?
  - In person delivery/Email
  - People publish their PGP fingerprints (hash of public key) on webpage, business card
  - Via a trusted third party, or a trusted certifying authority (referring)

- Unsigned keys are a security risk

PGP Key Management

- Rather than relying on certificate authorities
- In PGP every user is its own CA
  - can sign keys for users they know directly
- Forms a “web of trust”
  - trust keys have been signed
  - can trust keys others have signed if have a chain of signatures to them
- Key ring includes trust indicators
- Users can also revoke their keys
The Use of Trust

- PGP doesn’t require certificate, it is optional in PGP.
  - Anyone can issue a certificate to anyone else.
  - Each user decides whose certificates to trust in authenticating others.
- Trust can be relayed.
- In the public key ring:
  - Key legitimacy field (trustworthy of binding): based on signature trust(s)
  - Signature trust field: copy of Owner Trust of this signator
  - Owner trust field: user defined.

An example of the way in which signature trust and key legitimacy are related. (the node “You” refers to the user who maintains this public key rings.)

Revoking Public Keys

- The owner issues a key revocation certificate.
- Normal signature certificate with a revoke indicator.
- Corresponding private key is used to sign the certificate.
PGP Operation – Email Compatibility (skipped)

- Many electronic mail systems only allow messages made of ASCII text, not the 8-bit raw binary data that ciphertext is made of. To get around this problem, PGP supports ASCII radix-64 format for ciphertext messages
  - when using PGP will have binary data to send (encrypted message etc)
  - however email was designed only for text
- hence PGP must encode raw binary data into printable ASCII characters
- uses radix-64 algorithm
  - maps 3 bytes to 4 printable chars
  - also appends a CRC to detect transmission error
  - This format also acts as a form of transport armor, protecting it against corruption as it travels through intersystem gateways on Internet

E-mail Compatibility (skipped)

- The scheme used is radix-64 conversion (see appendix 5B of Stalling’s book).
- The use of radix-64 expands the message by 33%.

PKI: Public Key Infrastructure (Chapter 15)

- What is PKI? What is needed?
- Terminology
- PKI Trust Model
- PKIX & X.509

PKI

- The system which provides authentic public keys to applications is called a public key infrastructure
- Recall that with secret key cryptography the main problems were ones of
  - key management
  - key distribution
  - i.e., keys need to be distributed via secure channels
- In public key systems we replace these problems with those of key authentication,
  - which key belongs to whom
  - i.e., keys need to be distributed via authentic channels
- PKI provides the authentic channels used to distribute public keys.
Key Authentication

- Public Key Distribution
  - How do you trust the source of the public key?
  - Potential for masquerading

- Certification Authorities (CAs)
  - Keys are signed by the state, the corporation or somebody you trust

- Distribution of Trust
  - Your trust in a key is replaced by trust in a body, the CA

What is needed

- The set of Registration Authorities & Certification Authorities providing:
  - Registration of host/users (identity)
  - Certificate creation services (binding of identity & key)
  - Revocation services

- These make up the Public Key Infrastructure (PKI)

- Inevitably different PKIs will exist for different application domains

- Standards evolving to support PKI
  - e.g. X.509, PKIX, SPKI, ....
    - X.509 and PKIX mainly link identities to public keys, whereas SPKI links various attributes/authorizations to keys
  - SPKI = Simple Public Key Infrastructure

Digital Certificates

- It is similar to a passport
- A certificate is a person's public key and some identifying information, signed by an authority's private key
  - It can be verified using the authority's public key.
- Certificates are often chained
  - An ordered list of Certificates containing an end-user Subscriber Certificate and CA Certificates
- The value of certificates is in establishing trust. If you trust a certificate authority, then you trust all the certificates it issues.

Certificate Chaining

- A certificate essentially vouches for one public key with another public key.
- Each entity in the certificate chain with the exception of the bottom one is functioning as a certificate authority.
- The certificate contains extensions that define what specifically it can and cannot be used for
Access to Certificate

- Public key certificates will typically (although not always) be stored in repositories, and accessed as required.
- Certificate repositories may be separated from the CA which generates them.
- Certificate access protocols will typically be needed to access the certificates.

PKI: Terminology

- If Alice signs a certificate vouching for Bob's name and key, then Alice is the issuer and Bob is the subject.
- If Alice is evaluating a chain of certificates, she is the verifier, sometimes called the relying party.
- Anything that has a public key is known as a principal.
- A trust anchor is a public key that the verifier has decided through some means trusted to sign certificates. In a verifiable chain of certificates, a trust anchor will have signed the first certificate.
- PKI examples:
  - PGP
  - X509/PKIX
  - SPKI

PKI Trust Models: Monopoly Model

- Trust Models –
  - If Alice wants to send an encrypted message to Bob by his public key, defines where Alice gets her trust anchor, and what paths would create a legal chain from a trust anchor to the target name (Bob).

- Monopoly Model - the world chooses one organization, universally trusted by all companies, countries, universities, and other organizations to be a single CA for the world.
  - In reality, there is no one trusted organization.
  - It would be expensive and insecure to have a remote organization to certify your key. (How they know it was you?)
  - Once enough SW and HW were deployed so that it would be difficult for the world to switch organizations. (Could charge whatever it wanted for granting certificates.)

PKI Trust Models: Monopoly plus Registration Authorities

- As before except that single CA chooses other organizations (known as RAs) to securely check identities, and obtain and vouch for public keys.
- Based on information from RA, the CA issues a certificate because it trusts the RA.
- It is more convenient and secure to obtain certificates, since there are more places to go to get certified.
- However, all the other disadvantages of the monopoly model apply.
**PKI Trust Models: Delegated CAs**

- The trust anchor CA can issue certificates to other CAs, vouching for their keys and vouching for their trustworthiness as CAs.
- Users can then obtain certificates from one of delegated CAs instead of having to go to the trust anchor CA.
- The difference is in certificate paths.
- Security and operational properties similar to Monopoly plus RAs.

**PKI Trust Models: Oligarchy**

- This model is commonly used in browsers.
- Instead of having a single key, many trust anchors are used.
- A certificate issued by any one of them is accepted.
- Users can edit the list of trusted anchors.
- Creates competition with each other.
- Less secure than the monopoly model
  - It might be easy to trick a naive user into adding a bogus trust anchor into the set.
  - Users will not understand the concept of trust anchors.
  - There is no practical way for even a knowledgeable user to be able to examine the set of trust anchors and tell if someone has modified a set.
  - The trust anchor organizations are trusted by the product vendor, not by the user.

**PKI Trust Models: Anarchy Model**

- This is the model used by PGP.
- Each user is responsible for configuring some trust anchors.
  - For example, keys of people met who sent email containing their public keys.
  - Then anyone can sign a certificate for anyone else.
  - Eliminates the monopoly pricing, but is really unworkable on a large scale
  - The database would get unworkable large if it were deployed on Internet scale.

**PKI Trust Models: Name Constraints**

- The trustworthiness of a CA is not binary value where a CA would either be completely trusted or untrusted for everything.
- CA only is trusted for certifying some subset of users.
- MTU CA should be trusted for certifying MTU students but not for certifying, say, president@whitehouse.gov
- MTU CA should be trusted for certifying names in the namespace under mtu.edu, but not names mit.edu
  - The name by which you know someone determines whom you trust to certify that name.
- User might have multiple names, each name is a separate PKI entity.
PKI Trust Models: Top-Down Name Constraints

- This model is similar to the **monopoly model** in that everyone must be configured with a preordained, never changing **root key**, and that root CA **delegates** to other CA.

- However, the **delegated CAs** are only allowed to issue certificates for their **portions** of the namespace.

- Easy to find the path to a name - just follow the namespace from the root down.

- Other problems of the **monopoly model** - everyone has to agree upon a **root organization**, and that organization and its key would be prohibitively **expensive** to replace.

PKI Trust Models: Bottom-Up Name Constraints

- Similar to the design of Lotus Notes, it is not deployed
  - **By Digital in 1980’s**
  - **Reading assignment (pp377-380)**

- **Philosophy**: each organization can create its own PKI and then **link to others**.
  - A **hierarchical namespace** in which each node is represented by CA.
  - Not only does the parent certify the child’s name, but also the child certifies parent’s name.
  - In addition to **up-links and down-links, cross-links are allowed** (**cross-certificate**)

- It is possible to navigate the namespace from **any node**.

- An **ancestor** of a name is any **prefix** of that name, including the name itself.

- The **least common ancestor** is the node with the longest name which is a prefix of both names.

PKI Trust Models: Bottom-Up Name Constraints

- To find a path to a **target**, start at your **trust anchor** (user A/B/X wishes to find the key of user A/C/Y).
  - If it is an ancestor of the target name, **go down** from here to the name.
  - If not, look for a **cross-certificate** to an ancestor of the target.
  - If you don’t find a suitable **cross-certificate**, **go up** to the parent, look for cross-certificates to an ancestor of the target, and so forth, until you either find a suitable **cross-certificate** or get to the least common ancestor of the trust anchor and the target.
  - Once at an ancestor of the target, just follow **down-links** to the target.

- A/C/Y want to find the key of B/Y/Z/C?
Building Your Own CA

- It is best to stick to well established standards like X.509 and PKCS. X.509 is defined at
  - It contains references to PKCS standards.
- Standards aside, different organizations will implement PKI differently.
- Some will be content with the PKI services provided by large CAs as defined in the browser.
- Other organizations may need more flexibility and control than can be provided by a standard CA.
- They may need to implement their own CA.
- The main difference is who is the root and intermediate authority for your organization.

Certificate Revocation List (CRL)

- It is important to be able to revoke certificates (credit card?)
- X.509 based: list of serial numbers of unexpired revoked certificates and an issue time for the CRL
- OCRL: Online CRL
  - The PKIX standard protocol uses OCSP (online certificate status protocol): RFC 2560
  - Note that a given CRL is only valid for certificates from that CA.
- Verisign’s CRL can be downloaded from http://crl.verisign.com

X509 & PKIX

- PKIX is a profile of X.509
  - X509 defines a structure for public key certificates
  - PKIX specifies which X.509 options should be supported
- A CA assigns a unique name to each user and issues a signed certificate
  - Often name is the URL or Email address
- CAs are connected in a tree structure
  - Each CA issues a certificate for those beneath it
- X509 certificates are defined in standards using ASN.1
  - Abstract Syntax Notation Number One
- The basic structure is very simple

X509 & PKIX Certificate

- The following records are always in a certificate
  - X509 Version
  - Certificate Serial Number
  - CAs Signing Algorithm Identifier
    - Algorithm and Parameters
  - Issuer: The name of the CA
  - Validity
    - Not before and after date
  - Subject: Whose public key is being signed
  - Subjects Public Key
    - Algorithm & Parameters
    - Public Key Value
  - Issuer’s Signature on the subjects name and public key