# Credential Chain Discovery in $RT^T$ Trust Management Language

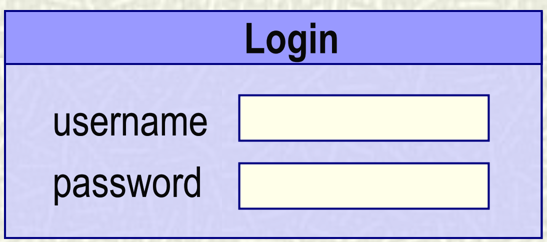
*Krzysztof Sacha, Warsaw University of Technology*

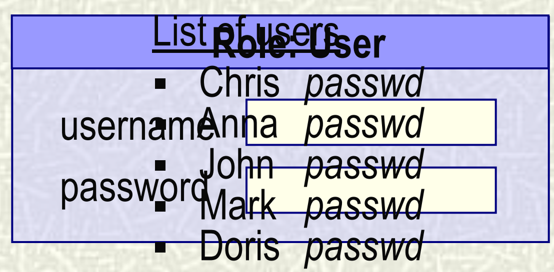
## Contents

- Introduction
- Trust management
- Role based Trust management language
- The semantics of $RT^T$ language
- Credential Graph
- Credential Chain
- Conclusions

# Introduction

- Access control
- Traditional approach based on identity
- Role-based access control
- The problem



# Trust management (1)

An Internet Bank adopts a policy of giving some special rates to the employees of accredited universities

The decision is based on a set of **credentials**, which state that:

- He/she is employed at a university.
- The university is an accredited university.

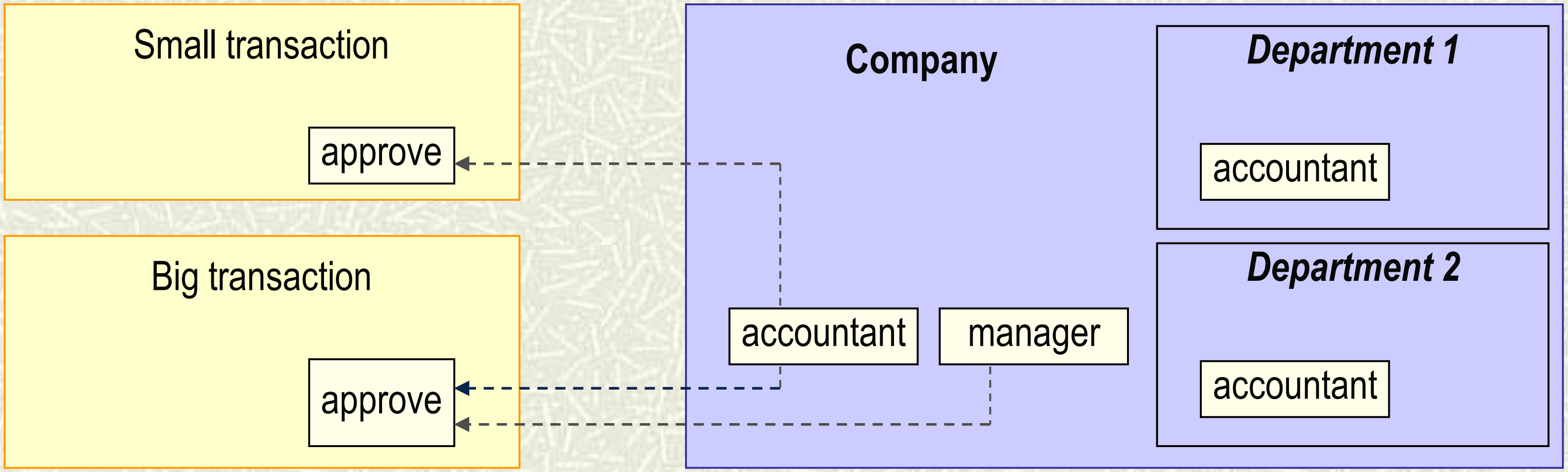
The implementation: electronic documents

# Trust management (2)

A bank, which supports a company, adopts the following security policy:

– a small transaction is authorized by an accountant,

– a big transaction is authorized by an accountant and a manager.



# Role-based Trust management language

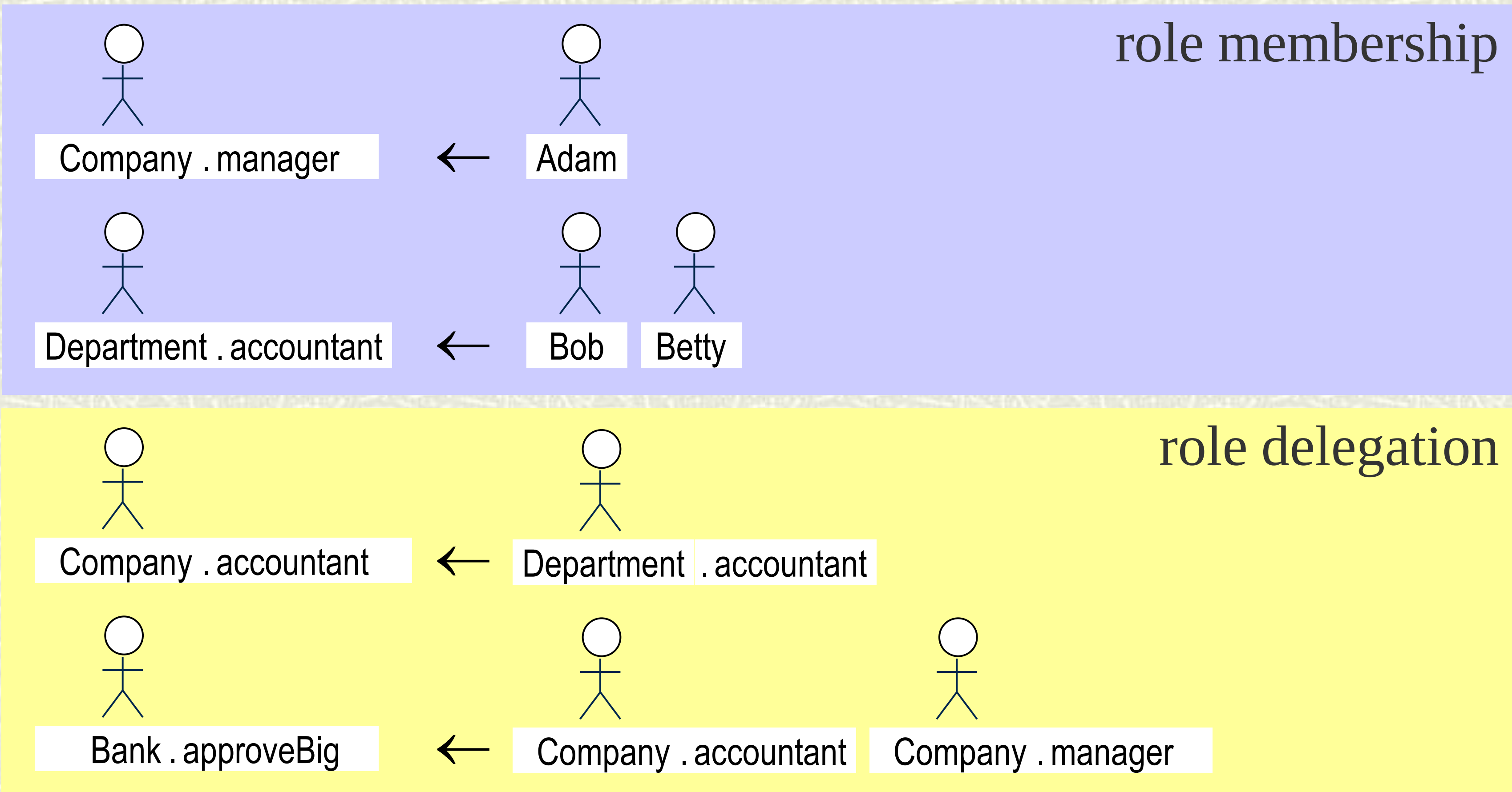
**Basic notions**

Entity – an individual (a person, an institution), who makes requests to access resources or decides on the access
(e.g.: *University*, *Chris*, *A*, *B*, *C*, *X*, *Y*, ...)

Role name – represents permissions to access resources
(e.g.: *student*, *accountant*, *r*, *s*, *t*, ...)

Role – represents a set of entities that have permissions issued by particular issuers
(*University.student*, *A.r*, *B.s*)

# RT credentials



# The syntax of $RT^T$ language

**<u>Types of credentials</u>**

(1) $A.r \leftarrow B$

(2) $A.r \leftarrow B.s$

(3) $A.r \leftarrow B.s.t$

(4) $A.r \leftarrow B.s \cap C.t$

(5) $A.r \leftarrow B.s \oplus C.t$

(6) $A.r \leftarrow B.s \otimes C.t$

(1) Simple membership

University.faculty ← {Chemistry}
University.faculty ← {Electronics}
Chemistry.student ← {John}

(5) Manifold role

Bank.approveBig ← Company.accountant ⊕ Company.manager

(4) Intersection inclusion

(6) Manifold role

Bank.approveBig ← Company.accountant ⊗ Company.manager

# The semantics of $RT^T$ (1)

$P$ – set of RT-credentials

$E$ – set of entities
$R$ – set of role names

$S_P \subseteq 2^E \times R \times 2^E$

**Instances of $S_P$**

( {Company}, manager, {Adam} )
( {Department}, accountant, {Bob} )
( {Department}, accountant, {Betty} )

( {Bank}, approveBig, {Bob,Adam} )
( {Bank}, approveBig, {Betty,Adam} )

$\hat{S}_P$ ( {Bank}, approveBig ) = { { Bob,Adam}, {Betty,Adam} }

$\hat{S}_P : 2^E \times R \rightarrow 2^F$ $\quad F = 2^E$

$\hat{S}_P ( A.r ) = \{ X \in 2^E : ( A, r, X ) \in S_P \}$

# The semantics of $RT^T$ (2)

The semantics of a set $P$ of $RT^T$ credentials is the smallest relation $S_P$, closed with respect to the following properties:

$(A, r, X) \in S_P$ for each $A.r \leftarrow X \in P$

If $A.r \leftarrow B.s \in P$ and $(B, s, X) \in S_P$ then $(A, r, X) \in S_P$

If $A.r \leftarrow B.s.t \in P$ and $(B, s, C) \in S_P$ and $(C, t, X) \in S_P$ then $(A, r, X) \in S_P$

If $A.r \leftarrow B.s \cap C.t \in P$ and $(B, s, X) \in S_P$ and $(C, t, X) \in S_P$ then $(A, r, X) \in S_P$

If $A.r \leftarrow B.s \oplus C.t \in P$ and $(B, s, X) \in S_P$ and $(C, t, Y) \in S_P$ then $(A, r, X \cup Y) \in S_P$

If $A.r \leftarrow B.s \otimes C.t \in P$ and $(B, s, X) \in S_P$ and $(C, t, Y) \in S_P$ and $X \cap Y = \phi$ then $(A, r, X \cup Y) \in S_P$

# Credential graph (1)

A graphical representation of the semantics of a set $P$ of credentials.

$$GP = ( N_P , E_P )$$

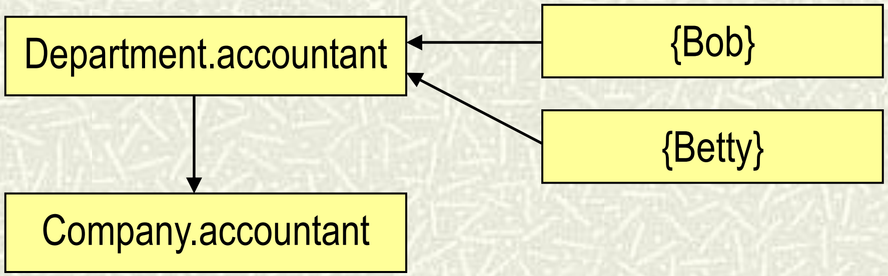
$N_P$ – nodes are role expressions, which meaning are sets of sets of entities

$E_P$ – edges reflect inclusion of those sets.

Department.accountant ← {Bob}

Department.accountant ← {Betty}

Compnay.accountant ← Department.accountant



# Credential graph (2)

**Theorem 1** (Soundness of the credential graph)

For each $n_1, n_2 \in N_P$, if $( n_1, n_2 ) \in E_P$ then $\hat{S}_P ( n_1 ) \subseteq \hat{S}_P ( n_2 )$

**Theorem 2** (Completeness of the credential graph)

If $( A, r, X ) \in S_P$ then $A.r$ , $X \in N_P$ and a path from $X$ to $A.r$ exists in $G_P$

# Credential chain (1)

Practical questions:

Who can play a role *A.r*?
Can *X* play the role *A.r*?

**Credential chain**:

A sub-graph of the credential graph,
which contains a path from *X* to *A.r*

# Credential chain (1)

1. Create a node, which represents the role in question. This node is active.
2. Select an active node (e.g. A.r), find all credentials A.r ← e, and for each credential create nodes representing e and roles in e.
3. Resolve node dependencies between the analyzed credentials.
4. All added nodes that represent roles are active. The node selected in step 2 becomes passive.

C.department ← { D1 }

C.department ← { D2 }

C.manager ← { Adam }

D1.accountant ← { Bob }

D2.accountant ← { Betty }

C.accountant ← C.department.accountant

Bank.approveBig ← C.manager ⊕ C.accountant

**Questions**:

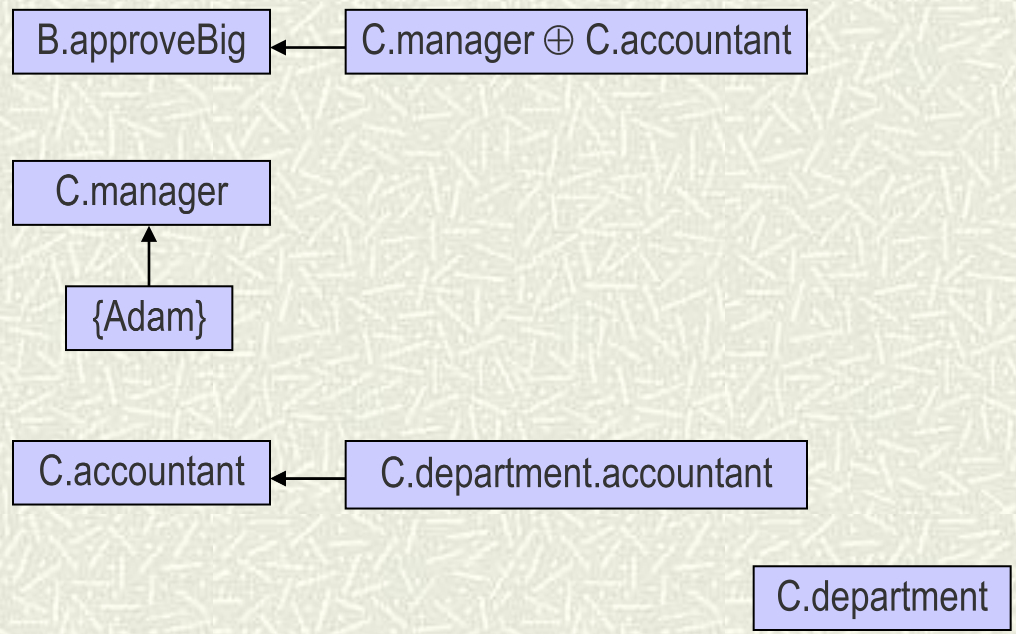
Who can approve a Big transaction?

Can Adam and Betty approve such a transaction?

# Credential chain (2)

C.department ← { D1 }
C.department ← { D2 }
C.manager ← { Adam }
D1.accountant ← { Bob }
D2.accountant ← { Betty }
C.accountant ← C.department.accountant
B.approveBig ← C.manager ⊕ C.accountant

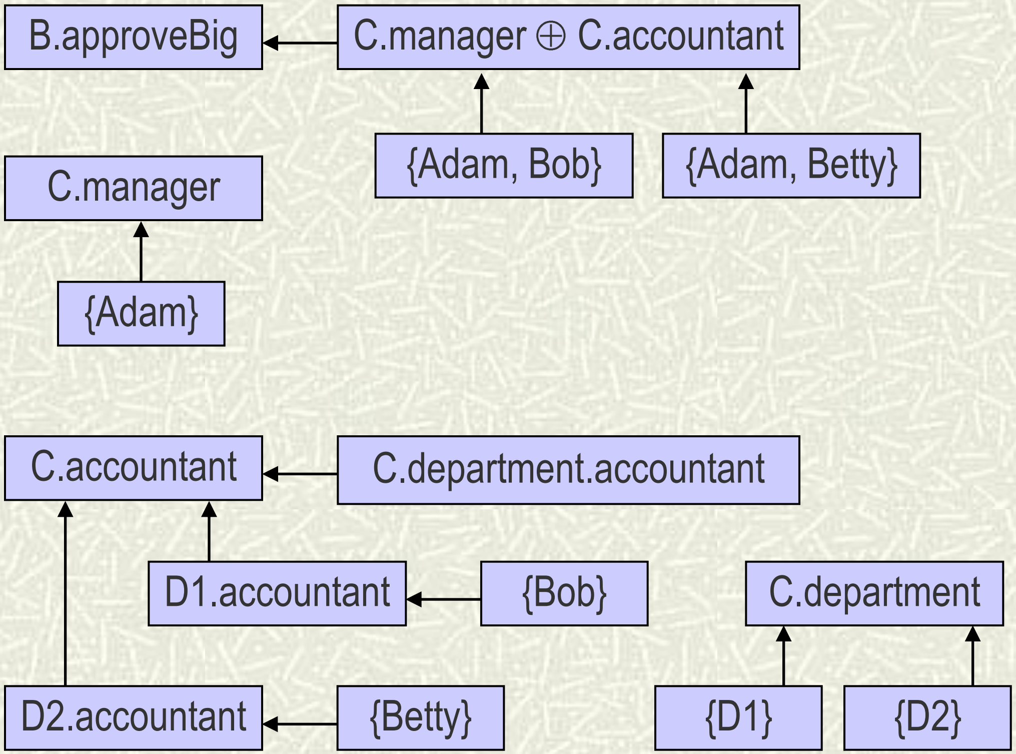
1. Create a node, which represents the role in question. This node is active.
2. Select an active node (e.g. A.r), find all credentials A.r ← e, and for each credential create nodes representing e and roles in e.
3. Resolve node dependencies between the analyzed credentials.
4. All added nodes that represent roles are active. The node selected in step 2 becomes passive.



# Credential chain (2)

C.department ← { D1 }
C.department ← { D2 }
C.manager ← { Adam }
D1.accountant ← { Bob }
D2.accountant ← { Betty }
C.accountant ← C.department.accountant
B.approveBig ← C.manager ⊕ C.accountant

1. Create a node, which represents the role in question. This node is active.
2. Select an active node (e.g. A.r), find all credentials A.r ← e, and for each credential create nodes representing e and roles in e.
3. Resolve node dependencies between the analyzed credentials.
4. All added nodes that represent roles are active. The node selected in step 2 becomes passive.



# Conclusions

- Trust management languages are an effective means for describing access control in distributed open systems.
- The main contribution of this work is an algorithm for creating a $RT^T$ credential chain.
- We are planning to implement a trust management server to resolve access control queries.