Bootstrapping Trust

Frank Pfenning
based on lecture notes by Matt Fredrikson

15-316 Software Foundations of Security & Privacy
November 14, 2024

Decentralized Security Policies

- Principals can affirm statements
- Both general policy and specific facts or challenges

  $\mathit{admin}\ \textbf{says}\ \forall A. \forall R. \mathrm{owns}(A, R) \rightarrow \mathrm{mayOpen}(A, R)$
  $\mathit{fp}\ \textbf{says}\ \mathrm{studentOf}(\mathit{derek}, \mathit{fp})$

  $\Gamma \vdash \mathit{admin}\ \textbf{says}\ \mathrm{mayOpen}(\mathit{derek}, \mathit{ghc6017})$
- Construct and verify proofs, using policy rules and facts
- How do we know that *admin* and *fp* really made the affirmations in the policy?

Eduroam Wifi Networks

- Principals *er* and *cmu*, *pitt*, ...
- Delegation

$$\begin{array}{l} er\ \mathbf{says}\ \forall x.\,(cmu\ \mathbf{says}\ \mathrm{isStudent}(x)) \to \mathrm{isStudent}(x) \\ er\ \mathbf{says}\ \forall x.\,(pitt\ \mathbf{says}\ \mathrm{isStudent}(x)) \to \mathrm{isStudent}(x) \end{array}$$

- Access

$$er\ \mathbf{says}\ \forall x.\,\mathrm{isStudent}(x) \to \mathrm{canAccess}(x)$$

- Challenge

$$\Gamma \vdash er\ \mathbf{says}\ \mathrm{canAccess}(derek)$$

Let's Build a Proof

$$
\begin{array}{lcl}
c_1 & : & \mathit{er}\ \textbf{says}\ \forall x.\,(\mathit{cmu}\ \textbf{says}\ \mathsf{isStudent}(x)) \to \mathsf{isStudent}(x) \\
c_2 & : & \mathit{er}\ \textbf{says}\ \forall x.\,\mathsf{isStudent}(x) \to \mathsf{canAccess}(x) \\
c_3 & : & \mathit{cmu}\ \textbf{says}\ \mathsf{isStudent}(\mathit{derek}) \\
\vdash & & \\
?M & : & \mathit{er}\ \textbf{says}\ \mathsf{canAccess}(\mathit{derek})
\end{array}
$$

$$
\begin{array}{l}
?M = \{ \\
\qquad \textbf{let}\ \{x_1\}_{er} = c_1\ \textbf{in} \\
\qquad \textbf{let}\ \{x_2\}_{er} = c_2\ \textbf{in} \\
\qquad \textbf{let}\ x_4 = x_1\ \mathit{derek}\ c_3\ \textbf{in} \\
\qquad x_2\ \mathit{derek}\ x_4 \\
\quad \}_{er}
\end{array}
$$

Digital Signature

- Digital signature scheme
  1. Creating a public/secret pair of keys $\langle pk/A, sk/A \rangle$
  2. Algorithm for signing a message, $\mathrm{sign}_{sk/A}(m)$
  3. Algorithm for verifying a message, $\mathrm{verify}_{pk/A}(s, m)$
- Unforgeability (with high probability)

$$\mathrm{verify}_{pk/A}(s, m) = \top \quad \text{iff} \quad s = \mathrm{sign}_{sk/A}(m)$$

- $\mathrm{sign}_{sk/A}(m)$ should reveal no information about sk/A.

Certificates

- Certificate asserting the association between a principal and its public key.

$$\text{cert}_{er \to cmu}(pk) \equiv \text{sign}_{sk/er}(\text{isKey}(cmu, pk))$$

- Message sequence from CMU to Eduroam access point

$$pk/cmu, \text{cert}_{er \to cmu}(pk/cmu), \text{sign}_{sk/cmu}(\text{isStudent}(derek))$$

- Can be sent over an insecure channel and still be trustworthy

Certificate Authorities

- Certificate Authorities (CAs) issue certificates associating principals with public keys
- Signed by their own public/secret key pair
- Need to trust CA's public key

Formalizing Certificates and Trust

- Formalize in authorization logic
- Axiom

$$\forall x.\, \forall pk.\, \mathsf{isKey}(x, pk) \to \mathsf{sign}_{sk/x}(P) \to x \textbf{ says } P$$

- Turn into a proof rule

$$\frac{\Gamma \vdash \mathsf{isKey}(A, pk/A) \quad \Gamma \vdash \mathsf{sign}_{sk/A}(P)}{\Gamma \vdash A \textbf{ says } P}\ \textbf{says/sign}$$

- We prove $\mathsf{sign}_{pk}(P)$ by signature verification

$$\frac{\mathsf{verify}_{pk/A}(\mathsf{sign}_{sk/A}(P), P) = \top}{\Gamma \vdash \mathsf{sign}_{sk/A}(P)}\ \textbf{verify}$$

Certificate Authorities

- Certificate authorities can certify

$$\forall x.\, \forall y.\, \forall pk.\, \mathrm{isCA}(x) \to x \textbf{ says } \mathrm{isKey}(y, pk) \to \mathrm{isKey}(y, pk)$$

- We can also turn this into a rule

$$\frac{\Gamma \vdash \mathrm{isCA}(A) \qquad \Gamma \vdash A \textbf{ says } \mathrm{isKey}(B, pk)}{\Gamma \vdash \mathrm{isKey}(B, pk)}\ \textbf{cert}$$

Example Revisited

- Message sequence

  $$pk/cmu, \mathrm{cert}_{er \to cmu}(pk/cmu), \mathrm{sign}_{sk/cmu}(\mathrm{isStudent}(derek))$$

- Can now prove

  $\forall x.\, \forall pk.\, \mathrm{isKey}(x, pk) \to \mathrm{sign}_{sk/x}(P) \to x\ \textbf{says}\ P \quad$ (for all P)
  $\forall x.\, \forall y.\, \forall pk.\, \mathrm{isCA}(x) \to x\ \textbf{says}\ \mathrm{isKey}(y, pk) \to \mathrm{isKey}(y, pk)$

  $\mathrm{sign}_{sk/er}(\mathrm{isKey}(cmu, pk/cmu))$
  $\mathrm{sign}_{sk/cmu}(\mathrm{isStudent}(derek))$
  $\mathrm{isCA}(er)$
  $\mathrm{isKey}(er, pk/er)$
  $\vdash$
  $\mathrm{canAccess}(derek)$

Failure Modes

- CMU's private key is compromised
- CA's private key is compromised
- Man-in-the-middle attack against encryption, C compromised

$$A \xleftarrow{pk/B,\mathrm{sign}_{sk/C}(\mathrm{isKey}(B,pk/B))} B$$

$$A \xrightarrow[\mathrm{enc}_{pk/B}(\text{“Hello!”})]{} B$$

$$A \xleftarrow{pk/B^*,\mathrm{sign}_{sk/C}(\mathrm{isKey}(B,pk/B^*))} M \xleftarrow{pk/B,\mathrm{sign}_{sk/C}(\mathrm{isKey}(B,pk/B))} B$$

$$A \xrightarrow[\mathrm{enc}_{pk/B^*}(\text{“Hello!”})]{} M \xrightarrow[\mathrm{enc}_{pk/B}(\text{“Goodbye!”})]{} B$$

Public Key Infrastructure (PKI)

- How are CAs assigned and managed?
- Centralized CA
  - Trusted by everyone
  - Obtaining key through physical contact
  - Bundle public keys for widely-known CAs with common software like browsers and operating systems
- Attacks
  - Downloading corrupted version of software (but: checksums)
  - Untrustworthy CAs (but: obtain multiple certificates)

Delegated Trust and Hierarchical CAs

- Root-issued CAs delegate to second-level CAs, etc.

$$\begin{aligned} &\mathit{CA}\ \textbf{says}\ (\forall x.\, \forall y.\, \forall pk.\ \mathit{CA}\ \textbf{says}\ \mathrm{trusts}(x) \\ &\qquad\qquad \to x\ \textbf{says}\ \mathrm{isKey}(y, pk) \to \mathrm{isKey}(y, pk)) \end{aligned}$$

- Certificate chains
- Widely deployed in web browsers and operating systems
- 2010 survey: 651 distinct CAs

Web of Trust

- Flat structure, deployed in *Pretty Good Privacy* (PGP) project
- Builds redundancy over time
- Failure distributed rather than centralized
- Requires user knowledge and initiative

Dealing with Certificate Compromise

- Expiration
  - Window of vulnerability between compromise and expiration
  - Certificate lifetimes can not be too short in practice
- Certificate revocation lists (CRLs)
  - Each certificate has unique serial number
  - CA distributes list of compromised serial numbers
  - Verifiers must cross-reference the list when checking a certificate
  - CRLs can grow to be quite large
- Online Certificate Status Protocol (OCSP)
  - Pull information about certificate status dynamically
  - Traffic and latency issues
  - Privacy issues (for CA and network snooping)
  - Recently deprecated
- Certificate pinning
  - Specify ("pin") which CAs can certify you

Summary

- Digital signature schemes at the root of authorization
- Requires association between principals and public keys
- Provided by Certification Authorities (CAs), including delegation
- Formalized in authorization logic
- Revocation lists (CRLs)