

Trust Management

Scott D. Stoller

CSE608: Advanced Computer Security



Outline

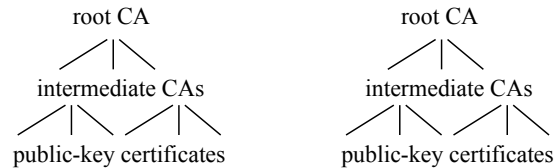
- Intro to Trust Management
- Trust Management for Relational Databases
- Rule-Based Trust Management

Public-Key Infrastructure (PKI)

- **Public-key certificate**: a certificate signed by a **certification authority (CA)**, containing a public key **K** and a principal's name **N**, and stating that **K** belongs to **N**.
- **Example**: [public key 1a4deb6c5c belongs to AMA] signed by VeriSign
- Who is trusted to **issue** public-key certificates?

Public-Key Infrastructure: X.509

- X.509 has a **hierarchical trust model**
- Trust is rooted at **root CAs**.
 - ◆ A list of them is embedded in your web browser.
- A CA can also issue certificates certifying other entities as CAs. This creates a **forest** of trust relationships.



Public-Key Infrastructure: PGP

- **Pretty Good Privacy (PGP)** is based on a **web of trust**.
- **Everyone** may issue public-key certificates.
- Each user specifies a **level of trust** in each issuer.
- Each user specifies the **total confidence** needed for a public-key↔name relationship to be considered **valid**.
 - ◆ **Example**: one certificate from an issuer trusted at level 10, or certificates from two issuers trusted at level 5 or higher.
- A user can also build confidence in such a relationship through successful communication using a particular public key.

Public-Key Infrastructure: X.509 vs. PGP

- X.509's hierarchical trust is appropriate for **business, government, etc.**
 - ◆ Accountability
 - ◆ Structure: known sources for certificates, revocation lists, etc.
- PGP's web of trust may be appropriate for **personal communication**.
 - ◆ Individuals will not spend time and money to get VeriSign certificates (\$995 for their cheapest 3-year certificate)
 - ◆ Currently, personal communication is authenticated mainly through non-technical means.

Essential features of policy frameworks for decentralized systems: attributes and relations

- Policy can use application-specific **attributes and relations**.
- **Example:** Nurses in the workgroup treating a patient can access the patient's medical record.
 - ◆ **Attributes:** isNurse(employee)
 - ◆ **Relations:** treatedBy(patient, workgroup).
- Encoding such policies as **identity-based** policies is
 - ◆ **impractical:** potential users are not known to resource owners in advance
 - ◆ **dangerous:** attributes can change
- In traditional RBAC, users are added to roles based on **identity**, not **attributes**.

September 2008

Scott Stoller, Stony Brook University

7

Essential features of policy frameworks for decentralized systems: attributes and relations

- Attributes and relations can be **defined** in terms of other attributes and relations.
 - ◆ **Example:** Nurses in the workgroup treating a patient can access the patient's medical record. A nurse is in the workgroup if a manager assigned the nurse to it.
- This allows **interactions** that are essential in decentralized systems.
- **Standard RBAC** does **not** support this. Each role is defined **independently** (aside from inheritance).
 - ◆ **Example:** RBAC does not support policies of the form $\text{role1.members} = \text{role2.members} \cap \text{role3.members}$

September 2008

Scott Stoller, Stony Brook University

8

Essential features of policy frameworks for decentralized systems: delegation

- Policy administration is **completely decentralized** at the top level. No globally trusted administrators. No root of trust.
- Policies interact through **delegation of authority**: rely on others for attribute values, access control decisions, etc.
- **Example:** Hospitals, doctor's offices, insurance companies, and government agencies share information (medical, financial, and personnel records) and have limited trust.
- **Example:** Student discount at research conference. Conference trusts Dept of Education (U.S., etc.) about universities. It trusts each university about enrollment.

September 2008

Scott Stoller, Stony Brook University

9

Delegation of Permissions

- **Delegation of permissions:** a principal with a permission can grant it to other principals (under some conditions).
 - ◆ **Example:** President of bank delegates permission to approve loans to VP while the President is on vacation.
 - ◆ **Example:** A gives B access to file. B gives access to C.
- **Delegation of authority** is more general.
 - ◆ Trust for information as well as authorization decisions.
 - ◆ Principals can delegate permissions they don't have.
 - **Example:** Office of Grants Management can grant faculty permission to approve expenditures from an account. OGM cannot approve expenditures itself.

September 2008

Scott Stoller, Stony Brook University

10

Essential Features of Trust Management

- Each policy statement is associated with a principal, called its **source** or **issuer**.
- Each principal's policy specifies which **sources it trusts** for which kinds of statements, thereby **delegating** some authority to those sources.
- Policies may refer to domain-specific **attributes** and **relationships** between principals, resources, and other objects.
- **Example:** Acme Hospital says "doc can access pat's medical record if AMA says doc is a licensed doctor and pat says doc is treating him." (patient consent)

September 2008

Scott Stoller, Stony Brook University

11

Outline

- Intro to Trust Management
- **Trust Management for Relational Databases**
- Rule-Based Trust Management

September 2008

Scott Stoller, Stony Brook University

12

Trust Management for Relational Databases

- **Current DB security**
 - ◆ User authentication: passwords or X.509 PKI
 - ◆ Identity-based assignment of users to roles.
 - ◆ Trust management is done in applications.
- **Goal:** a trust management system that integrates easily into existing DBs.
- **Benefits:** re-use across applications, complete mediation
- Scott D. Stoller. Trust Management and Trust Negotiation in an Extension of SQL. In *Proceedings of the 4th International Symposium on Trustworthy Global Computing (TGC)*. Springer-Verlag, 2008 (to appear).

September 2008

Scott Stoller, Stony Brook University

13

Base Concepts

- All certificates implicitly include the issuer's name and signature.
- **Public-key certificate**
 - ◆ Subject's name, subject's public key
- **Attribute certificate**
 - ◆ Subject's name (or public key), attribute names and values
 - ◆ **Example:** subject=Dr.Dan, physician=true, issuer=NHS
 - ◆ **Example:** subject=Dr.Dan, relation=consentToTreatment, patient=Pat, issuer=Pat

September 2008

Scott Stoller, Stony Brook University

14

Policy Language: create certtable

- Extend SQL with statements that define conditions under which a user can activate a role.
- The conditions use info obtained from public-key certificates and attribute certificates and stored in special tables.
- `create [per-user | shared] certtable name (column_def*) check (issuer_constraint [&& constraint]?)`
- `column_def ::= name type`
- `issuer_constraint ::= issuer in (select subject from ctv*) | issuer is pkfile`
- *ctv* is the name of a certtable or a view of certtable(s).

September 2008

Scott Stoller, Stony Brook University

15

create certtable: Example

- `create shared certtable Initial-Spine-admin (relation varchar equals 'register', role varchar equals 'Spine-admin') check (issuer is 'NHS.crt')`
- `create shared certtable Spine-admin (relation varchar equals 'register', role varchar equals 'Spine-admin') check (issuer in (select subject from Initial-Spine-admin, Spine-admin))`

September 2008

Scott Stoller, Stony Brook University

16

Certtables: Implicit Columns

- Every certtable implicitly contains the following columns:
 - ◆ `subject principal_type`
 - ◆ `issuer principal_type`
 - ◆ `expiration datetime`
- `principal_type` denotes the SQL data type used for identities of principals.
 - ◆ Example: `varchar(40)` for hex encodings of public keys
 - ◆ Example: `varchar(256)` for Distinguished Names

September 2008

Scott Stoller, Stony Brook University

17

Policy Language: insert_certificate

- `insert certificate [into ct] cert`
- Insertion of attribute certificate *c* into certtable *ct* succeeds if:
 1. *c* contains all attributes defined in *ct*
 2. attribute values in *c* satisfy the *constraint* in *ct*'s *check* clause
 3. *c*'s issuer satisfies the *issuer_constraint* in *ct*'s *check* clause
 4. the user has insert privilege for *ct*
- If “*into ct*” is omitted, try to insert *c* into all certtables.

September 2008

Scott Stoller, Stony Brook University

18

Policy Language: delete_certificate, tm_user()

- delete certificate from *ct* where *condition*
- Delete certificates satisfying *condition* from *ct*
- Succeeds if the user has delete privilege for *ct*

- `tm_user()` is a stored function that returns the user's trust management username (Distinguished Name or public key, depending on the design).
 - ◆ SQL's `user()` function returns the user's database username

Certtable: Example

- create per-user certtable Consent
(relation varchar(40) equals 'consentToTreatment',
patient principal_type)
check (issuer in (select subject from Patient))
 - ◆ Recall: subject of consent cert is the doctor.
 - ◆ Should also check: issuer is the patient or his/her agent.
- # contains health recs for patients treated by dr. using DB.
create view HealthRec-MyPatients as
select * from HealthRec
where HealthRec.patient in (select patient from Consent
where subject=tm_user())

Policy Language: ab_grant

- Attribute-based grant statements automatically grant privileges to users, based on their attributes.
 - ◆ Standard SQL grant statement:
grant *privilege* to [*user* | *role*] *grant_options*
- `ab_grant privilege` to (select *column* from *ctv*)
grant_options name name
- Grant *privilege* to the users whose identities are returned by the `select` statement.
- Privileges are **continually** granted and revoked as the result of the select statement changes, until the `ab_grant` is cancelled.

ab_grant: Example

- # A patient can annotate their own record items (S5.1.2)
create view ehr_patient_add_view as
select ehr.* from ehr
where ehr.patient = tm_user()
- `ab_grant update` (annotation) on `ehr_patient_add_view`
to (select subject from patient-activation)
name patient-annotate-policy

ab_grant: Example (continued)

- create per-user certtable patient-activation
(subject principal_type equals issuer,
activated_role varchar equals 'patient')
check (issuer in (select subject from Register-patient))

- # subjects are users registered as patients ...
create view Register-patient as ...

- "equals" clause is shorthand for part of the constraint in the "check" clause

Policy Language: ab_revoke

- `ab_revoke name`
- Cancel the named `ab_grant` statement, and revoke permissions already granted by it.
- Example: `ab_revoke patient-annotate-policy`

Summary of Policy Language

- create [per-user | shared] certtable *name*
(*column_def**) check (*issuer_constraint* [&& *constraint*]?)
- *column_def* ::= *name type*
- *issuer_constraint* ::= issuer in (select subject from *ctv**)
| issuer is *pkfile*
- insert certificate [into *ct*] *cert*
- delete certificate from *ct* where *condition*
- ab_grant *privilege* to (select *column* from *ctv*)
grant_options name name
- ab_revoke *name*

September 2008

Scott Stoller, Stony Brook University

25

Let's Try It!

1. NHS is trusted to declare that a principal is a physician.
 2. A physician is trusted to declare that a principal is an agent for a patient.
 3. A patient's agent can view the patient's HealthRec.
- create table HealthRec (patient, ...)

September 2008

Scott Stoller, Stony Brook University

26

A Solution

- create shared certtable Physician
(relation varchar equals 'register', role varchar equals 'Physician')
check (issuer is 'NHS.crt')
- create shared certtable Agent
(relation varchar equals 'agent', patient principal_type)
check (issuer in (select subject from Physician))
- create view HealthRec-AgentView as
select HealthRec.* from HealthRec
where HealthRec.patient in
(select patient from Agent where subject=tm_user())

September 2008

Scott Stoller, Stony Brook University

27

A Solution (continued)

- ab_grant select on HealthRec-AgentView to
(select subject from Agent)
- Let's Try A Variation!
- A patient is trusted to declare that another principal is his/her agent.

September 2008

Scott Stoller, Stony Brook University

28

A Solution for the Variation

- create shared certtable Agent
(relation varchar(40) equals 'agent', patient principal_type)
check (issuer in (select subject from Patient) &&
issuer=patient)
- Other statements are unchanged.

September 2008

Scott Stoller, Stony Brook University

29

Let's Try Another Example!

- SUNY Central Administration is trusted to identify the SUNY campuses.
- SUNY says X is a SUNY employee if a SUNY campus says X is an employee of that campus.
- SUNY says its employees can read the campus directory.

September 2008

Scott Stoller, Stony Brook University

30

A Solution

- create certtable Campus (relation equals 'SUNY_campus')
check (issuer is 'SUNYCentralAdmin.crt')
- create certtable Employee
(relation equals 'register', role equals 'Employee')
check (issuer in (select subject from Campus))
- ab_grant select on SUNYDirectory to
(select subject from Employee)

September 2008

Scott Stoller, Stony Brook University

31

Credential Discovery and request_privilege

- Current design assumes that the **user supplies** all necessary **certificates** using **insert_credential**
- The user needs to understand the **resource owner's policy** (ab_grants, views, certtables, etc.) to do this.
- This is **annoying**, because the user cares about **privileges**, not policies and certificates.
- Introduce a new policy statement:
- **request_privilege privilege**
- Trust manager attempts to provide the user with the specified privilege, by fetching certificates as needed.

September 2008

Scott Stoller, Stony Brook University

32

Policy Language: fetch_from

- Extend the policy language to support credential discovery.
- **create** [per-user | shared] certtable *name*
[*fetch from location*⁺?]
(*column_def*^{*}) check (*issuer_constraint* [&& *constraint*][?])
- *location* ::= issuer | subject | user
- **Example:** create shared certtable PDS-Register-patient
fetch from issuer
(relation varchar equals 'register',
role varchar equals 'patient')
check (issuer is 'PDS.crt')

September 2008

Scott Stoller, Stony Brook University

33

Trust Manager API: get_certificates

- **get_certificates**(*column*, *val*, *ct_def*) returns a set of certificates to the requester *r*. The requester wants certificates *c* that (1) can be inserted in a certtable defined by *ct_def* and (2) satisfy *c.column=val*.
- for each certtable *t* that contains all the columns in *ct_def*
for each certificate *c* in *t*
if (*c.column* == *val* &&
c satisfies the constraint in the **check** clause in *ct_def*)
include *c* in the return value
- What if *c* contains sensitive information?

September 2008

Scott Stoller, Stony Brook University

34

Trust Negotiation: Gradual Release of Sensitive Credentials

- Credentials may contain **sensitive** information.
- **Example** [Bhargava+ 2004, Winsborough+ 2004]: On-Line University (OLU) gives a discount to veterans. Joe reveals his veteran status only to IRS-certified non-profits.
- Joe → OLU: register(CS101)
- OLU → Joe: requestCredential: VA.veteran(Joe)
- Joe → OLU: requestCredential: IRS.nonProfit(OLU)
- OLU → Joe: IRS.nonProfit(OLU)
- Joe → OLU: VA.veteran(Joe) // OLU: give discount
- OLU → Joe: requestPayment: \$1000
- Joe → OLU: Citigroup.creditCard(Joe,1234-5678-9012)

September 2008

Scott Stoller, Stony Brook University

35

Trust Negotiation: Privacy Policy for Credentials

- Suppose we associate privacy policies with **credentials**.
- **Example** [Winsborough+ 2004]:
- Joe reveals his low-income credential only to non-profits.
- Joe → E-Realty: request house listings
- E-Realty → Joe: requestCredential: IRS.lowIncome(Joe)
- Joe → E-Realty: requestCred.: IRS.nonProfit(E-Realty)
- E-Realty is not non-profit. Rich is not low-income.
- Rich → E-Realty: request house listings
- E-Realty → Rich: requestCred.: IRS.lowIncome(Rich)
- Rich → E-Realty: nothing
- E-Realty infers (no proof) Joe is low-income, Rich isn't.

September 2008

Scott Stoller, Stony Brook University

36

Trust Negotiation: Privacy Policy for Attributes

- Associate a privacy policy with each **attribute**, regardless of whether user has that attribute or a credential for it.
- **Joe** → **E-Realty**: request house listings
- **E-Realty** → **Joe**: requestCredential: IRS.lowIncome(Joe)
- **Joe** → **E-Realty**: requestCred.: IRS.nonProfit(E-Realty)
- Rich is not low-income but has the same privacy policy.
- **Rich** → **E-Realty**: request house listings
- **E-Realty** → **Rich**: requestCred.: IRS.lowIncome(Rich)
- **Rich** → **E-Realty**: requestCred.: IRS.nonProfit(E-Realty)
- E-Realty learns nothing.

September 2008

Scott Stoller, Stony Brook University

37

Where to Get Privacy Policies for Attributes?

- **Where** does Rich get the privacy policy for IRS.lowIncome? Perhaps Rich never heard of IRS.lowIncome before E-Realty asked about it.
- **Issuers** should provide **standard privacy policies** for attributes, at well-known locations [Winsborough+ 2004].
- **Example**: When Rich sees request for IRS.lowIncome credential, he contacts IRS policy server and obtains and uses the IRS-recommended privacy policy for attribute IRS.lowIncome.
- If Rich doesn't bother to do this, then other people probably won't do it for other attributes, which Rich might want to keep private.

September 2008

Scott Stoller, Stony Brook University

38

Trust Negotiation Strategies

- Trust negotiation **policy** determines when a credential **may** be released.
- Trust negotiation **strategy** determines which releasable credentials **are** released.
- **Eager Strategy**: at each step, send all releasable credentials.
- **Targeted Strategy**: at each step, send releasable credentials that help achieve the current goal.

September 2008

Scott Stoller, Stony Brook University

39

Trust Negotiation Strategies: Example

- **Eager Strategy (fewer rounds of communication)**:
- **Joe** → **NP-Realty**: request house listings
- **NP-Realty** → **Joe**: requestCred.: IRS.lowIncome(Joe); IRS.nonProfit(NP-Realty), BBB.member(NP-Realty), ...
- **Joe** → **NP-Realty**: IRS.lowIncome(Joe)
- **Targeted Strategy (fewer credentials sent)**:
- **Joe** → **NP-Realty**: request house listings
- **NP-Realty** → **Joe**: requestCred.: IRS.lowIncome(Joe)
- **Joe** → **NP-Realty**: requestCred.: IRS.nonProfit(NP-Realty)
- **NP-Realty** → **Joe**: IRS.nonProfit(NP-Realty)
- **Joe** → **NP-Realty**: IRS.lowIncome(Joe)

September 2008

Scott Stoller, Stony Brook University

40

Trust Negotiation: Hidden Credentials

- The **hidden credentials** framework [Holt+ 2003, Frikken+ 2006] for trust negotiation provides **greater privacy**.
- The server does not learn anything about client's credentials, and the client does not learn anything about the server's access control policy, except what each can infer from the outcome of the negotiation (success or failure).
- Privacy policies for attributes are unnecessary in this framework, because credentials are **never revealed**.
- **Idea**: Server uses **identity-based encryption** to encrypt responses so that the client can decrypt them only if the client possesses appropriate credentials.

September 2008

Scott Stoller, Stony Brook University

41

Policy Language: release_to

- End general discussion of trust negotiation.
- Return to SQL-based trust management framework.
- Extend the policy language to support trust negotiation.
- **create** [per-user | shared] certtable *name* [fetch from *location*⁺?] [release to *release_target*]* (*column_def**) check (*issuer_constraint* [&& *constraint*]?)
- *release_target* ::= public | *pkfile* | *ctv* [for same *column*]?
 - ◆ “*ctv*” means “release to *p* in (select subject from *ctv*)”
 - ◆ “*ctv* for same *column*” means ...

September 2008

Scott Stoller, Stony Brook University

42

release_to: Example

- # subject is an agent registered for a patient by the patient.
- create shared certtable Register-agent-by-patient
release to register-agent-req-cred for same subject
(relation varchar equals 'register-agent',
pat principal_type equals issuer,
GP principal_type equals null,
agent principal_type equals subject)
check (issuer in (select subject from Register-patient))
- Release policy: release each agent credential to the agent.
- View register-agent-req-cred contains records for agents who have activated the agent role.

Trust Negotiation: Let's Try It!

1. Fetch Physician credentials from NHS or the physician.
2. Release Physician credentials to Patients.
3. Fetch Agent credentials from the agent or the physician.
4. Release Agent credentials to NHS and the agent.
 - create shared certtable Physician (relation ...,role ...)
check (issuer is 'NHS.crt')
 - create shared certtable Agent (relation ..., patient...)
check (issuer in (select subject from Physician))
 - create shared certtable Patients (...)

Summary of Policy Language

- create [per-user | shared] certtable *name*
[fetch from *location*⁺?]
[release to *release_target*?]
(*column_def**) check (*issuer_constraint* [&& *constraint*?])
- *location* ::= issuer | subject | user
- *release_target* ::= public | *pkfile* | *ctv* [for same *column*?]
- *column_def* ::= *name type*
- *issuer_constraint* ::= issuer in (select subject from *ctv**)
| issuer is *pkfile*