CS590U
Access Control: Theory and Practice

Lecture 20 (March 30)

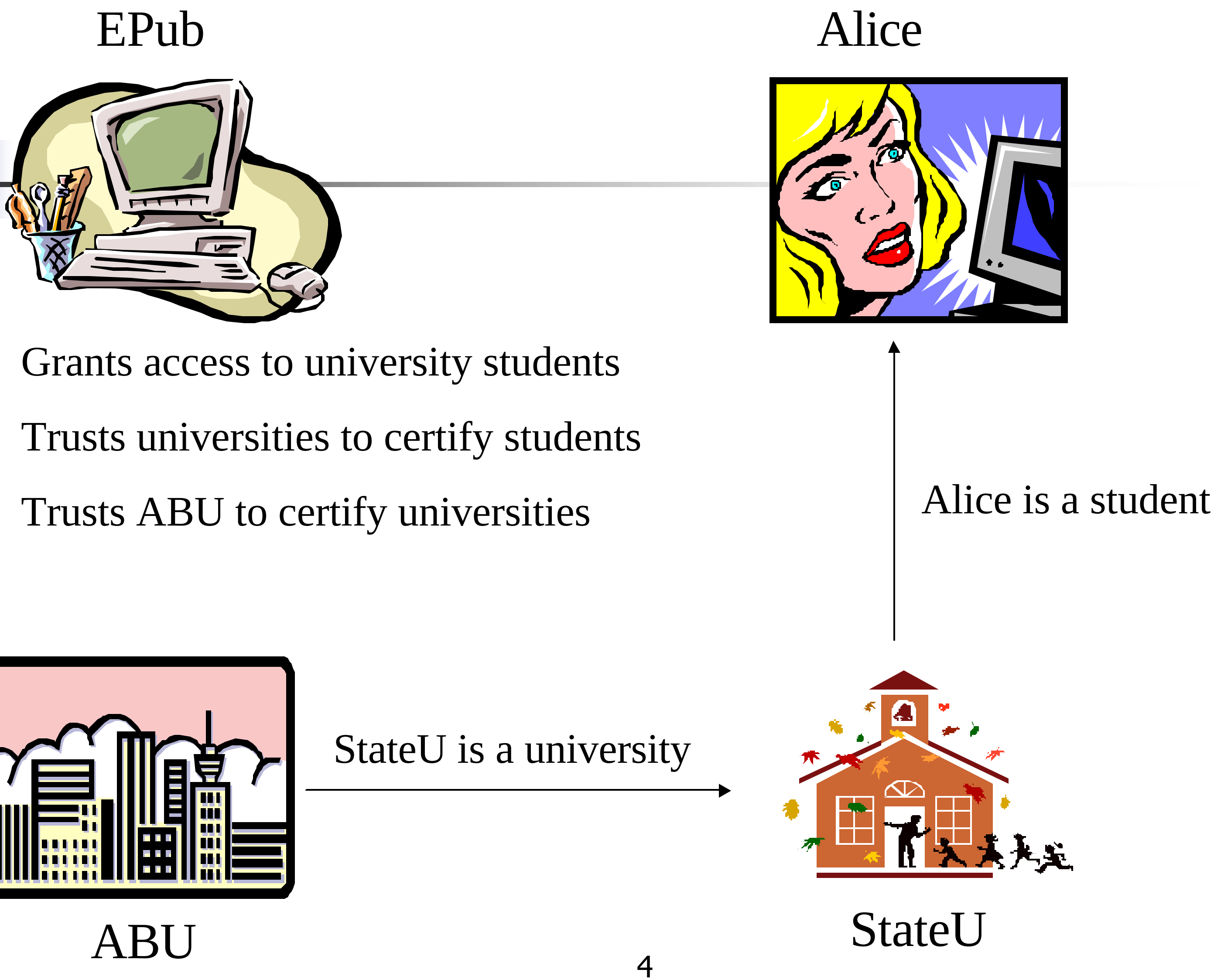
Overview of Trust Management

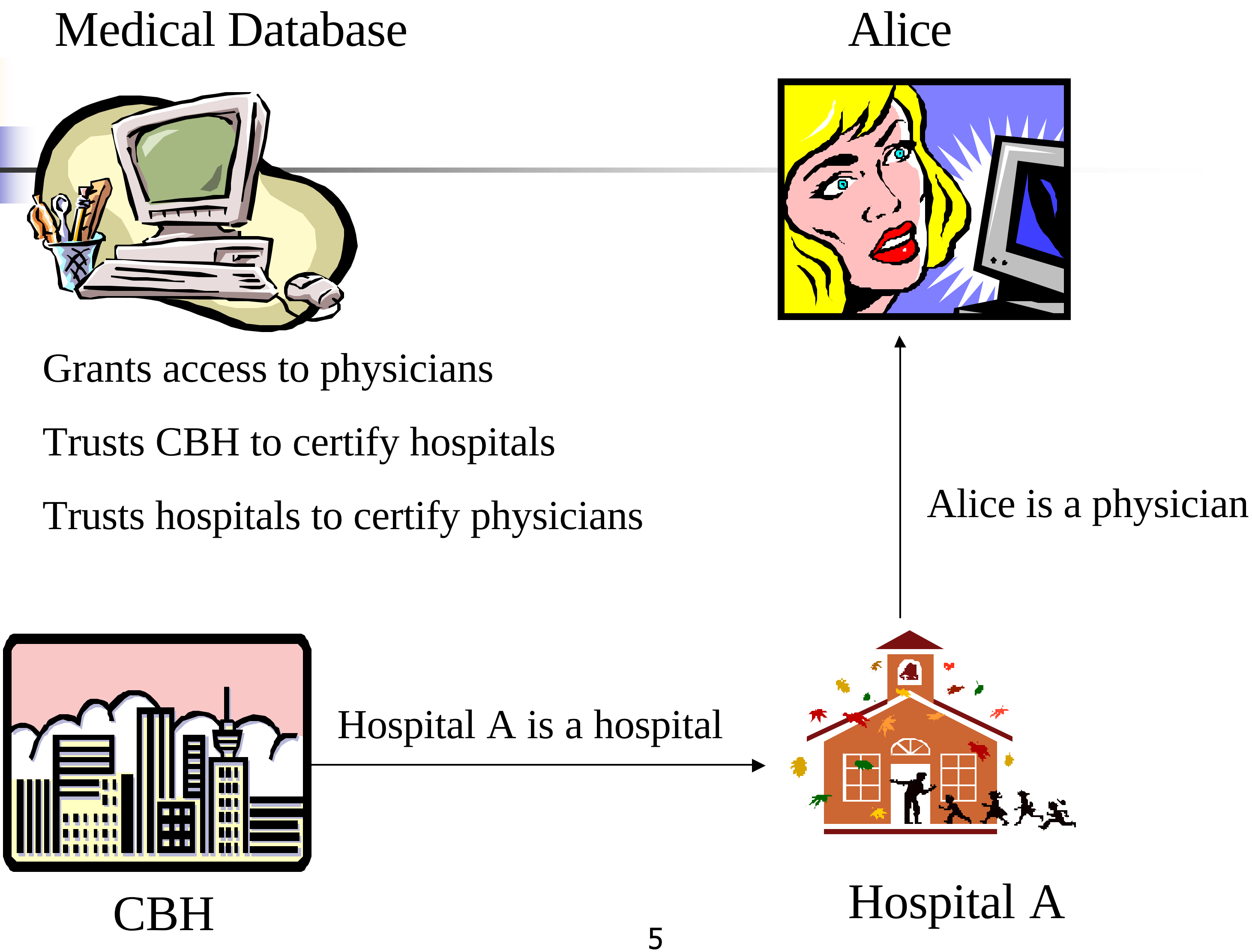
Distributed Authorization

- Flexible and scalable access control in large-scale, open, distributed, decentralized systems
  - electronic commerce:
    - transaction authorization
    - application-level / business-policy authorization
  - resource sharing in decentralized systems
    - coalitions, multi-centric collaborative systems
    - grid computing
  - health care
  - and so on

Characteristics of Distributed Authorization

- No central administration, each service makes its own decision
- No relationship between a service and a user prior to a request
  - knowing a user's name may not help
  - must rely on information from third-party to make authorization decision (delegation)
- Authorization information is distributed
- Communication channels may be insecure


EPub
Alice
Grants access to university students
Trusts universities to certify students
Trusts ABU to certify universities
Alice is a student
StateU is a university
ABU
StateU


Medical Database
Alice
Grants access to physicians
Trusts CBH to certify hospitals
Trusts hospitals to certify physicians
Alice is a physician
Hospital A is a hospital
CBH
Hospital A

The Trust-Management (TM) Approach

- Multicentric access control using delegation
  - access control decisions are based on distributed policy statements issued by multiple principals
  - policy statements contain
    - attributes of principals such as permissions, roles, qualifications, characteristics
    - trust relationships

Common characteristics of TM systems

- Use public-key certificates for non-local statements
- Treat public keys as principals to be authorized
  - authentication consists of verifying signatures
- Adopt a peer model
  - an entity can be an authorizer, a requester, or a credential provider (trusted 3rd party)
- Treat the authorization decision problem as an application-independent proof-of-compliance problem

Public-Key Certificates

- A certificate is a data record together with a digital signature
- A certificate is signed using K^{-1}
  - we say that it is issued by a public key K
- A certificate binds some information to another public key (the subject key)
- Can be verified by anyone who knows the issuer's public key
  - can one trust the issuer's public key?

Existing Kinds of Public Key Infrastructures (PKIs)

- X.509 certificates
  - certificates are **issued** (signed) by **certification authorities (CA's).**
  - **CA's may be arranged in a hierarchy**
  - certificates form a chain
  - used by numerous applications: SSL, IPSec, etc.
- PGP
  - everyone can issue certificates, which bind email addresses to public keys

Early Trust Management Langugaes

- PolicyMaker
  - Blaze, Feigenbaum & Lacy: "Decentralized Trust Management", S&P'96.
  - Blaze, Feigenbaum & Strauss: "Compliance-Checking in the PolicyMaker Trust Management System", FC'98.
- KeyNote
  - Blaze, Feigenbaum, Ioannidis & Keromytis: "The KeyNote Trust-Management System, Version 2", RFC 2714.
- SPKI (Simple Public Key Infrastructure) / SDSI (Simple Distributed Security Framework)
  - Rivest & Lampson: SDSI — A Simple Distributed Security Infrastructure, Web-page 1996.
  - Ellison et al.: SPKI Certificate Theory, RFC 2693.
  - Clarke et al.: Certificate Chain Discovery in SPKI/SDSI, JCS'01.

Datalog-based Trust Management Languages

- Delegation Logic
  - Li, Grosof & Feigenbaum: "Delegation Logic: A Logic-based Approach to Distributed Authorization", TISSEC'03. (Conference versions appeared in CSFW'99 and S&P'00)
- SD3 (Secure Dynamically Distributed Datalog)
  - Jim: "SD3: A Trust Management System with Certified Evaluation", S&P'01.
- Binder
  - DeTreville: "Binder, a Logic-Based Security Language", S&P'02.
- RT: A Family of Role-based Trust-management Languages
- PeerTrust

Other Closely Related Logic-based Security Languages

- ABLP logic (Abadi, Burrows, Lampson, et al.)
  - Lampson et al.: "Authentication in Distributed Systems: Theory and Practice", TOCS'92.
  - Abadi et al.: "A Calculus for Access Control in Distributed Systems", TOPLAS'93.
- QCM (Query Certificate Managers)
  - Gunter & Jim: "Policy-directed Certificate Retrieval", SPE'00
- AF logic
  - Appel & Felton: "Proof-Carrying Authentication", CCS'99

Issues in Designing Trust Management Languages

- Say what you want
  - succinctly and directly
  - with confidence that you said what you meant
- Enforcement
  - deduction, proof of compliance
- Policy development tools
  - manage policy lifecycle
  - analysis of safety, availability, and other security properties

Decentralized Trust Management

Matt Blaze, Joan Feigenbaum, Jack Lacy

Oakland'1996

The PolicyMaker Language

- A query has the form
  - $K_1, K_2, \cdots, K_n$ REUESTS ActionString
- Policies & credentials are encoded as assertions of the form
  - Source ASSERTS AuthorityStruct WHERE Filter
  - Source is either a public key or the keyword LOCAL
  - AuthorityStruct is a key, a list of keys, or a k-out-of-n threshold structure
  - Filter is a program that can be safety interpreted, it may be
    - a predicate, that returns yes/no
    - an annotator, returns yes/no and add to ActionString

Certificate chain discovery in **SPKI/SDSI**

Clarke et al.

JCS 2001

History of SPKI/SDSI

- SDSI (Simple Distributed Security Infrastructure)
  - SDSI 1.0 and 1.1
  - Rivest & Lampson 96
- SPKI (Simple Public Key Infrastructure)
  - SPKI 1.0 (Ellison 1996)
- SPKI/SDSI 2.0
  - RFC 2693 [1999]
  - [Clarke et al. JCS'01]

An Example in SDSI 2.0

- SDSI Certificates
  - (K_C access ⇨ K_C mit faculty secretary)
  - (K_C mit ⇨ K_M)
  - (K_M faculty ⇨ K_{EECS} faculty)
  - (K_{EECS} faculty ⇨ K_{Rivest})
  - (K_{Rivest} secretary ⇨ K_{Rivest} alice)
  - (K_{Rivest} alice ⇨ K_{Alice})
- From the above certificates, K_C concludes that K_{Alice} has access

4-tuple Reduction in RFC 2693

- Name strings can be reduced using 4-tuples
  - $(K_1\ A_1 \Rightarrow K_2)$ reduces "$K_1\ A_1\ A_2 \ldots A_n$"
    to "$K_2\ A_2 \ldots A_n$"
    - e.g., (K_C mit $\Rightarrow$ K_M) reduces "K_C mit faculty secretary" to "K_M faculty secretary"
  - $(K_1\ A_1 \Rightarrow K_2\ B_1 \ldots B_m)$
    reduces "$K_1\ A_1\ A_2 \ldots A_n$"
    to "$K_2\ B_1 \ldots B_m\ A_2 \ldots A_n$"
    - e.g., (K_M faculty $\Rightarrow$ K_{EECS} faculty) reduces "K_M faculty secretary" to "K_{EECS} faculty secretary"

Applying 4-tuple Reduction in the Example

- From (K_C access)
  to (K_C mit faculty secretary)
  to (K_M faculty secretary)
  to (K_{EECS} faculty secretary)
  to (K_{Rivest} secretary)
  to (K_{Rivest} alice)
  to (K_{Alice})

(K_C access ⇨ K_C mit faculty secretary) (K_C mit ⇨ K_M)
(K_M faculty ⇨ K_{EECS} faculty) (K_{EECS} faculty ⇨ K_{Rivest})
(K_{Rivest} secretary ⇨ K_{Rivest} alice) (K_{Rivest} alice ⇨ K_{Alice})

Papers on Semantics for SPKI/SDSI

- Develop specialized modal logics
  - Abadi: "On SDSI's Linked Local Name Spaces", CSFW'97, JCS'98.
  - Halpern & van der Meyden:
    - "A logic for SDSI's linked local name spaces", CSFW'99, JCS'01
    - "A Logical Reconstruction of SPKI", CSFW'01, JCS'03
  - Howell & Kotz: "A Formal Semantics for SPKI", ESORICS'00
- Other approaches
  - Li: "Local Names in SPKI/SDSI", CSFW'00
  - Jha & Reps: "Analysis of SPKI/SDSI Certificates Using Model Checking", CSFW'02
  - Li & Mitchell: "Understanding SPKI/SDSI Using First-Order Logic", CSFW'03, IJIS'2005

Concepts in SDSI

- Concepts
  - principals — K, K_1
  - identifiers — A, B, A_1
    e.g., mit, faculty, alice
  - local names — $K\ A$, $K_1\ A_1$
    e.g., K_M faculty, K_{Rivest} alice
  - name strings — $K\ A_1\ A_2 \ldots A_n$
    ω, ω_1
    e.g., K_C mit faculty secretary

Statements in SDSI

- 4-tuple (K, A, ω, V)
  - K is the issuer principal
  - A is an identifier
  - ω is a name string
  - V is the validity specification
- We write (K A ⇨ ω) for a 4-tuple
  - ignoring validity specification

A Rewriting Semantics for SDSI

- A set P of 4-tuples defines a set of rewriting rules, denoted by RS[P]
- Queries have the form "can ω_1 rewrite into ω_2?"
- Answer a query is not easy.
  - cannot naively search for all ways of rewriting ω_1, as there may be recursions
    - e.g., (K friend ⇨ K friend friend)
- What can we do?

Deduction Based on the Rewriting Semantics (1)

- Limit queries to the form "can ω_1 rewrite into K?"
  - In [Clarke et al.'01], the following closure mechanism is used
    - rewrite 4-tuples
      - e.g., apply (K_C mit ⇨ K_M)
        to rewrite (K_C access ⇨ K_C mit faculty secretary), one gets
        (K_C access ⇨ K_M faculty secretary)
    - compute the closure of a set of 4-tuples,
      - obtained by applying 4-tuples that rewrites to a principal
    - then use the resulting shortening 4-tuples to rewrite ω_1
  - Search is not goal-directed

Deduction Based on the Rewriting Semantics (2)

- Limit to queries like "can ω_1 rewrite into K?"
  - In [Li CSFW'00], the following XSB logic program is given

```
:- table(contains/2).
contains([P0, N0 | T], P2) :-
        contains([P0, N0], P1),
        contains([P1 | T], P2).
contains([P0, N0], P) :-
        credential([P0, N0], CN2),
        contains(CN2, P).
contains([P], P) :- isPrincipal(P).
```

Deduction Based on the Rewriting Semantics (3)

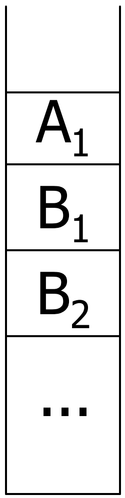
- [Li, Winsborough & Mitchell, JCS'03]
  - develop a graph-based search algorithm for a language RT_0, a superset of SDSI
    - combines bottom-up search and goal-directed top-down search with tabling specifically for the kind of rules in RT_0
    - can deal with distributed discovery
  - we will talk about this later

Deduction Based on the Rewriting Semantics (4)

- Use techniques for model checking pushdown systems [Jha & Reps CSFW'02]
  - SDSI rewriting systems correspond to string rewriting systems modeled by pushdown systems
  - algorithms for model checking pushdown systems can be used
    - takes time O(N^3), where N is the total size of the SDSI statements

SDSI and Pushdown Systems

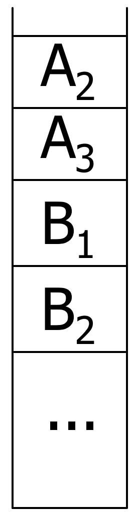
Stack:



State: K_1

Apply the rewriting rule:
K_1 A_1 to K_2 A_2 A_3

Stack:



State: K_2

A name string corresponds to a configuration

“rewrites into” equivalent to “reaches”

Recap of the Rewriting-based Semantics

- Defines answers to queries having the form "can ω_1 rewrite into ω_2?"
- Specialized algorithms (either developed for SDSI or for model checking pushdown systems) are needed
- Papers by Abadi and Halpern and van der Meyden try to come up with axiom systems for the rewriting semantics

Next Lecture

- Distributed Credential Chain Discovery in RT0