0x1A Great Papers in Computer Security

Vitaly Shmatikov

http://www.cs.utexas.edu/~shmat/courses/cs380s/

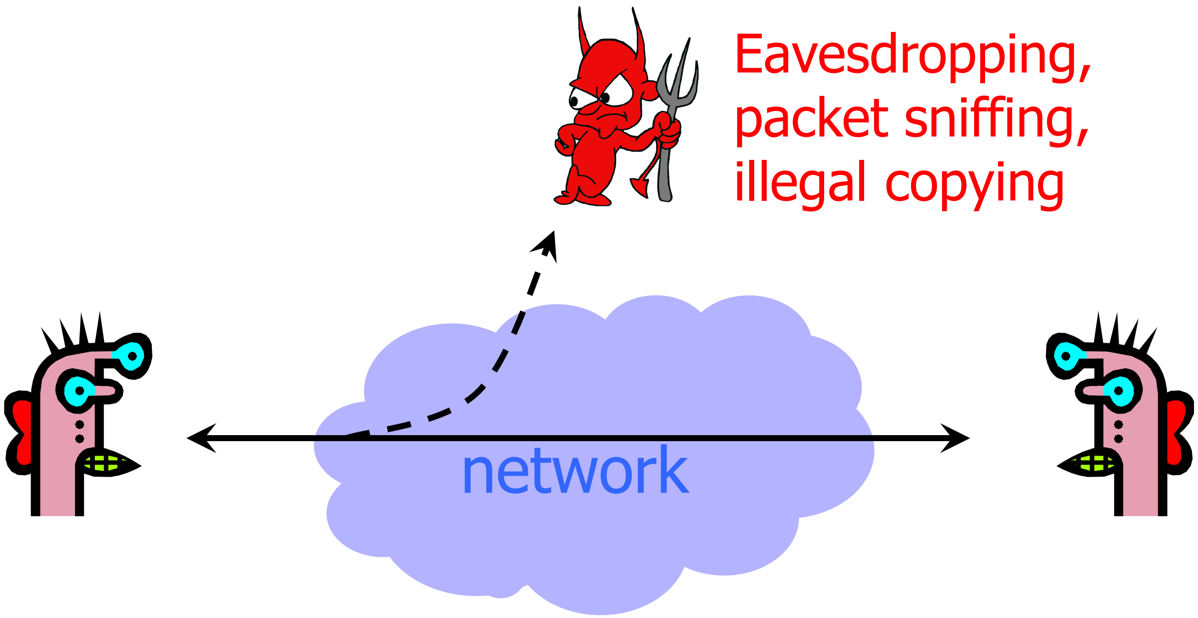
B. Lampson, M. Abadi, M. Burrows, E. Wobber

Authentication in Distributed Systems: Theory and Practice

(ACM Trans. Computer Systems 1992)

Confidentiality (Secrecy)

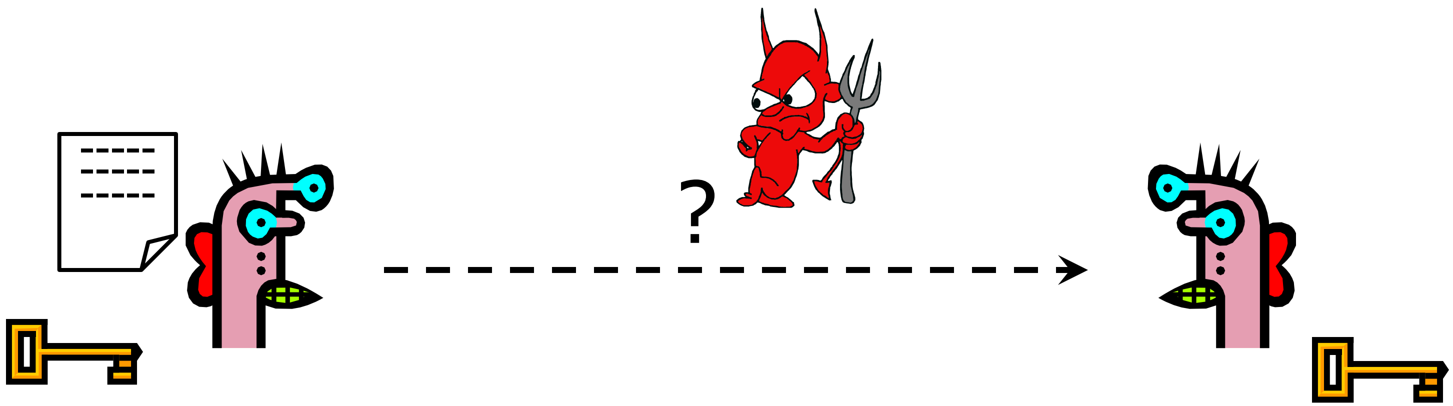
- Confidentiality is concealment of information



Q: Who is the receiver of the message?

(who might be able to read it)

Symmetric Encryption

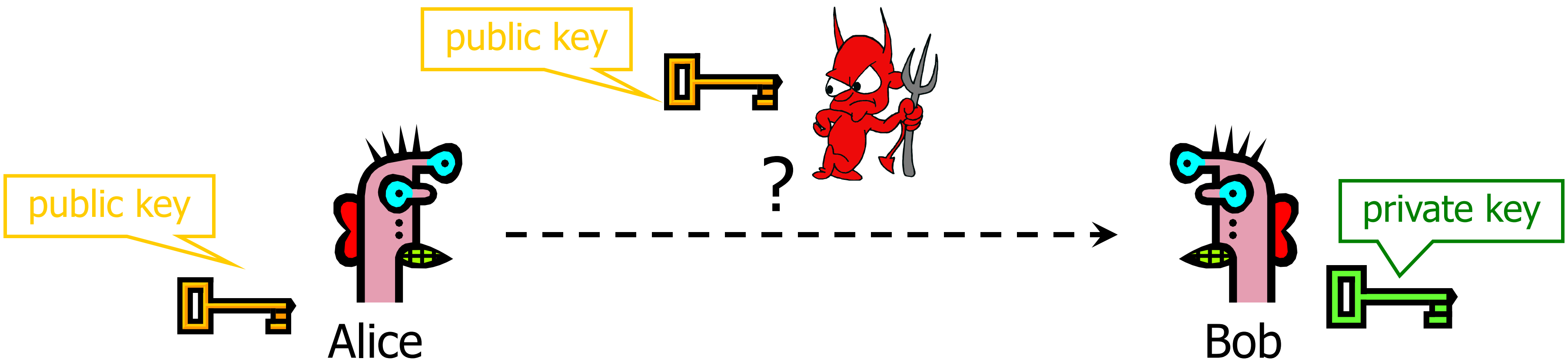


Given: both parties already know the same secret

Goal: send a message confidentially

How is this achieved in practice?

Public-Key Encryption



How is this achieved in practice?

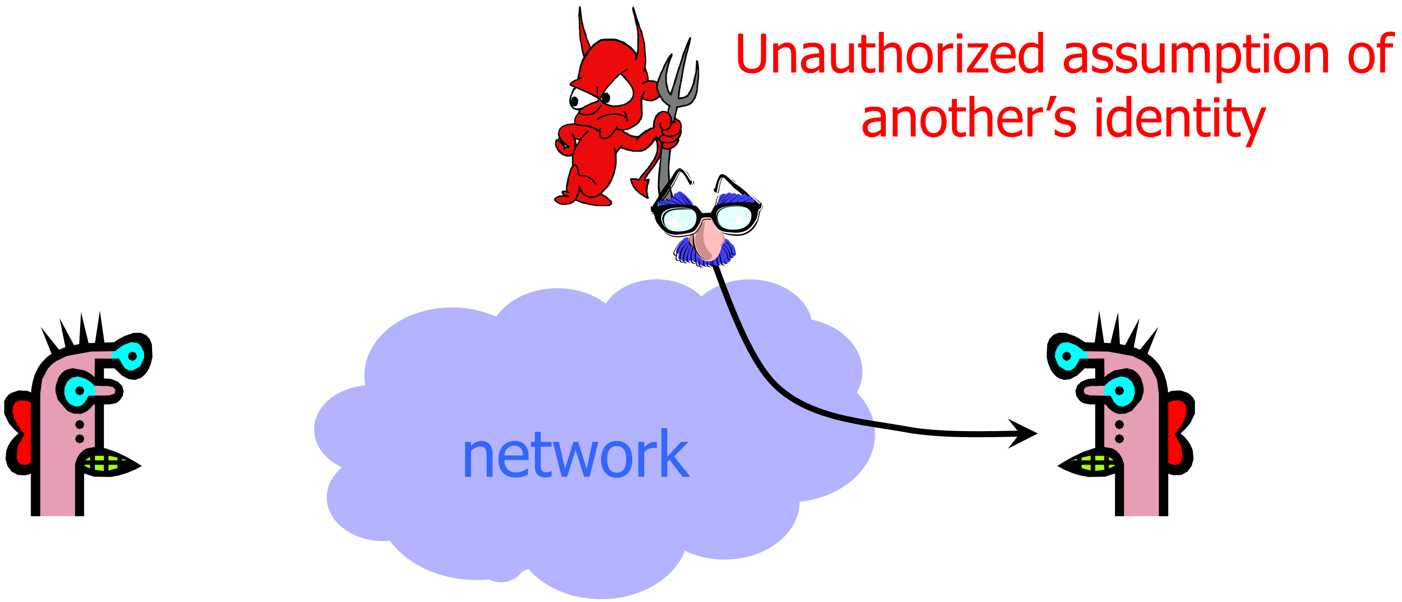
Given: Everybody knows Bob's public key

Only Bob knows the corresponding private key

Goal: Send a message to Bob confidentially

Authentication

- Authentication is identification and assurance of origin of information

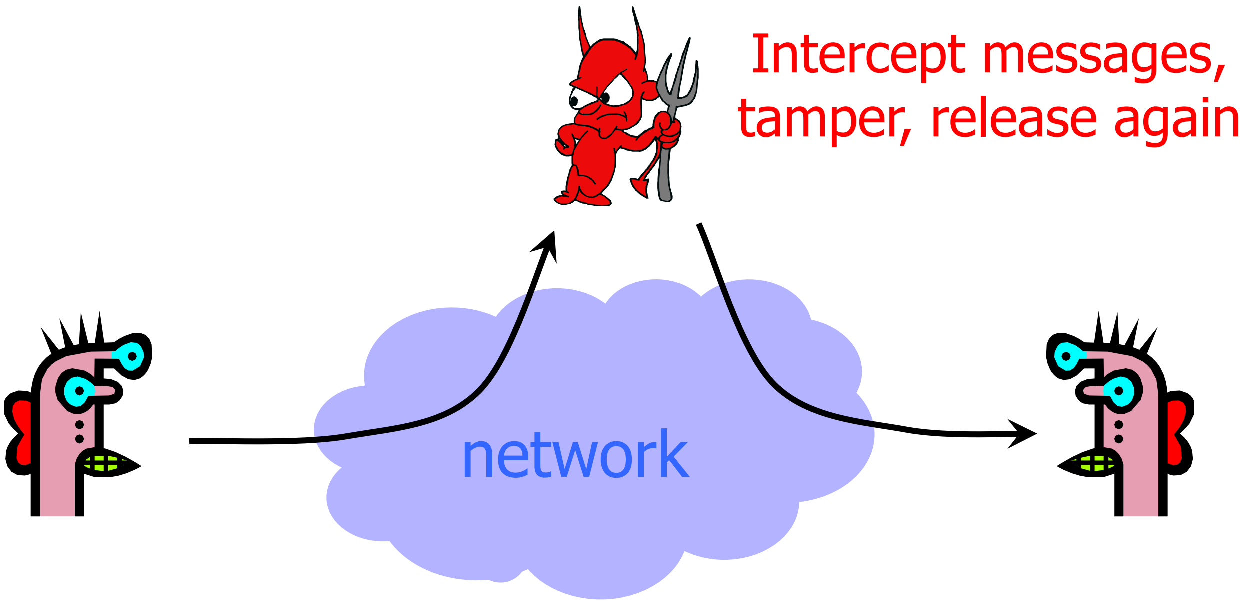


Q: Who is the sender of the message?

(who might have been able to create it)

Integrity

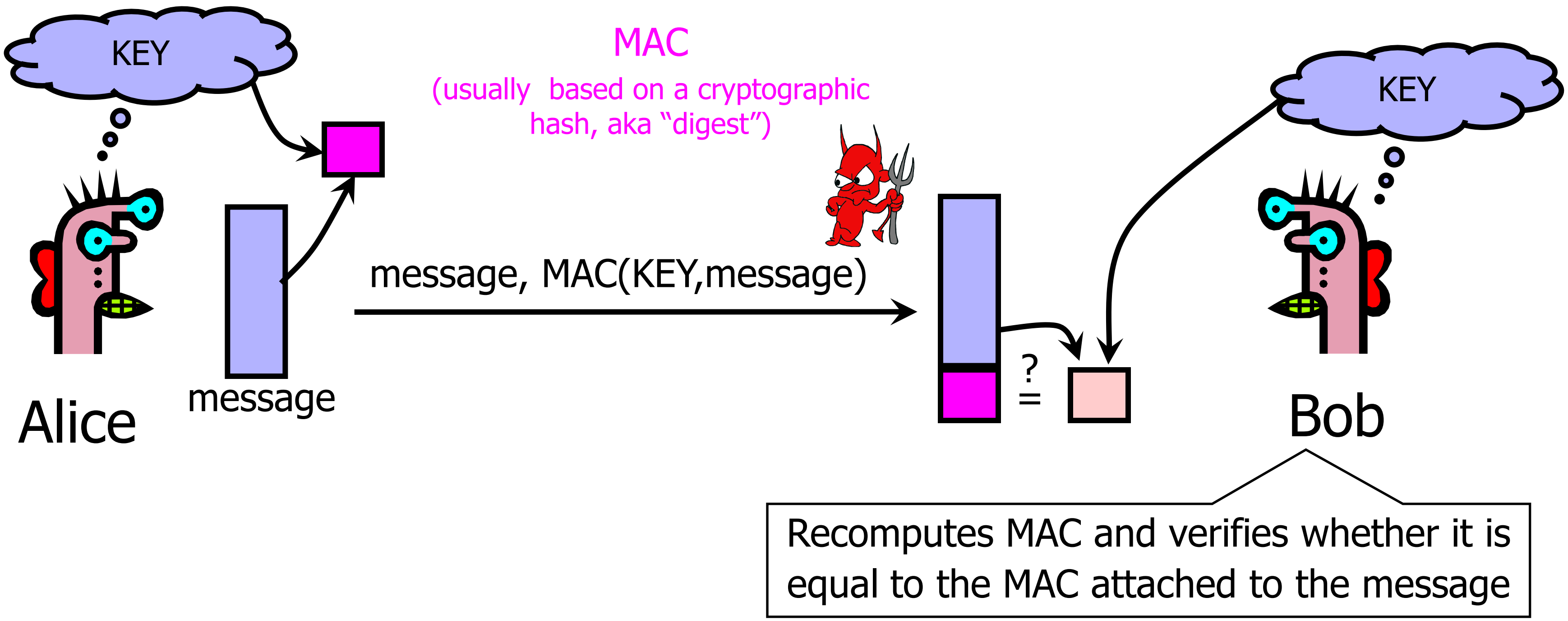
- Integrity is prevention of unauthorized changes



Q: Who is the sender of the message?

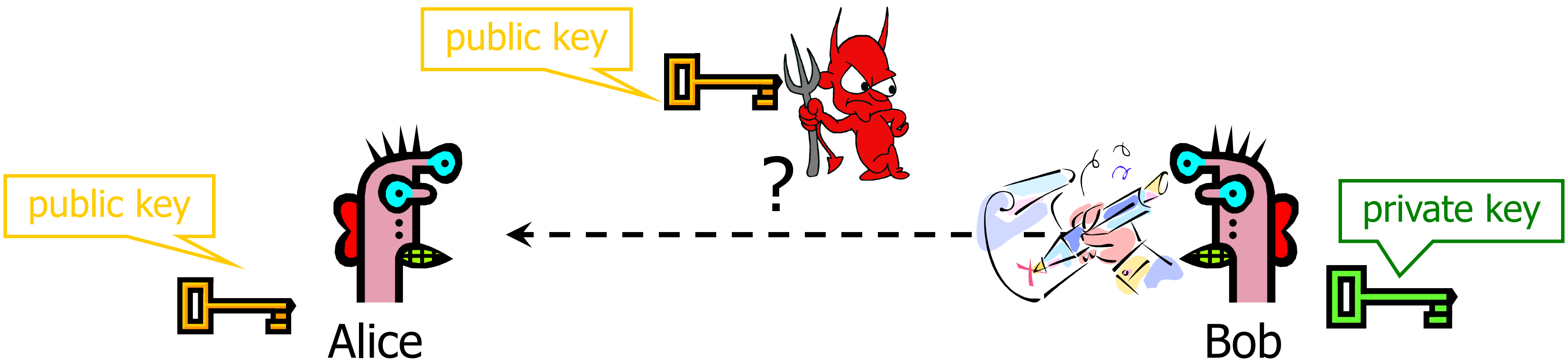
(who might have been able to modify it)

MAC: Message Authentication Code



Integrity and authentication: only someone who knows KEY can compute MAC for a given message

Digital Signature



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goal: Bob sends a "digitally signed" message

- To create a valid signature, must know the private key
- To verify a signature, enough to know the public key

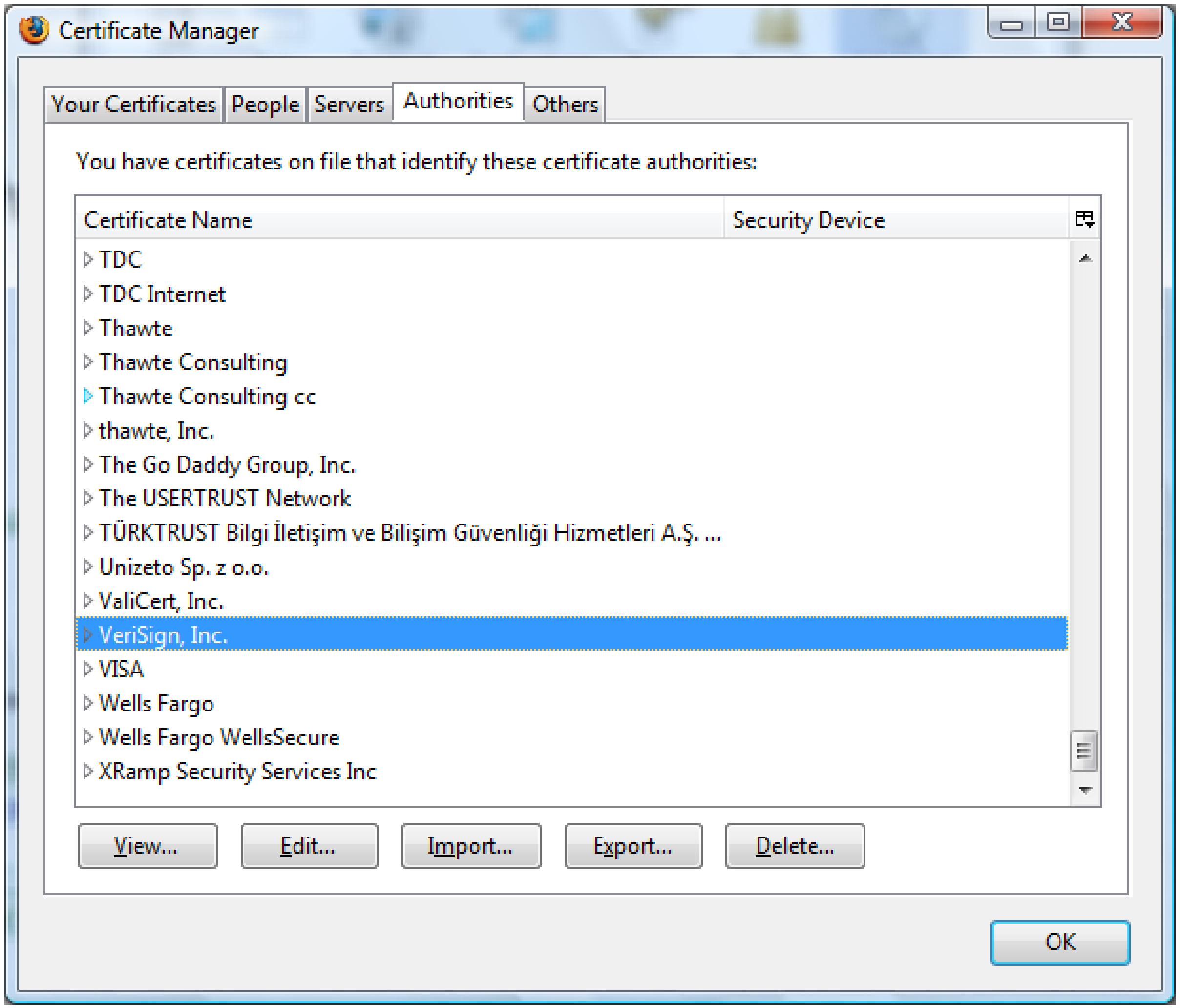
Distribution of Public Keys

- Public announcement or public directory
  - Risks: forgery, tampering
- Public-key certificate
  - Signed statement binding a public key to an identity
    - sig_{Alice}("Bob", PK_B)
- Common approach: certificate authority (CA)
  - An agency responsible for certifying public keys
  - Browsers are pre-configured with 100s of trusted CAs
    - 135 trusted CA certificates in Firefox 3
    - A public key for any website in the world will be accepted by the browser if certified by one of these CAs

Hierarchical Approach

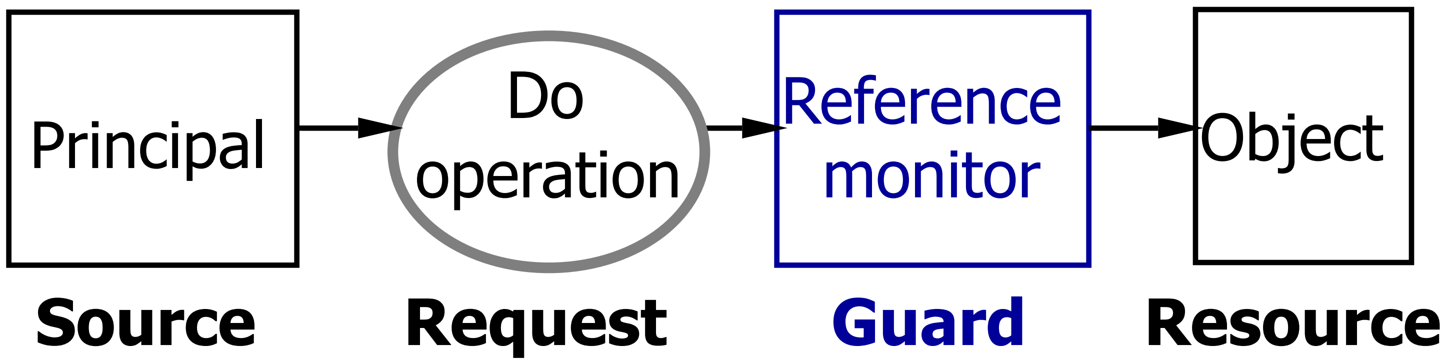
- Single CA certifying every public key is impractical
- Instead, use trusted root authorities
  - Everybody has root CAs' public keys
- A root authority signs certificates for lower-level authorities, lower-level authorities sign certificates for individual networks, and so on
  - Instead of a single certificate, use a certificate chain
    - $sig_{VeriSign}$("UT Austin", PK_{UT}), sig_{UT}("Vitaly S.", PK_V)
  - What happens if a root authority is ever compromised?

Trusted Certificate Authorities



The Access Control Model

- Guards control access to valued resources.



Goal: Decide whether to grant a request to access an object

Access Control in OS

- Assume secure channel from user
- Authenticate user by local password
- Map user to her user ID + group IDs
  - Local database for group memberships
- Access control by ACL on each resource
  - OS kernel is usually the reference monitor
  - Any RPC target can read IDs of its caller
- ACLs are lists of IDs
  - A program has IDs of its logged-in user

Distributed Systems Are Harder

- Autonomy
  - Path to a resource may involve untrusted machines
- Size
- Heterogeneity
  - Different kinds of channels: encryption, physically secure wires, inter-process channels within OS
- Fault tolerance
  - Components may be broken or inaccessible

Trusted Computing Base (TCB)

- Hardware and local operating system on each node
- Channels based on encryption

Authentication and Authorization

- Given a statement **s**, authentication answers the question "who said **s**?"
- Given an object **o**, authorization answers the question "who is trusted to access **o**?"

"who" refers to a principal

Principals and Subjects

- Principal and subject are both used to denote the active entity in an access operation
- Many different and confusing meanings
  - Principals are subjects in the TCSEC sense, but not all subjects are principals. [Gasser, 1989]
  - Principals are public keys. [SDSI, 1996]
  - The term principal represents a name associated with a subject. Since subjects may have multiple names, a subject essentially consists of a collection of principals. [Gong, 1999]

Principal = Abstraction of "Who"

- Authentication: Who sent a message?
- Authorization: Who is trusted?

- Principal — abstraction of "who"
  - People Lampson, Gray
  - Machines SN12672948, Jumbo
  - Services microsoft.com, Exchange
  - Groups UTCS, MS-Employees

Principals and Channels

- Principal says statements
  - Lampson **says** “read /MSR/Lampson/foo”
  - Microsoft-CA **says** “Lampson's key is #7438”
- Secure channel says messages (RPCs)
  - Has known possible receivers — Secrecy
  - Has known possible senders — Integrity

Implementing Secure Channels

- Within a node
  - Responsibility of OS (pipes, interprocess sockets, etc.)
- Between nodes
  - Secure wire - difficult to implement
  - Network - fantasy for most networks
  - Encryption - practical

Delegation

- Principal A speaks for B: $A \Rightarrow B$
  - Meaning: if A says something, B says it, too
    - Lampson $\Rightarrow$ MSR
    - Server-1 $\Rightarrow$ MSR-NFS
    - Key #7438 $\Rightarrow$ Lampson
- Handoff rule:

  if A says $B \Rightarrow A$, then $B \Rightarrow A$

Authorization with ACLs

- Access control lists (ACLs)
  - An object O has an ACL that says:
    principal P may access O with certain rights
    - Lampson may read and write O
    - MSR may append to O
- ACLs typically use names for principals
  - So that humans can read them

Names and Name Spaces

- A name is local to some name space
  - Examples of path names:
    - $K_{microsoft}$ / Lampson / friends
    - $K_{lampson}$ / friends
- A name space is defined by a key
- The key can bind names in its name space via public certificates
  - $K_{microsoft}$ **says** $K_{bwl} \Rightarrow K_{microsoft}$ / Lampson

Secure Channels

- The channel is defined by the public key
  - If only A knows the private key corresponding to a public key K, then $K \Rightarrow A$
    - Intuition: key K speaks for A because any signed message that passes verification with K must have come from A
- "K **says** s" is a message s which is signed by the private key corresponding to public key K
- More complex for symmetric keys

Authenticating a Channel

- Intuition: secure channel "speaks for" its sender
  - $C \Rightarrow P$ where C is the channel, P is the sender
- Trusted principal K_{ca} that "owns" sender P can authenticate channels from P by providing an appropriate certificate
  - K_{ca} **says** $K_{ws} \Rightarrow K_{ca}$ / WS
  - K_{ca} **says** $K_{bwl} \Rightarrow K_{ca}$ / Lampson

Checking Access

- Given a request an ACL — Q says read O; $P \Rightarrow$ read/write O
- Check that Q speaks for P — $Q \Rightarrow P$; rights are enough — read/write $\geq$ read

$Q \Rightarrow P \Rightarrow$ read/write O,

thus $Q \Rightarrow$ read/write O

Groups and Group Credentials

- A group is a principal; its members speak for it
  - Lampson $\Rightarrow$ MSR
  - Rashid $\Rightarrow$ MSR
- Certificates prove group membership
  - K_{MSR} says Lampson $\Rightarrow K_{MSR}$ / MSR

Auditing

- Formal proof for every access control decision
  - Can be written into the audit trail
- Premises are statements about channels or base assumptions made by the reference monitor
- Each proof step is justified by a signed statement (certificate) or a rule

Reasoning About Certificates

- Certificates are a general tool, but can be hard to reason about
- (Relatively) simple: SSL certificate
  - Trusted third party (CA) attests to binding between a public key and principal's name
- How can we reason formally about whether collection of certificates truly authenticates some principal to perform some operation on some object?

Strawman Authentication Model

- Scenario: user on a client workstation needs to authenticate to file server
  - User is a principal
  - User is authorized on file server to perform certain operations on certain file objects
- Strawman model
  - 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

Drawbacks of Strawman Model

- Public-key cryptography is slow
- Model is too rigid for distributed systems
  - Suppose user logs into second machine, now second machine needs to sign file server RPCs, too
  - If it sends messages to first machine for signing, how does first machine know they are authentic?
  - Rely on user – how does user know? What if user goes home, leaves computation running for hours?

Authentication in TAOS

- Each machine has identity: public/private key pair
- User lampson logs into machine X, signs certificate "lampson says X speaks for lampson"
  - True because X is executing lampson's programs
- X now can:
  - Open a secure channel to file server, thus file server knows it's talking to X (why?)
  - Present "lampson says X speaks for lampson" to file server, thus server knows X can speak for user (why?)
  - Send RPCs generated by lampson's programs to server

… all without machine X holding lampson's private key!

Authorizing Second Machine

- lampson logs into second machine (Y) via SSH, wants it to talk to file server on behalf of lampson
- SSH on X signs "X says Y can speak for lampson", gives this certificate to Y when lampson logs into Y
- Y presents proof to file server:
  - I'm Y
  - X says Y can speak for lampson
  - lampson says X can speak for lampson
- File server can check signatures and verify that request is authorized

Certificates

Certificates are true independently of channels and therefore can be

- … stored
- … passed to other parties
- … used to prove transitive trust relationships

Delegation of Authority

- Meaning of (A | B)
  - A signed a statement, claiming (no proof yet) that A is speaking for B
- Meaning of (A for B)
  - Logical conclusion that A is allowed to speak for B
    - (A | B) + delegation
  - Interpreted as B for purposes of access control, but preserves who actually signed the statement (A)

Scenario

- User Bob logs into workstation WS
- Need to authenticate requests from Bob’s login session to a remote file server FS
- Principals involved:
  - Workstation firmware, OS, Bob, channel from WS to FS

State Before Bob Logs In

- Workstation firmware knows long-term private signing key corresponding to public key K_{vax4}
- User knows his own long-term private signing key $PrivateKey_{bob}$
- File server has $PublicKey_{bob}$ in an ACL
  - … or, rather, "Bob" + Bob's public-key certificate

Workstation Boot: Generating K_{ws}

- At boot time, workstation firmware generates fresh public key K_{ws} and correspond. private key
  - Why not just use K_{vax4} directly?
    - Don't want it to be compromised because of frequent use
    - Don't want statements to survive reboot - certificates generated for a login session should die with the session
- 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 and not just use K_{ws} as workstation's true identity?
    - Want workstation's identity to survive reboots

State after Boot-up

- Why do workstations need identity at all?
  - So users can delegate to it!
- After boot-up, vax4's authentication agent knows
  - K_{ws}
  - Certificate: K_{vax4} says (K_{ws} speaks for K_{vax4})

… forgets K_{vax4}!

Logging In

- Login = user delegates authority to workstation
  - Want WS to be able to act for Bob
- Bob signs with his private key following certificate:

  "K_{bob} says ((K_{ws} | K_{bob}) speaks for (K_{ws} for K_{bob}))"
  - – Bob's private key 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?
  - Statements by K_{ws} are ambiguous, may be used out of context

Special principal:
"WS acting on behalf of Bob"

State After Bob's Login

- After delegation by Bob, vax4's authentication agent knows:
  - Private key corresponding to K_{ws}
  - K_{vax4} says (K_{ws} speaks for K_{vax4})
  - K_{bob} says ((K_{ws} | K_{bob}) speaks for (K_{ws} for K_{bob}))

Channels

- Channels are encrypted using symmetric-key ciphers and named by their symmetric key
- C_{bob} is a mnemonic to indicate intent that channel carries messages from Bob, but system must prove that this is indeed the case!
- File server knows "C_{bob} says RQ"
  - Meaning: file server received request RQ from someone who knows channel key C_{bob}
- But who knows channel key C_{bob}?
  - K_{ws}? K_{ws} on behalf of Bob? On behalf of someone else?

Channel Certificates (1)

- RQ is encrypted with C_{bob}, need to link it to Bob
- WS signs the channel certificate when the channel between WS and file server is first created

  (K_{ws} | K_{bob}) says (C_{bob} speaks for (K_{ws} for K_{bob}))
- Why not just have K_{bob} sign "C_{bob} speaks for K_{bob}"
  - Authentication agent doesn't hold the private key corresponding to K_{bob} (why?) and can't sign such statements

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 for K_{bob}
- Channel certificate says only what's needed and no more
  - K_{ws} says C_{bob} speaks for (K_{ws} speaking for Bob)
- But K_{ws} could sign this statement without Bob's agreement, so file server needs K_{ws} to prove that it is allowed to speak for Bob

All Certificates Together

- K_{vax4} says (K_{ws} speaks for K_{vax4})
- 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}))

Delegation Axiom

- Delegation axiom (informally): If Bob signs a certificate allowing K_{ws} to speak for Bob, then K_{ws} is allowed to speak for Bob
- Meaning of delegation certificate
  - If K_{ws} says it's speaking for Bob, believe it
  - This is different than "K_{ws} speaks for K_{bob}" (why?)
- File server takes "K_{bob} says ((K_{ws} | K_{bob}) speaks for (K_{ws} for K_{bob}))" and deduces, using delegation axiom, "(K_{ws} | K_{bob}) speaks for (K_{ws} for K_{bob})"

Proving Authenticity

- Combine

  $(K_{ws} \mid K_{bob})$ speaks for $(K_{ws}$ for $K_{bob})$ and

  $(K_{ws} \mid K_{bob})$ says $(C_{bob}$ speaks for $(K_{ws}$ for $K_{bob}))$

  to derive

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

  - Meaning: K_{ws} really does speak for K_{bob}, not just claiming to do so

- Conclusion: C_{bob} speaks for K_{ws} speaking for K_{bob}
- Therefore, $(K_{ws}$ for $K_{bob})$ says RQ