Logics for Access Control and Security

Morteza Amini
m_amini@ce.sharif.edu

Network Security Center
Department of Computer Engineering
Sharif University of Technology

16th October 2006

Outline

The Problem

- Three ingredients are essential for security in computing systems:
  1. A trusted computing system
  2. Authentication
  3. Authorization
- The ingredients are fairly understood in centralized systems. What about in Distributed Systems?
- Difficulties with DS:
  - Scale
  - Communication
  - Booting
  - Loading
  - Authentication
  - Authorization

Why Logics?

1. Clean Foundation: hence formal guarantees
2. Flexibility
3. Expressiveness: possible to describe protocols & policies at reasonable level of abstraction.
4. Independency from implementation: important in heterogeneous distributed systems
5. Declarativeness: Users not required any programming ability
6. Having Ability of Verification
7. ...

Applications of Logics in Security

1. Logic-based Policy Specification
2. Policy Composition Frameworks
   - Constraint-based Approach
   - Deontic Approach
   - Algebraic Approach
3. Policy Evaluation and Verification
4. Trust Management
5. Model Checking for Protocol Verification
6. Intrusion Detection

Outline

Basic Concepts

Principal

Participants in a distributed systems are called *agents*, and the symbols the represents agents in logical expressions are called *principals*.

- Users and Machines
- Channels
- Conjunction of principals $(A \wedge B)$
- Groups
- Principals in roles $(A\ as\ R)$
- Principals on behalf of principals $(B\ for\ A) or (B|A)$

Basic Concepts

$A \wedge B$: A and B as cosigners. A request from $A \wedge B$ is a request that both A and B make.

$A \vee B$: represents a group of which A and B are the sole members.

A *as* R: a principal A in role R.

$B|A$: the principal obtained when B speaks on behalf of A, not necessarily with a proof that A has delegated authority to B. we pronounce it B *quoting* A.
$B|A$ *says* s if B *says* that A *says* s.

Basic Concepts

$A \wedge B$: A and B as cosigners. A request from $A \wedge B$ is a request that both A and B make.

$A \vee B$: represents a group of which A and B are the sole members.

A *as* R: a principal A in role R.

$B|A$: the principal obtained when B speaks on behalf of A, not necessarily with a proof that A has delegated authority to B. we pronounce it B *quoting* A.
$B|A$ *says* s if B *says* that A *says* s.

Basic Concepts

$A \wedge B$: A and B as cosigners. A request from $A \wedge B$ is a request that both A and B make.

$A \vee B$: represents a group of which A and B are the sole members.

A *as* R: a principal A in role R.

$B|A$: the principal obtained when B speaks on behalf of A, not necessarily with a proof that A has delegated authority to B. we pronounce it B *quoting* A.
$B|A$ *says* s if B *says* that A *says* s.

Basic Concepts

$A \wedge B$: A and B as cosigners. A request from $A \wedge B$ is a request that both A and B make.

$A \vee B$: represents a group of which A and B are the sole members.

A *as* R: a principal A in role R.

$B|A$: the principal obtained when B speaks on behalf of A, not necessarily with a proof that A has delegated authority to B. we pronounce it B *quoting* A.
$B|A$ *says* s if B *says* that A *says* s.

Basic Concepts

B *for* A: the principal obtained when B speaks on behalf of A with appropriate delegation certificates.
B *for* A *says* s when A has delegated authority to B and B *says* that A *says* s.

$A \Rightarrow B$ (A *implies* B) or (A *speaks for* B): A is a member of group B. A is at least as powerful as B.

$$A \Rightarrow B \text{ iff } A = A \wedge B.$$

A *says* s: *says* is a modal operator. A *says* s is a formula that means the principal A believes that formula s is true.

$\supset$: logical implication.

Basic Concepts

B *for* A: the principal obtained when B speaks on behalf of A with appropriate delegation certificates.
B *for* A *says* s when A has delegated authority to B and B *says* that A *says* s.

$A \Rightarrow B$ (A *implies* B) or (A *speaks for* B): A is a member of group B. A is at least as powerful as B.

$$A \Rightarrow B \text{ iff } A = A \wedge B.$$

A *says* s: *says* is a modal operator. A *says* s is a formula that means the principal A believes that formula s is true.

$\supset$: logical implication.

Basic Concepts

B *for* A: the principal obtained when B speaks on behalf of A with appropriate delegation certificates.
B *for* A *says* s when A has delegated authority to B and B *says* that A *says* s.

$A \Rightarrow B$ (A *implies* B) or (A *speaks for* B): A is a member of group B. A is at least as powerful as B.

$$A \Rightarrow B \text{ iff } A = A \wedge B.$$

A *says* s: *says* is a modal operator. A *says* s is a formula that means the principal A believes that formula s is true.

$\supset$: logical implication.

Basic Concepts

B *for* A: the principal obtained when B speaks on behalf of A with appropriate delegation certificates.
B *for* A *says* s when A has delegated authority to B and B *says* that A *says* s.

$A \Rightarrow B$ (A *implies* B) or (A *speaks for* B): A is a member of group B. A is at least as powerful as B.

$$A \Rightarrow B \text{ iff } A = A \wedge B.$$

A *says* s: *says* is a modal operator. A *says* s is a formula that means the principal A believes that formula s is true.

$\supset$: logical implication.

Basic Concepts

A controls s: $(A \; says \; s) \supset s$.
ACL is a list of assertions like $(A \; controls \; s)$. If s is clear, ACL is a list of principals trusted on s

Corollary

$$\frac{B \;\; controls \;\; s \; \wedge \; A \Rightarrow B}{\therefore \; A \;\; controls \;\; s}$$

Basic Concepts

A controls s: $(A \ says \ s) \supset s$.

ACL is a list of assertions like $(A \ controls \ s)$. If s is clear, ACL is a list of principals trusted on s

Corollary

$$\frac{B \ \ controls \ \ s \ \wedge \ A \Rightarrow B}{\therefore \ A \ \ controls \ \ s}$$

The Basic Logic
A Calculus of Principals

Some Axioms

1. $\wedge$ and $|$ are primitive operators of calculus of principals.
2. $\wedge$ is associative, commutative, and idempotent
   - Principals form a semilattice under $\wedge$.
3. $|$ is associative.
   - Principals form a semigroup under $|$.
4. $|$ distributes over $\wedge$.
   - Multiplicativity implies monotonicity.

Corollary

Principals structure multiplicative semilattice semigroup, which is isomorphism with binary relations with union and composition.

The Basic Logic
A Calculus of Principals

Some Axioms

1. $\wedge$ and $|$ are primitive operators of calculus of principals.
2. $\wedge$ is associative, commutative, and idempotent
   - Principals form a semilattice under $\wedge$.
3. $|$ is associative.
   - Principals form a semigroup under $|$.
4. $|$ distributes over $\wedge$.
   - Multiplicativity implies monotonicity.

Corollary

Principals structure multiplicative semilattice semigroup, which is isomorphism with binary relations with union and composition.

The Basic Logic

A Logic of Principals and Their Statements

Syntax

The formulas are defined inductively as follows:

1. a countable supply of primitive propositions $p_0, p_1, \ldots$ are formulas;
2. if s and s' are formulas then so are $\neg s$ and $s \wedge s'$;
3. if A and B are principal expressions then $A \Rightarrow B$ is a formula;
4. if A is a principal expression and s is a formula then $A \textit{ says } s$ is a formula.

The Basic Logic

A Logic of Principals and Their Statements

Axioms

1. The basic axioms for normal modal logic:
   - if s is an instance of a propositional-logic tautology then $\vdash s$;
   - if $\vdash s$ and $\vdash (s \supset s')$ then $\vdash s'$;
   - $\vdash A \textit{ says } (s \supset s') \supset (A \textit{ says } s \supset A \textit{ says } s')$;
   - if $\vdash s$ then $\vdash A \textit{ says } s$, for every A.
2. The axioms of the calculus of principals:
   - if s is a valid formula of the calculus of principals then $\vdash s$.
3. The axioms connect the calculus to the modal logic:
   - $\vdash (A \wedge B) \textit{says } s \equiv (A \textit{ says } s) \wedge (B \textit{ says } s)$;
   - $\vdash (B|A) \textit{says } s \equiv B \textit{ says } A \textit{ says } s$;
   - $\vdash (A \Rightarrow B) \supset ((A \textit{ says } s) \supset (B \textit{ says } s))$.
     which is equivalent to $(A = B) \supset ((A \textit{ says } s) \equiv (B \textit{ says } s))$.

The Basic Logic
Semantics

Kripke Semantics

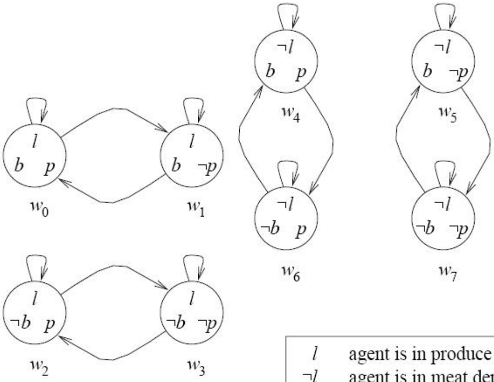
A structure $\mathcal{M}$ is a tuple $\langle \mathcal{W}, \omega_0, \mathcal{I}, \mathcal{J} \rangle$, where:

- $\mathcal{W}$ is a set (a set of possible worlds)
- ω_0 is distinguished element of $\mathcal{W}$
- $\mathcal{I}$ is an interpretation function
  $\mathcal{I} : PropositionSymbols \rightarrow \mathcal{P}(W)$.
  ($\mathcal{I}(s)$ is a set of worlds where the proposition symbol is true)
- $\mathcal{J}$ is an interpretation function
  $\mathcal{J} : Principals \rightarrow \mathcal{P}(\mathcal{W} \times \mathcal{W})$

The Basic Logic

Semantics

Example



l	agent is in produce department
$\neg l$	agent is in meat department
b	the bananas are yellow
$\neg b$	the bananas are green
p	the pork is fresh
$\neg p$	the pork is spoiled

The Basic Logic

Semantics

The meaning function $\mathcal{R}$ extends $\mathcal{J}$ as follows:

- $\mathcal{R}(A_i) = \mathcal{J}(A_i)$
- $\mathcal{R}(A \wedge B) = \mathcal{R}(A) \cup \mathcal{R}(B)$
- $\mathcal{R}(A|B) = \mathcal{R}(A) \circ \mathcal{R}(B)$

The Basic Logic
Semantics

The meaning function $\mathcal{E}$ extends $\mathcal{I}$ as follows:

- $\mathcal{E}(p_i) = \mathcal{I}(p_i)$
- $\mathcal{E}(\neg s) = \mathcal{W} - \mathcal{E}(s)$
- $\mathcal{E}(s \wedge s') = \mathcal{E}(s) \cap \mathcal{E}(s')$
- $\mathcal{E}(A \textit{ says } s) = \{\omega | \mathcal{R}(A)(\omega) \subseteq \mathcal{E}(s)\}$
- $\mathcal{E}(A \Rightarrow B) = \mathcal{W}$ if $\mathcal{R}(B) \subseteq \mathcal{R}(A)$ and $\emptyset$ otherwise
  $\mathcal{R}(C)(\omega) = \{\omega' | \omega \mathcal{R}(C) \omega'\}$

The Basic Logic

Semantics

Soundness

The axioms are sound, in the sense that if $\vdash s$ then $\models s$.

Completeness

Although useful for our application, the axioms are not complete.
For example, the formula
$C\ says\ (A \Rightarrow B)) \equiv ((A \Rightarrow B) \vee (C\ says\ false))$
is valid but not provable.

The Basic Logic

Semantics

Soundness

The axioms are sound, in the sense that if $\vdash s$ then $\models s$.

Completeness

Although useful for our application, the axioms are not complete.
For example, the formula
$C\ says\ (A \Rightarrow B)) \equiv ((A \Rightarrow B) \vee (C\ says\ false))$
is valid but not provable.

The Basic Logic
On Idempotence

- The idempotence of $|$ and *for* is intuitively needed.
  - $A|A = A$: A *says* A *says* s and A *says* s are equal.
  - $G|A$ in an ACL postulates $G|G|A$ and $G|G \Rightarrow G$.
  - Idempotence impose more complexity. e.g., it yields $(A \wedge B) \Rightarrow (B|A)$. On a request of $A \wedge B$ we need to check both $(A|B)$ and $(B|A)$.
- We unable to find a sensible condition on binary relations that would force idempotence and would be preserved by union and composition.

Corollary

The authors preferred to do without idempotence and rely on assumptions of the form $G|G \Rightarrow G$.

The Basic Logic
On Idempotence

- The idempotence of $|$ and *for* is intuitively needed.
  - $A|A = A$: A *says* A *says* s and A *says* s are equal.
  - $G|A$ in an ACL postulates $G|G|A$ and $G|G \Rightarrow G$.
  - Idempotence impose more complexity. e.g., it yields $(A \wedge B) \Rightarrow (B|A)$. On a request of $A \wedge B$ we need to check both $(A|B)$ and $(B|A)$.
- We unable to find a sensible condition on binary relations that would force idempotence and would be preserved by union and composition.

Corollary

The authors preferred to do without idempotence and rely on assumptions of the form $G|G \Rightarrow G$.

Roles

What Roles Are For?

- "Least Privilege" principle.
- For diminishing power of a user (A *as* R).
- For limiting untrusted software.

Roles, Groups, and Resources

- Roles may be related to Groups. e.g., G_{role} related to group G. A *as* G_{role} means A act in the role of member of G.
- We do allow roles related to groups but this relation is not formal.
- Roles may correspond to a set of resources.

Roles

What Roles Are For?

- "Least Privilege" principle.
- For diminishing power of a user (A *as* R).
- For limiting untrusted software.

Roles, Groups, and Resources

- Roles may be related to Groups. e.g., G_{role} related to group G. A *as* G_{role} means A act in the role of member of G.
- We do allow roles related to groups but this relation is not formal.
- Roles may correspond to a set of resources.

Roles
The Encoding

Definition (Role)

In the binary relation model, roles are subsets of the identity relations: $1 \Rightarrow R$.
A principal A in role R is defined as $A \textit{ as } R$ which is equal to $A|R$.

A special principal **1**, the *identity*, believes everything that is true and nothing that is not.

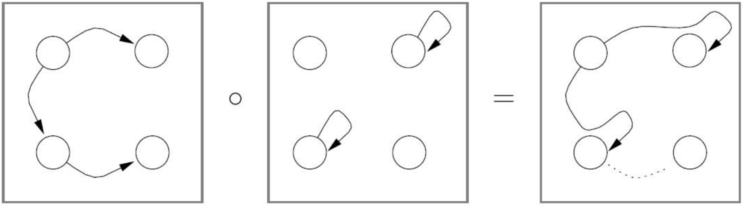
$$\mathcal{R}(1)(\omega) = \omega, \quad \forall \omega \in \mathcal{W}$$

Roles

The Encoding

Roles reduce privileges.

$$\mathcal{R}(A) \circ \mathcal{R}(R_1) \subseteq \mathcal{R}(A)$$



An arbitrary principal relation $\mathcal{R}(A)$ ...

... composed with a role relation $\mathcal{R}(R)$ ...

... gives a new relation that is always a subset of $\mathcal{R}(A)$.

Roles
The Encoding

Role Properties

- Monotonicity, multiplicativity, and associativity of $|$ offer formal advantages.
  - $|$ is monotonic: if $R \Rightarrow R'$ and $A \Rightarrow A'$, then $(A \ as \ R) \Rightarrow (A' \ as \ R')$.[role hierarchy is possible]
  - $|$ is multiplicative: $A \wedge B$ in a role R is identical to the conjunction of A and B both in role R. $(A \wedge B) \ as \ R = (A \ as \ R) \wedge (B \ as \ R)$. Similarly, $A \ as \ (R \wedge R') = (A \ as \ R) \wedge (A \ as \ R')$.
  - $|$ is associative: is useful for interaction between roles and delegation. $B|(A \ as \ R) = (B|A) \ as \ R$ and also $B \ for \ (A \ as \ R) = (B \ for \ A) \ as \ R$.

Roles
The Encoding

Role Properties

- All roles are:
  - idempotent $(R|R = R)$
  - commute with one another $(R|R' = R'|R)$.

  These yield the following:
  - $A \text{ as } R \text{ as } R = A \text{ as } R$
  - $A \text{ as } R \text{ as } R' = A \text{ as } R' \text{ as } R$
- We assume:

  $A \Rightarrow (A \text{ as } R)$ for all A.

Delegation

Definition (Delegation)

The ability of a principal A to give to another principal B the authority to act on A's behalf.

- Different mechanisms embody the concept of delegation in different settings.
- Delegation relies on authentication. It is often convenient to view delegation as independent.
- We consider 3 instances of delegations:
  - Delegation without certificates;
  - Delegation with certificates;
  - Delegation without delegate authentication.

Delegation

Definition (Delegation)

The ability of a principal A to give to another principal B the authority to act on A's behalf.

- Different mechanisms embody the concept of delegation in different settings.
- Delegation relies on authentication. It is often convenient to view delegation as independent.
- We consider 3 instances of delegations:
  - Delegation without certificates;
  - Delegation with certificates;
  - Delegation without delegate authentication.

Delegation

The Forms of Delegation

Assumptions for Delegation

- Synchronized clocks are available.
- All principals can perform digital signature.
- The formula K *says* X represents the certificate X encrypted under K^{-1}.
- A certificate authority S provides certificates for the principals' public keys.

Delegation
The Forms of Delegation

1. Delegation Without Certificate

B sends the signed request along with A's name: K_B *says* A *says* r

$\Downarrow$

When C receives r, must believe that K_S is S's public key: $K_S \Rightarrow S$

$\Downarrow$

C obtains a certificate encrypted under the inverse of K_S:
K_S *says* $(K_B \Rightarrow B)$

$\Downarrow$

These yield: S *says* $(K_B \Rightarrow B)$

Delegation

The Forms of Delegation

1. Delegation Without Certificate

B sends the signed request along with A's name: K_B *says* A *says* r

$\Downarrow$

When C receives r, must believe that K_S is S's public key: $K_S \Rightarrow S$

$\Downarrow$

C obtains a certificate encrypted under the inverse of K_S:
K_S *says* $(K_B \Rightarrow B)$

$\Downarrow$

These yield: S *says* $(K_B \Rightarrow B)$

Delegation
The Forms of Delegation

1. Delegation Without Certificate

B sends the signed request along with A's name: K_B *says* A *says* r

$\Downarrow$

When C receives r, must believe that K_S is S's public key: $K_S \Rightarrow S$

$\Downarrow$

C obtains a certificate encrypted under the inverse of K_S:
K_S *says* $(K_B \Rightarrow B)$

$\Downarrow$

These yield: S *says* $(K_B \Rightarrow B)$

Delegation

The Forms of Delegation

1. Delegation Without Certificate

B sends the signed request along with A's name: K_B *says* A *says* r

$\Downarrow$

When C receives r, must believe that K_S is S's public key: $K_S \Rightarrow S$

$\Downarrow$

C obtains a certificate encrypted under the inverse of K_S:
K_S *says* $(K_B \Rightarrow B)$

$\Downarrow$

These yield: S *says* $(K_B \Rightarrow B)$

$\Downarrow$

C trusts S for such statement: S *controls* $(K_B \Rightarrow B)$

$\Downarrow$

We obtain that C has B's key: $K_B \Rightarrow B$

$\Downarrow$

C sees a message as: K_B *says* A *says* r.

$\Downarrow$

C obtains: B *says* A *says* $r = (B|A)$ *says* r.

$\Downarrow$

C checks ACL for r. If $B|A$ exist in the ACL, r access is granted.

$\Downarrow$

C trusts S for such statement: S *controls* $(K_B \Rightarrow B)$

$\Downarrow$

We obtain that C has B's key: $K_B \Rightarrow B$

$\Downarrow$

C sees a message as: K_B *says* A *says* r.

$\Downarrow$

C obtains: B *says* A *says* $r = (B|A)$ *says* r.

$\Downarrow$

C checks ACL for r. If $B|A$ exist in the ACL, r access is granted.

⇓

C trusts S for such statement: S *controls* $(K_B \Rightarrow B)$

⇓

We obtain that C has B's key: $K_B \Rightarrow B$

⇓

C sees a message as: K_B *says* A *says* r.

⇓

C obtains: B *says* A *says* $r = (B|A)$ *says* r.

⇓

C checks ACL for r. If $B|A$ exist in the ACL, r access is granted.

$\Downarrow$

C trusts S for such statement: S *controls* $(K_B \Rightarrow B)$

$\Downarrow$

We obtain that C has B's key: $K_B \Rightarrow B$

$\Downarrow$

C sees a message as: K_B *says* A *says* r.

$\Downarrow$

C obtains: B *says* A *says* $r = (B|A)$ *says* r.

$\Downarrow$

C checks ACL for r. If $B|A$ exist in the ACL, r access is granted.

$\Downarrow$

C trusts S for such statement: S *controls* $(K_B \Rightarrow B)$

$\Downarrow$

We obtain that C has B's key: $K_B \Rightarrow B$

$\Downarrow$

C sees a message as: K_B *says* A *says* r.

$\Downarrow$

C obtains: B *says* A *says* $r = (B|A)$ *says* r.

$\Downarrow$

C checks ACL for r. If $B|A$ exist in the ACL, r access is granted.

Delegation
The Forms of Delegation

It is preferable for A to issue a delegation certificate that proves the delegation to B.

2. Delegation With Certificate

After mutual authentication, A issues a certificate to B under A's key: K_A *says* $(B$ *serves* $A)$

$\Downarrow$

Checking of public-key certificates from S yields: A *says* $(B$ *serves* $A)$

Delegation
The Forms of Delegation

It is preferable for A to issue a delegation certificate that proves the delegation to B.

2. Delegation With Certificate

After mutual authentication, A issues a certificate to B under A's key: K_A *says* $(B$ *serves* $A)$

$$\Downarrow$$

Checking of public-key certificates from S yields:
A *says* $(B$ *serves* $A)$

$\Downarrow$

If C trusts A on this statement, C gets:
$((B|A)\ says\ r) \wedge (B\ serves\ A)$

$\Downarrow$

In our theories, this implies: $(B\ for\ A)\ says\ r$

$\Downarrow$

C checks ACL for r. If $B\ for\ A$ exist in the ACL, r access is granted.

$\Downarrow$

If C trusts A on this statement, C gets:
$((B|A)\ says\ r) \wedge (B\ serves\ A)$

$\Downarrow$

In our theories, this implies: $(B\ for\ A)\ says\ r$

$\Downarrow$

C checks ACL for r. If $B\ for\ A$ exist in the ACL, r access is granted.

$\Downarrow$

If C trusts A on this statement, C gets:
$((B|A)\ says\ r) \wedge (B\ serves\ A)$

$\Downarrow$

In our theories, this implies: $(B\ for\ A)\ says\ r$

$\Downarrow$

C checks ACL for r. If $B\ for\ A$ exist in the ACL, r access is granted.

Delegation

The Forms of Delegation

Omitting the authentication between B and C, leaving the responsibility of authenticating B solely to A.

3. Delegation Without Delegate Authentication

After mutual authentication, A issues a certificate to B under A's key. The cert. includes a key K_d and B's name.[K_d^-1 is secret key]

$\Downarrow$

B can present A's certificate to C along with request under the delegation key K_d.

$\Downarrow$

C knows that B has requested r on behalf of A. Now C checks the ACL for r.

Delegation

The Forms of Delegation

Omitting the authentication between B and C, leaving the responsibility of authenticating B solely to A.

3. Delegation Without Delegate Authentication

After mutual authentication, A issues a certificate to B under A's key. The cert. includes a key K_d and B's name.[K_d^-1 is secret key]

⇓

B can present A's certificate to C along with request under the delegation key K_d.

⇓

C knows that B has requested r on behalf of A. Now C checks the ACL for r.

Delegation

The Forms of Delegation

Omitting the authentication between B and C, leaving the responsibility of authenticating B solely to A.

3. Delegation Without Delegate Authentication

After mutual authentication, A issues a certificate to B under A's key. The cert. includes a key K_d and B's name.[K_d^-1 is secret key]

⇓

B can present A's certificate to C along with request under the delegation key K_d.

⇓

C knows that B has requested r on behalf of A. Now C checks the ACL for r.

Delegation
The Encoding

Delegation Properties

- *for* is monotonic
- *for* is multiplicative. This follows:
  $(B \wedge B')\ \mathit{for}\ (A \wedge A') = (B\ \mathit{for}\ A) \wedge (B\ \mathit{for}\ A') \wedge (B'\ \mathit{for}\ A) \wedge (B'\ \mathit{for}\ A')$
- $B\ \mathit{for}\ A$ is always defined, even if A has not delegated to B. We have:
  - $(B|A) \wedge (C\ \mathit{for}\ A) \Rightarrow ((B \wedge C)\ \mathit{for}\ A)$
  - $(B|A) \wedge (C\ \mathit{for}\ A) \Rightarrow (B\ \mathit{for}\ A)$

Delegation

The Encoding

Delegation Properties

- *for* possesses a weak associativity property:

$$(C \textit{ for } (B \textit{ for } A)) \Rightarrow (C \textit{ for } B) \textit{ for } A$$

- $(A \wedge (B|A)) \Rightarrow (B \textit{ for } A)$, because $|$ is multiplicative.
- If $A = A|A$ then $A = A \textit{ for } A$, the idempotence of $|$ implies the idempotence of *for*: $A \textit{ for } A = A$.

Extensions

The Basic logic can be extended in many ways, such as:

1. Adding Intersection
2. Adding Subtraction
3. Adding Variables

Extensions
Intersection

An intersection operation $\cap$ permits the construction of groups from groups.

- It can only applied to atomic symbols. $(A_j|A_i) \cap A_k$ is not valid
- Conjunction is strictly weaker than intersection: $(A \cap B) \Rightarrow (A \wedge B)$, $(A \wedge B) \nRightarrow (A \cap B)$
- Application: restricting access to only a particular member of a group.
  The ACL entry $A \wedge G$ grants access to
  - A , if A is a member of G,
  - $A \wedge B$, if A and B are members of G.

  In contrast, $A \cap G$ just grant access to
  - A , if A is a member of G.

Extensions
Subtraction

Group subtraction $(-)$ provides the ability to specify that certain named individuals or subgroups with in a supergroup should be denied access.

- $G" - G$ means all members of $G"$, except for those member of $G"$ only via G.
- In distributed systems nonmembership may not be available.
- It is not very useful in distributed systems, just for groups which are managed by centralized subsystems.

Extensions
Variables

Adding variables can increase the expressiveness, but more complexity.

- Variables can be included in ACLs.
- Example:s Existing $(y|x)$ *controls* s in an ACL, results in granting access to $(B|A)$, by matching
  - x with A,
  - y with B.

Access Control Decision Algorithm

- The calculus of principals is undecidable.
- There are decidable variant of this caculus.

Access Control Decision Algorithm

A General Access Control Problem

The Parts of an Instance of A.C. Decision Problem

- A principal P that is making the request.

  [An expression in the calculus of principals]

- A statement s represents what is being requested or asserted.

  [The service provider does not need to derive it logically from other statements.]

- Assumptions state implications among principals.

  [Assumptions about group membership (e.g., $P_i \Rightarrow G_i$) and idempotence (e.g., $G|G \Rightarrow G$)]

- Roles $R_0, R_1, \ldots$

  [Certain atomic symbols which may be obvious from their name]

- An ACL for s.

  [A list of expressions $E_0, E_1, \ldots$ of principals that are trusted on s.]

Access Control Decision Algorithm

A General Access Control Problem

The Basic Problem of Access Control

Deciding whether by having

- $\bigwedge_i (P_i \Rightarrow G_i)$, derived from the assumptions
- $\bigwedge_i (E_i \textit{ controls } s)$, derived from the ACL

can we imply $P \textit{ controls } s$.

Outline

The Problem

Problem

- Information are governed by Policies.
- When information is shared, it is necessary compare, contrast, and compare the underlying security policies.
- Problems arise in doing so is diversity and incompatibility of
  - requirements,
  - security models,
  - security policies,
  - enforcement mechanisms.

Proposed Solution

Presenting a policy composition framework at the propositional level for access control.

The Problem

Problem

- Information are governed by Policies.
- When information is shared, it is necessary compare, contrast, and compare the underlying security policies.
- Problems arise in doing so is diversity and incompatibility of
  - requirements,
  - security models,
  - security policies,
  - enforcement mechanisms.

Proposed Solution

Presenting a policy composition framework at the propositional level for access control.

Basic Concepts

Definition (Permission and Permission Set)

- A permission is an ordered pair $(object, \pm action)$.
  Example: $(file1, +read)$, or $(file1, -write)$
- A permission set is a set of such permissions.
  Example: $\{(file1, +read), (file1, -write)\}$

Basic Concepts

Definition (Permission and Permission Set)

- A permission is an ordered pair $(object, \pm action)$.
  Example: $(file1, +read)$, or $(file1, -write)$
- A permission set is a set of such permissions.
  Example: $\{(file1, +read), (file1, -write)\}$

Basic Concepts

Definition (Nondeterministic Transformers)

- Permission Set Transformation: $(s, PermSet) \mapsto (s, PermSet')$
- Nondeterministic Transformers of Permission Sets:
  $(s, SetOfPermSet) \mapsto (s, SetOfPermSet')$
  Example:
  $(A, \emptyset) \mapsto (A, \{\{(f1, +r), (f1, -w)\}, \{(f1, -r), (f1, +w)\}\})$
  It allows A two choices:
  $\{(f1, +r), (f1, -w)\}$ or
  $\{(f1, -r), (f1, +w)\}$

Basic Concepts

Definition (Policy)

A policy is interpreted as nondeterministic transformers on permission set assignments to subjects.
Operations on policies are interpreted as relational or set theoretical operations on such nondeterministic transformers.

Basic Concepts

Internal vs. External Operator

There are two types of policy composition operators. Suppose $P_1 = (C_1, \emptyset) \mapsto (C_1, \{(check, +read), (check, +write)\})$ and $P_2 = (C_1, \emptyset) \mapsto (C_1, \{(check, +read), (check, +approval)\})$.

1. Internal Operators: E.g., internal union, $P_1 \cup P_2 = (C_1, \emptyset) \mapsto (C_1, \{(check, +read), (check, +write), (check, +approval)\})$.
2. External Operators: E.g., external union, $P_1 \sqcup P_2 = (C_1, \emptyset) \mapsto (C_1, \{\{(check, +read), (check, +write)\}, \{(check, +read), (check, +approval)\}\})$.

Basic Concepts

Internal vs. External Operator

There are two types of policy composition operators. Suppose $P_1 = (C_1, \emptyset) \mapsto (C_1, \{(check, +read), (check, +write)\})$ and $P_2 = (C_1, \emptyset) \mapsto (C_1, \{(check, +read), (check, +approval)\})$.

1. Internal Operators: E.g., internal union, $P_1 \cup P_2 = (C_1, \emptyset) \mapsto (C_1, \{(check, +read), (check, +write), (check, +approval)\})$.
2. External Operators: E.g., external union, $P_1 \sqcup P_2 = (C_1, \emptyset) \mapsto (C_1, \{\{(check, +read), (check, +write)\}, \{(check, +read), (check, +approval)\}\})$.

Basic Concepts

Individual vs. Set Proposition

For conditional authorization we have two types of propositions:

1. Individual Proposition: applies to an object.
   E.g., *Can read any check with a face value greater than $ 10,000.*
2. Set Proposition: applies to a set of objects.
   E.g., *if the total value of all checks to be read by a clerk is more than $10,000.*

Basic Concepts

Individual vs. Set Proposition

For conditional authorization we have two types of propositions:

1. Individual Proposition: applies to an object.
   E.g., *Can read any check with a face value greater than $ 10,000.*
2. Set Proposition: applies to a set of objects.
   E.g., *if the total value of all checks to be read by a clerk is more than $10,000.*

Syntax
Definition

- P_{atomic} is terminal symbol taken from a set of atomic policies $\mathcal{POL}$
- ϕ_{atomic} is a terminal symbol taken from a set of atomic propositions $\mathcal{PROP}$
- Φ_{atomic} is a terminal symbol taken from a set of atomic set (second order) propositions $\mathcal{SETP}$
- Definition of policies, propositions and set propositions:

$$
\begin{aligned}
P := & P_{atomic} | P \sqcup P | P \sqcap P | P \boxminus P | \Lsh P | (\phi :: P) | (P \shortparallel \phi) | P \cup P | \\
& P \cap P | P - P | \neg P | (\phi : P) | (p \shortmid \phi) | \odot P | P ; P | P^* | \\
& min(P) | max(P) | oCom(P) | cCom(P) \\
\phi := & \phi_{atomic} | \phi \wedge \phi | \phi \vee \phi | \neg \phi \\
\Phi := & \Phi_{atomic} | \Phi \wedge \Phi | \Phi \vee \Phi | \neg \Phi
\end{aligned}
$$

Syntax
Introduction to the Notations

- $\sqcup$ $\rightsquigarrow$ external union
- $\sqcap$ $\rightsquigarrow$ external intersection
- $\boxminus$ $\rightsquigarrow$ external difference
- $\Lsh$ $\rightsquigarrow$ external negation
- $\phi ::$ $\rightsquigarrow$ external scoping
- $\shortparallel \phi$ $\rightsquigarrow$ external provisioning
- $;$ $\rightsquigarrow$ sequential composition
- $*$ $\rightsquigarrow$ closure (extension of sequential composition)
- $\cup$ $\rightsquigarrow$ internal union

Syntax

Introduction to the Notations

- $\cap$ $\leadsto$ internal intersection
- $-$ $\leadsto$ internal difference
- $\neg$ $\leadsto$ internal negation
- $\phi :$ $\leadsto$ internal scoping
- $\imath\,\phi$ $\leadsto$ internal provisioning
- $\odot$ $\leadsto$ invalidate permissions
- min, max $\leadsto$ for conflict resolution, denial/permission take precedence
- $oCom, cCom$ $\leadsto$ open and close policy

Semantics

Basic Building Blocks of the Semantics

Definition (Subjects, Objects, and Permissions)

(1) *Subjects*: Let $\mathcal{S} = \{s_i, i \in \mathbb{N}\}$ be a set of subjects.

(2) *Objects*: Let $\mathcal{O} = \{o_i, i \in \mathbb{N}\}$ be a set of objects.

(3) *Signed Actions*: Let $\mathcal{A} = \{a_i, i \in \mathbb{N}\}$ be a set of action terms. Then $\mathcal{A}^{\pm} = \mathcal{A}^{+} \cup \mathcal{A}^{-}$, where $\mathcal{A}^{+} = \{+a : a \in \mathcal{A}\}$ and $\mathcal{A}^{-} = \{-a : a \in \mathcal{A}\}$ is the set of signed action terms.

(4) *Roles*: Let $\mathcal{R} = \{R_i, i \in \mathbb{N}\}$ be a set of roles.

Semantics

Basic Building Blocks of the Semantics

Definition (Subjects, Objects, and Permissions)

(1) *Subjects*: Let $\mathcal{S} = \{s_i, i \in \mathbb{N}\}$ be a set of subjects.

(2) *Objects*: Let $\mathcal{O} = \{o_i, i \in \mathbb{N}\}$ be a set of objects.

(3) *Signed Actions*: Let $\mathcal{A} = \{a_i, i \in \mathbb{N}\}$ be a set of action terms. Then $\mathcal{A}^{\pm} = \mathcal{A}^{+} \cup \mathcal{A}^{-}$, where $\mathcal{A}^{+} = \{+a : a \in \mathcal{A}\}$ and $\mathcal{A}^{-} = \{-a : a \in \mathcal{A}\}$ is the set of signed action terms.

(4) *Roles*: Let $\mathcal{R} = \{R_i, i \in \mathbb{N}\}$ be a set of roles.

Semantics

Basic Building Blocks of the Semantics

Definition (Subjects, Objects, and Permissions)

(1) *Subjects*: Let $\mathcal{S} = \{s_i, i \in \mathbb{N}\}$ be a set of subjects.

(2) *Objects*: Let $\mathcal{O} = \{o_i, i \in \mathbb{N}\}$ be a set of objects.

(3) *Signed Actions*: Let $\mathcal{A} = \{a_i, i \in \mathbb{N}\}$ be a set of action terms. Then $\mathcal{A}^{\pm} = \mathcal{A}^{+} \cup \mathcal{A}^{-}$, where $\mathcal{A}^{+} = \{+a : a \in \mathcal{A}\}$ and $\mathcal{A}^{-} = \{-a : a \in \mathcal{A}\}$ is the set of signed action terms.

(4) *Roles*: Let $\mathcal{R} = \{R_i, i \in \mathbb{N}\}$ be a set of roles.

Semantics

Basic Building Blocks of the Semantics

Definition (Subjects, Objects, and Permissions)

(1) *Subjects*: Let $\mathcal{S} = \{s_i, i \in \mathbb{N}\}$ be a set of subjects.

(2) *Objects*: Let $\mathcal{O} = \{o_i, i \in \mathbb{N}\}$ be a set of objects.

(3) *Signed Actions*: Let $\mathcal{A} = \{a_i, i \in \mathbb{N}\}$ be a set of action terms. Then $\mathcal{A}^{\pm} = \mathcal{A}^{+} \cup \mathcal{A}^{-}$, where $\mathcal{A}^{+} = \{+a : a \in \mathcal{A}\}$ and $\mathcal{A}^{-} = \{-a : a \in \mathcal{A}\}$ is the set of signed action terms.

(4) *Roles*: Let $\mathcal{R} = \{R_i, i \in \mathbb{N}\}$ be a set of roles.

Semantics

Basic Building Blocks of the Semantics

Definition (Subjects, Objects, and Permissions)

(5) *Authorizations*: $(s, PermSet)$ is an authorization if one of the following conditions holds:

- s is either a subject or a role and $PermSet \subseteq \mathcal{O} \times \mathcal{A}^{\pm}$
- s is a subject and $PermSet$ is a role. The notation $\mathcal{AU}(\mathcal{S}, \mathcal{R}, \mathcal{O}, \mathcal{A})$ denotes the set of all authorizations.

(6) *Permission-Prohibition Triples*: $(s, o, \pm a)$ where $s \in \mathcal{S}, o \in \mathcal{O}, a \in \mathcal{A}^{\pm}$. The notation $\mathcal{T}(\mathcal{S}, \mathcal{R}, \mathcal{O}, \mathcal{A})$ denotes the set of all permission-prohibition triples.

Semantics

Basic Building Blocks of the Semantics

Definition (Subjects, Objects, and Permissions)

(5) *Authorizations*: $(s, PermSet)$ is an authorization if one of the following conditions holds:

- s is either a subject or a role and $PermSet \subseteq \mathcal{O} \times \mathcal{A}^{\pm}$
- s is a subject and $PermSet$ is a role. The notation $\mathcal{AU}(\mathcal{S}, \mathcal{R}, \mathcal{O}, \mathcal{A})$ denotes the set of all authorizations.

(6) *Permission-Prohibition Triples*: $(s, o, \pm a)$ where $s \in \mathcal{S}, o \in \mathcal{O}, a \in \mathcal{A}^{\pm}$. The notation $\mathcal{T}(\mathcal{S}, \mathcal{R}, \mathcal{O}, \mathcal{A})$ denotes the set of all permission-prohibition triples.

Semantics

Basic Building Blocks of the Semantics

Definition (State)

A state is a pair of mappings $(M_{prop}, M_{setProp})$, where $M_{prop} : \mathcal{PROP} \mapsto \mathcal{P}(\mathcal{T})$ and $M_{setProp} : \mathcal{SETP} \mapsto \mathcal{P}(\mathcal{P}(\mathcal{T}))$.

Definition (Interpreting Atomic Policies)

An interpretation of atomic policies $M_{AtPolicy}$ is a mapping from $STATES \times \mathcal{POL} \times (\mathcal{S} \cup \mathcal{R}) \times \mathcal{P}(\mathcal{O} \times \mathcal{A}^{\pm}) \mapsto (\mathcal{S} \cup \mathcal{R}) \times \mathcal{P}(\mathcal{P}(\mathcal{O} \times \mathcal{A}^{\pm}))$ satisfying the condition that $s' = s$ for any $(s', PermSet') \in M_{AtPolicy}(St)(p)(s, PermSet)$.

Semantics

Basic Building Blocks of the Semantics

Definition (State)

A state is a pair of mappings $(M_{prop}, M_{setProp})$, where $M_{prop} : \mathcal{PROP} \mapsto \mathcal{P}(\mathcal{T})$ and $M_{setProp} : \mathcal{SETP} \mapsto \mathcal{P}(\mathcal{P}(\mathcal{T}))$.

Definition (Interpreting Atomic Policies)

An interpretation of atomic policies $M_{AtPolicy}$ is a mapping from $STATES \times \mathcal{POL} \times (\mathcal{S} \cup \mathcal{R}) \times \mathcal{P}(\mathcal{O} \times \mathcal{A}^{\pm}) \mapsto (\mathcal{S} \cup \mathcal{R}) \times \mathcal{P}(\mathcal{P}(\mathcal{O} \times \mathcal{A}^{\pm}))$ satisfying the condition that $s' = s$ for any $(s', PermSet') \in M_{AtPolicy}(St)(p)(s, PermSet)$.

Semantics

Basic Building Blocks of the Semantics

Definition (Negating Permissions Sets)

If $PS \subseteq \mathcal{O} \times \mathcal{A}_{\pm}$ is a permission set, then
$-PS = \{(o, -a) : (o, +a) \in PS\} \cup \{(o, +a) : (o, -a) \in PS\}$.
If $r \in \mathcal{R}$ is a role, then $(o, -a) \in -r$ iff $(o, +a) \in r$ and $(o, +a) \in -r$ iff $(o, -a) \in r$.

Semantics

Interpreting Policy Operators

Definition (Interpreting Policy Operators)

An interpretation $M_{AtPolicy}$ of atomic policies is extended to an interpretation M_{policy} nonatomic policies using the following definition:

(1) $M_{policy}(St)(\mathrm{p}) = M_{AtPolicy}(St)(\mathrm{p})$ for all atomic policies p and states St.

(2) $M_{policy}(St)(\mathrm{p} \sqcup \mathrm{q})(s, PermSet) =$
$M_{policy}(St)(\mathrm{p})(s, PermSet) \cup M_{policy}(St)(\mathrm{q})(s, PermSet)$.

(3) $M_{policy}(St)(\mathrm{p} \sqcap \mathrm{q})(St)(s, PermSet) =$
$M_{policy}(St)(\mathrm{p})(s, PermSet) \cap M_{policy}(St)(\mathrm{q})(s, PermSet)$.

(4) $M_{policy}(St)(\mathrm{p} \boxminus \mathrm{q})(s, PermSet) =$
$M_{policy}(St)(\mathrm{p})(s, PermSet) \setminus M_{policy}(St)(\mathrm{q})(s, PermSet)$.

Semantics

Interpreting Policy Operators

(5) $M_{policy}(St)(\Lsh \mathrm{p})(s, PermSet) = \{(s, PS) : PS \in \mathcal{PS}\} \setminus M_{policy}(St)(\mathrm{p})(s, PermSet)$.

(6) $M_{policy}(St)(\phi :: \mathrm{p})(s, PermSet) = M_{policy}(St)(\mathrm{p})(s, PermSet)$, if $(s, PermSet) \in M_{setProp}(St)(\phi)$, and $\emptyset$ otherwise.

(7) $M_{policy}(St)(\mathrm{p})(\mathrm{p} \| \phi)(s, PermSet) = M_{prop}(St)(\mathrm{p})(s, PermSet) \cap M_{setProp}(St)(\phi)$.

(8) $M_{policy}(St)(\mathrm{p}; \mathrm{q})(s, PermSet) = \{(s, PermSet_1) \in M_{policy}(St')(\mathrm{q})(s, PermSet_2)$: for some $(s, PermSet_2) \in M_{policy}(St)(\mathrm{p})(s, PermSet)\}$.

(9) To define $M_{policy}(St)(\mathrm{p}^*)(s, PermSet)$, inductively define $M_{policy}(St)(\mathrm{p}^n)$ using the following rules:
- (a) $M_{policy}(St)(\mathrm{p}^1) = M_{policy}(St)(\mathrm{p})$.
- (b) $M_{policy}(St)(\mathrm{p}^{n+1}) = M_{policy}(St)((\mathrm{p}; \mathrm{p}^n) \cup \mathrm{p}^n)$.
- (c) $M_{policy}(St)(\mathrm{p}^*) = \bigcup_{n \in \omega} M_{policy}(St)(\mathrm{p}^n)$.

(10) $M_{policy(St)}(\mathrm{p} \cup \mathrm{q})(s, PermSet) = \{(s, PermSet_{\mathrm{p}} \cup PermSet_{\mathrm{q}}) : (s, PermSet_{\mathrm{p}}) \in M_{policy}(St)(\mathrm{p})(s, PermSet_{\mathrm{p}})$ and $(s, PermSet_{\mathrm{q}}) \in M_{policy}(St)(\mathrm{q})(s, PermSet_{\mathrm{q}})\}$

Semantics

Interpreting Policy Operators

(11) $M_{policy}(St)(\mathrm{p} \cap \mathrm{q})(s, PermSet) = \{(s, PermSet_{\mathrm{p}} \cap PermSet_{\mathrm{q}}) : (s, PermSet_{\mathrm{p}}) \in M_{policy}(St)(\mathrm{p})(s, PermSet_{\mathrm{p}})$ and $(s, PermSet_{\mathrm{q}}) \in M_{policy}(St)(\mathrm{q})(s, PermSet_{\mathrm{q}})\}$

(12) $M_{policy}(St)(\mathrm{p} - \mathrm{q})(s, PermSet) = \{(s, PermSet_{\mathrm{p}} \setminus PermSet_{\mathrm{q}}) : (s, PermSet_{\mathrm{p}}) \in M_{policy}(St)(\mathrm{p})(s, PermSet_{\mathrm{p}})$ and $(s, PermSet_{\mathrm{q}}) \in M_{policy}(St)(\mathrm{q})(s, PermSet_{\mathrm{q}})\}$

(13) $M_{policy}(St)(\neg \mathrm{p})(s, PermSet) = \{(s, -PermSet') : (s, PermSet) \in M_{policy}(St)(\mathrm{p})(s, PermSet,')\}$.

(14) $M_{policy}(St)(\phi : \mathrm{p})(s, PermSet\{(o, a) \notin M_{prop}(St)(\phi)\}) = M$ if $M_{policy}(St)(\mathrm{p})(s, PermSet) = M$.

(15) $M_{policy}(St)(\mathrm{p} \setminus \phi)(s, PermSet) = M$ provided that $M_{policy}(St)(\mathrm{p})(s, PermSet) = \{(s, PermSt'\{(o, a) : (s, o, a) \notin M_{prop}(St')(\phi)\})\}$, where $M_{policy}(St)(\mathrm{p}) = St'$.

(16) $M_{policy}(St)(\mathsf{max}(\mathrm{p}))(s, PermSet) = \{(s, PermSet_1) : PermSet_1 = PermSet_2 \setminus \{(o, -a) : (o, +a), (o, -a) \in PermSet_2\}$ for some $(s, PermSet_2) \in M_{policy}(St)(\mathrm{p})(s, PermSet)\}$.

Semantics

Interpreting Policy Operators

(17) $M_{policy}(St)(\mathsf{min}(\mathrm{p}))(s, PermSet) = \{(s, PermSet_1) : PermSet_1 = PermSet_2 \setminus \{(o,+a) : (o,+a),(o,-a) \in PermSet_2\}$ for some $(s, PermSet_2) \in M_{policy}(St)(\mathrm{p})(s, PermSet)\}$.

(18) $M_{policy}(St)(\odot(\mathrm{p}))(s, PermSet) = (s, \emptyset)$.

(19) $M_{policy}(St)(\mathsf{cCom}(\mathrm{p}))(St)(s, PermSet) = \{(s, PermSet_1) : PermSet_1 = PermSet_2 \cup \{(o,-a) : (o,-a),(o,+a) \notin PermSet_2\}$ for some $(s, PermSet_2) \in M_{policy}(St)(\mathrm{p})(s, PermSet)$.

(20) $M_{policy}(St)(\mathsf{oComp})(s, PermSet) = \{(s, PermSet_1) : PermSet_1 = PermSet_2 \cup \{(o,+a) : (o,-a),(o,+a) \notin PermSet_2\}$ for some $(s, PermSet_2) \in M_{policy}(St)(\mathrm{p})(s, PermSet)$.

The Algebra of Operators
The Algebra of External Operators

Theorem (1)

External operator namely, $(\mathcal{POL}, \sqcup, \sqcap, \urcorner, 1_{pol}, 0_{pol})$ *form a Boolean algebra under the following interpretation of* 1_{pol} *and* 0_{pol}

$$M_{policy}(St)(1_{pol})(s, PermSet) = (s, \mathcal{P}(PS))$$
$$M_{policy}(St)(0_{pol})(s, PermSet) = (s, \emptyset)$$

for each $s \in \mathcal{S} \cup \mathcal{R}$ *and* $St \in STATES$.

In summary, theorem 1 says that we can manipulate policy operators just as operators in propositional logic and therefore use the same disjunctive or conjunctive normal forms etc.

The Algebra of Operators

The Algebra of External Operators

Theorem (1)

External operator namely, $(\mathcal{POL}, \sqcup, \sqcap, \urcorner, 1_{pol}, 0_{pol})$ *form a Boolean algebra under the following interpretation of* 1_{pol} *and* 0_{pol}

$$M_{policy}(St)(1_{pol})(s, PermSet) = (s, \mathcal{P}(PS))$$
$$M_{policy}(St)(0_{pol})(s, PermSet) = (s, \emptyset)$$

for each $s \in \mathcal{S} \cup \mathcal{R}$ *and* $St \in STATES$.

In summary, theorem 1 says that we can manipulate policy operators just as operators in propositional logic and therefore use the same disjunctive or conjunctive normal forms etc.

The Algebra of Operators

The Algebra of External Operators

Theorem (2)

Let $p, p_1, p_2,$ and p_3 be policy terms and $I = (M_{policy}, M_{prop}, M_{setProp})$ be an interpretation. Then the following properties are valid in I:

1. *Idempotency of conjunctions and disjunctions:*
   - $p \sqcup p = p$
   - $p \sqcap p = p$
2. *Distributivity of composition over unions and intersections:*
   - $(p_1 \sqcup p_2); p_3 = (p_1; p_2) \sqcup (p1; p_3)$
   - $p_1; (p_2 \sqcup p_3) = (p_1; p_2) \sqcup (p_1; p_3)$
   - $(p_1 \sqcap p_2); p_3 \subseteq (p_1; p_3) \sqcap (p_2; p_3)$
   - $p_1; (p_2 \sqcap p_3) \subseteq (p_1; p_2) \sqcap (p_1; p_3)$

The Algebra of Operators

The Algebra of External Operators

Theorem (2)

3. *properties of the scoping operator*
   - $\Phi :: (\Psi :: p) == (\Phi \wedge \Psi) :: p$
   - $\Phi :: (p_1 \sqcup p_2) = (\Phi :: p_1) \sqcup (\Phi :: p_2)$
   - $\Phi :: (p_1 \sqcap p_2) = (\Phi :: p_1) \sqcap (\Phi :: p_2)$
   - $\Phi :: (p_1 ; p_2) = (\Phi :: p_1); p_2$
   - $\Phi :: (p_1 \boxminus p_2) = (\Phi :: p_1) \boxminus (\Phi :: p_2)$
4. *Properties of the provisioning operator*
   - $(p \shortparallel \Phi) \shortparallel \Psi = p \shortparallel (\Phi \wedge \Psi)$
   - $(p_1 \sqcup p_2) \shortparallel \Phi = (p_1 \shortparallel \Phi) \sqcup (p_2 \shortparallel \Phi)$
   - $(p_1 \sqcap p_2) \shortparallel \Phi = (p_1 \shortparallel \Phi) \sqcap (p_2 \shortparallel \Phi)$
   - $(p_1 ; p_2) \shortparallel \Phi = p_1 ; (p_2 \shortparallel \Phi)$
   - $(p_1 \boxminus p_2) \shortparallel \Phi = (p_1 \shortparallel \Phi) \boxminus (p_2 \shortparallel \Phi)$

The Algebra of Operators

The Algebra of External Operators

- In summery, theorem 2 shows how policy operators interact with each other.
- It also shows that scoping and provisioning operators distribute over unions, intersections, and sequencing operators. So, they can be used to express composed policies.

The Algebra of Operators
The Algebra of Internal Operators

- Internal operators do not satisfy $p \cup p = p$ and $p \cap p = p$, So they do not form a Boolean algebra.
- There are some properties for internal operators which are hold for all policies.

Determinism, Consistency,and Completeness

Determinism

Determinism

1. Determinism with respect to Authorizations
2. Determinism with respect to an Interpretation
3. Deterministic Policy
4. Ultra Deterministic Policy

Determinism is closed under some (and not all) internal and external operators.

Determinism, Consistency,and Completeness

Consistency

Consistency

1. Contradictory Permission Sets
2. Policies Consistence for Authorizations
3. Policies Consistent with respect to an Interpretation
4. Consistent Policy

Consistency is closed under some (and not all) internal and external operators.

Determinism, Consistency,and Completeness

Completeness

Completeness

1. Complete Permission Sets
2. Policies Complete for Authorizations
3. Policies Complete with respect to an Interpretation
4. Complete Policy

Completeness is closed under some (and not all) internal and external operators.

Outline

- *Advantages of using logics in security*
  [Clean Foundation, Flexibility, Declarativeness, Ability of Verification, Independency from Implementation.]
- *A sample of logic-based policy specification*: A calculus for access control
- *A sample of algebraic approach in policy composition*: A propositional algebra for access control

Outline

References

(1) Piero A. Bonatti, Pierangela Samarati, **Logics for Authorization and A Security**, First chapter of Logics for Emerging Applications of Databases, J. Chomicki, R. Vander Meyden, G. Saake (eds.), LNCS, Springer-Verlag, 2003.

(2) Martin Abadi, Michael Brrows, Butler Lampson, **A Calculus for Access Control**, ACM Transactions on Programming Languages and Systems, Vol 15, No 4, pp. 706-734, Sep 1993.

(3) Jon Howell, David Kotz, **A Formal Semantics for SPKI**, Technical Report TR 2000-363, Department of Computer Science, Dartmouth College, March 2000.

References

(4) Duminda Wijesekera, Sushil Jajodia, **A Propositional Algebra for Access Control**, ACM Transaction on Information and System Security, Vol 6, No 2, pp. 286-325, May 2003.

(5) Piero Bonatti, Sabrina De Capitani di, Pierangela Samarati, **A Modular Approach to Composing Access Control Policies**, In Proceedings of the 7th ACM Conference on Computer and Communication Security (CCS2000), pp. 164-173, Athens, Greece, Nov 2000.

(6) Thomas Y.C. Woo, Simon S. Lam, **Authorization in Distributed Systems: A New Approach**, Journal of Computer Security, Vol 2, No 3, pp. 107-136, 1993.

Thanks for your attention ...

Questions?