



# Trust Management

## CS 6463 Lecture 10

Prof. William Winsborough  
February 20, 2006

# Business

- Preliminary project proposal
  - Due Wednesday, March 8
- Advertisement
  - I have funding for PhD students working in trust management or information flow

# Example: Expressivity in Credentials

- Deferring a Guaranteed Student Loan
  - `BankWon.deferGSL` ← `FAB.accredited.fulltimeStudent`
  - `FAB.accredited` ← `StateU`
  - `StateU.fulltimeStudent` ← `URegistrar.fulltimeLoad`
  - `StateU.fulltimeStudent` ← `URegistrar.parttimeLoad`  $\cap$   
`StateU.gradOfficer.phdCandidate`
  - `URegistrar.parttimeLoad` ← `Bob`
  - `StateU.gradOfficer` ← `Carol`
  - `Carol.phdCandidate` ← `Bob`

# $RT_0$ Syntax

- Basic structure is a role (i.e., an attribute):  $A.r$ 
  - $A$  is a principal (authority for  $A.r$ ),  $r$  is a role name
- Four types of policy statement
  - $A.r \leftarrow D$   
Role  $A.r$  contains principal  $D$  as a member
  - $A.r \leftarrow B.r_1$   
 $A.r$  contains role  $B.r_1$  as a subset
  - $A.r \leftarrow A.r_1.r_2$   
 $A.r$  contains  $B.r_2$  as a subset, for each  $B$  in  $A.r_1$
  - $A.r \leftarrow A_1.r_1 \cap A_2.r_2$   
 $A.r$  contains the intersection
- A credential is a statement signed by  $A$ , the credential issuer and the authority over  $A.r$
- The first 3 statement types give a language equivalent to pure SDSI

# Set-based Semantics for $RT_0$

- Given as set of credentials  $C$ , the semantics of  $C$  is  $S_C : \text{Role} \rightarrow \wp(\text{Entity})$  given by the least function  $\text{rmem} : \text{Role} \rightarrow \wp(\text{Entity})$  that satisfies the following system of set inequalities:

$$\{ \text{rmem}(A.r) \supseteq \text{expr}[\text{rmem}](e) \mid A.r \leftarrow e \in C \}$$

- Where

- $\text{expr}[\text{rmem}](B) = \{B\}$
- $\text{expr}[\text{rmem}](A.r) = \text{rmem}(A.r)$
- $\text{expr}[\text{rmem}](A.r_1.r_2) = \bigcup_{B \in \text{rmem}(A.r_1)} \text{rmem}(B.r_2)$
- $\text{expr}[\text{rmem}](f_1 \wedge f_2 \wedge \dots \wedge f_k) = \bigcap_{1 \leq j \leq k} \text{expr}[\text{rmem}](f_j)$

# Calculating $\mathcal{S}_C$

- Let  $SF = \text{Role} \rightarrow \wp(\text{Entity})$ 
  - $SF$  is the function type that is the space of all semantic functions
- Let  $\Psi_C: SF \rightarrow SF$  be given by
$$\Psi_C(\text{rmem})(A.r) = \bigcup_{A.R \leftarrow e \in C} \text{expr}[\text{rmem}](e)$$
- $\Psi_C$  is monotonic and  $SF$  is a complete lattice
  - [Tarski 1955] The least fixpoint of  $\Psi_C$  exists
- Note that the least solution to the set of set containments on the previous slide is identical to the least fixpoint of  $\Psi_C$
- The least fixpoint can be calculated by taking the limit of  $\Psi_C^i: SF$  where  $\Psi_C^0(A.r) = \emptyset$  and
$$\Psi_C^{i+1}(A.r) = \bigcup_{A.R \leftarrow e \in C} \text{expr}[\Psi_C^i](e)$$

# Credential Chain Discovery

## ■ Outline

- Sound and complete evaluation model
  - Based on graphical model (“credential graph”)
- Efficient search for proof of authorization
- Provides a basis for discovering chains when credentials are stored in a distributed manner
  - Evaluation steps can be interleaved with credential retrieval steps

# Algorithmic Contributions

- Search algorithms:
  - Worst case efficiency as good as any existing algorithm
    - Backward.  $O(N^3)$  time,  $N$  = number of credentials
    - Forward.  $O(N^2M)$  time,  $M$  = sum of credential sizes
    - Both directions.  $O(N^2M)$  time
  - Well suited to the application
    - Efficient when there are lots of unrelated credentials
    - Changes to credential pool do not degrade performance
    - Graph search can drive credential discovery

# Example: Student ACM Discount

- $\text{EPub.studentACM} \leftarrow \text{EOrg.student} \cap \text{ACM.member}$
- $\text{EOrg.student} \leftarrow \text{EOrg.university.student}$
- $\text{EOrg.university} \leftarrow \text{FAB.accredited}$

*Credential Discovered  
in Backward Direction*

- $\text{FAB.accredited} \leftarrow \text{StateU}$
- $\text{StateU.student} \leftarrow \text{URegistrar.parttimeLoad}$
- $\text{URegistrar.parttimeLoad} \leftarrow \text{Alice}$
- $\text{ACM.member} \leftarrow \text{Alice}$

*Credential Discovered  
in Forward Direction*

# Credential Graph

EPub gives a double subscription discount:

EPub.studentacm

$\text{EOrg.student} \cap \text{ACM.member}$

EOrg.student

EOrg.university

EOrg.university.student

FAB.accredited

StateU.student

StateU

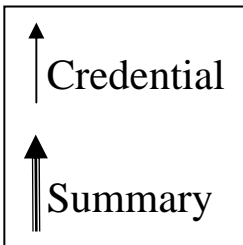
URegistrar.parttimeLoad

ACM.member

Alice

- Type 1: FAB.accredited  $\leftarrow$  StateU
- Type 2: EOrg.university  $\leftarrow$  FAB.accredited
- Type 3: EOrg.student  $\leftarrow$  EOrg.university.student
- Type 4: EPub.studentacm  $\leftarrow$   $\text{EOrg.student} \cap \text{ACM.member}$

## Key



# Credential Graph Organizes Chain Discovery

## Key

