

Trust Management

Scott D. Stoller

CSE608: Advanced Computer Security



Outline

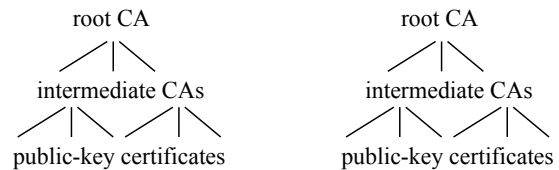
- Intro to Trust Management
- Trust Management for Relational Databases
- Rule-Based Trust Management

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**.

October 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 $\text{role1.members} = \text{role2.members} \cap \text{role3.members}$

October 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.

October 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.

October 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)

October 2008

Scott Stoller, Stony Brook University

11

Outline

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

October 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).

October 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

October 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).

October 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))`

October 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

October 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.

October 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

October 2008

Scott Stoller, Stony Brook University

19

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())

October 2008

Scott Stoller, Stony Brook University

20

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.

October 2008

Scott Stoller, Stony Brook University

21

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

October 2008

Scott Stoller, Stony Brook University

22

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

October 2008

Scott Stoller, Stony Brook University

23

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`

October 2008

Scott Stoller, Stony Brook University

24

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*

October 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, ...)

October 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())

October 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.

October 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.

October 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.

October 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)

October 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.

October 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')

October 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?

October 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)

October 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.

October 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.

October 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.

October 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.

October 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)

October 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.

October 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 ...

October 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.

October 2008

Scott Stoller, Stony Brook University

43

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 (...)

October 2008

Scott Stoller, Stony Brook University

44

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*

October 2008

Scott Stoller, Stony Brook University

45

A Solution

- create shared certtable Physician (relation ...,role ...)
fetch from issuer, subject
release to Patient
check (issuer is 'NHS.crt')
- create shared certtable Agent (relation ..., patient...)
fetch from subject, issuer
release to 'NHS.pub'
release to Agent for same subject
check (issuer in (select subject from Physician))

October 2008

Scott Stoller, Stony Brook University

46

Certificate Fetching During Insertion

- create shared certtable t1 (...) fetch from **subject**
check (issuer in (select subject from t2) && ...)
- create shared certtable t2 (...) fetch from **issuer**
check (issuer in (select subject from t3))
- ab_grant select on EHR to (select subject from t1)
- User u: request_privilege select on EHR
- Suppose t1 does not contain a row with subject u.
- Trust manager calls get_certificates("subject",u, def of t1)
on u's home location L.
- Suppose L replies with a certificate c1.
- Suppose t2 does not contain a row with subject c1.issuer.

October 2008

Scott Stoller, Stony Brook University

47

Certificate Fetching During Insertion (2)

- Trust manager could discard c1.
 - ◆ **Aside:** It might need to discard c1 for other reasons, e.g., because c does not satisfy the constraint in t1's definition. **Why can't the trust manager at L check it?**
- **Trust manager can do better! How?**
- For each subject s in t3, call get_certificates("subject", c1.issuer, def of t2) on home location of s. Try to insert the resulting certificates c2 in t2.
 - ◆ **Note:** Same issues arise for "insert_certificate c1".
- If this doesn't work, what else can the trust manager do?
- Suppose insert fails because c2.issuer is not a subject in t3.

October 2008

Scott Stoller, Stony Brook University

48

Certificate Fetching During Insertion (3)

- Trust manager can request certificates c3 with subject c2.issuer to insert in t3. Request them from where?
- create shared certtable t3 (...) fetch from issuer check (issuer in (select subject from t4))
- Request them from all possible issuers for t3, i.e., all subjects in t4.
- If insertion of c1 still fails, trust manager could try to fetch certs c4 to insert in t4, and then request certs to insert in t3 from their subjects.
- create shared certtable t4 (...) fetch from subject ...
- We don't know the desired subject of c4, so we give up.

October 2008

Scott Stoller, Stony Brook University

49

Certificate Fetching During get_certificates

- Recall: trust manager at u's home location L receives the call get_certificates("subject",u, def of t1).
- create shared certtable mytbl (...) fetch from issuer check (issuer in (select subject from mytbl2))
- Suppose $columns(t1) \subseteq columns(mytbl)$, so certs in mytbl are candidates for the reply.
- Could certificate fetching help?
- Yes, trust manager could fetch certificates for mytbl from subjects in mytbl2 and include them in the reply.

October 2008

Scott Stoller, Stony Brook University

50

Outline

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

October 2008

Scott Stoller, Stony Brook University

51

Simple Rule-Based Trust Management Language

- Essentially Datalog. Start simple. Extend later.
- Atom: issuer.relation(arguments)
- Argument: constant, variable, or constant(arguments)
 - ◆ Relation names and variables start with lowercase.
 - ◆ Constants start with uppercase.
 - ◆ Restrict the use of arguments so constants have bounded depth. In other words, allow tuples, not lists.
- Rule: atom :- atom1, atom2, ...
 - ◆ If atom1 and atom2 and ... hold, then atom holds.
- Fact: a rule with no hypotheses.
- Policy: a collection of facts and rules.

October 2008

Scott Stoller, Stony Brook University

52

Simple Trust Management Language: Example

- By convention, issuer.allow(principal, operation(resource)) means issuer authorizes principal to perform operation on resource. This notation is similar to [Becker+ 2004].
- The default issuer of an atom in a rule is the owner of the policy database containing the rule.
- 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."
AcmeHospital.allow(doc, Read(EPR(pat)) :-
AMA.doctor(doc),
pat.consentToTreatment(doc).

October 2008

Scott Stoller, Stony Brook University

53

Simple Trust Management Language: Example

- SUNY says its employees can read the campus directory.
- SUNY.allow(e, Read(Directory)) :- SUNY.employee(e)
- SUNY says X is a SUNY employee if a SUNY campus says X is a campus employee.
SUNY.employee(e) :- SUNY.campus(c), c.employee(e)
- In this example, the conclusion of a rule is used as a premise of another rule.
- Variables that appear in premises and not in the conclusion are, in effect, existentially quantified.

October 2008

Scott Stoller, Stony Brook University

54

Compliance Checking: Bottom-Up Algorithm

Input: a fact (the goal). **Output:** whether the goal is derivable.

Boolean derivable(goal)

while (there exist

rule "c :- p1,p2,..." in policy and

facts f1, f2, ... in policy and substitution σ

such that $\sigma(p1)=f1$, $\sigma(p2)=f2$, ...

and $\sigma(c)$ not in policy)

add $\sigma(c)$ to policy

return (goal in policy)

- Pretend for now that policy is a global set.

Compliance Checking: Goal-Directed Alg.

Boolean derivable(goal)

for each rule r and substitution σ s.t. $\sigma(r.conclusion)=goal$

for each premise p of r

if derivable($\sigma(p)$) continue; // try next premise

else break; // this rule failed; try next rule

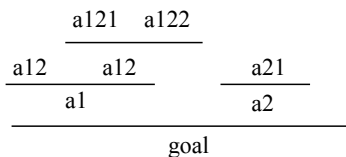
return true // we proved the goal using this rule

// no rule succeeded

return false

Proof of Compliance

- The goal-directed algorithm can easily be extended to provide a **proof** that the goal is derivable.
- A **proof** is a tree formed from instantiated rules, with facts from the policy at the leaves, and with the goal at the root.



Goal-Directed Algorithm: Tabling

- The simple goal-directed algorithm on previous slide may:
 - re-derive the same goal many times
 - **Example:** a12 and a21 could be the same.
 - diverge on recursive policies
- Goal-directed evaluation with **tabling**:
 - Cache all derived goals.
 - Look in the cache for an **existing goal** that **unifies** with the **new goal** before attempting to derive the new goal.

Outline

- Introduction and Motivation
- **Design Issues and Features**
 - Re-Delegation
 - External Data
 - Proof Search
 - Separation of Duty
 - Credential Gathering
 - Separation of Privilege
 - Policy Changes
 - Roles
 - Constraints
 - Global and Local Names
- Trust Management Frameworks
- Sample Application Domains
- Research Directions

Re-Delegation

- If A delegates a permission to B, can B re-delegate it to C?
- **Example:** Conference's policy for reviewing papers
- A PC member can submit a review of a paper.
- $allow(pcmem, Submit(Review(p))) :- PCmember(pcmem)$
- A PC member can designate a subreviewer.
- $allow(subrev, Submit(Review(p))) :- PCmember(pcmem), pcmem.subreviewer(subrev)$
- Can subreviewer S1 delegate to sub-sub-reviewer S2? No.
- $S1.subreviewer(S2)$ doesn't work, because $Conf.PCmember(S1)$ doesn't hold.
- S2 could write S1's review, though.

Re-Delegation

- Re-delegation is allowed if relations are defined recursively.
- Conf: `allow(rev, Submit(Review(p))) :- PCmember(rev)`
- If `rev` can submit review, `rev` can designate subreviewer.
- `allow(subrev, Submit(Review(p))) :-
allow(rev, Submit(Review(p))),
rev.allow(subrev, Submit(Review(p)))`
- This allows delegation chains of arbitrary length.
- To allow delegation chains up to a specified length, use a subreviewer relation parameterized by the allowed delegation depth.

October 2008

Scott Stoller, Stony Brook University

61

In compliance checking, who searches for proof?

- **Resource owner**, i.e., the policy enforcement mechanism
 - ◆ Example: medical records database server
- **Requester** (needs resource owner's policy) [Bauer+05]
 - ◆ Appropriate for embedded devices
 - ◆ Example: Lock with Bluetooth on Mike's office door.
 - ◆ The lock's policy is: `allow(e, Open()) :-
Mike.allow(e, Open(OfficeDoor))`
 - ◆ e's cell phone needs to present a proof of `Mike.allow(e, Open(OfficeDoor))`.
 - ◆ The cell phone can communicate with Mike and his delegates. The lock can't.

October 2008

Scott Stoller, Stony Brook University

62

Credential Gathering

- **Credential**: signed certificate containing a policy statement, usually a fact (in some systems, a fact or rule).
- To **import** a credential `[iss.r(args)]` signed by `K`: if `K` is `iss`'s key, and the signature is valid, then add `iss.r(args)` to the policy; otherwise, the credential is invalid.
- Compliance checking requires credentials for all subgoals with remote issuers.
- Example:
`AcmeHospital.allow(doc, Read(EPR(pat))) :-
AMA.doctor(doc),
pat.consentToTreatment(doc).`

October 2008

Scott Stoller, Stony Brook University

63

Where To Get Credentials?

- From **issuer**
 - ◆ Example: request `AMA.doctor(doc)` from AMA
- From **requester**
 - ◆ Example: request `AMA.doctor(doc)` from `doc`
 - ◆ `doc` may have it or may request it from AMA.
- From a location specified in the policy (instead of hard-coding the decision in the evaluation algorithm).
 - ◆ Details on the next slide.

October 2008

Scott Stoller, Stony Brook University

64

Policy-Directed Credential Gathering

- Each **premise** is labeled with a **location** (e.g., an Internet address) where the credential should be obtained. Location can be a variable. Default location is issuer.
- Notation: `location◇issuer.relation(args)`.
- Example: request `AMA.doctor` credential from `doctor`.
`AcmeHospital.allow(doc, Read(EPR(pat))) :-
doc◇AMA.doctor(doc), pat.consentToTreatment(doc).`
- Example: request `AMA.doctor` credential from `AMA`.
- `AcmeHospital.allow(doc, Read(EPR(pat))) :-
AMA◇AMA.doctor(doc), pat.consentToTreatment(doc).`
- Can include both of these rules in the policy.

October 2008

Scott Stoller, Stony Brook University

65

Policy Changes

- Derived facts may be cached locally and may be sent in credentials and cached at other sites, for efficiency and fault-tolerance.
 - ◆ Example: Hospital caches `AMA.doctor` credential.
- These facts may become invalid due to:
 - ◆ Deletion of facts or rules.
 - ◆ Addition of facts or rules, if the policy language is **non-monotonic** (adding a fact or rule can invalidate facts), i.e., can express negation or equivalent.

October 2008

Scott Stoller, Stony Brook University

66

Policy Changes

- This is a standard problem with caching in distributed systems.
- Standard techniques apply, such as:
 - ◆ Re-validation before each use
 - Requires communication but avoids re-checking the signature.
 - ◆ Expiration time
 - This is a partial solution
 - ◆ Revocation list
 - Issuer needs to know where certs it issued are cached

October 2008

Scott Stoller, Stony Brook University

67

Constraints

- **Constraint:** a premise that uses an externally defined relation on a data type. Common examples include:
- **Numerical inequalities**
 - ◆ **Example:** `allow(empl,Read(file)) :- securityLevel(empl,m), securityLevel(file,n), m ≥ n.`
- **Prefix-of** relation on sequences (e.g., pathnames)
 - ◆ **Example:** `allow(stu, Read(file)) :- Registrar.enrolled(stu, CSE306), /CSE306/project/ prefix-of file.`
- These relations can't be defined in Datalog.

October 2008

Scott Stoller, Stony Brook University

68

External Data

- Policy may depend on **external data**.
- **Example:** personnel database: employees, their department and rank
- **Example:** EHR database: author of each entry
- Storing this info in **policy database** would be inefficient.
- How does the policy access it?
 - ◆ Request **credentials** from the DBMS. This is inefficient and unnecessary, assuming DBMS is local and trusted.
 - ◆ Use a **connector** that makes the DBMS look like part of the policy database. Neat, because policy language and DBMS are both relational.

October 2008

Scott Stoller, Stony Brook University

69

External Data: Connector to DBMS

- Each **table** corresponds to a **relation**.
- Each **record** corresponds to a **fact**.
- Connector generates **SQL queries** to retrieve relevant data.
- **Example:** `allow(e, read(Budget(dept)) :- deptSeniorPers(sp, dept), sp.allow(e, read(Budget(dept))).`
- `sp` is **unbound** when `deptSeniorPers` is evaluated.
- If `deptSeniorPers` is external, the generated **SQL query** finds and returns all senior personnel of the department.
- If results from DBMS are **cached**, they must be **invalidated** if a **DBMS update** changes the relevant data.

October 2008

Scott Stoller, Stony Brook University

70

External Functions

- Manipulate data
 - ◆ **Example:** selectors for compound data structures
- Provide environment and context information
 - ◆ **Example:** `allow(stu, Read(file)) :- Registrar.enrolled(stu,CSE306), /CSE306/project/ prefix-of file, currentTime() > 09:00.1feb2006.`
- Provide simple interface to external data (file, DBMS, ...).
 - ◆ Arguments must be ground (constants) at call time.
 - ◆ **Example:** Note: `author` is an external function.
`allow(e, Update(rec)) :- employee(e), e=author(rec).`

October 2008

Scott Stoller, Stony Brook University

71

Object-Based Separation of Duty

- Separation of duty limits the set of permissions of a single user. This helps prevent fraud, which requires collusion.
- **Example:** A single employee may perform at most 1 of the 3 steps involved in a purchase: issue purchase order, verify receipt of goods, issue payment.
- **Object-based separation of duty** allows an employee to perform at most 1 of these operations for a single purchase.
- **Example:** `allow(e, IssuePayment(trans)) :- acctgClerk(e), e ≠ getPurchClerk(trans), e ≠ getRcvClerk(trans)`
- `getPurchClerk(trans)`: clerk who issued the PO for `trans`

October 2008

Scott Stoller, Stony Brook University

72

Separation of Privilege

- **Separation of privilege:** an action is permitted only if a specified number of authorized users request it.
- `issuer.allow2(principal1,principal2,operation(resource))` means `issuer` authorizes `principal1` and `principal2` jointly (together) to perform `operation` on `resource`.
- **Example:** `allow2(clerk, mgr, IssuePayment(amount)) :- AcctgClerk(clerk), AcctgManager(mgr), clerk ≠ mgr, amount < 1,000,000.`
`allow(clerk, IssuePayment(amount)) :- AcctgClerk(clerk), amount < 10,000.`
- **Alternative:** Decompose the action into multiple actions.
- **Example:** `clerk: InitiatePayment, mgr: ApprovePayment.`

October 2008

Scott Stoller, Stony Brook University

73

Roles

- **Parameterized role:** `r(args)`. Abbreviate `r()` as `r`.
- **Example:** `Manager(department), Guardian(patient)`
- “`p` is a member of `r(args)`” can be represented as
 - ◆ `r(p, args)` roles as relations
 - ◆ `member(p, r(args))` roles as values
- Permission-role relation is defined by rules like:
 - ◆ `permit(p, oper(resource)) :- r(p,args), ...`
 - ◆ `permit(p, oper(resource)) :- member(p,r(args)), ...`
- Roles as values allows variables that range over roles, but this can be simulated with roles as relations.

October 2008

Scott Stoller, Stony Brook University

74

Role Hierarchy

- **Role hierarchy** can be expressed by rules for inheritance of membership.
- **Example:** `Manager ≥ Employee` is expressed by `member(e, Employee) :- member(e, Manager).`
- Role hierarchy could be expressed instead by rules for inheritance of permissions if we made the permission-role relation `PR(action,role)` explicit.

October 2008

Scott Stoller, Stony Brook University

75

Global and Local Names

- **Local names:** names of attributes and relations include the name of a principal, called its `source` or `issuer`.
 - ◆ **Example:** `AMA.doctor(Dan), BMA.doctor(Dan)`
 - ◆ Statements about `src.r` may be issued only by `src`.
- **Global names:** shared namespace for attributes & relations.
 - ◆ Less structured, but more flexible.
 - ◆ **Example:** `AcmeHosp.member(Dan, Doctor(AMA))`
- **RT** [Li+ 2003]: local. **Cassandra** [Becker+ 2004]: global.
- An **ontology** can provide common meaning for names.
- Local names for **principals**, e.g., [SBU President]. Used in **SPKI/SDSI**. Can be simulated using parameterized roles.

October 2008

Scott Stoller, Stony Brook University

76

Outline

- Introduction and Motivation
- Design Issues and Features
- **Trust Management Frameworks**
 - ◆ Several frameworks are listed on the next slide.
 - ◆ We'll discuss a few representative frameworks.
- Sample Application Domains
- Research Directions

October 2008

Scott Stoller, Stony Brook University

77

Some Trust Management Frameworks

- Authentication in Distributed Systems, Taos [Burrows, Abadi, Lampson, Wobber, 1992-1994]
- **PolicyMaker**, Keynote [Blaze, Feigenbaum, et al., 1996-9]
- **SPKI/SDSI** [Rivest, Lampson, et al., 1997-1999]
 - ◆ Simple PKI / Simple Distributed Security Infrastructure
- Delegation Logic, RT [Li et al., 2000-present]
- SD3 [Jim 2001]
- **Binder** [DeTreville 2002]
- **TrustBuilder** [Seamons, Winslett, et al., 2002-present]
- **PeerTrust** [Nejdl, Olmedilla, et al., 2003-present]
- **Cassandra** [Becker and Sewell, 2004-2005]

October 2008

Scott Stoller, Stony Brook University

78

PolicyMaker [Blaze, Feigenbaum, et al.]

- PolicyMaker is a blackboard-based trust management **architecture**. The **blackboard** contains
 - ◆ requests (goals): action/statement to be authorized
 - ◆ acceptance records: issuer allows action/statement
- **Policy**: functions that read requests and acceptance records from the blackboard and write acceptance records.
 - ◆ Use any safe functional programming lang. (SafeAWK)
- PolicyMaker is **flexible** but offers **minimal functionality**.
 - ◆ Application gathers credentials, verifies signatures, etc.
 - ◆ AWK interpreter (or ...) evaluates policy functions

October 2008

Scott Stoller, Stony Brook University

79

SPKI/SDSI [Rivest, Lampson, et al.] Name Certificates

- Local names for principals. The meaning of local names is given by name certificates that relate local names to each other and to global identifiers (public keys).
- **Format**: [K, name, subject, validitySpec] signed by K
- **Meaning**: local name K name refers to subject
- **subject** may be a name or a public key
- **Example**: [K-SBU-CS, Chair, K-Ari, exp. june 2010]
- A local name mapped to multiple keys is a group name.
- **Example**: [K-SUNY, Student, K-Joe, exp. may 2008]
[K-SUNY, Student, K-Mary, exp. may 2008]
- **validitySpec**: expiration date, CRL location, ...

October 2008

Scott Stoller, Stony Brook University

80

SPKI/SDSI: Authorization Certificate

- **Format**: [K, subject, deleg, tag, validitySpec] signed by K
 - ◆ Not using official syntax.
- **Meaning**: issuer K gives permission tag to subject.
 - ◆ deleg indicates whether subject can delegate the permission (in addition to using it himself).
- **subject** may be a name (defined by other certificates), a public key, a threshold structure, or an object hash (ignore)
 - ◆ A **threshold structure** [{K1,K2,...}, n] means any n of the listed keys can together authorize the delegated action (separation of privilege).

October 2008

Scott Stoller, Stony Brook University

81

SPKI/SDSI: Authorization Certif. Examples

- [K-SBU, [K-SBU Faculty], false, readDir, exp. 2010]
- **Example with delegation**:
 - [K-SBU, [K-SBU Faculty], true, (getRoster *), exp. 2010]
- **Name Certificate**: [K-SBU, Faculty, Scott, exp. 2009]
- [K-Scott, K-Tom, false, (getRoster CSE101), exp. 2008]
 - ◆ Tom is the TA. He can't delegate this permission.
- Authorization certificates define **one relation**: allows (delegates).
- Role-based policies can be expressed using groups

October 2008

Scott Stoller, Stony Brook University

82

Limitations of SPKI/SDSI

- Delegation and authorization are not distinguished: a principal must **have** a permission in order to **delegate** it.
 - ◆ **Example** [De Treville 2002]: DMV must be a licensed driver in order to be authorized to license drivers.
- Only unary relations on principals, expressed as groups, are supported.
 - ◆ Can't express policies like: "Nurses in the workgroup treating a patient can access the patient's medical record", which uses relation **treatingWorkgroup(pat,grp)**
- No **variables** (parameters) in tags. No **conjunction** of groups/attributes. No **trust negotiation**.

October 2008

Scott Stoller, Stony Brook University

83

Binder [DeTreville 2002]

- Our simple policy language (without constraints) is very similar to Binder's.
- Authorization relation:
 - ◆ issuer says can(principal,operation,resource)
- **Nicely written** paper; recommended as an **introduction** to rule-based trust management
- Allows **communication of rules** (discussed later), although the details are unspecified
- Does not consider **trust negotiation**.

October 2008

Scott Stoller, Stony Brook University

84

Cassandra [Becker and Sewell, 2004-2005]

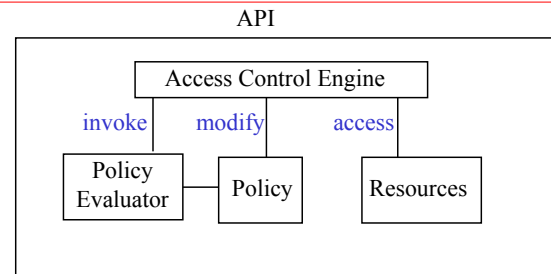
- Lang: Datalog + constraints + aggregation (non-monotonic)
- Role-based
- Parameterized roles, parameterized actions (permissions)
- Global names (simulate local names using issuer and other parameters)
- Goal-directed evaluation with memoization (tabling)
- Policy-controlled credential gathering
- Role activation (but not sessions; details below)
- Trust negotiation
- External functions

October 2008

Scott Stoller, Stony Brook University

85

Cassandra Architecture



- Policy: local policy + cached credentials
- Alternative architecture: return authorization decisions to the application

October 2008

Scott Stoller, Stony Brook University

86

Cassandra Predicates

- **Syntax for Predicates:** `location◇issuer.pred(args)`
- **Special Predicates** (special significance in the API):
- `permits(entity, action)`: entity is authorized to perform action
- `canActivate(entity, role)`: entity can activate (is a member of) role
- `hasActivated(entity, role)`: entity has activated role
 - ◆ Role activations are recorded as facts in the policy.
 - ◆ No explicit notion of sessions; implicitly, there is one session per Cassandra instance.

October 2008

Scott Stoller, Stony Brook University

87

Facts as Role Activations

- Many kinds of facts are expressed as role activations. An entity says `fact(args)` by activating the "role" `fact(args)`.
 - ◆ This moderately simplifies the API.
- **Example:** A patient consents to treatment by Dr. Dan by activating the role `ConsentToTreatment(Dan)`.
- **Example:** The manager of department `dept` appoints an employee `e` in `dept` by activating `AppointEmployee(e,dept)`
- `canActivate(mgr, AppointEmployee(e, dept)) :- hasActivated(mgr, Manager(dept))`
- `canActivate(e, Employee(dept)) :- hasActivated(mgr, AppointEmployee(e, dept))`

October 2008

Scott Stoller, Stony Brook University

88

Cassandra Predicates: canDeactivate

- `canDeactivate(entity, entity1, role)`: entity is authorized to deactivate `entity1`'s activation of `role`
- **Example:**

```

canDeactivate(pat, pat, ConsentToTreatment(doc)) :- true.
canDeactivate(grdn, pat, ConsentToTreatment(doc)) :-
  hasActivated(grdn, Guardian(pat)).
  
```
- Cassandra does not consider administrative policy, so there is no notion of who is authorized to remove an entity from a role (by changing the policy).

October 2008

Scott Stoller, Stony Brook University

89

Cassandra Predicates: isDeactivated

- `isDeactivated(entity,role)`: entity's activation of `role` is being deactivated. Used to trigger other role deactivations.
- **Example:** A user being removed from the `Employee` role should also be removed from the `Manager` role.


```

isDeactivated(e, Manager()) :-
  isDeactivated(e, Employee()).
  
```
- **Example:** A doctor being removed from "on duty at hospital" should also be removed from "attending doctor".


```

isDeactivated(doc, AttendingDoctor(pat)) :-
  isDeactivated(doc, OnDuty()).
  
```
- Could add premise `hasActivated(doc,AttendingDoctor(pat))`

October 2008

Scott Stoller, Stony Brook University

90

Cassandra Predicates: isDeactivated (2)

- Example where deactivation of a user's role triggers deactivation of a different user's role:
- `isDeactivated(e, Employee(dept)) :-`
`isDeactivated (mgr, AppointEmployee(e,dept)),`
`COUNT(hasActivated(x, AppointEmployee(e, dept)))=1`

October 2008

Scott Stoller, Stony Brook University

91

Cassandra Predicates: canRequestCredential

- This predicate expresses trust negotiation policy.
- `canRequestCredential(entity, issuer.r(args))`: entity is authorized to request credentials that match issuer.r(args).
 - ◆ Abbreviate as `canRequestCred`
- **Example**: OLU allows a registered student to request a credential showing this.
- `stu` is registered in semester `sem` \leftrightarrow `stu` has activated `Student(sem)`.
- `canRequestCred(stu,OLU.hasActivated(stu,Student(sem))) :- hasActivated(stu,Student(sem))`.
 - ◆ Response: "here's the certif" or "unauthorized request"

October 2008

Scott Stoller, Stony Brook University

92

Cassandra Predicates: canRequestCredential

- Continuing the example, consider an alternative rule:
- `canRequestCred(stu,OLU.hasActivated(stu,Student(sem))) :- true`.
 - ◆ Same effect as previous policy, except response to requests by non-student asking about own status is "You are not registered as a student."
- OLU allows a student's parent to request that credential.
- `canRequestCred(par,OLU.hasActivated(stu,Student(sem))) :- parentOf(par,stu)`.

October 2008

Scott Stoller, Stony Brook University

93

Cassandra Predicates: canRequestCredential

- A student can delegate authority to get his Student credential to anyone (e.g., potential employer).
- **OLU's policy**:
`canRequestCred(e,OLU.hasActivated(stu,Student(sem))) :- stu.canRequestOLUreg(e)`.
- **Joe Cool's policy**: `JoeCool.canRequestOLUreg(Google)`.
- An entity can have a `canRequestCredential` policy for credentials (attributes) it does not have.
- **Example**: Every user can have the policy
- `canRequestCredential(c, IRS.lowIncome(y)) :- IRS.nonProfit(c)`

October 2008

Scott Stoller, Stony Brook University

94

Cassandra API: doAction, activate

S: site where the operation is invoked.

P: S's policy.

P |- fact: fact is derivable from P.

e: the entity invoking the operation.

e:doAction(a)

if P |- S.permits(e,a)

execute action a.

e:activate(r)

if P |- S.canActivate(e,r) and not P |- S.hasActivated(e,r)

add S.hasActivated(e,r) to P

October 2008

Scott Stoller, Stony Brook University

95

Cassandra API: deActivate

e:deactivate(e1,r)

if P |- S.hasActivated(e1,r) and P |- S.canDeactivate(e,e1,r)

add S.isDeactivated(e1,r) to P

D = { [e2,r2] | P |- S.isDeactivated(e2,r2) }

// note: D contains (e1,r)

for [e2,r2] in D

remove S.hasActivated(e2,r2) from P (if present)

remove S.isDeactivated(e1,r) from P

To handle rules like `isDeactivated(...) :- ¬hasActivated(...)`, another loop is needed.

October 2008

Scott Stoller, Stony Brook University

96

deActivate: Example

Initial policy P:

```
isDeactivated(e, Manager()) :- isDeactivated(e, Employee())
hasActivated(Mike, Employee())
hasActivated(Mike, Manager())
canDeactivate(Charles, Mike, Employee())
```

Charles:deActivate(Mike,Employee())

Add isDeactivated(Mike, Employee()) to P.

Then P |- isDeactivated(Mike, Manager()).

So D = { [Mike,Employee()], [Mike,Manager()] }

Cassandra API: requestCredential

- may be invoked directly or via a remote premise.
 - iss.r(args) must be ground. Ignore constraint creds here.
- ```
e:requestCredential(iss.r(args))
if P |- S.canRequestCredential(e, iss.r(args))
if iss = S
 if P |- S.r(args) return S.r(args) signed by S
 else return "not S.r(args)"
else // iss ≠ S. Forward cached credential, if any.
 if P contains iss.r(args) return iss.r(args) signed by iss
 else return "unauthorized request"
```

## Cassandra: Constraints and Aggregation

- Constraints over integers, sets, other domains.
- Aggregation operators:
- Group: collect the facts that match a pattern S.r(args) into a set
- Count: count the number of facts that match a pattern S.r(args)
- Variables in S.r(args) not used elsewhere in query may have any value
- S is the local entity; otherwise, answer may be incomplete.

## Non-Monotonic Policies

- Aggregation + constraints allow non-monotonic policies
- Example: Dynamic Separation of Duty: no user may have the Doctor and Patient roles active concurrently.  

```
canActivate(doc, Doctor()) :-
 AMA.Doctor(doc),
 COUNT(hasActivated(doc, Patient())) = 0
```
- This is my own syntax. Cassandra's syntax is more general but harder to read.
- Non-monotonic because adding hasActivated(Dan, Patient()) changes canActivate(Dan, Doctor()) from true to false.

## Example: Chinese Wall

- To avoid conflict of interest, a consultant can work on at most one project involving each industry sector. (This simple policy does not consider information flow.)
- Example: can't work on projects for Intel and AMD, both in semiconductor sector.
- industrySector(proj, sec): proj involves industry sector sec.
- Example: industrySector(IntelReengg, Semiconductor).
- employeeSector(emp, sec): employee emp is working on a project in sector sec.
- employeeSector(emp, sec) :-  
 hasActivated(mgr, AppointEmployee(emp, proj)),  
 industrySector(proj, sec)

## Example: Chinese Wall

- canActivate(mgr, AppointEmployee(emp, proj)) :-  
 Manager(mgr, proj), industrySector(proj, sec),  
 COUNT(employeeSector(emp, sec)) = 0.

## Let's Try It!

1. OLU: A student can read his or her own academic record.
  2. OLU: Someone claiming a student as a dependent on his tax return, according to IRS, can read the student's record.
  3. IRS: Information about dependents is provided to universities registered with the Education Dept (ED).
  4. ED: Info about registered universities is avail to everyone.
  5. OLU: A faculty can assign a grade for a student enrolled in a class he is teaching.
- **Actions:** ReadRec(stu), AssignGrade(class,stu)
  - **Relations:** taxDependent(dependent, filer), isUniv(univ), teaching(faculty, class), enrolled(student, class), canActivate(e, role), canReqCred(e, cred), permits(e, act)

October 2008

Scott Stoller, Stony Brook University

103

## A Solution

1. OLU: permits(stu, ReadRec(stu)) :- true.
2. OLU: permits(p, ReadRec(stu)) :-  
IRS $\diamond$ IRS.taxDependent(stu, p).
3. IRS: canRequestCredential(u, taxDependent(x, y)) :-  
ED $\diamond$ ED.isUniv(u).
4. ED: canRequestCredential(x, isUniv(u)) :- true.
5. OLU: permits(fac, AssignGrade(class, stu)) :-  
teaching(fac,class), enrolled(stu,class).

October 2008

Scott Stoller, Stony Brook University

104

## Let's Try the TM-SQL Example Again!

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 EHR.

**Hint:** Interpret "x is a physician" as "x can activate the Physician role".

October 2008

Scott Stoller, Stony Brook University

105

## A Solution

- canActivate(u, Physician()) :-  
NHS.hasActivated(x, RegisterPhysician(u))
  - Why not the following?  
canActivate(u, Physician()) :- NHS.isPhysician(u)
- canActivate(u, RegisterAgent(ag,pat)) :-  
hasActivated(u, Physician)
- canActivate(ag, Agent(pat)) :-  
hasActivated(p, RegisterAgent(ag,pat))
- permit(u, ReadEHR(pat)) :- hasActivated(u, Agent(pat))
- [See notes on Extended API for Cassandra](#)

October 2008

Scott Stoller, Stony Brook University

106

## Outline

- Introduction and Motivation
- Design Issues and Features
- Trust Management Frameworks
- **Sample Application Domains**
  - Electronic Health Records (EHR)
  - Other application domains
- Research Directions

October 2008

Scott Stoller, Stony Brook University

107

## Electronic Health Records (EHR)

- Promising application for RBAC and trust management
- People and organizations with limited trust must share sensitive information: patients, doctors, nurses, hospitals, billing companies, insurance companies, government agencies (e.g., Medicaid, FDA), professional societies (e.g., AMA), medical researchers, etc.
- More interactive information sharing will increase the need for trust management.
- **Case study [Becker 2005]: Output Based Specification for Integrated Care Record Service, version 2, 2003.**  
Developed by the National Health Service (NHS) of the United Kingdom.

October 2008

Scott Stoller, Stony Brook University

108

## EHR Policy

- **Spine**: a nationwide EHR system
  - ◆ one electronic health record (EHR) per patient
  - ◆ multiple items per record
- **Registration Authority (RA)**: issues credentials for clinicians, with name, affiliation, specialty, etc.
  - ◆ typically for one organization, but may be regional
- **Local health organizations**: hospitals, doctors' offices, etc.
  - ◆ one electronic patient record (EPR) per patient, with full data
- **Patient Demographic System (PDS)**
  - ◆ One nationwide PDS

October 2008

Scott Stoller, Stony Brook University

109

## Registration Authority Policy

- **Main Role**: RA-manager
- A manager registers a clinician by activating NHS-Clinician-cert(...).
- **canActivate**(mgr, NHS-clinician-cert(org, cli, spcty, start, end)) :-  
hasActivated(mgr, RA-manager()),  
hasActivated(y, NHS-health-org-cert(org, start2, end2)),  
start in [start2, end2],  
end in [start2, end2],  
start < end

October 2008

Scott Stoller, Stony Brook University

110

## Spine Policy: Main Roles and Main Actions

- **Clinician**
  - ◆ request consent to treatment, read and update EHR
  - ◆ emergency access to EHR
  - ◆ refer a patient to another clinician
  - ◆ approve requests to seal items
  - ◆ appoint agents for patient (if patient is unable to)
  - ◆ conceal items (from patient or other clinicians)
- **Administrator**
  - ◆ register new patients, clinicians, and administrators
  - ◆ unregister old ones

October 2008

Scott Stoller, Stony Brook University

111

## Spine Policy: Main Roles and Main Actions

- **Patient**
  - ◆ one-off (i.e., one-time) consent to policy
  - ◆ consent to treatment
  - ◆ appoint agents
  - ◆ read items in EHR
  - ◆ request that items in EHR be concealed
- **Agent**: someone who can act on a patient's behalf
- **Third party**: a third party whose consent is needed for an action. **Example**: Joe needs his father's consent to access item in Joe's EHR describing his father's cardiac disease.

October 2008

Scott Stoller, Stony Brook University

112

## Spine Policy: Activate Spine-clinician Role

- A NHS-certified clinician can activate Spine-clinician role.
- **canActivate**(cli, Spine-clinician(ra, org, spcty)) :-  
ra.hasActivated(x, NHS-clinician-cert(org, cli, spcty, start, end)), // a similar rule has location ra for this premise  
canActivate(ra, Registration-authority()),  
no-main-role-active(cli),  
Current-time() in [start, end]
- **canActivate**(ra, Registration-authority()) :-  
NHS.hasActivated(x, NHS-registration-authority(ra, start, end)),  
Current-time() in [start, end]

October 2008

Scott Stoller, Stony Brook University

113

## Spine Policy: Author Can Read Item

- The author of an EHR item can always read it, provided the patient has given one-off consent, even if the patient has sealed the item.
- **permits**(cli, Read-spine-record-item(pat, id)) :-  
hasActivated(cli, Spine-clinician(ra, org, spcty)),  
hasActivated(x, One-off-consent(pat)),  
Get-spine-record-org(pat, id) = org,  
Get-spine-record-author(pat, id) = cli

October 2008

Scott Stoller, Stony Brook University

114

## Spine Policy: Treating Clinician Can Read Item

- A treating clinician can read item if patient has given one-off consent, item is not sealed by patient, and the item's subjects are permitted for the clinician's specialty.
- `permits(cli, Read-spine-record-item(pat, id)) :- hasActivated(cli, Spine-clinician(ra, org, spcty)), hasActivated(x, One-off-consent(pat)), canActivate(cli, Treating-clinician(pat, org, spcty)), count-concealed-by-spine-patient(n, a, b), n = 0, a = (pat, id), b = (org, cli, spcty), Get-spine-record-subjects(pat, id) ⊆ Permitted-subjects(spcty)`

October 2008

Scott Stoller, Stony Brook University

115

## Spine Policy: Activate Treating-clinician

- `cli` can activate `Treating-clinician(pat, org, spcty)` if
  - ◆ `pat` consented to treatment by `cli`, or
  - ◆ `pat` consented to treatment by workgroup containing `cli`, or
  - ◆ a clinician treating `pat` referred `pat` to `cli`, or
  - ◆ there is an emergency situation (audited later)

October 2008

Scott Stoller, Stony Brook University

116

## Patient Demographic System: Main Roles

- **PDS-manager**
  - ◆ Register and unregister people
- **Patient**
- **Agent**
- **Professional-user**
  - ◆ Clinicians, Caldicott guardians, etc.

October 2008

Scott Stoller, Stony Brook University

117

## Patient Demographic System: Activate Agent

- An agent can activate the `Agent(pat)` role if the agent is registered as a patient at the PDS, and the Spine confirms that he is an agent for `pat`.
- `canActivate(ag, Agent(pat)) :- hasActivated(x, Register-patient(ag)), no-main-role-active(ag), Spine◇Spine.canActivate(ag, Agent(pat))`

October 2008

Scott Stoller, Stony Brook University

118

## Local Health Organization: Staff Roles

- **Clinician(spcty)**
  - ◆ as in Spine
- **Caldicott-guardian()**
  - ◆ patient advocate and ombudsman. can give consent on behalf of a patient and, in exceptional cases, override a patient's decisions.
- **HR-manager()**
  - ◆ Register and unregister patients and staff
- **Receptionist()**
  - ◆ Register patients

October 2008

Scott Stoller, Stony Brook University

119

## Local Health Organization: Non-Staff Roles

- **Patient**
- **Agent**
- **Ext-treating-clinician**
  - ◆ external clinician who needs access to patient's local EPR
- **Third-party**

October 2008

Scott Stoller, Stony Brook University

120

### Local Health Organization Policy: Activate Ext-treating-clinician

- An clinician can activate `Ext-treating-clinician` if the referring clinician has activated the `Consent-to-referral` role and the clinician is certified by an RA certified by NHS.
- `canActivate(cli, Ext-treating-clinician(pat, ra, org, spcty)) :- hasActivated(ref, Consent-to-referral(pat, ra, org, cli, spcty)), no-main-role-active(cli), ra◇ra.hasActivated(y, NHS-clinician-cert(org, cli, spcty, start, end)), canActivate(ra, Registration-authority())`

October 2008

Scott Stoller, Stony Brook University

121

### Local Health Organization Policy: Deactivate Ext-treating-clinician

- `Ext-treating-clinician` is deactivated if patient (or patient's agent) cancels the referral by deactivating the referring clinician's `Consent-to-referral`.
  - `other-referral-consents(...)` holds if, e.g., an agent of the patient has given consent for the referral.
- `isDeactivated(cli, Ext-treating-clinician(pat, ra, org, spcty)) :- isDeactivated(x, Consent-to-referral(pat, ra, org, cli, spcty)), other-referral-consents(0, x, pat, ra, org, cli, spcty)`

October 2008

Scott Stoller, Stony Brook University

122

### Other Potential Application Domains

- **Military**: cooperation with
  - ◆ other armed services (Army, Navy, Air Force, Marines)
  - ◆ government agencies
  - ◆ highly trusted coalition partners
  - ◆ less trusted coalition partners
- **Collaborative Engineering Design** [Bhargava+, 2004]
  - ◆ multi-vendor bids for large engineering contracts
  - ◆ collaborators need to share designs and design knowledge, status of technologies, etc.

October 2008

Scott Stoller, Stony Brook University

123

### Other Potential Application Domains

- **Supply Chain Management** [Bhargava+, 2004]
  - ◆ For tight integration, a company must give its suppliers, customers, and its customers' customers (to increase their confidence in its ability to deliver) some access to its order entry, order status, sales forecast, and production planning systems.

October 2008

Scott Stoller, Stony Brook University

124

### Outline

- Introduction and Motivation
- Design Issues and Features
- Trust Management Frameworks
- Sample Application Domains
- **Research Directions**
  - ◆ State-dependent policies
  - ◆ Communication of rules
  - ◆ Administrative policy
  - ◆ Policy analysis
  - ◆ Trust for service provision

October 2008

Scott Stoller, Stony Brook University

125

### State-Dependent Policies

- **Approach 1**: The application should know how and when to update the state.
- **Example**: A teaching assistant (TA) may change a student's grade for an assignment at most once.
 

`permit(ta, ChangeGrade(class, stu, assignment)) :- TeachingAssistant(ta, class), COUNT(changedGrade(ta, class, stu, assignment)) = 0.`

  - ◆ Application should remember to add the fact `changedGrade(...)` when a grade is changed.
- The state may be **internal** (as in this example) or **external**.

October 2008

Scott Stoller, Stony Brook University

126

## State-Dependent Policies

- **Approach 2:** Required updates (and other side-effects) are specified as part of the policy.
- `permit(principal, operation(resource), effect)`: as before, and `effect` should be executed together with the operation.
- In query, first two args are constants, third arg is a variable
- **Example:**  
`permit(ta, ChangeGrade(class,stu,assignment),  
addFact(changedGrade(ta,class,stu,assignment))) :-  
TeachingAssistant(ta,class),  
COUNT(changedGrade(ta,class,stu,assignment)) = 0.`
- Digital rights management uses state-dependent policies.

October 2008

Scott Stoller, Stony Brook University

127

## Communication of Rules

- Policy evaluator may gather rules, as well as facts, from other sites. This can be more efficient.
- **Example:** SUNY students get discount at Textbooks.com.
- **Textbooks.com:**  
`getDiscount(stu) :- SUNY $\diamond$ SUNY.student(stu).`
- **SUNY:** `student(stu) :- stu $\diamond$ SBU.student(stu).`  
`student(stu) :- stu $\diamond$ UAlbany.student(stu).`
- **JoeCool:** `SBU.student(JoeCool).`
- Without rule communication: for each student, Textbooks.com asks SUNY which asks student.

October 2008

Scott Stoller, Stony Brook University

128

## Communication of Rules

- With rule communication: Textbooks.com imports rules from SUNY.
- **Textbooks.com:** `SUNY.student(stu) :- SBU.student(stu).`  
`SUNY.student(stu) :- UAlbany.student(stu)`
- Only **imported** rules have **conclusions** with **issuer**  $\neq$  **self**.
- Textbooks.com asks each student for campus credential and applies an imported rule to infer SUNY credential.
- This reduces communication overhead and delays.
- Facts concluded using imported rules cannot be exported.
  - ◆ **Example:** `[SUNY.student(JoeCool)]` signed by Textbooks.com is an invalid credential.

October 2008

Scott Stoller, Stony Brook University

129

## Which Rules to Send?

- Textbooks.com asks SUNY for rules relevant to `SUNY.student`.
- **SUNY:** `student(stu) :- SUNYSB.student(stu), approved(stu).`  
`approved(stu) :- ...`
- **Which rules should SUNY send?**
  - ◆ Rules with conclusion `student(stu)`.
  - ◆ Rules with conclusion `student(stu)` and rules they depend on, recursively following dependencies.
- Some of the rules may have premises with remote issuers. Should all of them send relevant rules, too?

October 2008

Scott Stoller, Stony Brook University

130

## Which Rules to Send?

- **Binder** [DeTreville 2002] does not address this issue.
- **Secure Dynamically Distributed Datalog (SD3)** [Jim 2001] sends (in one step) all rules and facts that could be useful for answering the query. This minimizes communication delays but may send unnecessary rules and facts.
- **Open Issues:**
  - ◆ Privacy policies for rules
  - ◆ Policies that control which rules are sent, instead of a fixed algorithm.

October 2008

Scott Stoller, Stony Brook University

131

## Administrative Policy

- **Administrative policy:** security policy that controls changes to the security policy.
- Introduce **actions** that update the policy
  - ◆ `addFact(fact), addRule(rule), removeFact(fact), removeRule(rule)`
- Develop **authorization rules** for those actions.
- **Example:** `allow(hrm, addFact(paymentMgr(e))) :- humanResourcesMgr(hrm), employee(e).`
- `paymentMgr` may be internal (policy) data or external data.
- This extension to the API eliminates the need to encode such facts as role activations.

October 2008

Scott Stoller, Stony Brook University

132

## Administrative Policy: Static Separation of Duty

- **Separation of duty** limits the set of permissions of a single user. This helps prevent fraud, which requires collusion.
- **Example:** A single employee may perform at most 1 of the 3 steps involved in a purchase: issue purchase order, verify receipt of goods, issue payment.
- **Static separation of duty** allows an employee to be a member of at most 1 of the corresponding roles (purchasing clerk, receiving clerk, accounting clerk).
- **Example:** `allow(so, addFact(member(e, PurchClerk))) :- COUNT(member(e, RcvClerk))=0, COUNT(member(e, AcctgClerk))=0.`

October 2008

Scott Stoller, Stony Brook University

133

## Policy Analysis: Sample Analysis Questions

- Is a given principal allowed to perform a given action?
- Which principals are allowed to perform a given action?
- What is the effect of adding a given rule or fact?
  - ◆ i.e., what new actions can each principal perform?
- What is the effect of removing a given rule or fact?
  - ◆ i.e., what allowed actions does each principal lose?
- Is every principal that is allowed to perform a given action also allowed to perform another given action?
- Analysis algorithms for SPKI/SDSI [Jha+ 2004], XACML [Fisler+ 2005]

October 2008

Scott Stoller, Stony Brook University

134

## Policy Analysis with Administrative Policy

- Given:
  - ◆ a policy, including an administrative policy
  - ◆ a set of (less trusted) administrators
- Ask questions about the policies reachable from the current policy through changes those administrators can perform.
- Does  $Q$  hold for some such policy?
- Does  $Q$  hold for every such policy?
- $Q$  is a yes/no question from the previous slide.

October 2008

Scott Stoller, Stony Brook University

135

## Policy Analysis with Administrative Policy: Example

- **Example:** Can an administrator for the CPU division give an employee access to resources in the Memory division?
- Such delegation of administrative control can be indirect.
  - ◆ **Example:** The Memory division allows employees on the company's re-engineering team to access Memory division resources. An administrator in the CPU division can appoint an employee (representing his division) to the re-engineering team.
- Need to analyze possible delegation chains.
- This is computationally hard in general for rule-based languages [Li+ 2005, Sasturkar+ 2006].
- We identified some useful tractable cases [Stoller+ 2007].

October 2008

Scott Stoller, Stony Brook University

136

## Trust for Service Provision

- **Access control:** Who do I trust to **access** my resource/service?
  - ◆ Usually a **boolean** answer is desired.
- **Service provision:** Who do I trust to **provide** this resource/service to me?
  - ◆ Often desire a **quantitative evaluation** of providers
  - ◆ Final **decision** depends on trust level, cost, speed, etc.
  - ◆ Trust levels may be **discrete** (e.g., low/med/high) or **continuous** (e.g., a number between 0 and 1)
  - ◆ A **confidence level** for the trust evaluation can also be maintained.

October 2008

Scott Stoller, Stony Brook University

137

## Trust for Service Provision

- **Examples:** Trust in
  - ◆ Internet content ratings from different organizations
  - ◆ Internet storage providers (xdrive.com, idrive.com, netdrive.com, ...)
    - Availability, privacy, ...
- **Extend rule-based policy languages to support this**
  - ◆ Add relations, e.g., `acceptService(service, provider, ...)`
  - ◆ Add trust level as a parameter of appropriate relations, e.g., `acceptService(service, provider, trustLevel)`

October 2008

Scott Stoller, Stony Brook University

138

## Recap: 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** of and **relationships** between principals, resources, and other objects.
- **Example:** `AcmeHospital.allow(doc, Read(EPR(pat)) :-  
AMA.doctor(doc),  
pat.consentToTreatment(doc).`