Bridge Certification Architecture

A Brief Overview

by

Tim Sigmon

May, 2000
Outline

- PKI refresher (e.g., public key cryptography, digital signatures, certificates)
- Bridge certification architecture motivation
  - trust models
  - cross-certification
- BCA pilot project
  - BCA itself
  - application issues and demo
PKI refresher

- infrastructure that enables and supports a range of security services
  - confidentiality, authentication, integrity, non-repudiation

- Symmetric key cryptography
  - same key used to encrypt and decrypt
  - problem: key sharing

- Public key cryptography
  - public-private keys that are mathematically linked
  - ~500 times slower than symmetric key cryptography
  - used for symmetric key exchange and digital signatures
Digital Signature example

Diagram:
- Original data
- Hashing algorithm
  - One-way hash
  - Private key encryption
  - Digital signature
- Network
- Hashing algorithm
  - Digital signature
  - Public key decryption
  - One-way hash
- Identical hashes validate data integrity
Signed Email example

- (show example of sending/receiving digitally signed email using Netscape Messenger)
- (uses S/MIME)
Problem: relying party needs to verify a digital signature

- To do this, needs to have an assured copy of the signer’s public key
  - signer’s identity must be assured
  - integrity of public key must be assured

- Potential options for obtaining public keys
  - signer personally gives their public key to relying party
  - relying party obtains the desired public key by other “out of band” means that they trust, e.g., transitive relationships, signing parties, etc.

- But, what about strangers? what about integrity of the public key?
Public Key (or Digital) Certificates

- **Purpose**: validate both the integrity of a public key and the identity of the owner
- **How**: bind identifying attributes to a public key (and therefore to the keyholder of the corresponding private key)
- **Binding is done** (i.e., digitally signed) by a trusted third party (Certification Authority)
- **It is this third party's credibility** that provides "trust"
X.509 v3 Certificates

- Subject’s/owner’s identifying info (e.g., name)
- Subject’s/owner’s public key
- Validity dates (not before, not after)
- Serial number
- Level of assurance
- Issuer’s (CA’s) name and signature
- Extensions
Distribution of Certificates

- since certs carry public info and are integrity-protected, they can be distributed and shared by any and all means, e.g.,
  - distribute via floppies or other removable media
  - publish on web sites
  - distribute via email (e.g., S/MIME)
  - directory lookups (e.g., LDAP, X.500)

- distribution via directories is the ultimate solution

- however, many important applications and uses of digital signatures can be implemented without the implementation or use of sophisticated directories
Trust Domain

- Relying party needs to determine whether to trust the certificate before using the public key it contains to verify the original signature
  - validity dates, revocation lists, policy constraints
  - verify the cert’s signature, trust the CA?
- Trust domain is defined by the root (or self-signed) certificates that the relying party knows and trusts (for reasons outside of PKI)
- **Very Important:** Root certificates are not integrity-protected since they are self-signed
Simple Hierarchical Trust

Relying Party
  e.g., email or web form application

1) path construction
   CA₁ → EEₐ
   CA₁ → CA₁

2) path validation

3) signature verification

Trust Domain

CA₁ → CA₁

Signed doc and CA₁ → EEₐ
Trust Domain Expansion

- **Hierarchical CA’s**

```
CA_1 → CA_1^{(Governor)}
   ↘                      ↗
   CA_1 → CA_2^{(SoED)}   CA_1 → CA_3^{(DGIF)}
     ↘                                 ↘
     CA_2 → CA_4                CA_2 → CA_5^{(UVa)}
                   ↘                                      ↘
                   CA_5 → CA_6^{(Darden)}     CA_5 → EE_1^{(tms)}
                              ↘                                                  ↘
                              EE_1                                        EE_1
```

Note: relying party follows issuer chain to verify cert of EE_2

- CA_5 → EE_1
- CA_2 → CA_5
- CA_1 → CA_2
- CA_1 → CA_1 ← trusted
Trust Domain Expansion

- Hierarchical CA’s

\[
\begin{array}{c}
CA_1 \rightarrow CA_1 \\
CA_1 \rightarrow CA_2 & CA_1 \rightarrow CA_3 \\
CA_2 \rightarrow CA_4 & CA_2 \rightarrow CA_5 \\
| & |
CA_5 \rightarrow CA_6 & CA_5 \rightarrow EE_1
\end{array}
\]

Note: if CA_1’s private key is compromised, the entire hierarchy collapses

- Multiple root certificates
  – disservice of Microsoft and Netscape

\[
\begin{array}{c}
CA_A \rightarrow CA_A \\
| \downarrow \ldots \downarrow | & | \downarrow \ldots \downarrow |
CA_B \rightarrow CA_B & CA_C \rightarrow CA_C \\
| \downarrow \ldots \downarrow | & | \downarrow \ldots \downarrow |
\ldots & \ldots \\
\ldots & \ldots \\
CA_Z \rightarrow CA_Z
\end{array}
\]
Trust Domain Expansion (cont’d)

- Cross certification
  - two CA’s issue certificates to each other (a cross-certificate pair), i.e., sign each other’s public keys

- $N^2$ problem if $N$ CA’s want to cross-certify with each other
Bridge Certification Architecture

- addresses the $N^2$ problem by providing a central cross-certification hub for a group of CA’s who wish to interoperate
- each CA does one cross-certification with the bridge CA

- Certificate path processing (construction & validation)

CA$_5$→EE$_2$
CA$_{bridge}$→CA$_5$
CA$_1$→CA$_{bridge}$
CA$_1$→CA$_1$ ←trusted
BCA Pilot Implementation

- OpenSSL (www.openssl.org) and OpenCA (www.openca.org) open source software running on Red Hat Linux
- Bridge is booted only to create cross certificates; remains turned off in secure location most of the time
- Cross certificates stored in LDAP directories (using crossCertificatePair attribute) and/or stored with relying parties
Example application

- Yuji Shinozaki is developing an example application (digitally signed web forms) to illustrate use of BCA
- chose server-based app instead of email
  - current relying party software can only verify hierarchical trust relationships
- as part of federal bridge project, Cygnacom developed Certificate Path Library (CPL) that handles general trust relationships
- use CPL to help verify digitally-signed web forms in a cross-certification environment
Example application