Logic and Logic Programming in Distributed Access Control (Part One)

Ninghui Li
Department of Computer Science
and CERIAS
Purdue University

Outline

- A brief introduction to trust management
- Logic-based semantics for SDSI

The Trust-Management (TM) Approach

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

Common characteristics of TM systems

- Use public-key certificates for non-local statements
- Treat public keys as principals to be authorized
  - authentication consists of verifying signatures

Digital Signature Scheme

- Key space: a set of key pairs (K, K^{-1})
  - K is the verification key and is publicly available
  - K^{-1} is the signing key and is kept private
- A signing algorithm **sig**
  - **sig**(K^{-1}, M) outputs a digital signature on M
- A verification algorithm **ver**
  - **ver(**K, M, σ) outputs yes or no
  - **ver**$(K, M, \text{sig}(K^{-1}, M)) = \text{yes}$
  - w/o knowing K^{-1}, it is difficult to find σ s.t. **ver**(K,M,σ)=yes

Public-Key Certificates

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

Early Trust Management Langugaes

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

Datalog-based Trust Management Languages

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

Other Closely Related Logic-based Security Languages

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

History of SPKI/SDSI

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

An Example in SDSI 2.0

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

4-tuple Reduction in RFC 2693

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

Applying 4-tuple Reduction in the Example

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

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

Papers on Semantics for SPKI/SDSI

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

What is a Semantics?

- Elements of a semantics
  - syntax for statements
  - syntax for queries
  - an entailment relation that determines whether a query Q is true given a set P of statements

Why a Formal Semantics?

- What can we gain by a formal semantics
  - understand what queries can be answered
  - defines the entailment relation in a way that is precise, easy to understand, and easy to compute
- How can one say a semantics is good
  - subjective metrics:
    - simple, natural, close to original intention
  - defines answers to a broad class of queries
  - can use existing work to provide efficient deduction procedures for answering those queries

Concepts in SDSI

- Concepts

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

Statements in SDSI

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

A Rewriting Semantics for SDSI

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

Deduction Based on the Rewriting Semantics (1)

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

Deduction Based on the Rewriting Semantics (2)

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

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

Deduction Based on the Rewriting Semantics (3)

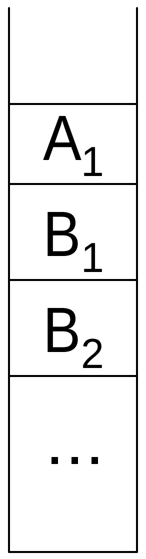
- [Li, Winsborough & Mitchell, CCS'01, JCS'03]
  - develop a graph-based search algorithm for a language RT_0, a superset of SDSI
    - combines bottom-up search and goal-directed top-down search with tabling specifically for the kind of rules in RT_0
    - can deal with distributed discovery

Deduction Based on the Rewriting Semantics (4)

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

SDSI and Pushdown Systems

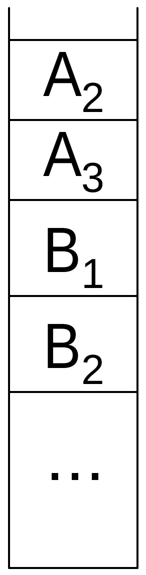
Stack:



State: K_1

Apply the rewriting rule:
$K_1\ A_1$ to $K_2\ A_2\ A_3$

Stack:



State: K_2

A name string corresponds to a configuration

“rewrites into” equivalent to “reaches”

Recap of the Rewriting-based Semantics

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

Set-based Semantic Intuitions

- Each name string is bound to a set of principals
- (K A ⇨ ω) means the local name “K A” is bound to a superset of the principal set that ω is bound to

Defining Set-based Semantics (1)

- A valuation *V* maps each local name to a set of principals
- A valuation *V* can be extended to map each name string to a set of principals
  - $\underline{V}(\mathrm{K}) = \{ \mathrm{K} \}$
  - $\underline{V}(\mathrm{K\ A}) = V(\mathrm{K\ A})$
  - $\underline{V}(\mathrm{K\ B}_1 \ldots \mathrm{B}_m) = \underset{j = 1..n}{\text{È}}\ \underline{V}(\mathrm{K}_j\ \mathrm{B}_2 \ldots \mathrm{B}_m)$
    - where m>1 and $V(\mathrm{K\ B}_1) = \{\mathrm{K}_1, \mathrm{K}_2, \ldots, \mathrm{K}_n\}$

Defining Set-based Semantics (2)

- A 4-tuple (K A ⇨ ω) is the following constraint
  - V (K A) $\supseteq$ $\underline{V}$ (ω)
- The semantics of P is the least valuation V_P that satisfies all the constraints
- Queries
  - "can ω rewrite into K?" answered by checking whether "K $\in$ $\underline{V}_P$ (ω)".
- Does not define answers to "can ω_1 rewrite into ω_2".
  - asking whether $\underline{V}_P$ (ω_1) $\supseteq$ $\underline{V}_P$ (ω_2) is incorrect

Relationship Between the Rewriting Semantics and the Set Semantics

- Theorem: Given P, ω_1, and ω_2, ω_1 rewrites into ω_2 using P if and only if for any P' $\supseteq$ P, $\underline{V}_{P'}(\omega_1) \supseteq \underline{V}_{P'}(\omega_2)$.

- Corrolary: Given P, ω, and K, ω rewrites into K using P if and only if $\underline{V}_{P}(\omega) \supseteq \{ K \}$

A Logic-Programming-based Semantics Derived from the Set-based Semantics

- Translate each 4-tuple into a LP clause
  - Using a ternary predicate m
    - m(K, A, K') is true if K' $\in$ *V* (K A)
  - (K A ⇨ K') to m(K, A, K')
  - (K A ⇨ K_1 A_1) to m(K, A, ?x) :- m(K_1, A_1, ?x)
  - (K A ⇨ K_1 A_1 A_2)
    to m(K,A,?x) :- m(K_1,A_1,?y_1), m(?y_1,A_2,?x)
  - (K A ⇨ K_1 A_1 A_2 A_3)
    to m(K,A,?x) :- m(K_1,A_1,?y_1), m(?y_1,A_2,?y_2), m(?y_2,A_3,?x)
- The minimal Herbrand model determines the semantics

An Alternative Way of Defining the LP-based Semantics (1)

- Define a macro **contains**
  - **contains**[ω][K’] means that K’ Î *V* (ω)
    - **contains**[K][K’] ≡ (K= K’)
    - **contains**[K A][K’] ≡ m(K, A, K’)
    - **contains**[K A_1 A_2 … A_n][K’] ≡ $*y* (m(K, A_1, *y*) Ù **contains**[*y* A_2 … A_n][K’]) where n>1

An Alternative Way of Defining the LP-based Semantics (2)

- Translates a 4-tuple (K A ⇨ ω) into a FOL sentence
  - $\forall z$ (**contains**[K A][z] $\Leftarrow$ **contains**[ω][z])
- This sentence is also a Datalog clause
- A set P of 4-tuples defines a Datalog program, denoted by SP[P]
  - The minimal Herbrand model of SP[P] defines the semantics

An Example of Translation

From (K_C access ⇨ K_C mit faculty secretary)

to $\forall z$ (**contains**[K_C access][z] $\Leftarrow$
  contains[K_C mit faculty secretary][z])

to $\forall z$ (**m(**K_C, access, z) $\Leftarrow$
  $\$y_1$ (m(K_C, mit, y_1) Ù **contains**[y_1 faculty secretary][z])

to $\forall z$ $\forall y_1$ (**m(**K_C, access, z) $\Leftarrow$
  m(K_C, mit, y_1) Ù
  $\exists y_2$ (m(y_1, faculty, y_2) Ù **contains**[y_2 secretary] [z])

to $\forall z$ $\forall y_1$ $\forall y_2$ (**m(**K_C, access, z) $\Leftarrow$
  m(K_C, mit, y_1) Ù
  m(y_1, faculty, y_2) Ù
  m(y_2, secretary, z]))

Set semantics is equivalent to LP semantics

- The least Herbrand model of SP[P] is equivalent to the least valuation, i.e.,
  - K’ $\in$ V_P (K A) iff. m(K,A,K’) is in the least Herbrand model of SP[P]
- Same limitation as set-based semantics
  - does not define answers to containment between arbitrary name strings

A First-Order Logic (FOL) Semantics

- A set P of 4-tuples defines a FOL theory, denoted by Th[P]
- A query is a FOL formula
  - "ω_1 rewrites into ω_2" is translated into $\forall z\ (\mathbf{contains}[\omega_1][z] \Leftarrow \mathbf{contains}[\omega_2][z])$
  - Other FOL formulas can also be used as queries
- Logical implication determines semantics

FOL Semantics is Extension of LP Semantics

- LP semantics is FOL semantics with queries limited to LP queries
  - m(K,A,K') is in the least Herbrand model of SP[P] iff. Th[P] |= m(K,A,K')

Equivalence of Rewriting Semantics and FOL Semantics

- Theorem: for string rewriting queries, the string rewriting semantics is equivalent to the FOL semantics
  - Given a set P of 4-tuples, it is possible to rewrite ω_1 into ω_2 using the 4-tuples in P if and only if
    Th[P] |= " z (**contains**[ω_1][z] $\Leftarrow$
    contains[ω_2][z])

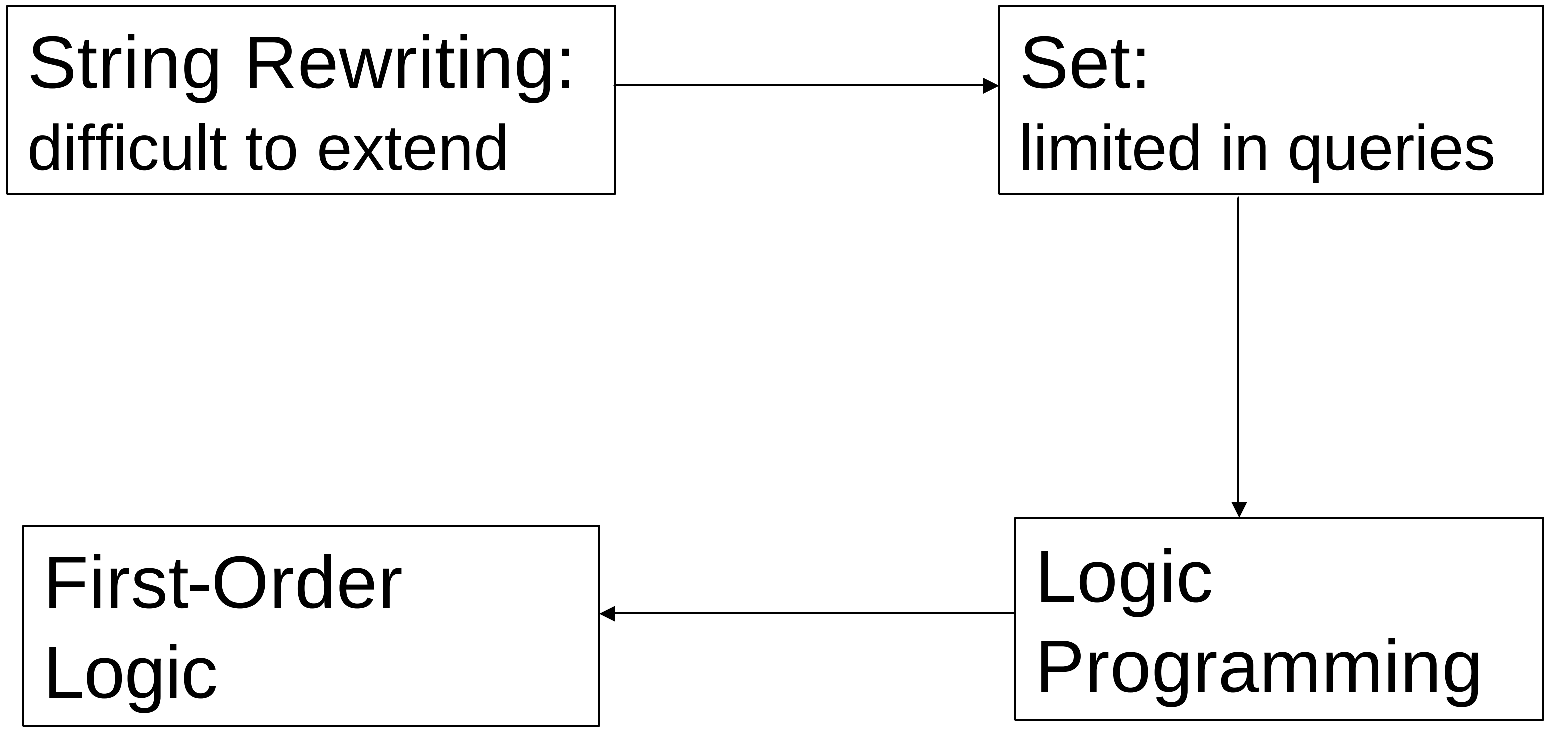
Advantages of FOL semantics: Computation efficiency

- A large class of queries can be answered efficiently using logic programs
  - including rewriting queries
  - e.g., whether ω rewrites into $K\ B_1\ B_2$ under P can be answered by determining whether $SP[P \cup (K'\ A' \Rightarrow \omega) \cup (K\ B_1 \Rightarrow K'_1) \cup (K'_1\ B_2 \Rightarrow K'_2)] \models m(K', A', K'_2)$
    - where K', K'_1, and K'_2 are new principals
    - this proof procedure is sound and complete
      - this result also follows from results in proof theory regarding Harrop Hereditary formulas

Advantages of FOL semantics: Extensibility

- Additional kinds of queries can be formulated and answered, e.g.,
  - $\forall z\ (\mathrm{m}(K_1, A_1, z) \Leftarrow \mathrm{m}(K_1, A_2, z))$
    $\Leftarrow \exists z\ (\mathrm{m}(K_2, A_1, z) \wedge \mathrm{m}(K_2, A_2, z))$
- Additional forms of statements can be easily handled, e.g.,
  - $(K\ A \Rightarrow K_1\ A_1 \cap K_2\ A_2)$ maps to
    $\forall z\ (\mathrm{m}(K,A,z) \Leftarrow \mathrm{m}(K_1,A_1,z)$ Ù $\mathrm{m}(K_2,A_2,z))$

Summary: 4 Semantics



Advantages of FOL Semantics: Summary

- Simple
  - captures the set-based intuition
  - defined using standard FOL
- Extensible
  - additional policy language features can be handled easily
  - allow more meaningful queries
- Computation efficiency