Bridge Certification Architecture

A Brief Demo

by

Tim Sigmon and Yuji Shinozaki

June, 2000
Trust Domain

- Trust domain is defined by the root (or self-signed) certificate(s) 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

- BCA provides for expansion of trust domain
  - without need for potentially expensive processes to add additional root certs to all relying parties
  - solves order $N^2$ cross-certification problem
BCA Pilot Implementation

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

- Yuji Shinozaki has developed 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 follow issuer chains (e.g., hierarchical trust relationships)
  - we cache all needed certs (including cross certs) at application (server); no need for directory
  - Yuji has implemented more general path construction as part of the server-based app
- Note: for federal bridge project, Cygnacom developed Certificate Path Library (CPL) that handles very general trust relationships
Digital Signature Demo in a bridge cross-certification environment

Relying Party
e.g., web form application

1) path construction
- CA₁ → Tim
- CA₁ → CA₁
- CA₂ → CA₂

Trust Domain
- CA₁ → CA₁

Signed form data and CA₁ → Tim

CA₁ → BCA

CA₂ → BCA
Digital Signature Demo in a bridge cross-certification environment

Relying Party
e.g., web form application

1) path construction
   CA₁ → Tim
   BCA → CA₁
   CA₂ → BCA
   CA₂ → CA₂

2) path validation

3) signature verification

Trust Domain
CA₂ → CA₂
BCA Demo

Simple Hierarchical Trust

Relying Party

1) path construction
   \( CA_1 \rightarrow EE_A \)
   \( CA_1 \rightarrow CA_1 \)

2) path validation

3) signature verification

Trust Domain

\( CA_1 \rightarrow CA_1 \)

Signed doc and \( CA_1 \rightarrow EE_A \)

\( CA_1 \rightarrow EE_A \)

Trust Domain

\( CA_1 \rightarrow CA_1 \)

Simple Hierarchical Trust

Relying Party

1) path construction
   \( CA_1 \rightarrow EE_A \)
   \( CA_1 \rightarrow CA_1 \)

2) path validation

3) signature verification

Trust Domain

\( CA_1 \rightarrow CA_1 \)

Signed doc and \( CA_1 \rightarrow EE_A \)

Trust Domain

\( CA_1 \rightarrow CA_1 \)

Simple Hierarchical Trust

Relying Party

1) path construction
   \( CA_1 \rightarrow EE_A \)
   \( CA_1 \rightarrow CA_1 \)

2) path validation

3) signature verification

Trust Domain

\( CA_1 \rightarrow CA_1 \)

Signed doc and \( CA_1 \rightarrow EE_A \)

Trust Domain

\( CA_1 \rightarrow CA_1 \)
Trust Domain Expansion

- Hierarchical CA’s

Note: relying party follows issuer chain to verify cert of EE₁

CA₅→EE₁
CA₂→CA₅
CA₁→CA₂
CA₁→CA₁ ←trusted
Trust Domain Expansion

- **Hierarchical CA’s**

  ![Diagram of Hierarchical CA's]

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

- **Multiple root certificates**
  - disservice of Microsoft and Netscape

  ![Diagram of Multiple root certificates]
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)

  \[ \text{CA}_5 \rightarrow \text{EE}_2 \]
  \[ \text{CA}_{\text{bridge}} \rightarrow \text{CA}_5 \]
  \[ \text{CA}_1 \rightarrow \text{CA}_{\text{bridge}} \]
  \[ \text{CA}_1 \rightarrow \text{CA}_1 \leftrightarrow \text{trusted} \]