Computer Security in the Real World

Butler Lampson

Microsoft

Security: The Goal

Computers are as secure as real world systems, and people believe it.

This is hard because:

- Computers can do a lot of damage fast.
- There are many places for things to go wrong.
- Networks enable
  - » Anonymous attacks from anywhere
  - » Automated infection
  - » Hostile code and hostile hosts
- People don’t trust new things.

Real-World Security

It’s about value, locks, and punishment.

- Locks good enough that bad guys don’t break in very often.
- Police and courts good enough that bad guys that do break in get caught and punished often enough.
- Less interference with daily life than value of loss.

Security is expensive—buy only what you need.

Elements of Security

Policy: *Specifying* security
What is it supposed to do?

Mechanism: *Implementing* security
How does it do it?

Assurance: *Correctness* of security
Does it really work?

Dangers

Vandalism or sabotage that	
– damages information	*integrity*
– disrupts service	*availability*
Theft of money	*integrity*
Theft of information	*secrecy*
Loss of privacy	*secrecy*

Vulnerabilities

Bad (buggy or hostile) *programs*

Bad (careless or hostile) *people*
giving instructions to good programs

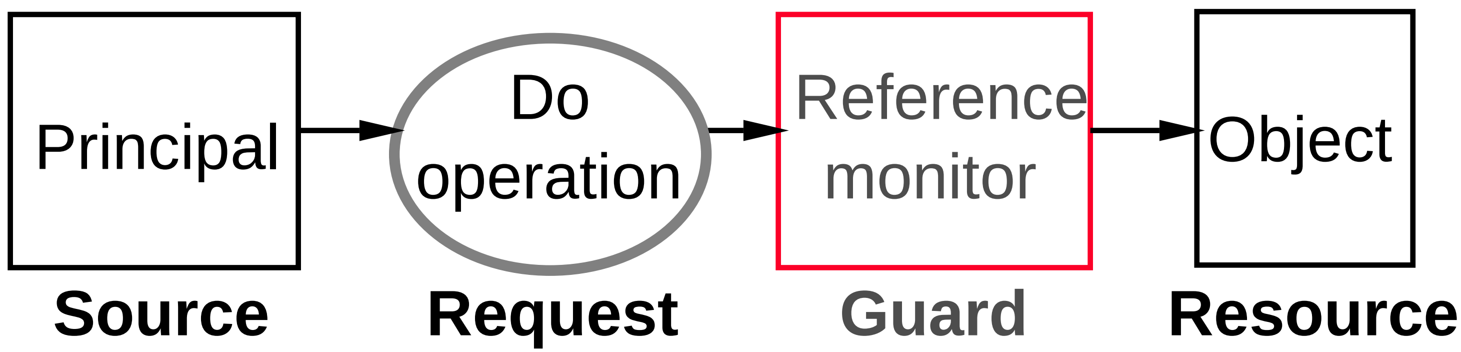
Bad guy interfering with *communications*

Defensive strategies

- Keep everybody out
  - Isolation
- Keep the bad guy out
  - Code signing, firewalls
- Let him in, but keep him from doing damage
  - Sandboxing, access control
- Catch him and prosecute him
  - Auditing, police

The Access Control Model

Guards control access to valued resources.



Mechanisms—The Gold Standard

Authenticating principals

– Mainly people, but also channels, servers, programs

Authorizing access.

– Usually for groups of principals

Auditing

Assurance

– Trusted computing base

Assurance: Making Security Work

Trusted computing base

- Limit what has to work to ensure security
  - » Ideally, TCB is small and simple
- Includes hardware and software
- Also includes configuration, usually overlooked
  - » What software has privileges
  - » Database of users, passwords, privileges, groups
  - » Network information (trusted hosts, …)
  - » Access controls on system resources
  - » . . .

The unavoidable price of reliability is simplicity.—Hoare

Assurance: Configuration

Users—keep it simple

– At most three levels: self, friends, others

» Three places to put objects

– Everything else done automatically with policies

Administrators—keep it simple

– Work by defining policies. Examples:

» Each user has a private home folder

» Each user belongs to one workgroup with a private folder

» System folders contain vendor-approved releases

» All executable programs are signed by a trusted party

Today’s systems don’t support this very well

Assurance: Defense in Depth

Network, with a firewall

Operating system, with sandboxing

- Basic OS (such as NT)
- Higher-level OS (such as Java)

Application that checks authorization directly

All need authentication

Why We Don’t Have “Real” Security

A. People don’t buy it:

- Danger is small, so it’s OK to buy features instead.
- Security is expensive.
  - Configuring security is a lot of work.
  - Secure systems do less because they’re older.
- Security is a pain.
  - It stops you from doing things.
  - Users have to authenticate themselves.

B. Systems are complicated, so they have bugs.

Standard Operating System Security

Assume secure channel from user (without proof)

Authenticate user by local password

– Assign local user and group SIDs

Access control by ACLs: lists of SIDs and permissions

– Reference monitor is the OS, or any RPC target

Domains: same, but authenticate by RPC to controller

Web servers: same, but *simplified*

– Establish secure channel with SSL
– Authenticate user by local password (or certificate)
– ACL on right to enter, or on user's private state

End-to-End Security

- Authenticate secure channels
- Work uniformly between organizations
  – Microsoft can securely accept Intel’s authentication
  – Groups can have members from different organizations
- Delegate authority to groups or systems
- Audit all security decisions

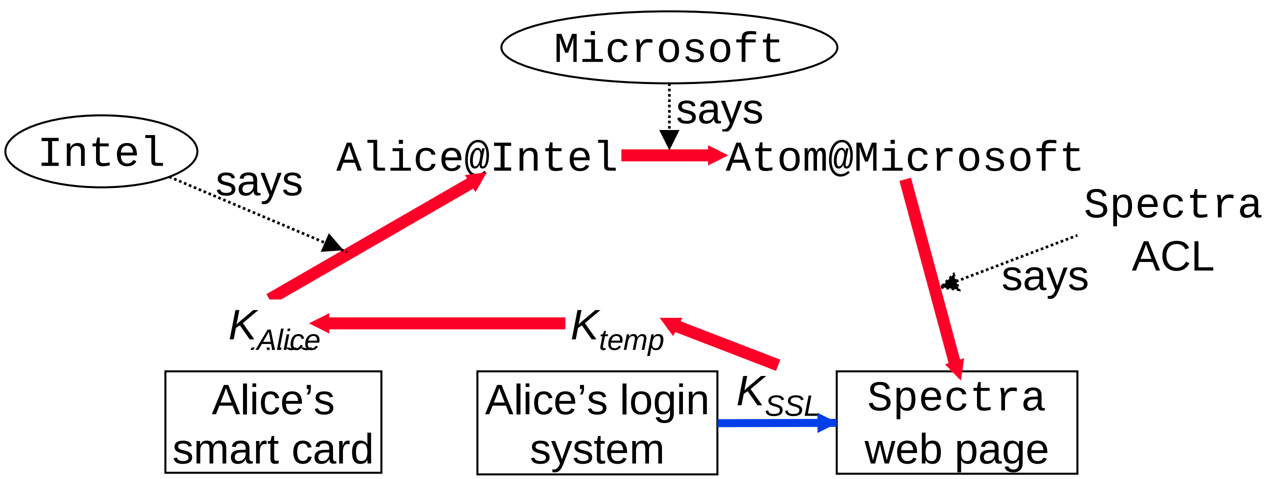
End-to-End example

`Alice` is at Intel, working on `Atom`, a joint Intel-Microsoft project

`Alice` connects to `Spectra`, Atom's web page, with SSL

Chain of responsibility:

– $K_{SSL} \Rightarrow K_{temp} \Rightarrow K_{Alice} \Rightarrow$ `Alice@Intel` $\Rightarrow$ `Atom@Microsoft` $\Rightarrow_{r/w}$ `Spectra`



Principals

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

Principal — abstraction of “who”:

– People	`Alice`, `Bob`
– Services	`microsoft.com`, `Exchange`
– Groups	`UW-CS`, `MS-Employees`
– Secure channels	key `#678532E89A7692F`, `console`

Principals say things:

- “Read file `foo`”
- “Alice’s key is `#678532E89A7692F`”

Speaks For

Principal *A* speaks for *B*: $A \Rightarrow_T B$

- Meaning: if *A* says something in set *T*, *B* says it too.
  - » Thus *A* is stronger than *B*, or responsible for *B*, about *T*
- Examples
  - » `Alice` ⇒ `Atom` *group of people*
  - » `Key #7438` ⇒ `Alice` *key for* `Alice`

Delegation rule: If *A* says "$B \Rightarrow A$" then $B \Rightarrow A$

- We trust *A* to delegate its own authority.
  - » Why should *A* delegate to *B*? Needs case by case analysis.
- Need a secure channel from *A* for "*A* says"
  - » Easy if *A* is a key.
  - » Channel can be off-line (certificate) or on-line (Kerberos)

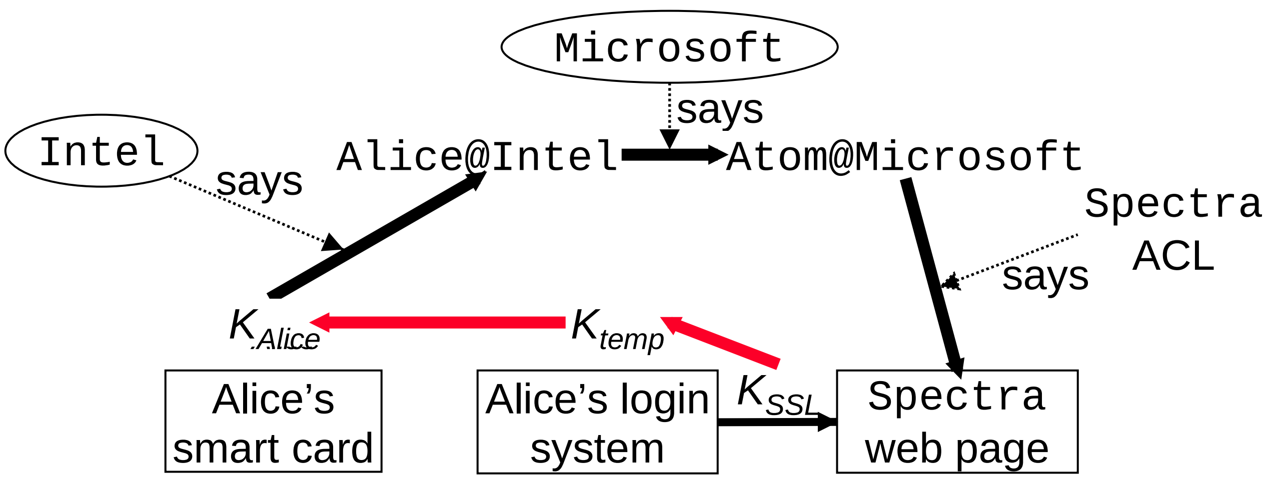
Authenticating Channels

Chain of responsibility:

$$K_{SSL} \Rightarrow K_{temp} \Rightarrow K_{Alice} \Rightarrow \texttt{Alice@Intel} \Rightarrow \ldots$$

K_{temp} says (SSL setup)

K_{Alice} says (via smart card)



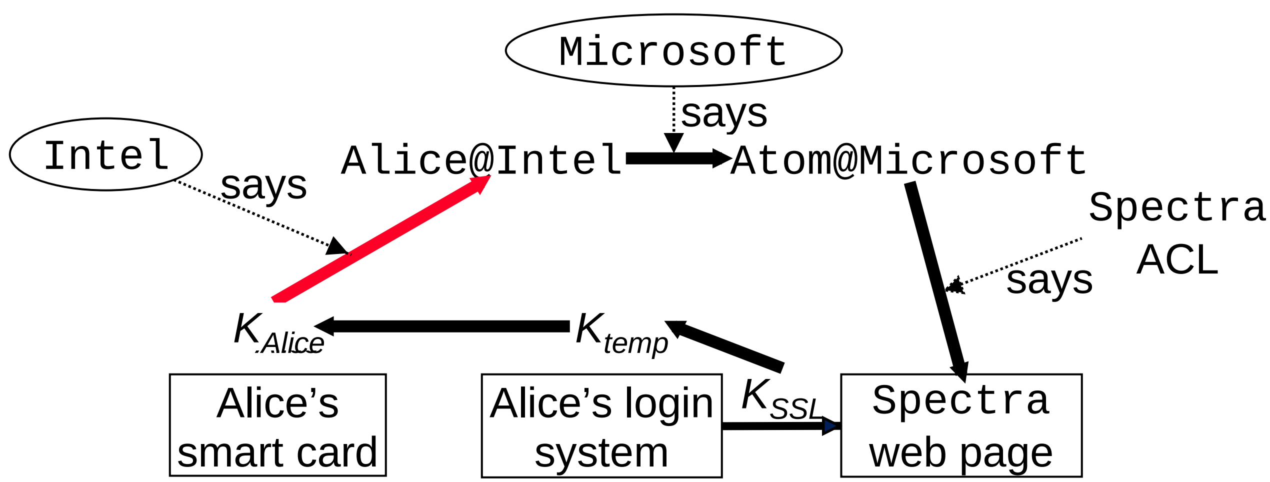
Authenticating Names: SDSI/SPKI

A name is in some name space, defined by a key

The key speaks for *any* name in its name space

- $K_{Intel} \Rightarrow K_{Intel}$ / `Alice` (which is just `Alice@Intel`)
- K_{Intel} **says**

$\ldots K_{temp} \Rightarrow K_{Alice} \Rightarrow$ `Alice@Intel` $\Rightarrow \ldots$



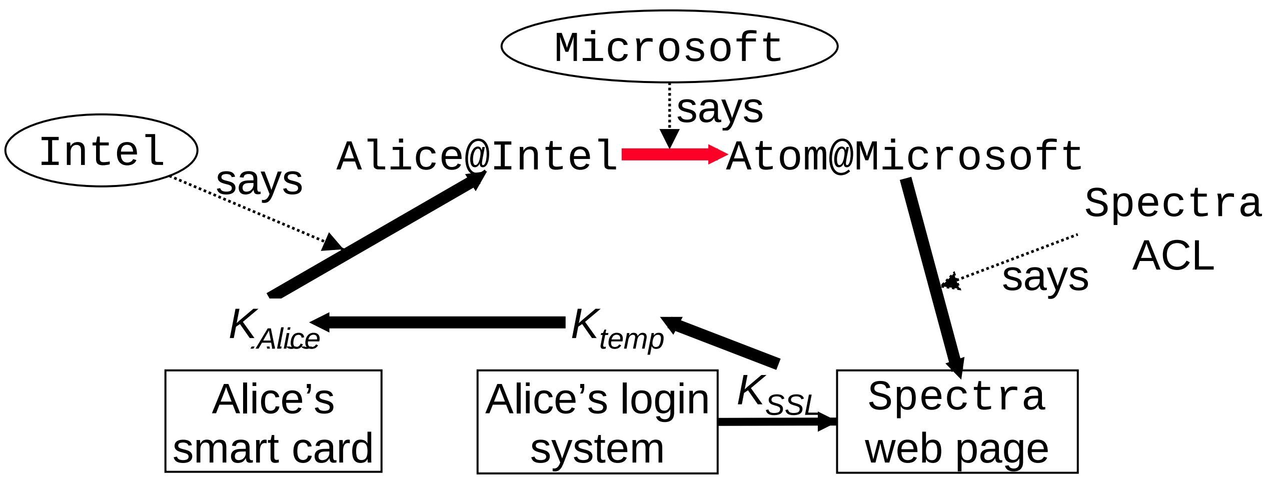
Authenticating Groups

A group is a principal; its members speak for it

- `Alice@Intel` $\Rightarrow$ `Atom@Microsoft`
- `Bob@Microsoft` $\Rightarrow$ `Atom@Microsoft`
- …

Evidence for groups: Just like names and keys.

… $K_{Alice} \Rightarrow$ `Alice@Intel` $\Rightarrow$ `Atom@Microsoft` $\Rightarrow_{\text{r/w}}$ …



Authorization with ACLs

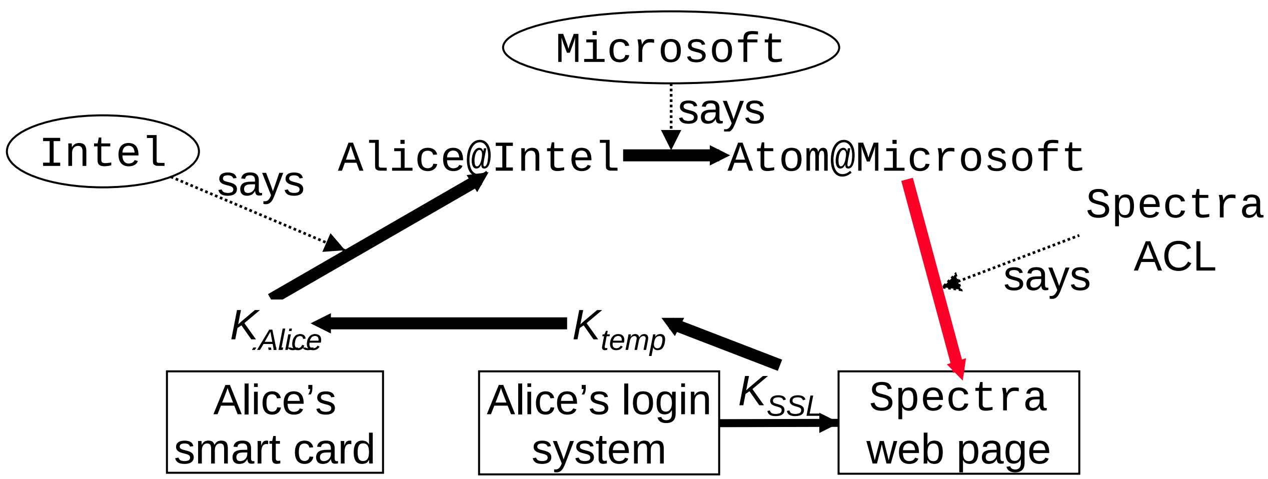
View a resource object *O* as a principal

An ACL entry for *P* means *P* can speak for *O*

- Permissions limit the set of things *P* can say for *O*

If `Spectra`'s ACL says `Atom` can `r/w`, that means `Spectra` says

`... Alice@Intel` $\Rightarrow$ `Atom@Microsoft` $\Rightarrow_{\text{r/w}}$ `Spectra`

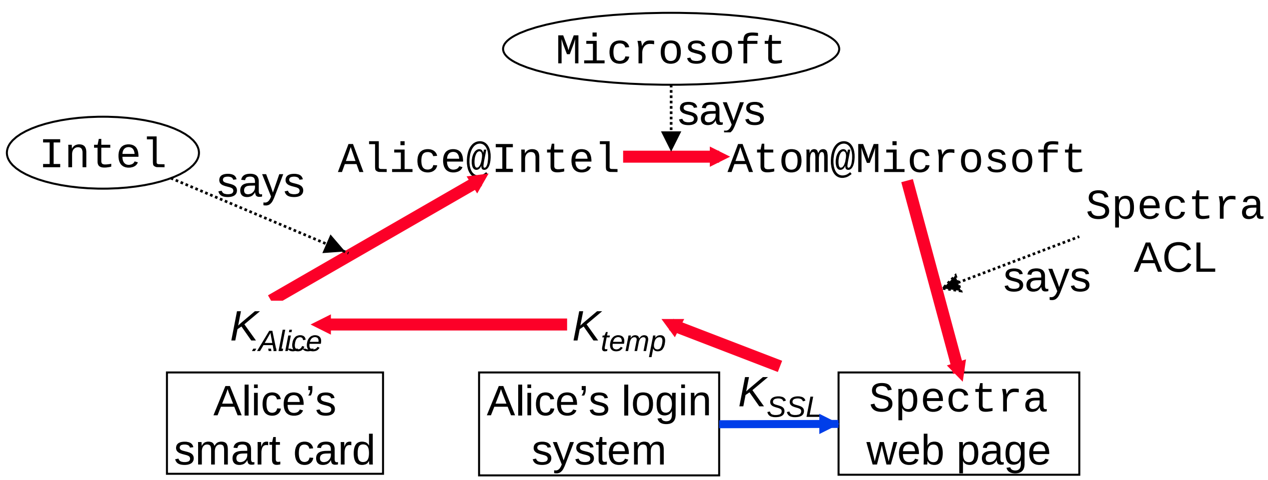


End-to-End Example: Summary

Request on SSL channel: K_{SSL} says "read `Spectra`"

Chain of responsibility:

$K_{SSL} \Rightarrow K_{temp} \Rightarrow K_{Alice} \Rightarrow$ `Alice@Intel` $\Rightarrow$ `Atom@Microsoft` $\Rightarrow_{\text{r/w}}$ `Spectra`



Compatibility with Local OS?

(1) Put network principals on OS ACLs

(2) Let network principal speak for local one

- `Alice@Intel` $\Rightarrow$ `Alice@microsoft`
- Use network authentication
  - » replacing local or domain authentication
- Users and ACLs stay the same

(3) Assign SIDs to network principals

- Do this automatically
- Use network authentication as before

Authenticating Systems

A digest X can authenticate a **program** `Word`:

- $K_{Microsoft}$ **says** "If image I has digest X then I is `Word`"
  formally $\boxed{X \Rightarrow K_{Microsoft} / \texttt{Word}}$

A **system** N can speak for another system `Word`:

- $K_{Microsoft}$ **says** $\boxed{N \Rightarrow K_{Microsoft} / \texttt{Word}}$

The first cert makes N want to run I if N likes `Word`, and it makes N assert the running I is Word

The second cert lets N convince others that N is authorized to run `Word`

Auditing

Checking access:

– Given	a request	K_{Alice} **says** “`read Spectra`”
	an ACL	`Atom` may `r/w Spectra`
– Check	K_{Alice} speaks for `Atom`	$K_{Alice} \Rightarrow$ `Atom`
	rights suffice	`r/w` $\geq$ `read`

Auditing: Each step is justified by

– A signed statement (certificate), or

– A delgation rule

Implement: Tools and Assurance

Gold standard

- *Authentication* Who said it?
- *Authorization* Who is trusted?
- *Auditing* What happened?

End-to-end authorization

- Principals: keys, names, groups
- Speaks for: delegation, chain of responsibility

Assurance: Trusted computing base

- Keep it small and simple.
- Include configuration, not just code.

References

Why "real" security is hard

- Ross Anderson: www.cl.cam.ac.uk/users/rja14
- Bruce Schneier, *Secrets and Lies*

Distributed system security

- Lampson et al. *TOCS* **10**, 4 (Nov. 1992)
- Wobber et al. *TOCS* **12**, 1 (Feb. 1994)

Simple Distributed Security Infrastructure (SDSI)

- theory.lcs.mit.edu/~cis/sdsi.html

Simple Public Key Infrastructure (SPKI)

- www.cis.ohio-state.edu/htbin/rfc/rfc2693.html