The TAOS Authentication System: Reasoning Formally About Security

Brad Karp
UCL Computer Science

CS GZ03 / M030
7th December, 2009

Motivation: Building Correct Authentication Systems

- We've studied cryptographic primitives
- We've studied certificates, and how they're used in SSL
  - Trusted third party, CA, attests to binding between public key and principal's name
  - One party can authenticate other using certificate
- Certificates are more general tool, but can be hard to reason about
- How can we reason formally about whether collection of certificates truly authenticates some principal to complete some operation on some object?

Motivation: Flexible Authentication Systems

- Suppose want to authenticate user on client workstation to file server
  - User is principal
  - User authorized on file server to perform certain operations on certain file objects
- Simple model:
  - Use public-key cryptography
  - Install user's public key on file server
  - User holds private key on client workstation while logged in
  - User signs each RPC sent to file server using his private key

Motivation: Drawbacks of Simple Authentication Model

- Very slow (TAOS took 250 ms per RSA sig)
- Rigid:
  - What if I ssh into second machine?
  - 2nd box must sign RPCs to file server, too
  - Does it send messages back to 1st box for signing? How would user know they're authentic?
  - What if user goes home, leaves simulation running for hours?

Motivation: SSL Rigid, Too

- Does SSL work here?
- Assume both sides (client and server) authenticate by presenting certificates
- Fast: symmetric-key ciphers for session data
- But workstation must hold private key for every connection
- What if I ssh into second machine?
  - Want it to be able to use file server, too
  - Would have to give second machine my private key!

Outline of TAOS Authentication (1)

- Give each machine an identity: public/ private key pair
- User bkarp logs into machine X, signs certificate:
  - "bkarp **says** X **speaks for** bkarp."
  - Reflects reality; X executes bkarp's programs
  - In paper, **speaks for** written as $\Rightarrow$
  - Y **says** X just means "Y signs statement X with K_Y" (note paper refers to public key when signing!)

Outline of TAOS Authentication (2)

- Now machine X can:
  - Open SSL-like secure channel from self to server; file server knows it's talking to X
  - Present "bkarp **says** X **speaks for** bkarp" to file server; file server knows X can speak for user
  - Send RPCs generated by bkarp's programs to file servers
  - All without machine X holding bkarp's private key!

Authorizing 2nd Machine with TAOS

- Consider ssh by bkarp to 2nd machine
- Want Y to talk to file server for bkarp
- ssh on X signs "X says Y can speak for bkarp"
- Gives this certificate to Y when bkarp logs into Y
- Now Y presents proof outline to file server:
  - I'm Y
  - X says Y can speak for bkarp
  - bkarp says X can speak for bkarp
- File server can check signatures and verify that RPCs authorized!

Why Can't SSL Authorize 2^{nd} Machine?

- SSL for exactly two principals, tied to channels
- If X says something to Y, Y can't prove anything to Z
- In fact, Y can't verify anything after X closes its connection to Y
- SSL too rigid to support distributed systems with > 2 parties

TAOS's Central Strengths

- Certificates are true independent of channels
- …so can be stored, passed to other parties
- …and used to prove transitive trust relationships

Axioms in the TAOS Logic (2.1 in paper)

- speaks for:
  - if (A **speaks for** B) and (A **says** S)
    then (B **says** S)
- handoff axiom:
  - if A **says** (B **speaks for** A)
    then (B **speaks for** A)
- delegation axiom:
  - if A **says** (B | A) **speaks for** (B for A))
    then (B | A) **speaks for** (B for A))

Applying Handoff and Delegation

- Handoff: given
  A **says** (B **speaks for** A) and B **says** S
      then A **says** S
- Delegation: given
  A **says** (B | A) **speaks for** (B **for** A)) and
  B **says** A **says** S
      then (B **for** A) **says** S

Applying Handoff and Delegation

- Handoff: given
  A **says** (B **speaks for** A) and B **says** S
  then A **says** S
- Delegation: given
  A **says** (B | A) **speaks for** (B **for** A)) and
  B **says** A **says** S
  then (B **for** A) **says** S

Delegation more specific than handoff; records both principals, the trustor and trustee
Better for auditing…

Using Logic to Reason About Authentication

- Consider example in Section 2.2 of TAOS paper:
  - User Bob logs into workstation WS
  - Logic used to authenticate requests from Bob’s login session to a remote file server FS
- What principals are involved?
  - Workstation firmware, OS, Bob, Channel
- Keep track of who knows:
  - Private keys
  - Signed certificates
  - Channel keys

State Before Bob Logs In

- Workstation firmware knows K_{vax4}
- User knows K_{bob}'s private "half"
- File server has K_{bob}'s public "half" in an ACL

Workstation Boot Time: Generating K_{ws}

- At boot, workstation firmware generates fresh public/private key, K_{ws}
- **Why not just use K_{vax4} directly?**
  - Don't want it to be stolen
  - Don't want statements to survive reboot (i.e., certificates generated for login sessions)
- Firmware signs:
  "K_{vax4} **says** (K_{ws} **speaks for** K_{vax4})"
- K_{vax4} never used again (until reboot)
- Why bother preserving K_{vax4}'s identity?
  - Why not just use K_{ws} as workstation's true identity?
  - Want workstation's identity to survive reboots

Boot Time: Generating K_{ws} (2)

- Why bother with roles ("K_{vax4} **as** OS")?
  - User might not trust some versions of OS, or some OS
  - Want to allow OS type/version to be visible in ACLs
  - Assuming a role amounts to reducing access rights
- Now vax4's authentication agent knows:

  K_{ws} (but forgets K_{vax4})

  (K_{vax4} **as** OS) **says** (K_{ws} **speaks for** (K_{vax4} **as** OS))
- Why does vax4 need an identity at all?
  - So Bob can delegate to it!

Login: Delegation of Authority to Workstation by User

- Want ws to be able to act for Bob
- Bob signs with his private key, K_{bob}:
  K_{bob} **says** ((K_{ws} | K_{bob}) **speaks for** (K_{ws} **for** K_{bob}))
- Private half of K_{bob} not used again until next login!
- Why not "K_{bob} **says** (K_{ws} **speaks for** K_{bob})"?
  - If K_{ws} signs something, on whose behalf was it?
  - So statements by K_{ws} ambiguous, and perhaps usable out of context

Delegation at Login (2)

- What does (A | B) mean?
  - That A is doing the signing
  - That A is claiming (no proof yet) that A is speaking for B
  - Really means that A says in its signed statement that it's speaking for B
- What does (A **for** B) mean?
  - Logical conclusion that A allowed to speak for B
  - i.e., (A | B) plus delegation, like one on previous slide (see delegation axiom on p. 4 of paper)
  - By default, interpreted as B for purposes of ACLs
  - But for those who care, preserves who actually signed (A)

Delegation at Login (3)

- After delegation by Bob, vax4's authentication agent knows:

  K_{ws} private half

  (K_{vax4} **as** OS) **says** (K_{ws} **speaks for** (K_{vax4} **as** OS))

  K_{bob} **says** ((K_{ws} | K_{bob}) **speaks for** (K_{ws} **for** K_{bob}))

TAOS Channels

- TAOS uses symmetric-key ciphers to encrypt channels between hosts
- Channels named by their symmetric key
  - Name has global meaning
- C_{bob} doesn't imply anything about Bob
  - Only a mnemonic used by authors to indicate intent that C_{bob} carries messages from Bob
  - System must establish proof that this is case
- File server knows:
  - C_{bob} **says** RQ (where RQ a file server request)
  - i.e., "received request from someone who knows key C_{bob}"
- But who knows key C_{bob}?
  - K_{ws}?
  - K_{ws} on behalf of Bob?
  - K_{ws} on behalf of someone else?

Proving Authenticity: Channel Certificates

- ws signs channel certificate when channel between ws and file server first created:
  (K_{ws} | K_{bob}) **says** (C_{bob} **speaks for** (K_{ws} **for** K_{bob}))
- Goal: link RQ encrypted with C_{bob} to Bob
- Why not just have K_{bob} sign:
  - "C_{bob} speaks for K_{bob}"
  - This is what SSL client-side certificates do.
  - But in TAOS, authentication agent doesn't hold K_{bob}'s private half—and that's a good thing…

Channel Certificates (2):

- Why not have K_{ws} sign:
  - "C_{bob} **speaks for** K_{ws}"
  - Along with pre-signed "K_{ws} **speaks for** K_{bob}"
  - C_{bob} doesn't speak for K_{ws} in general! Only K_{bob}.
- Channel certificate is in fact nicely restricted:
  - States what we mean, and no more
  - vax4 **says** C_{bob} **speaks for** (vax4 **speaking for** Bob)
- But vax4 could sign this statement without Bob's agreement!
- So file server needs further evidence:
  - Is vax4 allowed to speak for Bob?

Using Logic to Prove Authenticity

- Suppose ws sends all certificates to file server:

  (K_{vax4} **as** OS) **says** (K_{ws} **speaks for** (K_{vax4} **as** OS))

  K_{bob} **says** ((K_{ws} | K_{bob}) **speaks for** (K_{ws} **for** K_{bob}))

  (K_{ws} | K_{bob}) **says** (C_{bob} **speaks for** (K_{ws} **for** K_{bob}))

- Now file server can reason about meaning of C_{bob} **says** RQ

Using Logic to Prove Authenticity (2)

- File server can take

  K_{bob} **says** ((K_{ws} | K_{bob}) **speaks for** (K_{ws} **for** K_{bob}))

- and deduce, using delegation axiom:

  (K_{ws} | K_{bob}) **speaks for** (K_{ws} **for** K_{bob})

- Informally, delegation axiom just means:
  - If Bob signs certificate allowing K_{ws} to speak for Bob, then K_{ws} is allowed to speak for Bob
- Really, delegation certificate means:
  - If K_{ws} says it's speaking for Bob, believe it.
  - This is different than "K_{ws} speaks for K_{bob}"!

Using Logic to Prove Authenticity (3)

- Now, combine:

  (K_{ws} | K_{bob}) **speaks for** (K_{ws} **for** K_{bob})

  (K_{ws} | K_{bob}) **says** (C_{bob} **speaks for** (K_{ws} **for** K_{bob}))

- And thus derive:

  (K_{ws} **for** K_{bob}) **says** (C_{bob} **speaks for** (K_{ws} **for** K_{bob}))

- In other words:
  - K_{ws} really does speak for K_{bob}; it's not just claiming to do so
- So we can conclude that C_{bob} speaks for K_{ws} speaking for K_{bob}
- And thus:

  (K_{ws} for K_{bob}) says RQ

TAOS: Summary

- Certificates allow flexible authentication
  - Can survive longer than a channel
  - Allow delegation of authority
  - Can be combined using formal logic
- Central ideas:
  - **says** and **speaks for**
  - handoff, delegation axioms
  - useful tools for reasoning formally about authentication in any distributed system!