Title Goes Here

Understanding Authentication and Access Control in Distributed Systems

Mike Reiter

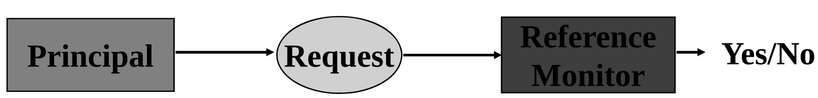
University of North Carolina at Chapel Hill

Partially based on: Lampson, Abadi, Burrows and Wobber. Authentication in distributed systems: Theory and practice. *ACM TOCS* 10(4), November 1992.

1

Access Control

- Principal makes a request for an object
- Reference monitor grants or denies the request



Ex: Editor — Send file — File server

Ex: Host — Route packet — Firewall

- Authentication: Determining who made request
- Authorization: Determining whether requestor is trusted to access an object
  - The “decision” the reference monitor must make

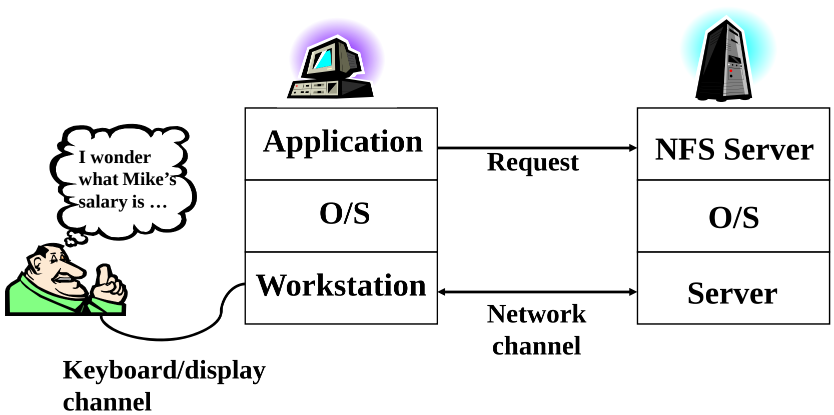
2

Authenticating a Channel

- Each request arrives on some channel, e.g.,
  - Kernel call from a user process
  - Network connection
  - A channel defined by a cryptographic key
- Reference monitor must authenticate the channel, i.e., determine whom the request is from
- Easy in a centralized system
  - OS implements all channels and knows the principal responsible for each process
- Harder in a distributed system
  - Request may have traversed different, not-equally-trusted machines
  - Different types of channels
  - Some parts of the system may be faulty or broken

3

The Challenge



- Whom is the request "from"?
  - The user? The workstation? The application?
  - All of the above?

4

Our Approach to Studying the Problem

- Explain authentication and access control using a logic
- The logic forces us to make assumptions explicit and teaches us how to think about access control

- Logic helps us to reason about principals and the statements they make
- Principals can be
  - Keys
  - People
  - Machines
  - Principals in roles
  - Groups
  - …

5

Trusted Computing Base (TCB)

- Logic will help us identify the "trusted computing base", i.e., the collection of hardware and software that security depends on
  - Compromise or failure of a TCB element may result in an incorrect "Yes" access-control decision
- Thus, TCB should be as small as possible
  - Must be carefully tested, analyzed and protected

- Benign failure of an untrusted (non-TCB) element may produce more "No" answers, not more "Yes" ones
  - This is called "fail secure" or "fail safe"
- Ex: An untrusted server holding a digitally signed credential
  - Failure prevents credential from being retrieved (more "Nos")
  - Cannot undetectably modify the credential (due to the signature)

6

The Logic

- The logic is inhabited by
  - Terms that denote principals and strings
  - Formulas that are either "true" or "false"
- Terms:

  t ::= s | p

  p ::= **key**(s) | $p.s$

  where s ranges over strings and p over principals
- Formulas:

  ϕ ::= s **signed** ϕ' | p **says** ϕ'

  ϕ' ::= **action**(s) | p **speaksfor** p | **delegate**(p, p, s)

  where s ranges over strings and p over principals

7

A Logic of Authorization (cont.)

- Inference rules

$$\frac{pubkey\ \textbf{signed}\ F}{\textbf{key}(pubkey)\ \textbf{says}\ F} \qquad \text{(says-I)}$$

$$\frac{A\ \text{says}\ (A.S\ \textbf{says}\ F)}{A.S\ \textbf{says}\ F} \qquad \text{(says-LN)}$$

8

A Logic of Authorization (cont.)

- Inference rules

$$\frac{F}{A\ \textbf{says}\ F} \qquad \text{(says-I2)}$$

$$\frac{A\ \textbf{says}\ (F \rightarrow G) \qquad A\ \textbf{says}\ F}{A\ \textbf{says}\ G} \qquad \text{(impl-E)}$$

9

A Logic of Authorization (cont.)

- Inference rules

$$\frac{A\ \textbf{says}\ (B\ \textbf{speaksfor}\ A) \qquad B\ \textbf{says}\ F}{A\ \textbf{says}\ F} \qquad \text{(speaksfor-E)}$$

$$\frac{A\ \textbf{says}\ (B\ \textbf{speaksfor}\ A.S) \qquad B\ \textbf{says}\ F}{A.S\ \textbf{says}\ F} \qquad \text{(speaksfor-E2)}$$

$$\frac{A\ \textbf{says}\ \textbf{delegates}(A, B, U) \qquad B\ \textbf{says}\ \textbf{action}(U)}{A\ \textbf{says}\ \textbf{action}(U)} \qquad \text{(delegate-E)}$$

10

Message Authentication Codes (Informal Defn)

- A message authentication code (MAC) scheme is a triple $\langle \boldsymbol{G}, \boldsymbol{T}, \boldsymbol{V} \rangle$ of efficiently computable functions
  - G outputs a "secret key" K
  $$K \leftarrow G(\cdot)$$
  - T takes a key K and "message" m as input, and outputs a "tag" t
  $$t \leftarrow T_K(m)$$
  - V takes a message m, tag t and key K as input, and outputs a bit b
  $$b \leftarrow V_K(m, t)$$
  - If $t \leftarrow T_K(m)$ then $V_K(m, t)$ outputs 1 ("valid")
  - Given only message/tag pairs $\{\langle m_i, T_K(m_i)\rangle\}_i$, it is computationally infeasible to compute $\langle m, t\rangle$ such that
  $$V_K(m, t) = 1$$
  for any new $m \neq m_i$

11

Digital Signatures (Informal Definition)

- A digital signature scheme is a triple $\langle \boldsymbol{G}, \boldsymbol{S}, \boldsymbol{V} \rangle$ of efficiently computable algorithms
  - G outputs a "public key" K and a "private key" K^{-1}
  $$\langle K, K^{-1}\rangle \leftarrow G(\cdot)$$
  - S takes a "message" m and K^{-1} as input and outputs a "signature" σ
  $$\sigma \leftarrow S_{K^{-1}}(m)$$
  - V takes a message m, signature σ and public key K as input, and outputs a bit b
  $$b \leftarrow V_K(m, \sigma)$$
  - If $\sigma \leftarrow S_{K^{-1}}(m)$ then $V_K(m, \sigma)$ outputs 1 ("valid")
  - Given only K and message/signature pairs $\{\langle m_i, S_{K^{-1}}(m_i)\rangle\}_i$, it is computationally infeasible to compute $\langle m, \sigma\rangle$ such that
  $$V_K(m, \sigma) = 1$$
  any new $m \neq m_i$

12

Hash Functions

- A hash function is an efficiently computable function $\boldsymbol{h}$ that maps an input $\boldsymbol{x}$ of arbitrary bit length to an output

$$\boldsymbol{y} \leftarrow \boldsymbol{h}(\boldsymbol{x})$$

  of fixed bit length
  - Preimage resistance: Given only y, it is computationally infeasible to find any x' such that $h(x') = y$.
  - 2nd preimage resistance: Given x, it is computationally infeasible to find any $x' \neq x$ such that $h(x') = h(x)$.
  - Collision resistance: It is computationally infeasible to find any two distinct inputs x, x' such that $h(x) = h(x')$.

13

Cryptographic Keys as Channels

- Let $\boldsymbol{t}$ be a MAC tag on message $\boldsymbol{x}$ such that $\boldsymbol{V_K(x, t) = 1}$
- Let $\boldsymbol{\sigma}$ be a digital signature on $\boldsymbol{x}$ such that $\boldsymbol{V_K(x, \sigma) = 1}$
- Interpret $\boldsymbol{t}$ or $\boldsymbol{\sigma}$ as "***K* signed *x***" (for respective $\boldsymbol{K}$)

- Sometimes, public identifiers are needed for keys (channels)
  - If K is a public key, then $\mathrm{id}(K) = K$
  - If K is a secret key, then $\mathrm{id}(K) = h(K)$ works if h is a preimage resistant, 2nd preimage resistant, and collision-resistant function
- "**id(*K*) signed *x***" can be used in place of "***K* signed *x***" when encoded in a system, if necessary

14

Authenticating a Channel

- Reference monitor receives a request ***C* says *s***
- An access-control list usually specifies named principals
- Thus, reference monitor must collect certificates to prove that ***C* speaksfor *A*** for some ***A*** on the access control list

- Two general methods
  - Push: The sender on the channel *C* collects *A*'s credentials and presents them to authenticate the channel to the receiver.
  - Pull: The receiver looks up *A* in some database to get credentials for *A* when it needs to authenticate the sender.

15

Certification Authorities

- Credentials typically come from "certification authorities"
- A certification authority is a named principal ***CA***
- ***CA*** issues statements of the form

$$K_{CA} \textbf{ signed (key(}K_A\textbf{) speaksfor key(}K_{CA}\textbf{).}A\textbf{)}$$

- If K_{CA} is a public key, this statement is called a *certificate*
  - But K_{CA} can be a symmetric key, too

16

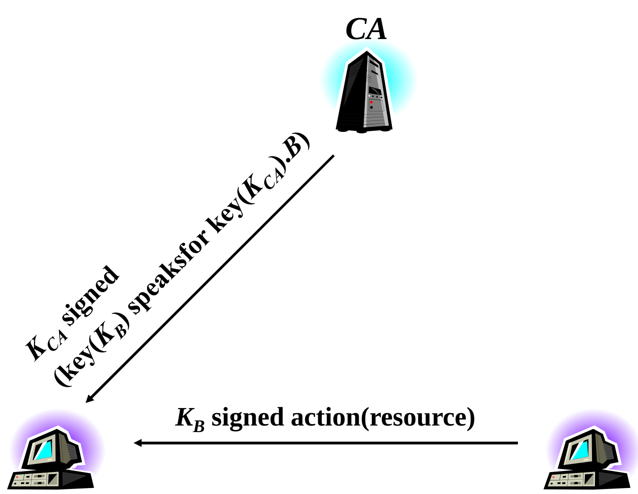
An Example Proof

1. K_{CA} signed (key(K_A) speaksfor key(K_{CA}).A)
2. K_A signed action(resource)

3.	key(K_{CA}) says (key(K_A) speaksfor key(K_{CA}).A)	says-I(1)
4.	key(K_A) says action(resource)	says-I(2)
5.	key(K_{CA}).A says action(resource)	speaksfor-E2(3, 4)

17

A Certification Authority



Infers key(K_{CA}).B says action(resource)

18

Groups

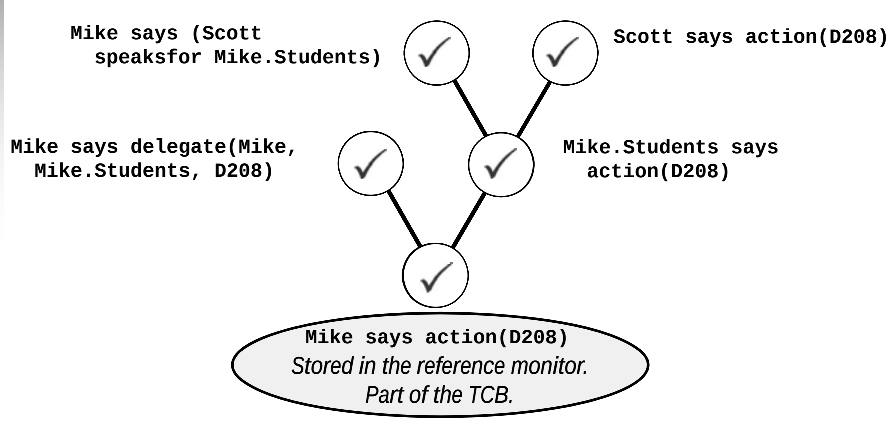
- A group is a principal whose members speak for it
- Simplest way to define a group G is for a defining CA to issue certificates

$$\text{key}(K_{CA})\ \textbf{says}\ P_1\ \textbf{speaksfor}\ \text{key}(K_{CA}).G$$
$$\text{key}(K_{CA})\ \textbf{says}\ P_2\ \textbf{speaksfor}\ \text{key}(K_{CA}).G$$
$$\ldots$$

for group members P_1, P_2, …

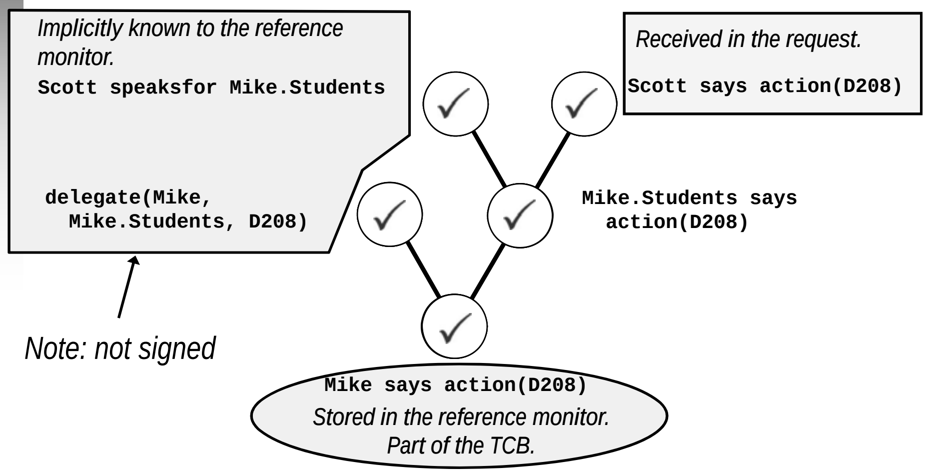
19

Example Proof



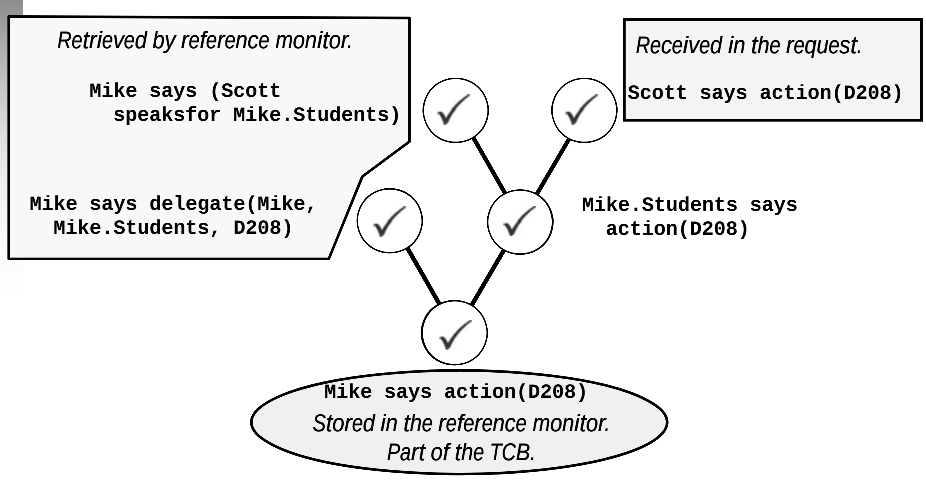
20

Traditional Access Control Lists



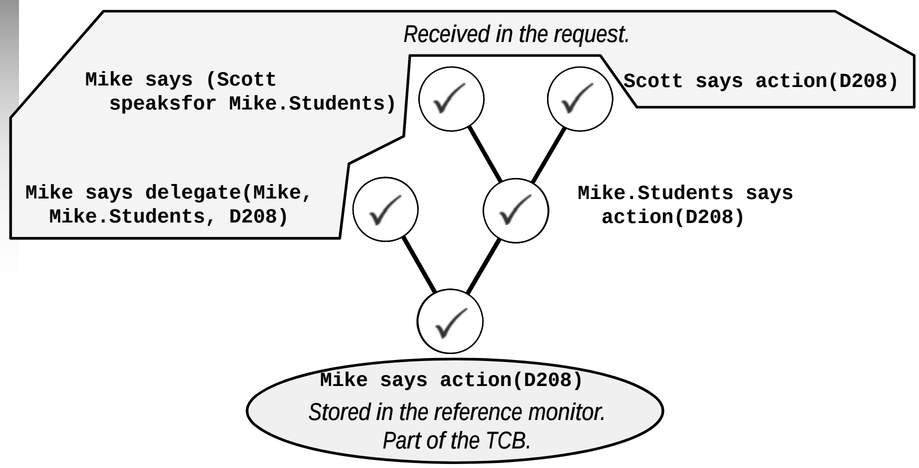
21

A “Pull” Approach



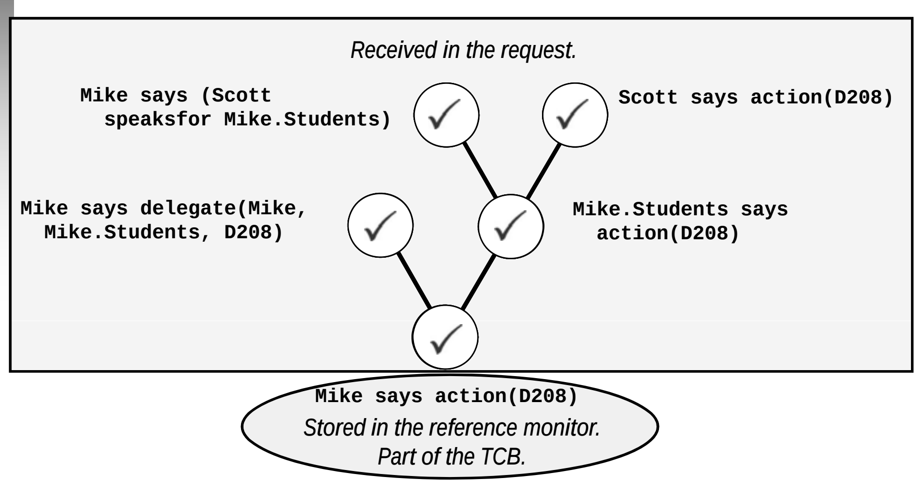
22

A “Push” Approach



23

A “Proof Carrying” Approach



24

Roles

- Suppose a principal wants to *limit* its authority
  - Reiter “as” GamePlayer
  - Reiter “as” SysAdmin
- Intuition: ***A* “as” *R*** should be weaker than ***A***
- ***A*** can accomplish this by enabling statements of the form

$$\boldsymbol{A.R} \textbf{ says } \boldsymbol{F}$$

to be created

25

Programs as an Application of Roles

- Acting in a role is like acting according to some program
- If node ***N*** is running program with text ***I***, then ***N*** can make

  $$\boldsymbol{N.I} \textbf{ says } \boldsymbol{F}$$

  for a statement ***F*** made by the process running ***I***
- Instead of using the whole program ***I***, ***N*** can instead make

  $$\boldsymbol{N.D} \textbf{ says } \boldsymbol{F}$$

  where $\boldsymbol{D = h(I)}$ for ***h*** a collision-resistant and 2nd preimage resistant hash function, and using

  $$\boldsymbol{D} \textbf{ speaksfor } \boldsymbol{P}$$

  where ***P*** is the program name

26

Loading Programs

- To load program named ***P***, node ***N***
  - Creates a process *pr*
  - Reads text *I* of file *P* from the file system
  - Finds credentials for *D* **speaksfor** *P* and checks $h(I) = D$
  - Copies *I* into *pr*
  - Gives *pr* ability to write to channel *C*
  - Emit: *N* **says** *C* **speaksfor** *N.P*
- Now ***pr*** can issue requests on channel ***C***
  - Will be granted if *N.P* is on ACL

27

Virus Control

- Some viruses alter texts of programs in the file system
  - If I' is the infected program text, then $D' = h(I')$ will be different from $D = h(I)$, and so *D* **speaksfor** *P* will not apply
- Certification authority ***CA*** can issue certificates

K_{CA} **signed** *P* **speaksfor** key(K_{CA}).trustedSW

K_{CA} **signed** *N* **speaksfor** key(K_{CA}).trustedNodes

K_{CA} **signed** (*P* **speaksfor** key(K_{CA}).trustedSW $\wedge$
N **speaksfor** key(K_{CA}).trustedNodes $\rightarrow$
N.P **speaksfor** key(K_{CA}).trustedNode.trustedSW)

where **trustedSW** and **trustedNodes** are group names, ***P*** is a program name, and ***N*** is a node name

28

Secure Booting

- 'trustedNodes' should be computers that
  - run operating systems validated before booting
  - validate other software before loading it
- Validating O/S during boot is like validating other software
  - Machine W holds $h(I)$ in boot ROM, where I is O/S image
    - i.e., $h(I)$ **speaksfor** P
- To create a channel C such that C **speaksfor** $W.P$, W can
  - Generate a new signature key pair $K_{W.P}, K_{W.P}^{-1}$, and
  - Give $K_{W.P}^{-1}$ to P, along with K_W **signed** key($K_{W.P}$) **speaksfor** key(K_W).P
- Private key for K_W must be protected in secure hardware
  - Otherwise, O/S can read it

29

Example: TCG

- Historically, PC manufacturers have chosen flexibility over security
  - User can modify the PC in any way she likes
  - PC does not have hardware protection for boot procedure, does not validate O/S before loading it, does not validate other programs
- Today this is changing with efforts like the Trusted Computing Group (TCG; www.trustedcomputing.org)
  - Alliance formed in Jan 1999 by Compaq, HP, IBM, Intel & Microsoft
  - More than 150 companies by 2002
  - Developing a standard for a "trusted platform" (TP), based on principles similar to those we've discussed
  - Scope of specs is at hardware, O/S and BIOS levels
    - Main spec released in Aug 2000 (v1.0) and Feb 2001 (v1.1)
    - PC-specific spec released in Sep 2001

30

Example: TCG

- Some goals of TP
  - Enable local and remote users to obtain reliable information about the software running on the platform
  - Provide a basis for secure key storage
  - Enable conditional release of secret information to the TP based on the software running
- TP enabled by a "trusted processing module" (TPM)
  - A hardware processing component that is isolated from software attacks and at least partially resistant to hardware tampering
- Each TPM is equipped with a different private key K_{TPM}^{-1} and a certificate

  K_{TPME} **says** key(K_{TPM}) **speaksfor** key(K_{TPME}).TrustedProcessingModules

  signed by a "trusted platform module entity" (TPME)
  - TrustedProcessingModules is a group

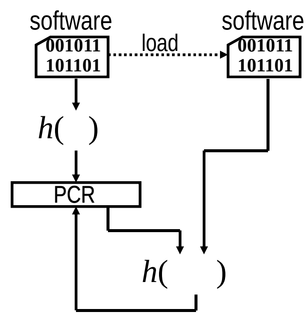
TCG "Roots of Trust"

TCPA specifies two logical "roots of trust"

- Root of trust for measurement (RTM): A platform-dependent component that starts "measurement" of software running
  - In a PC, the RTM is the platform itself, which is acceptable only if the RTM cannot be subverted before or during its operation
  - In practice, this means that the RTM must run first (or everything that is run before it is trusted)
    - e.g., BIOS boot block, called the "core root of trust for measurement" (CRTM)
- Root of trust for reporting (RTR): A platform-independent component that stores "measurements" as they happen, in such a way that measurements cannot be "undone"
  - RTR is implemented by the TPM

TPM Platform Configuration Registers

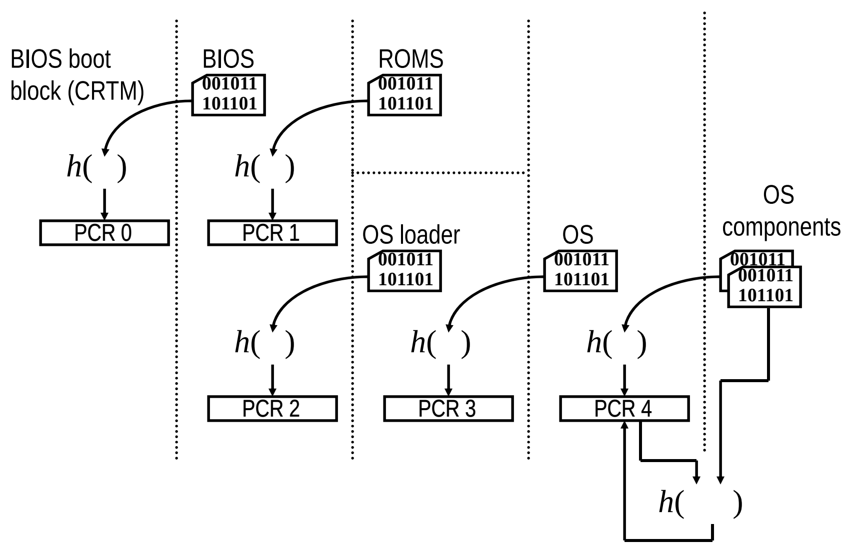
- TPM (version 1.1) contains sixteen 20-byte “platform configuration registers” (PCRs)
  - 20 bytes in order to store a SHA-1 hash value
- Each PCR records the last in a sequence of hashes of the software that has been loaded and run



- PCR is updated before newly loaded software gets control
- PCR cannot be erased except by reboot (or protected processor instruction in v1.2 TPMs)
- In this way, PCR contains record of software running

33

TCPA Authenticated Boot



34

TCG Secure Boot

- Non-volatile “data integrity registers” (DIRs) are loaded with expected PCR values
  - DIRs are contained within TPM and require owner authorization to write
- If a PCR value, when computed, doesn’t match corresponding DIR value, then boot is canceled

35

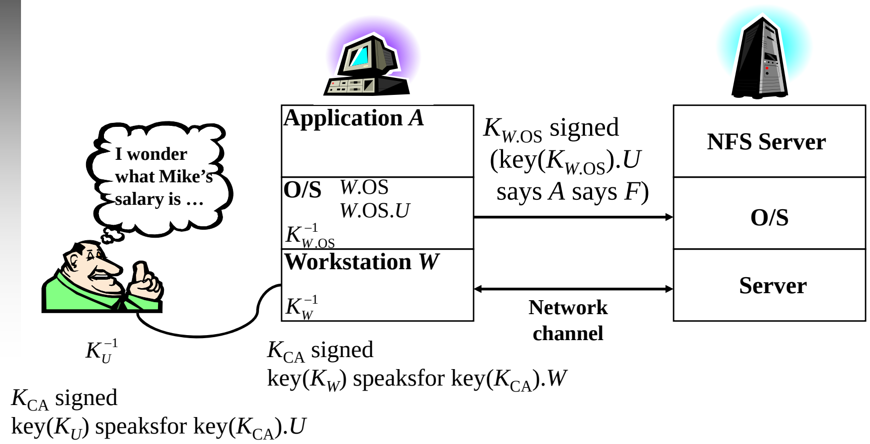
TCG Integrity Challenge and Response

- Remote machine can query TPM for contents of PCRs
- TPM responds with signed PCR values
  - Think of it as signed with K_{TPM}

    K_{TPM} **signed** PCRvals = …
  - (In reality, is not signed with K_{TPM} but another “identity key” is used to enhance privacy)
- TP additionally responds with records (hints) of what is “summarized” in the PCR values
  - Records could contain software itself, but more likely contains name, supplier, version, and URL for software
  - Enables remote machine to reconstruct and check PCR values
  - Records not trusted and so are stored outside TPM

36

Example



K_W signed key($K_{W.\mathrm{OS}}$) speaksfor key(K_{CA}).*W*.OS
K_U signed key(K_{CA}).*W*.OS.*U* speaksfor key(K_{CA}).*U*

37

Example (cont.)

1. **K_{CA} signed key(K_W) speaksfor key(K_{CA}).*W***
2. **K_{CA} signed key(K_U) speaksfor key(K_{CA}).*U***
3. **K_W signed key($K_{W.\mathrm{OS}}$) speaksfor key(K_{CA}).*W*.OS**
4. **K_U signed key(K_{CA}).*W*.OS.*U* speaksfor key(K_{CA}).*U***
5. **$K_{W.\mathrm{OS}}$ signed (key($K_{W.\mathrm{OS}}$).*U* speaksfor key(K_{CA}).*W*.OS.*U*)**
6. **$K_{W.\mathrm{OS}}$ signed (key($K_{W.\mathrm{OS}}$).*U* says *A* says *F*)**
7. **key(K_{CA}) says key(K_W) speaksfor key(K_{CA}).*W*** — **says-I(1)**
8. **key(K_{CA}) says key(K_U) speaksfor key(K_{CA}).*U*** — **says-I(2)**
9. **key(K_W) says key($K_{W.\mathrm{OS}}$) speaksfor key(K_{CA}).*W*.OS** — **says-I(3)**
10. **key(K_U) says key(K_{CA}).*W*.OS.*U* speaksfor key(K_{CA}).*U*** — **says-I(4)**
11. **key($K_{W.\mathrm{OS}}$) says (key($K_{W.\mathrm{OS}}$).*U* speaksfor key(K_{CA}).*W*.OS.*U*)** — **says-I(5)**
12. **key($K_{W.\mathrm{OS}}$) says (key($K_{W.\mathrm{OS}}$).*U* says *A* says *F*)** — **says-I(6)**

38

Example (cont.)

13. key(K_{CA}).*W* says key($K_{W.OS}$) speaksfor key(K_{CA}).*W*.OS — speaksfor-E2(7, 9)
14. key(K_{CA}).*U* says (key(K_{CA}).*W*.OS.*U* speaksfor key(K_{CA}).*U*) — speaksfor-E2(8, 10)
15. key(K_{CA}).*W*.OS says (key($K_{W.OS}$).*U* speaksfor key(K_{CA}).*W*.OS.*U*) — speaksfor-E2(13, 11)
16. key($K_{W.OS}$).*U* says *A* says *F* — says-LN(12)
17. key(K_{CA}).*W*.OS.*U* says *A* says *F* — speaksfor-E2(15, 16)
18. key(K_{CA}).*U* says *A* says *F* — speaksfor-E(14, 17)

39

Example: Web Server Authentication (1)

- What happens when you access https://www.foo.com?
- A protocol called Secure Sockets Layer (SSL) or Transport Layer Security (TLS) is used to authenticate the web server
  - Also performs other functions that are not important for the moment

HTTP	FTP	SMTP
SSL or TLS		
TCP		
IP		

40

Example: Web Server Authentication (2)

- As part of SSL/TLS, web server sends a certificate

  K_{CA} **signed (key(**$K_{\text{www.foo.com}}$**) speaksfor key(**K_{CA}**).'www.foo.com')**
  to browser
- Browser is shipped with public keys for numerous *CA*s:

  K_{CA1} , K_{CA2} , K_{CA3} , …

  - Mozilla Firefox ships with over 80 *CA* keys loaded
  - Reportedly these represent 34 organizations from 15 countries: BE, BM, DE, DK, ES, FI, GB, IE, JP, NL, PL, SE, US, WW, ZA
- Should we really trust that key(K_{CA}).'www.foo.com' is the "right" www.foo.com for all 80 CAs?

41

What if $K^{-1}_{\text{www.foo.com}}$ Is Compromised?

- In SSL/TLS, the certificate is sent from the web server
  - CA sends long-lived certificate to web server in advance
  - Web server stores it, and forwards it in SSL/TLS handoff protocol
- This structure has a benefit
  - K^{-1}_{CA} can be kept offline and made more secure
- What if $K^{-1}_{\text{www.foo.com}}$ is exposed?
  - *CA* may wish to revoke the statement (certificate)

    K_{CA} **signed** (key($K_{\text{www.foo.com}}$) **speaksfor** key(K_{CA}).'www.foo.com')

42

Certificate Countersigning

- For rapid certificate revocation, there needs to be some online authority *O* that vouches for it
  - Compromise of *O* can keep a certificate "alive" longer than it should be, but cannot make new certificates
- ***CA*** makes a weaker certificate

  K_{CA} **signed ((key(K_O) says key(K_A) speaksfor key(K_O).A)**
  → key(K_A) speaksfor key(K_{CA}).A)
- ***O*** "countersigns" with

  K_O **signed (date() < '2008.07.31'**
  → key(K_A) speaksfor key(K_O).A)

43

Certificate Countersigning

1. K_{CA} signed ((key(K_O) says key(K_A) speaksfor key(K_O).A) → key(K_A) speaksfor key(K_{CA}).A)
2. K_O signed (date() < '2008.07.31' → key(K_A) speaksfor key(K_O).A)
3. key(K_{CA}) says ((key(K_O) says key(K_A) speaksfor key(K_O).A) → key(K_A) speaksfor key(K_{CA}).A) — says-I(1)
4. key(K_O) says (date() < '2008.07.31' → key(K_A) speaksfor key(K_O).A) — says-I(2)
5. date() < '2008.07.31'
6. key(K_O) says (date() < '2008.07.31') — says-I2(5)
7. key(K_O) says key(K_A) speaksfor key(K_O).A — impl-E(4, 6)
8. key(K_{CA}) says (key(K_O) says key(K_A) speaksfor key(K_O).A) — says-I2(7)
9. key(K_{CA}) says key(K_A) speaksfor key(K_{CA}).A — impl-E(3, 8)

44

Certificate Revocation Lists

- Certificate Revocation Lists (CRLs) are an alternative to countersignatures by an online authority
  - Also more commonly used
- Each CA periodically produces a digitally signed statement recanting listed certificates

K_{CA} says "certificates $\underbrace{\text{134, 538, and 977}}_{\textbf{Certificate serial numbers}}$ are invalid"

- CRLs must have limited lifetimes
- All certificate serial numbers must be included in *one* CRL

45

Revisiting Trust of *CA*

- Trusting that for all ***CA***s, $\mathbf{key}(K_{CA}).A$ is the "correct" ***A*** is too strong
  - Remember that Firefox comes shipped with more than 80 of them!
- A better approach would reduce this trust
- If principal names are hierarchical, then this is natural
  - Many naming schemes are hierarchical, but the most well known one is the Domain Name System ("DNS")

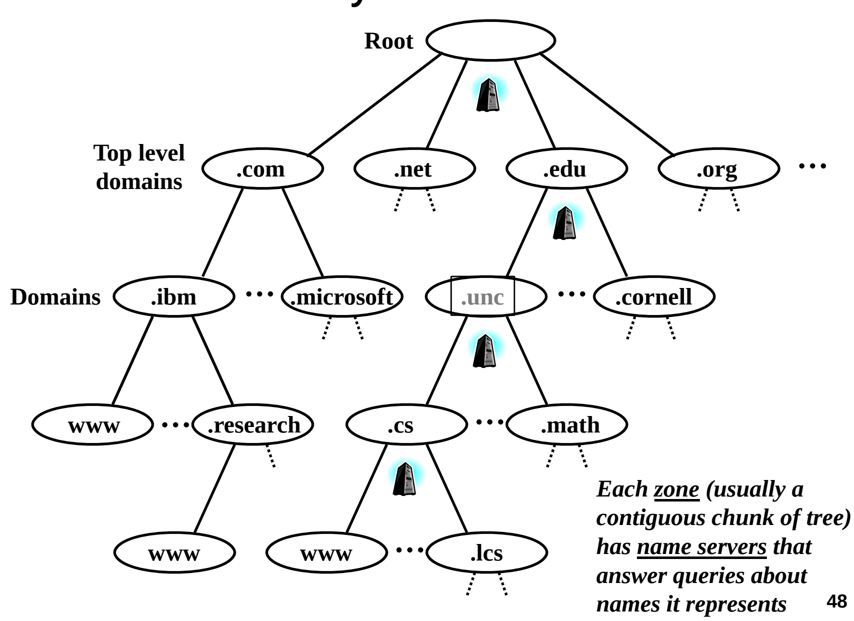
46

Example: DNS Security

- DNS translates between human-readable hostnames and IP addresses
  - Ex: translates www.foo.com to 208.228.229.218
  - Originally specified in RFC 1034 and RFC 1035, and revised by many since
- DNS Security (“DNSSEC”) specifies extensions to DNS to make DNS more secure
  - “Owned” by the DNSEXT working group in IETF
  - Specified in RFC 2065 (January 1997), probably revised since

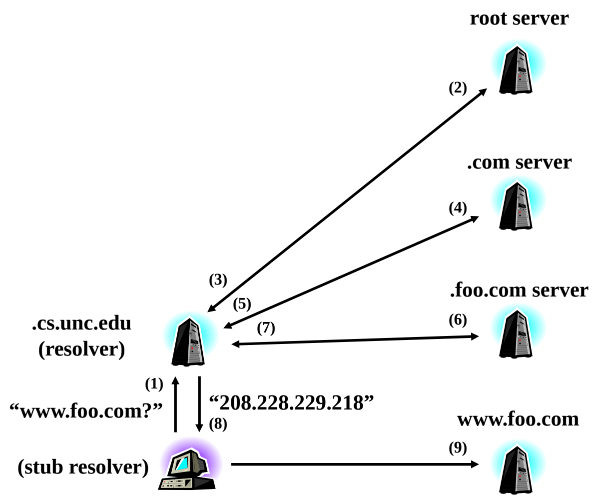
47

DNS Name Hierarchy



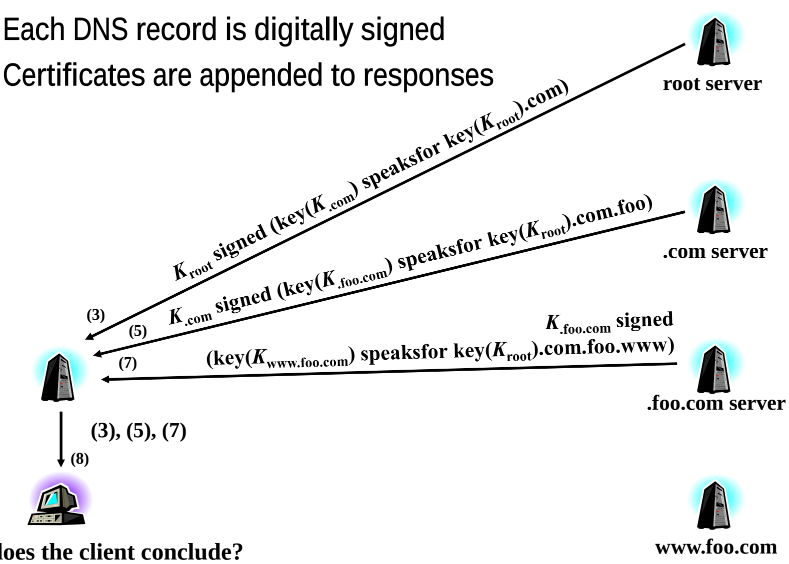
48

DNS Name Resolution



DNSSEC

- Each DNS record is digitally signed
- Certificates are appended to responses



What does the client conclude?

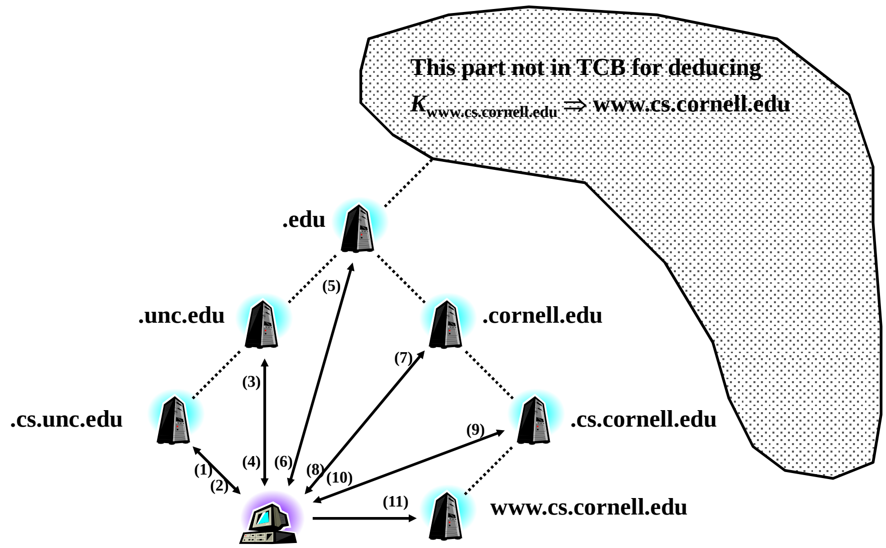
Example Proof

1. K_{root} **signed (key(**$K_{.com}$**) speaksfor key(**K_{root}**).com)**
2. $K_{.com}$ **signed (key(**$K_{.foo.com}$**) speaksfor key(**K_{root}**).com.foo)**
3. $K_{.foo.com}$ **signed (key(**$K_{www.foo.com}$**) speaksfor key(**K_{root}**).com.foo.www)**
4. $K_{www.foo.com}$ **signed** F
5. **key(**K_{root}**) says (key(**$K_{.com}$**) speaksfor key(**K_{root}**).com)** — **says-I(1)**
6. **key(**$K_{.com}$**) says (key(**$K_{.foo.com}$**) speaksfor key(**K_{root}**).com.foo)** — **says-I(2)**
7. **key(**$K_{.foo.com}$**) says (key(**$K_{www.foo.com}$**) speaksfor key(**K_{root}**).com.foo.www)** — **says-I(3)**
8. **key(**$K_{www.foo.com}$**) says** F — **says-I(4)**
9. **key(**K_{root}**).com says (key(**$K_{.foo.com}$**) speaksfor key(**K_{root}**).com.foo)** — **speaksfor-E2(5, 6)**
10. **key(**K_{root}**).com.foo says (key(**$K_{www.foo.com}$**) speaksfor key(**K_{root}**).com.foo.www)** — **speaksfor-E2(9, 7)**
11. **key(**K_{root}**).com.foo.www says** F — **speaksfor-E2(10, 8)**

What Went Wrong?

- We didn't reduce the trust on the root
  - But that's real life: DNSSEC root is in TCB for every DNS name
- Is this bad? … The answer depends on your perspective
- Optimist: DNS already requires a trusted root, at least DNSSEC is better (but not in this sense)
- Pessimist: Could have done better
  - But probably not without changing how DNS works
  - So, let's try changing how DNS works

Eliminating a Globally Trusted Authority



53

Extensions to the Logic

$$\frac{A \text{ says ascend}(\text{key}(K_{B.C}), B.C.D) \qquad \text{key}(K_{B.C}) \text{ says ascend}(\text{key}(K_B), B.C)}{A \text{ says ascend}(\text{key}(K_B), B.C)} \quad \text{(ascent)}$$

- If $C \neq D$

$$\frac{A \text{ says ascend}(\text{key}(K_B), B.C) \qquad \text{key}(K_B) \text{ says descend}(\text{key}(K_{B.D}), B.D)}{A \text{ says descend}(\text{key}(K_{B.D}), B.D)} \quad \text{(a2d)}$$

54

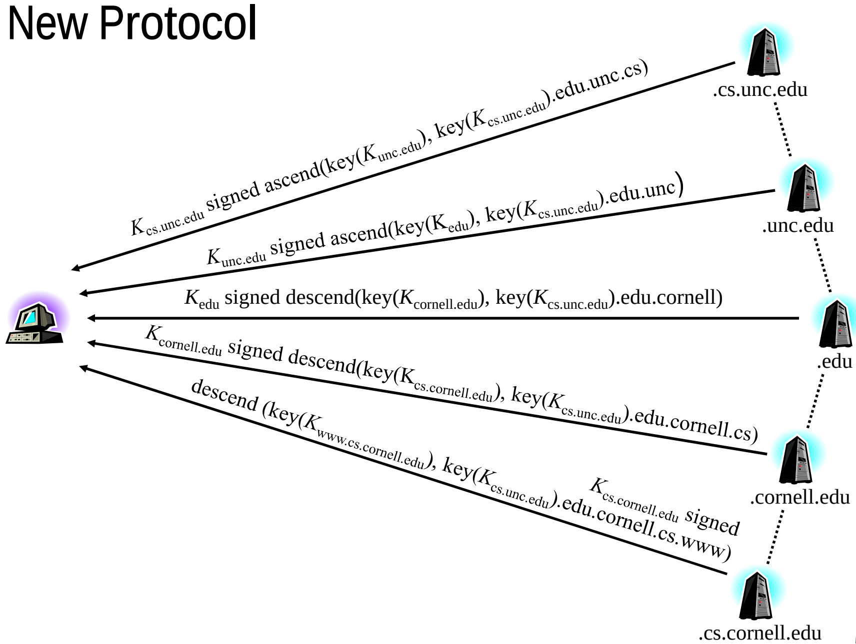
Extensions to the Logic (cont.)

$$\frac{A \text{ says descend}(\text{key}(K_B), B) \qquad \text{key}(K_B) \text{ says descend}(\text{key}(K_{B.C}), B.C)}{A \text{ says descend}(\text{key}(K_{B.C}), B.C)} \quad \text{(descent)}$$

$$\frac{A \text{ says descend}(\text{key}(K_B), B)}{A \text{ says key}(K_B) \text{ speaksfor } B} \quad \text{(resolve)}$$

55

New Protocol



56

Analysis

1. $K_{cs.unc.edu}$ **signed ascend(key($K_{unc.edu}$), key($K_{cs.unc.edu}$).edu.unc.cs)**
2. $K_{unc.edu}$ **signed ascend(key(K_{edu}), key($K_{cs.unc.edu}$).edu.unc)**
3. K_{edu} **signed descend(key($K_{cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell)**
4. $K_{cornell.edu}$ **signed**
   descend(key($K_{cs.cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell.cs)
5. $K_{cs.cornell.edu}$ **signed**
   descend (key($K_{www.cs.cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell.cs.www)
6. $K_{www.cs.cornell.edu}$ **signed** F

7. **key($K_{cs.unc.edu}$) says ascend(key($K_{unc.edu}$), key($K_{cs.unc.edu}$).edu.unc.cs)** **says-I(1)**
8. **key($K_{unc.edu}$) says ascend(key(K_{edu}), key($K_{cs.unc.edu}$).edu.unc)** **says-I(2)**
9. **key(K_{edu}) says descend(key($K_{cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell)** **says-I(3)**

57

Analysis (cont.)

10. **key($K_{cornell.edu}$) says descend(key($K_{cs.cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell.cs)** **says-I(4)**
11. **key($K_{cs.cornell.edu}$) says descend (key($K_{www.cs.cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell.cs.www)** **says-I(5)**
12. **key($K_{www.cs.cornell.edu}$) says** F **says-I(6)**

13. **key($K_{cs.unc.edu}$) says ascend(key(K_{edu}), key($K_{cs.unc.edu}$).edu.unc)** **ascent(7, 8)**
14. **key($K_{cs.unc.edu}$) says**
    descend(key($K_{cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell) **a2d(13, 9)**
15. **key($K_{cs.unc.edu}$) says descend(key($K_{cs.cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell.cs)** **descent(14, 10)**
16. **key($K_{cs.unc.edu}$) says descend (key($K_{www.cs.cornell.edu}$), key($K_{cs.unc.edu}$).edu.cornell.cs.www)** **descent(15, 11)**

58

Analysis (cont.)

17. key($K_{cs.unc.edu}$) says key($K_{www.cs.cornell.edu}$) speaksfor key($K_{cs.unc.edu}$).edu.cornell.cs.www — resolve(16)
18. key($K_{cs.unc.edu}$).edu.cornell.cs.www says F — speaksfor-E2(12, 17)

59

Bibliography

- Lampson, Abadi, Burrows and Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems* 10(4), November 1992.

60