Extended Trust Models

- the strict hierarchy may work well within a single organization
- it may be desirable to "connect" root CAs of two or more different PKI domains, a process sometimes called *PKI networking*
- this creates a "super-PKI" made up of users in different domains
- in the *mesh configuration*, all the root CAs will cross-certify each other
- n root CAs will lead to $n(n-1)$ cross-certifications
- an alternative approach is the *hub-and-spoke configuration*, in which the n root CAs each cross-certify with a new *hub CA*
- this reduces the number of cross-certifications to $2n$

Certificate Path Discovery in the Mesh Configuration

- in order to validate *Bob*'s certificate, *Alice* must be able to find a path of certificates from her trust anchor to *Bob*
- this process is called *path discovery*
- suppose that *Alice*'s trust anchor is CA_{root_i} and *Bob*'s trust anchor is CA_{root_j}, where $i \neq j$
- *Bob* sends *Alice* the certificates in the path from CA_{root_j} to *Bob*
- therefore, *Alice* needs to find the certificate of CA_{root_j} that has been signed by CA_{root_i}
- ideally, this certificate could be looked up in a directory which is maintained by CA_{root_i}
- now *Alice* has all the certificates in a path from CA_{root_i} to *Bob*

Certificate Path Discovery in the Hub and Spoke Configuration

- again, we assume that *Bob* sends *Alice* the certificates in the path from CA_{root_j} to *Bob*
- *Alice* needs to find the certificate path from CA_{root_i} to CA_{hub} to CA_{root_j}
- one "reasonable" way in which this could be done is as follows:
  - first, *Alice* obtains a copy of CA_{hub}'s certificate, that has been signed by CA_{root_i}, from CA_{root_i}
  - then *Alice* looks up the certificate of CA_{root_j} that has been signed by CA_{hub}, in a directory which is maintained by CA_{hub}
- now *Alice* has all the certificates in a path from CA_{root_i} to *Bob*

The Web Browser Model

- most web browsers (e.g., Netscape, Explorer) come pre-configured with a set of "independent" root *CA*s, all of which are treated as trust anchors
- e.g., in Netscape, this list can be found via pull-down menus in Communicator/Tools/Security Info/Certificates/Signers
- there may be on the order of 100 such root *CA*s in a typical web browser
- there is no cross-certification between these root *CA*s
- conceptually, the trust model can be thought of as a single strict hierarchy with a "virtual" root *CA* (i.e., the browser itself)

Security Issues with the Web Browser Model

- there are many root CAs of undetermined security
- the user can edit this list, but most users do not have the technical know-how to do this in a knowledgeable manner
- no mechanism exists to revoke a root CA from a web browser
- there is no legal relationship or implied contract between a user and the set of CAs configured in a web browser
- if a certificate is not signed by a known root CA, the user can choose to accept it anyway via a pop-up box in an SSL session
- verifying a certificate of a root CA (using the web browser software) does not imply that the certificate is authentic, because the certificate is self-signed

Certificate Revocation

- *revocation* means that a certificate becomes invalid before its normal expiration date
- a PKI needs a mechanism to verify that certificates have *not* been revoked
- a *certificate revocation list* (CRL) is a list of the serial numbers of all certificates that are revoked but not expired
- the CRL is signed by a *CA*
- CRLs must be updated periodically
- *delta CRLs* contain the changes (new revocations) that have occurred since the most recent previously issued (delta) CRL
- an alternative approach to CRLs is an *online certificate status protocol* (OCSP)

PGP Certificates

- in PGP, every user is his or her own CA
- a PGP certificate contains an email address (ID), a public key (PK), and one or more signatures on this (ID, PK) pair
- e.g., Alice might create a self-signed certficate

$$\mathsf{Cert}(Alice) = (\text{data}, \text{signatures})$$

that contains the following information:

$$\begin{aligned} \text{data} &= (ID = \texttt{alice@uwaterloo.ca}, PK = 12345) \\ \text{signatures} &= sig_{Alice}(\text{data}) \end{aligned}$$

- later, Bob might attach his signature to Alice's certificate, so

$$\text{signatures} = (sig_{Alice}(\text{data}), sig_{Bob}(\text{data}))$$

PGP Keyrings

- Alice keeps a collection of certificates, which she has obtained from various sources, in a data structure called a *keyring*
- associated with each certificate in the keyring is an *owner trust field* (OTF) and a *key legitimacy field*
- the *key legitimacy field* (KLF) specifies whether a particular public key will regarded as valid by Alice
- the KLF has the value *valid*, *marginally valid* or *invalid*
- the value of the owner trust field is determined by Alice, according to her own judgement; it indicates the extent to which Alice trusts the owner of that key to sign other keys
- the OTF has the value *implicitly trusted*, *completely trusted*, *partially trusted* or *untrusted*

PGP Trust Model

- Alice's own OTF has the value "implicitly trusted"
- if Alice sets Bob's OTF to the value "completely trusted" in her keyring, she believes that
  - Bob's public key is valid
  - Bob is an individual who would be careful not to sign any invalid (ID, PK) pairs
- the PGP software computes each KLF using the following rule:
- a key is *valid* if it is signed by
  - at least one key for which the OTF has the value "trusted" (implicitly or completely), or
  - at least two keys in for which the OTF has the value "partially trusted"

Example: Alice's Keyring

owner	OTF	signatures	KLF
Alice	implicit	Alice, Bob, Doris, Eve	
Bob	complete	Alice, Eve, Ginger	
Charlie	partial	Bob, Charlie, Eve	
Doris	untrusted	Doris, Eve, Janet	
Eve	untrusted	Doris, Eve, Fred	
Fred	partial	Bob, Eve, Fred	
Ginger	partial	Charlie, Eve, Fred	
Harry	untrusted	Eve, Harry, Irene	
Irene	untrusted	Eve, Fred, Harry, Irene	
Janet	complete	Alice, Bob, Eve, Janet	

Example: The KLFs in Alice's Keyring

owner	OTF	signatures	KLF
Alice	implicit	Alice, Bob, Doris, Eve	valid
Bob	complete	Alice, Eve, Ginger	valid
Charlie	partial	Bob, Charlie, Eve	valid
Doris	untrusted	Doris, Eve, Janet	valid
Eve	untrusted	Doris, Eve, Fred	marginally valid
Fred	partial	Bob, Eve, Fred	valid
Ginger	partial	Charlie, Eve, Fred	valid
Harry	untrusted	Eve, Harry, Irene	invalid
Irene	untrusted	Eve, Fred, Harry, Irene	marginally valid
Janet	complete	Alice, Bob, Eve, Janet	valid

Problems with a PGP-type PKI

- PGP is not an infrastructure:
  - users have to set trust levels themselves
  - revocation must be done by the certificate owners themselves
  - this requires technologically informed users
- there is no mechanism to revoke bogus certificates
- PGP is not easily scalable to large networks or user communities
- it is difficult to enforce uniform policies or standards, so a PGP-type PKI is not suitable for most organizations

The Future of PKI?

There are many potential problems associated with practical large-scale deployments of PKIs:

- lack of recognized authoritative bodies (commercial, government, academic)
- lack of standardization (entity naming, certificate formats, cryptographic algorithms, revocation, path discovery, directories, etc.) makes interoperability difficult to achieve
- differing PKI needs in different environments — no one-size-fits-all solution may be possible
- lack of PKI-compatible applications (chicken-and-egg problem)
- focus on reasonable expectations — recognize that the primary purpose of PKIs is key management, and try to solve that problem first