Policy-driven Distributed Authorization:

Status and Prospects

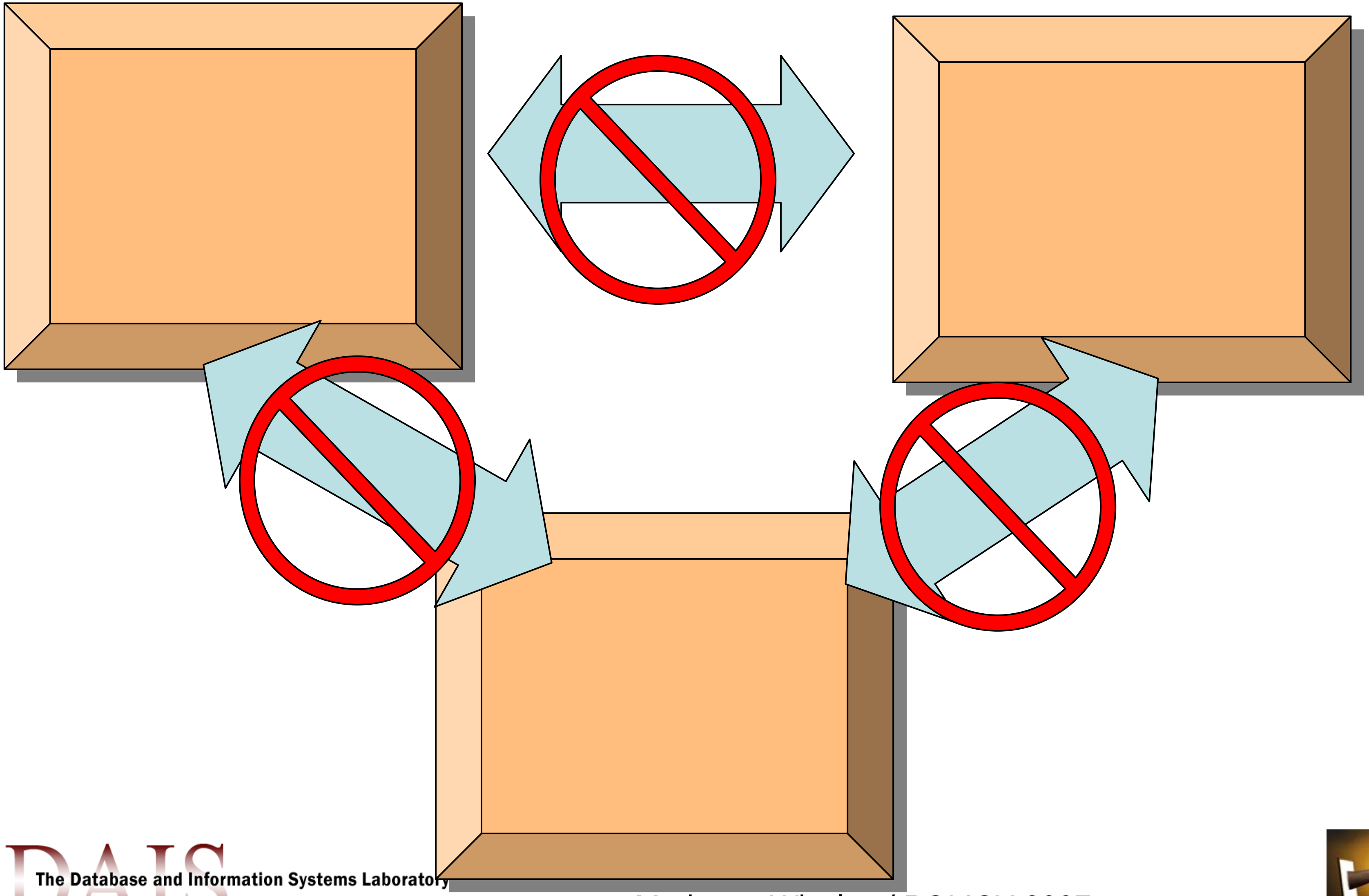
(sanitized version)

Marianne Winslett

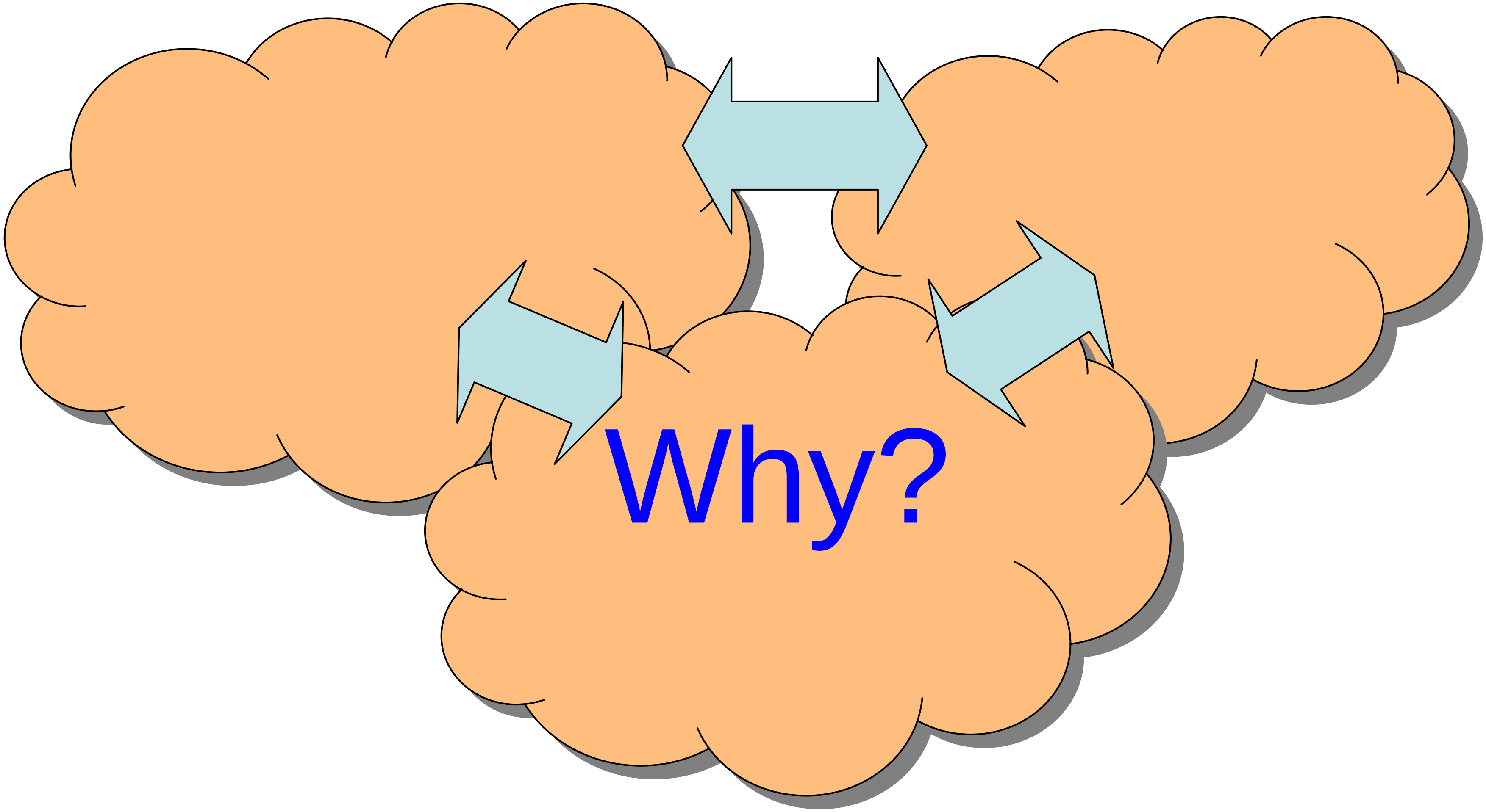
University of Illinois

A tale of two trends

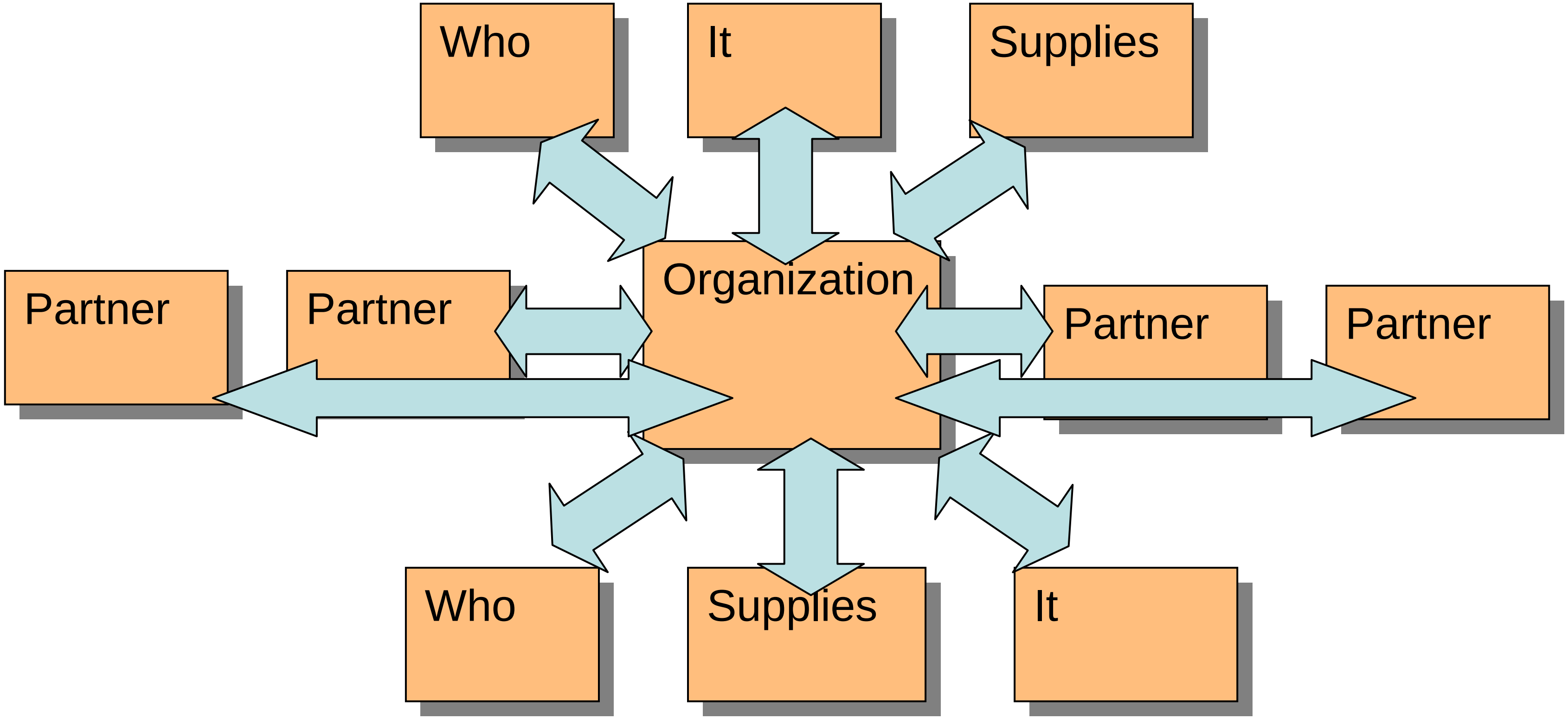
Organizational boundaries used to be solid



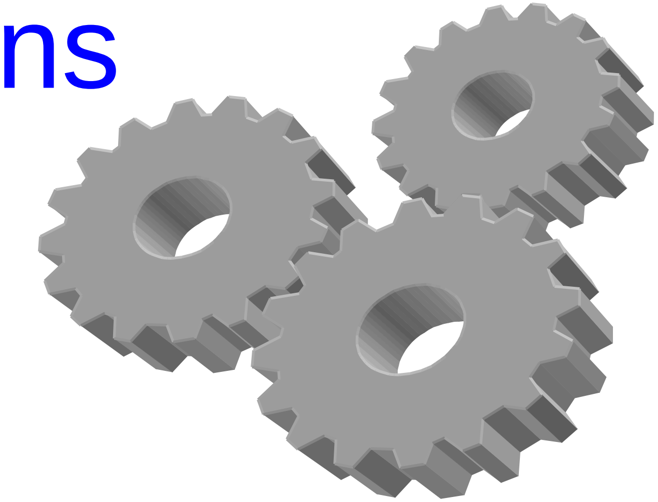
Now boundaries are fuzzy



Competitive pressures are dissolving boundaries



Example: supply chains



Walmart

Supplier

Supplier

Supplier

Supplier

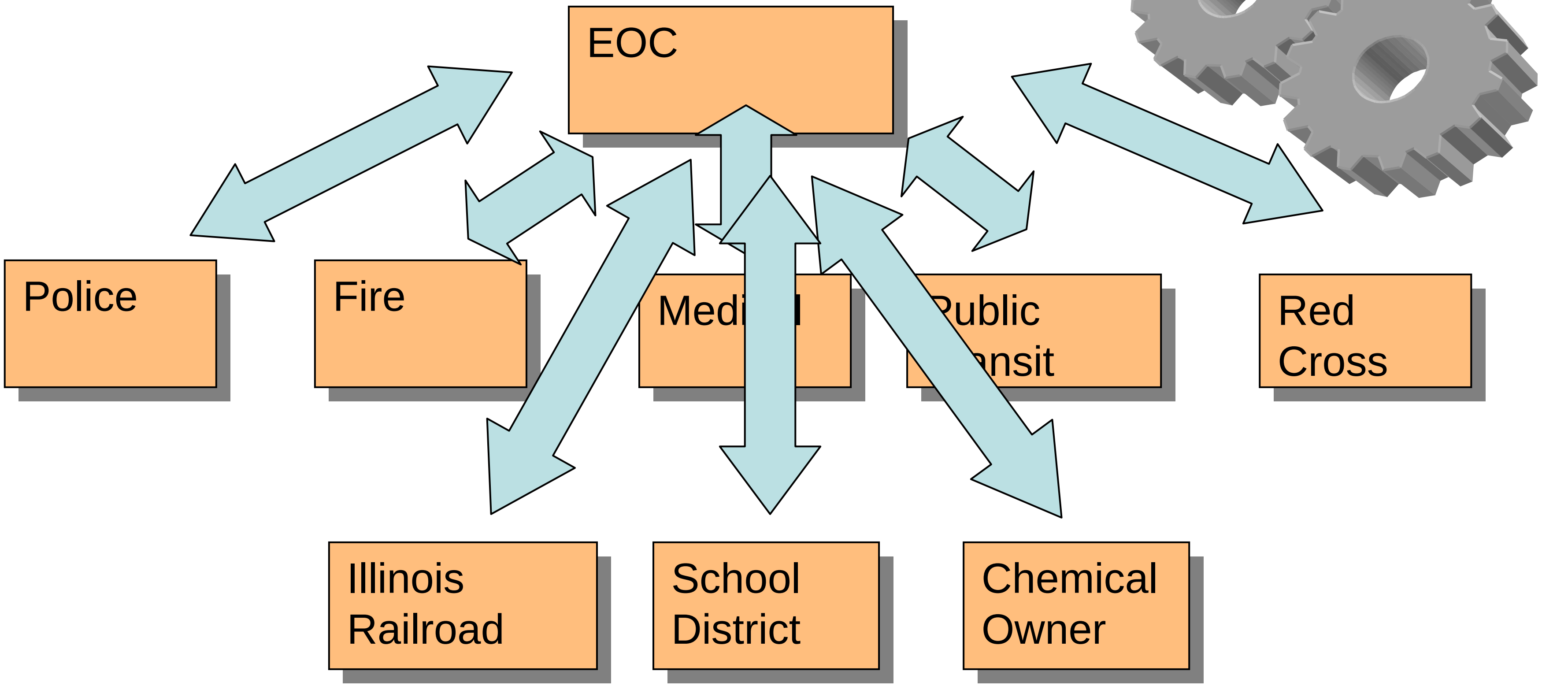
Supplier

2nd level Supplier

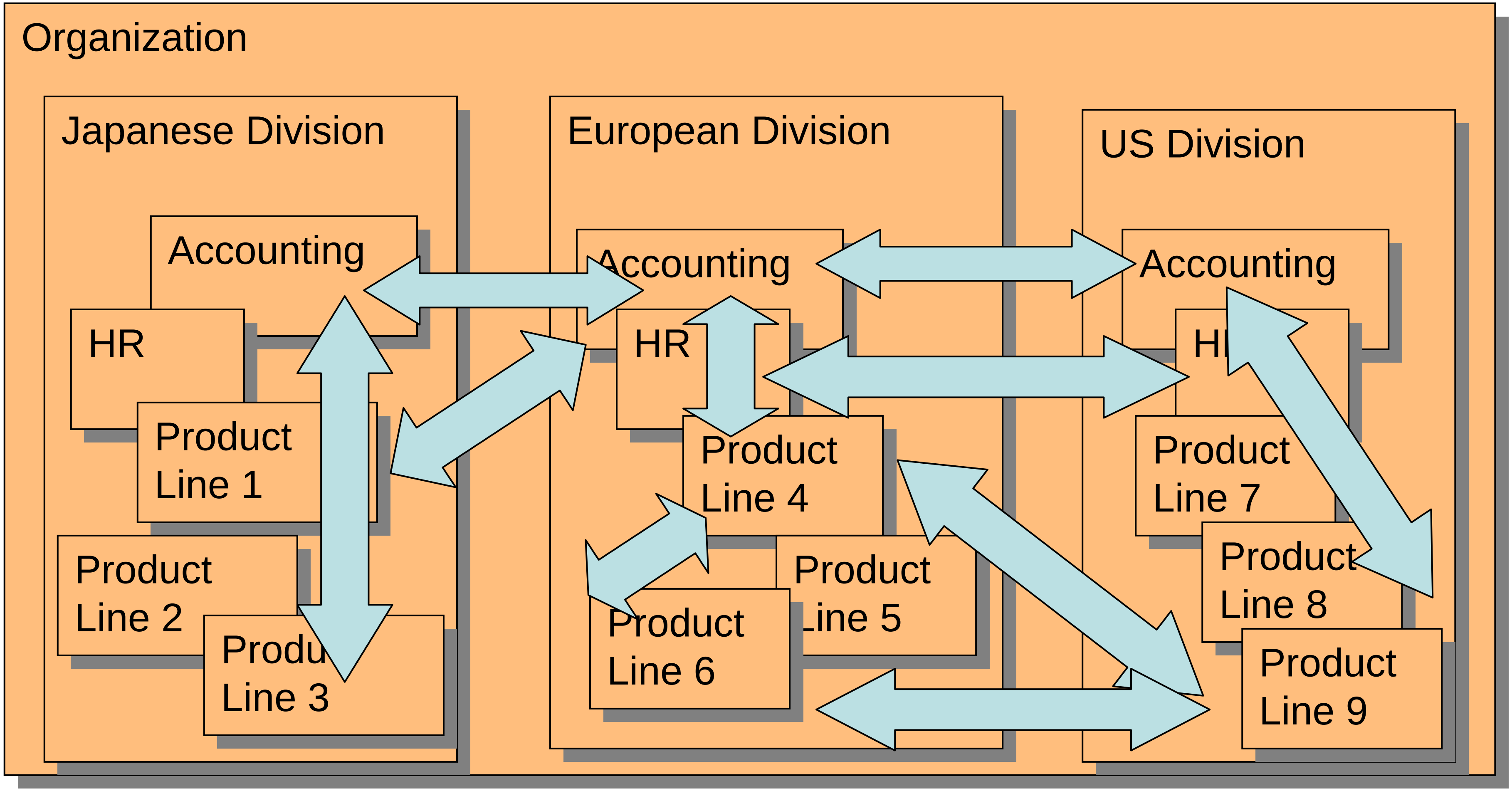
2nd level Supplier

2nd level Supplier

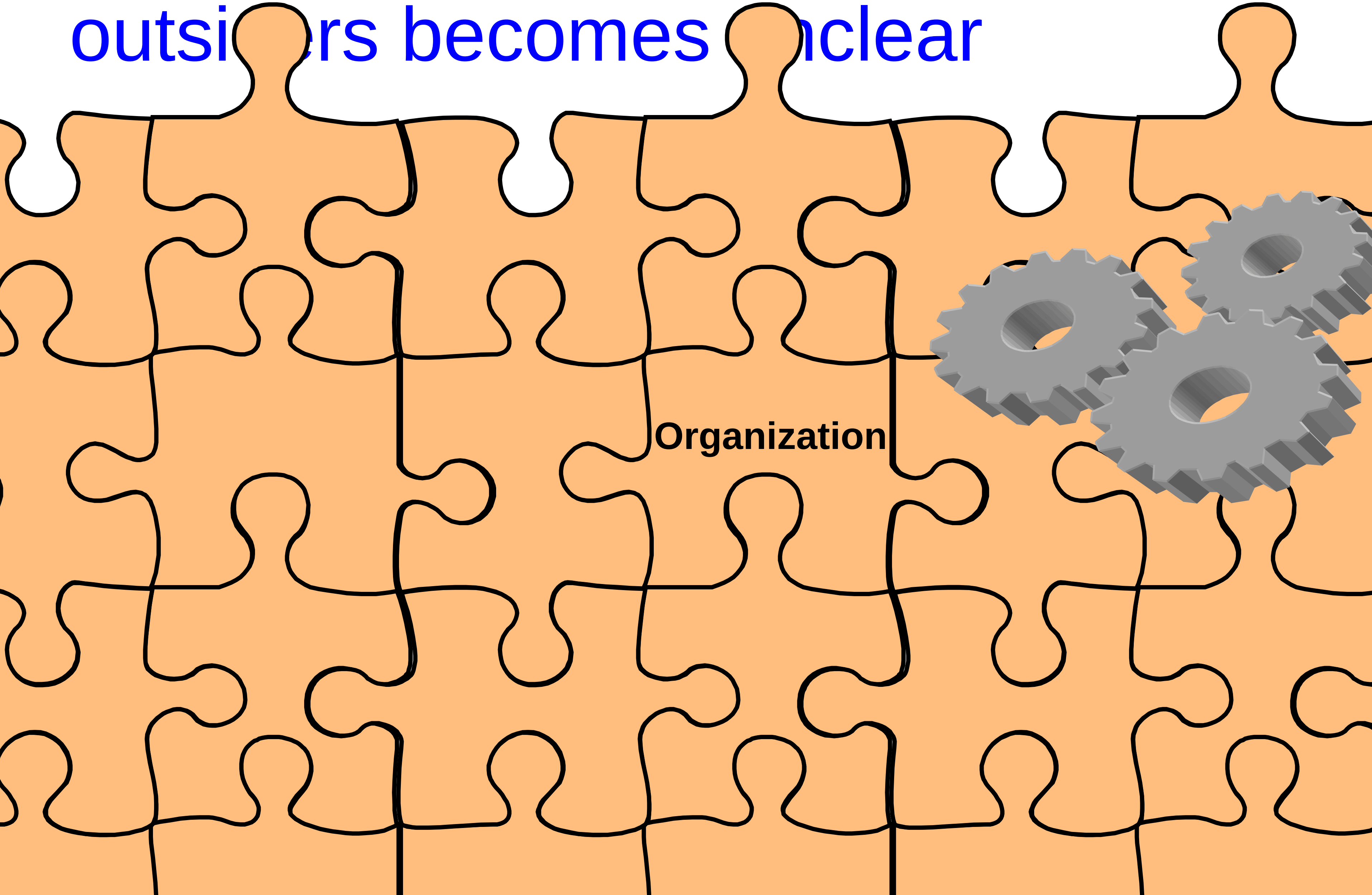
Example: first responders



Example: any large enterprise



Distinction between insiders and outsiders becomes unclear



Corporations are also facing new pressures for accountability

Global Crossing

HIPAA

ENRON

OSHA

FERPA

FDA

Sarbanes-Oxley

SEC Rule 15a

SEC Rule 15a

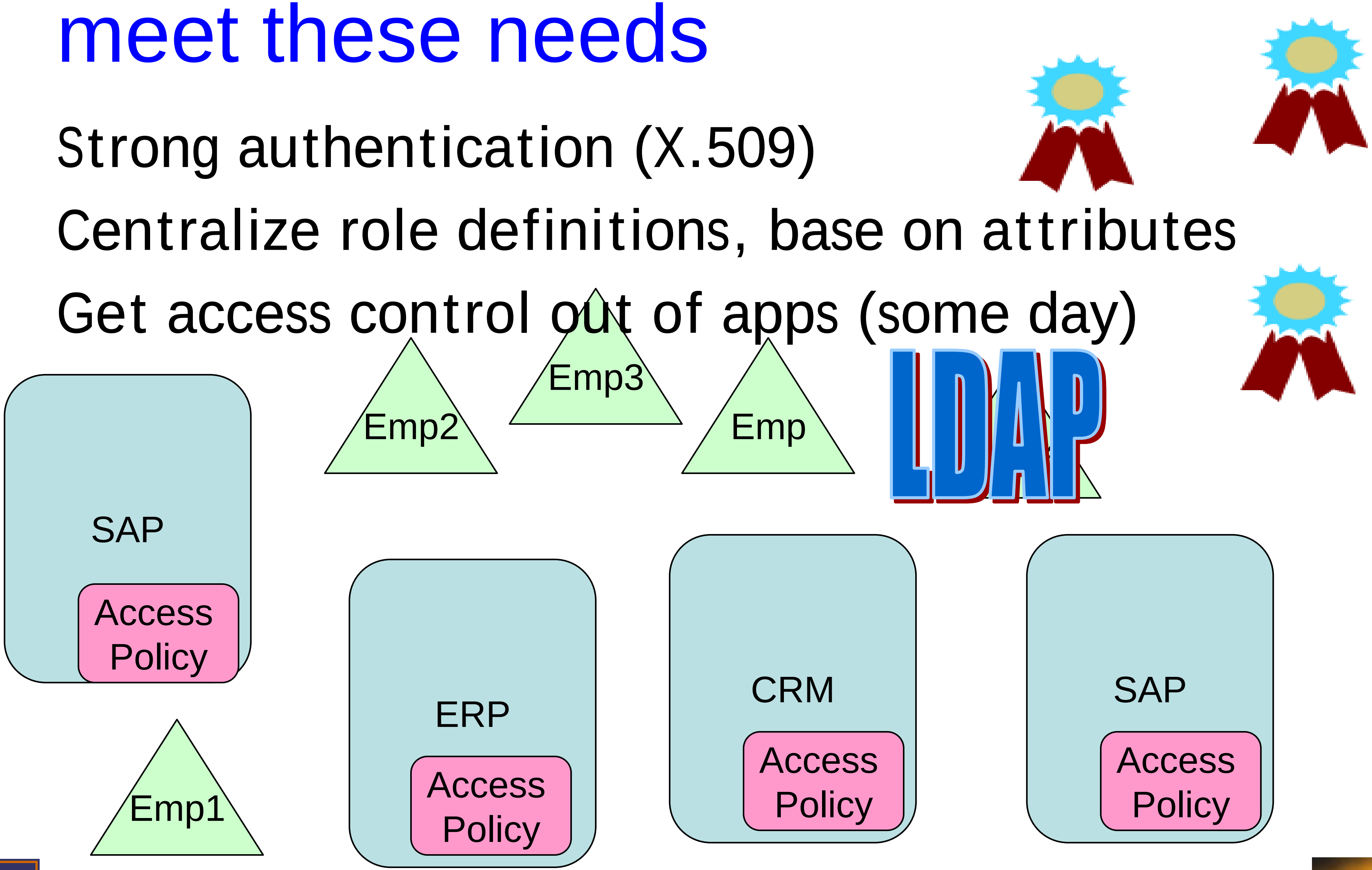
Accountability includes knowing who can/did do what to your data when

Industry is taking several steps to meet these needs

Strong authentication (X.509)

Centralize role definitions, base on attributes

Get access control out of apps (some day)



So enterprises are moving toward attribute-based access control

- Based off centralized LDAP + X.509
- Avoids inconsistency due to distribution
- Easier to maintain, compared to ACLs

HR

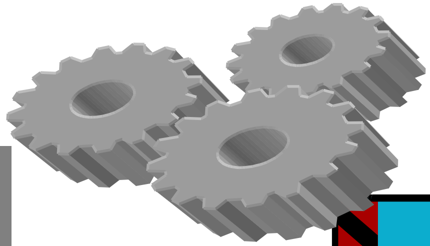
HR

HR

Walmart

Walmart's supplier

Walmart's supplier's supplier



Less insider threat

Doesn’t this sound like a good thing?

Why this scares me:

Automated exploitation of policy errors

Why this scares me:

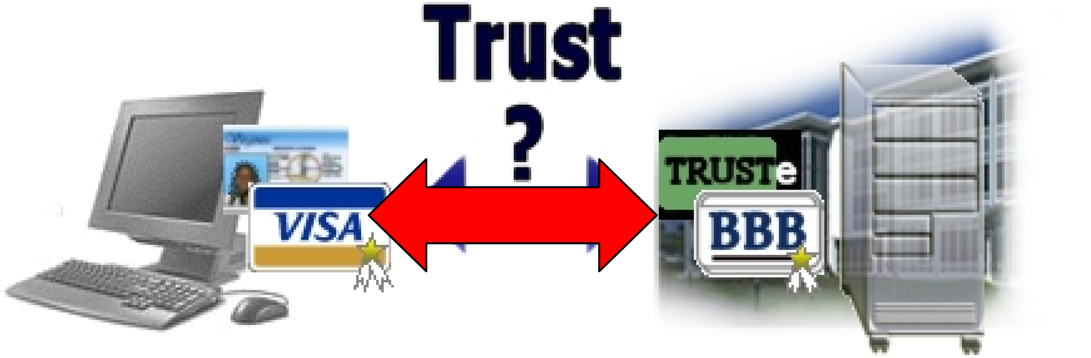
Centralized authorization services can be attacked

Why this scares me:

Industrial policy languages were not intended for rigorous analysis or user-friendliness

Analysis tools

Do things look more promising outside of industry?

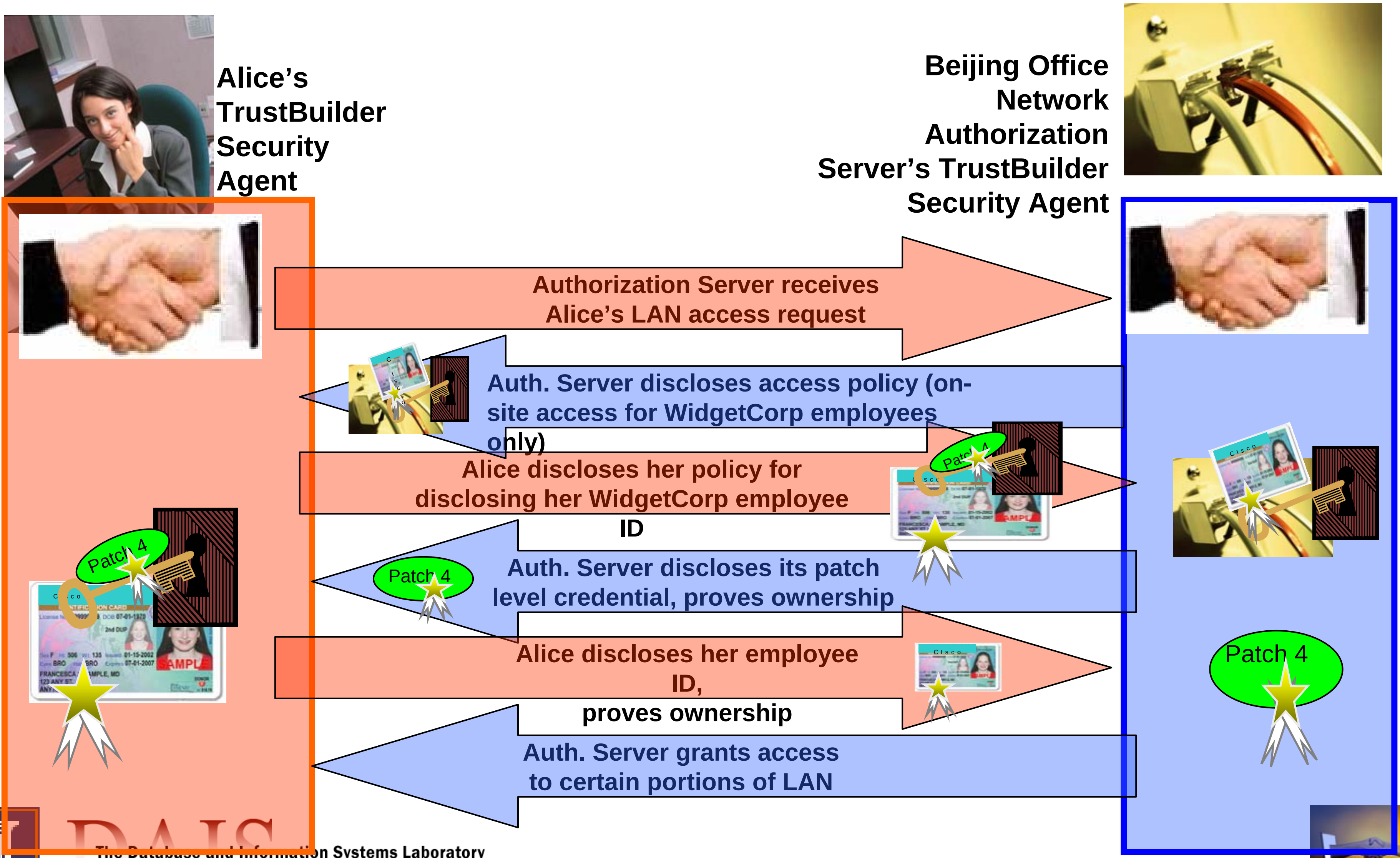


Bilateral trust

Sensitive policies and credentials

We understand this theory pretty well

Trust-negotiation-like approaches will inevitably come into use



Alice's TrustBuilder Security Agent

Beijing Office Network Authorization Server's TrustBuilder Security Agent

Authorization Server receives Alice's LAN access request

Auth. Server discloses access policy (on-site access for WidgetCorp employees only)

Alice discloses her policy for disclosing her WidgetCorp employee ID

Auth. Server discloses its patch level credential, proves ownership

Alice discloses her employee ID, proves ownership

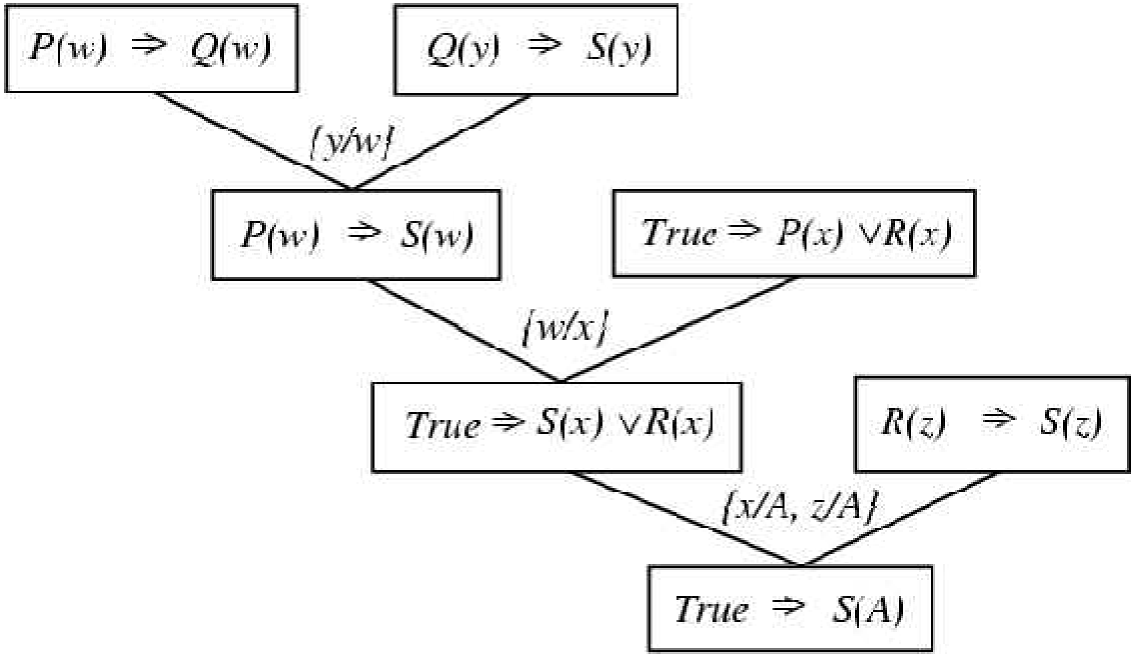
Auth. Server grants access to certain portions of LAN

But this only means more policies, more complex decisions to explain

> “Ohhhhhh . . .
> Look at that, Schuster . . . Dogs are so cute when they try to comprehend quantum mechanics”
>
> --Gary Larson

Traditional access control is transparent; TN is not

You are in the right group



Great ideas can fail if they don’t consider the human factor

The success of attribute-based policies for security and privacy, and ultimately the open and compliant systems they enable, relies on the ability of humans to comprehend and manage these policies.

Policy HCI is my #1 open problem

- Real-world case studies of policy management activities, to learn how users think about these activities
- User interfaces to help people understand and modify large, complex sets of policies

Example: Allegis policy middleware company

- Software for cross-organizational access to customer relationship management applications
- Allegis does not allow its clients to update their policies themselves

- Only policy specialists can be trusted to understand and update the policies correctly
- Even they may struggle to specify, modify, and comprehend complex policies--- note CRM focus

Large policies are as complex as any software

Declarative policy languages are not a panacea

- Consider hundreds of pages of (declarative) SQL

```
SELECT a1.Name, a1.Sales, SUM(a2.Sales)/(SELECT SUM(Sales)
    FROM Total_Sales) Pct_To_Total
    FROM Total_Sales a1, Total_Sales a2
    WHERE a1.Sales <= a2.sales or (a1.Sales=a2.Sales and
    a1.Name = a2.Name)
    GROUP BY a1.Name, a1.Sales
    ORDER BY a1.Sales DESC, a1.Name DESC;
```

...

- And any bugs may be found and exploited automatically

What if companies manage their own policies, as is natural with ABAC?

- How can a decision-maker with limited technical expertise quickly understand a particular policy that suddenly becomes crucial?
- What if the company's policy admin quits or is sick? How can a new hire quickly understand policies?

Ordinary users:

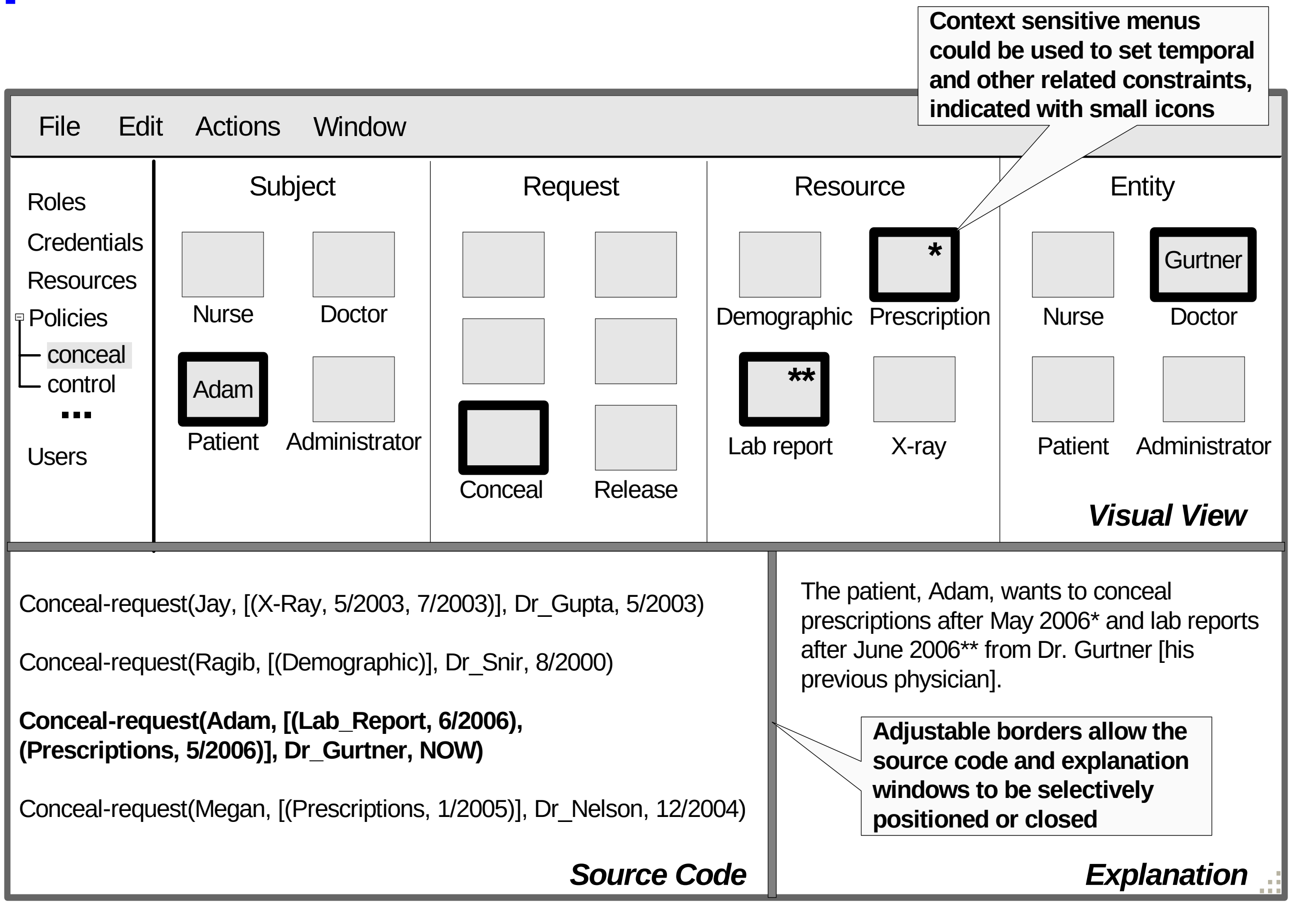
- Why was this decision made?
- How can I get it reversed?
- What if I ...

A proof is not an explanation

- Proofs are fundamental in TN
- But almost no one can understand a proof
- Need heuristics to turn proofs into explanations, both for ordinary users and administrators

- An explanation of why you didn’t get access, or how to get access, or what these policies say, doesn’t start from a proof

A possible solution: visual metaphors



Context sensitive menus could be used to set temporal and other related constraints, indicated with small icons

Conceal-request(Jay, [(X-Ray, 5/2003, 7/2003)], Dr_Gupta, 5/2003)

Conceal-request(Ragib, [(Demographic)], Dr_Snir, 8/2000)

Conceal-request(Adam, [(Lab_Report, 6/2006), (Prescriptions, 5/2006)], Dr_Gurtner, NOW)

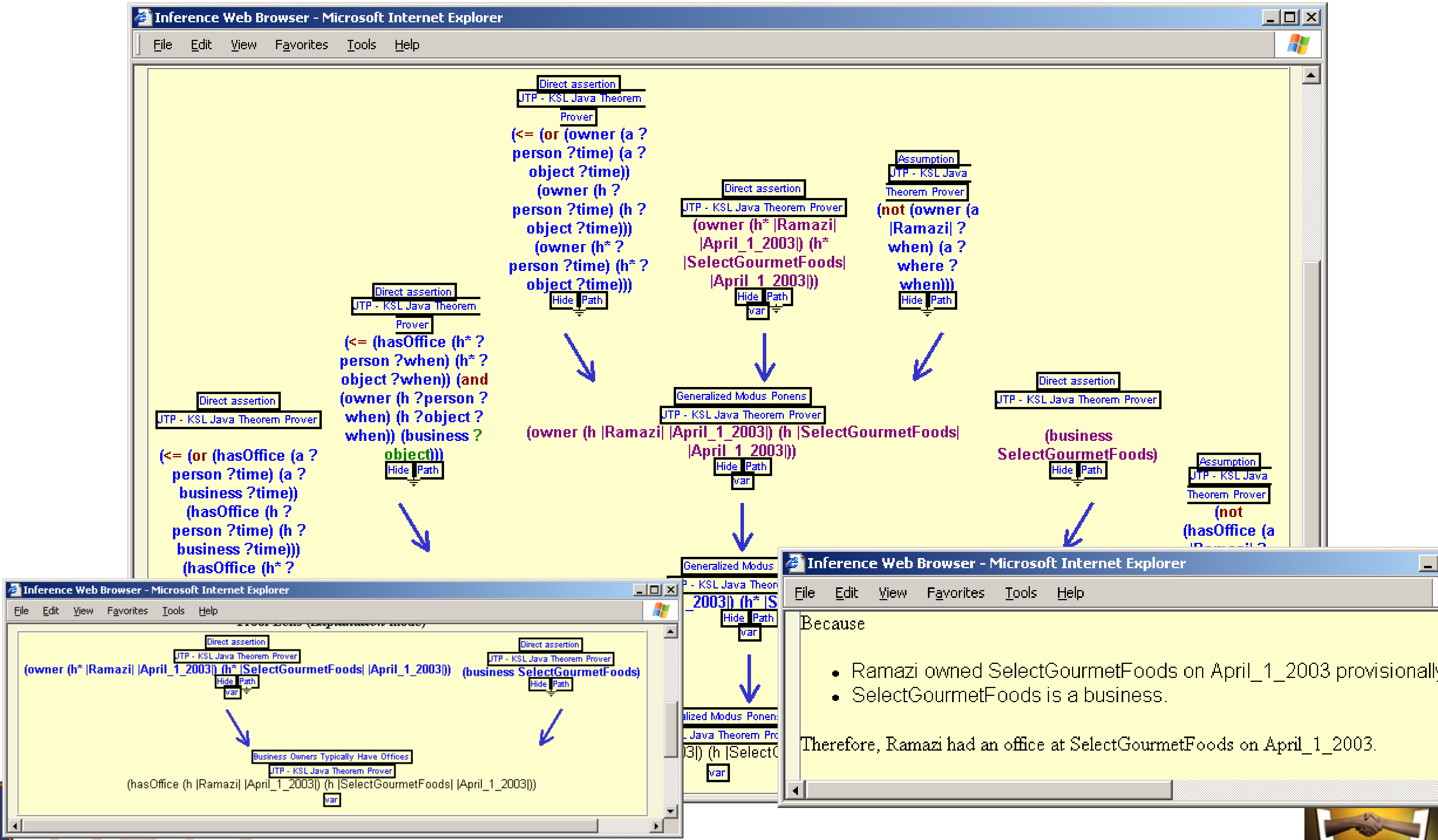
Conceal-request(Megan, [(Prescriptions, 1/2005)], Dr_Nelson, 12/2004)

The patient, Adam, wants to conceal prescriptions after May 2006* and lab reports after June 2006** from Dr. Gurtner [his previous physician].

Adjustable borders allow the source code and explanation windows to be selectively positioned or closed

Figure . Early design schematic for a visual interface for managing security policies.

A possible solution: use AI to convert proofs into explanations


Inference Web Browser - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Direct assertion
UTP - KSL Java Theorem Prover
(<= (or (owner (a ?person ?time) (a ?object ?time)) (owner (h ?person ?time) (h ?object ?time))) (owner (h* ?person ?time) (h* ?object ?time)))
Hide Path
Direct assertion
UTP - KSL Java Theorem Prover
(owner (h* |Ramazi| |April_1_2003|) (h* |SelectGourmetFoods| |April_1_2003|))
Hide Path var
Assumption
UTP - KSL Java Theorem Prover
(not (owner (a |Ramazi| ?when) (a ?where ?when)))
Hide Path
Direct assertion
UTP - KSL Java Theorem Prover
(<= (hasOffice (h* ?person ?when) (h* ?object ?when)) (and (owner (h ?person ?when) (h ?object ?when)) (business ?object)))
Hide Path
Generalized Modus Ponens
UTP - KSL Java Theorem Prover
(owner (h |Ramazi| |April_1_2003|) (h |SelectGourmetFoods| |April_1_2003|))
Hide Path var
Direct assertion
UTP - KSL Java Theorem Prover
(business SelectGourmetFoods)
Hide Path
Direct assertion
UTP - KSL Java Theorem Prover
(<= (or (hasOffice (a ?person ?time) (a ?business ?time)) (hasOffice (h ?person ?time) (h ?business ?time))) (hasOffice (h* ?
Assumption
UTP - KSL Java Theorem Prover
(not (hasOffice (a |Ramazi| ?
Generalized Modus
UTP - KSL Java Theor
_2003|) (h* |S
Hide Path var
alized Modus Ponen
Java Theorem Pro
03|) (h |Select
var
Inference Web Browser - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Direct assertion
UTP - KSL Java Theorem Prover
(owner (h* |Ramazi| |April_1_2003|) (h* |SelectGourmetFoods| |April_1_2003|))
Hide Path var
Direct assertion
UTP - KSL Java Theorem Prover
(business SelectGourmetFoods)
Hide Path
Business Owners Typically Have Offices
UTP - KSL Java Theorem Prover
(hasOffice (h |Ramazi| |April_1_2003|) (h |SelectGourmetFoods| |April_1_2003|))
var
Inference Web Browser - Microsoft Internet Explorer
File Edit View Favorites Tools Help

Because

- Ramazi owned SelectGourmetFoods on April_1_2003 provisionally
- SelectGourmetFoods is a business.

Therefore, Ramazi had an office at SelectGourmetFoods on April_1_2003.

Policy analysis is the #2 open problem

We need to develop tools for analyzing *large* sets of policies

- Safety
- Availability
- What-if?
- Why?

both for policy administrators and ordinary users

even in heterogeneous systems.

Challenges #1 & #2 should keep us busy for the next decade!

Lack of real-world experience is challenge #3

- Cassandra health care policies
- Shibboleth installations--- but only one-shot unilateral trust, with a closed set of organizations

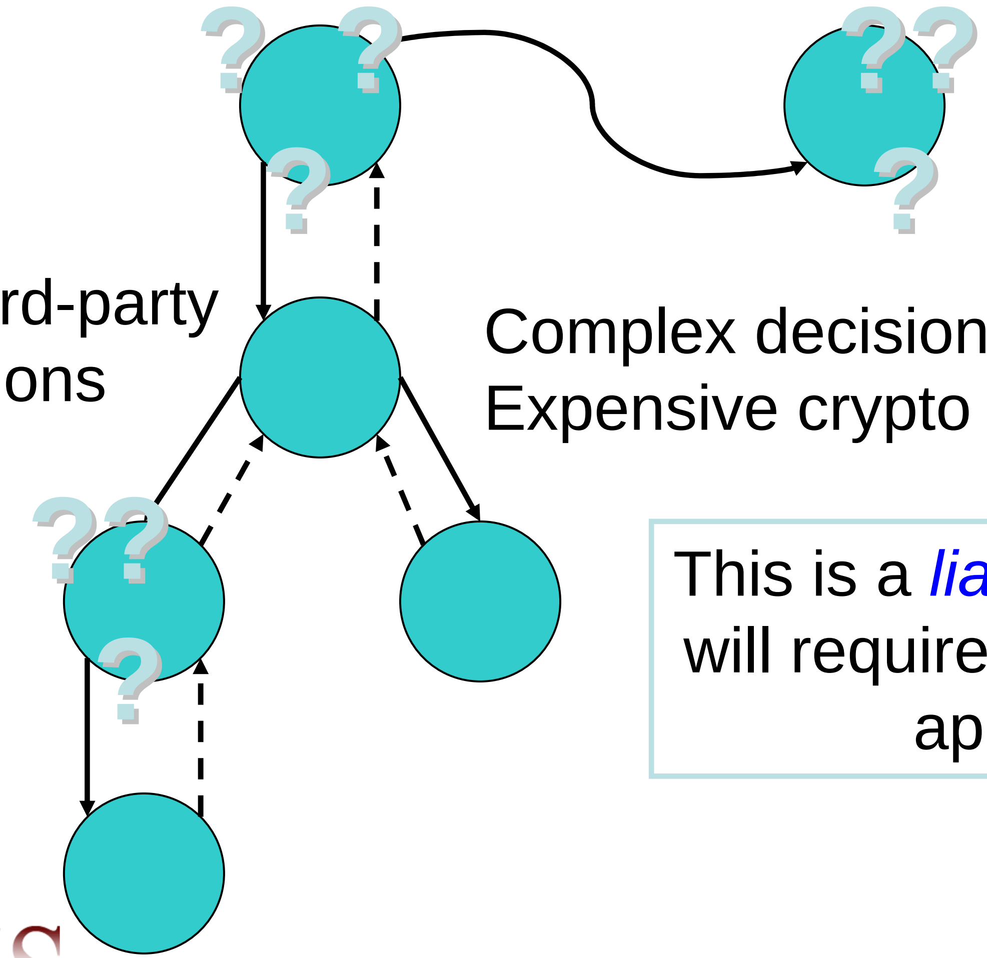
We need more feedback from the real world to ensure that we are addressing the most important problems in policy-based authorization!

Vulnerability to attack is #4

- Centralized authorization servers are attractive target
- TN is heavyweight → DDoS is *so* easy

TN is heavyweight

Multiple rounds of exchange



(Nested) third-party interactions

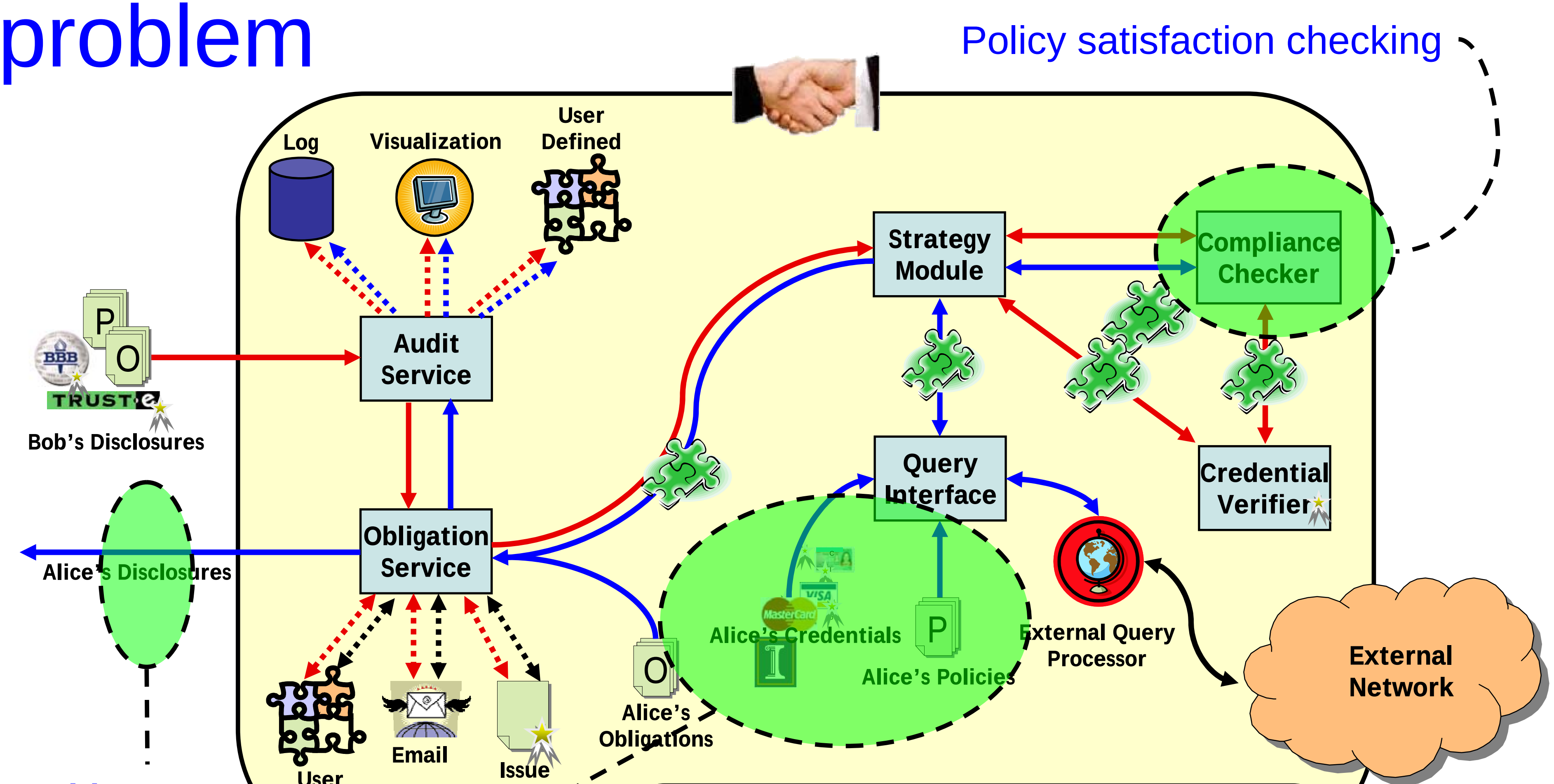
Complex decision making processes
Expensive crypto

This is a *liability*. Solutions will require a multi-faceted approach.

Poor understanding of systems issues is #5

- How should we build the policy engine?
  - Certainly not a Datalog theorem prover!
  - How can we integrate it with strategic decisions?
  - How can we make the policy engine reusable in other contexts (e.g., for analysis)?
- How can we make a TN implementation flexible?

TrustBuilder2 addresses the flexibility problem



Message serialization

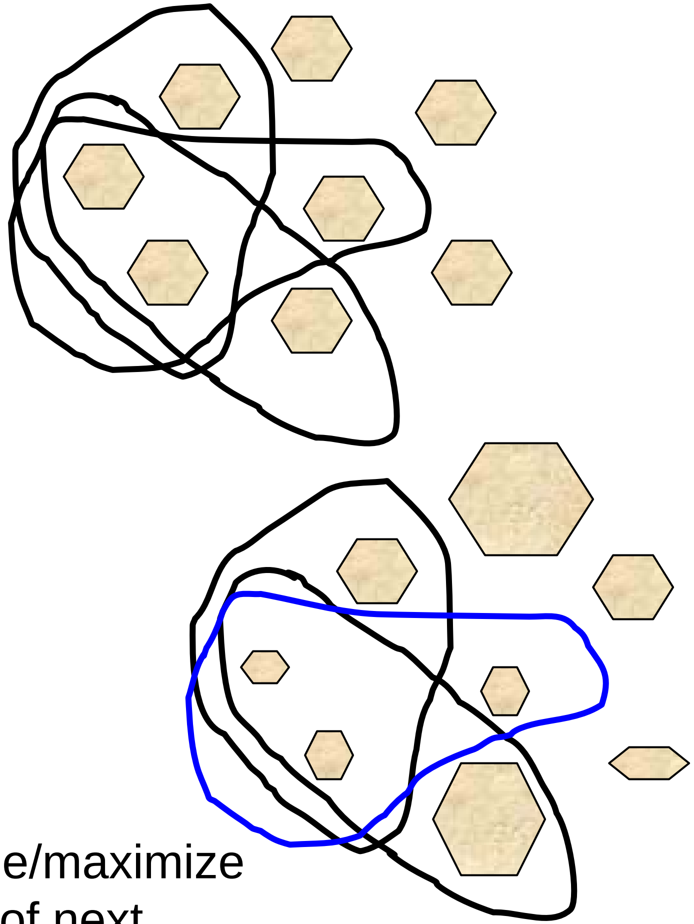
Caching

User-Supplied Plug-ins

- Anomaly Detection
- Logging, visualization, or instrumentation
- Policy composition or rewriting
- State inspection
- Disclosure modification
- Etc.

Policy compliance checking is slow

Theorem Prover



Minimize/maximize
“value” of next
disclosure

Choice of "best" way to satisfy a policy depends on strategic goals

- Service availability

  - e.g., closeness to ideal completeness

- Privacy preservation
  - e.g., control leaks or minimize "value" of disclosed credentials

- Computational overheads

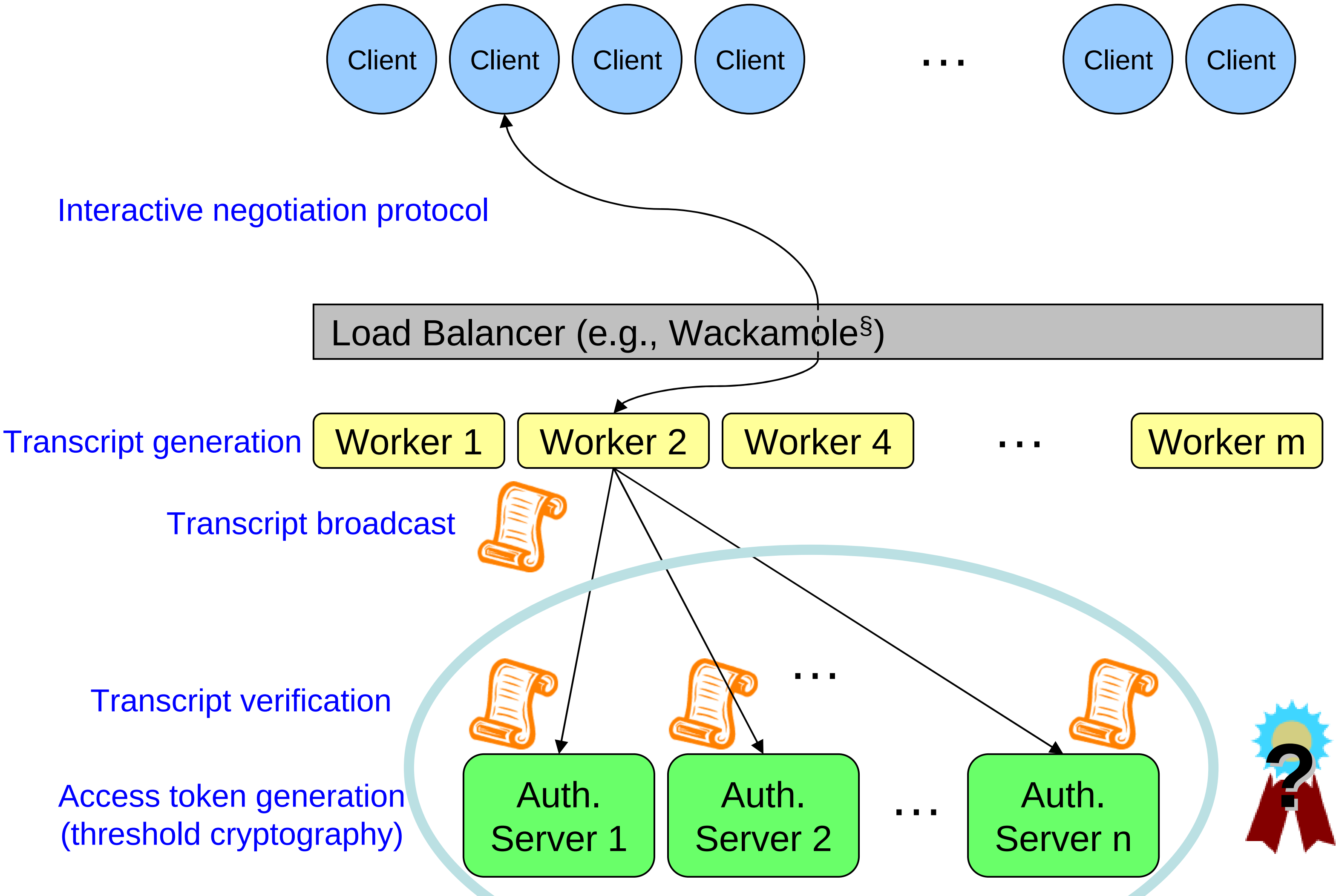
- Storage requirements

Rete is fast for compliance checking

Less than 4 seconds to find hundreds of satisfying sets, pick the one with minimal weight (new work)

Ships with Trustbuilder2!

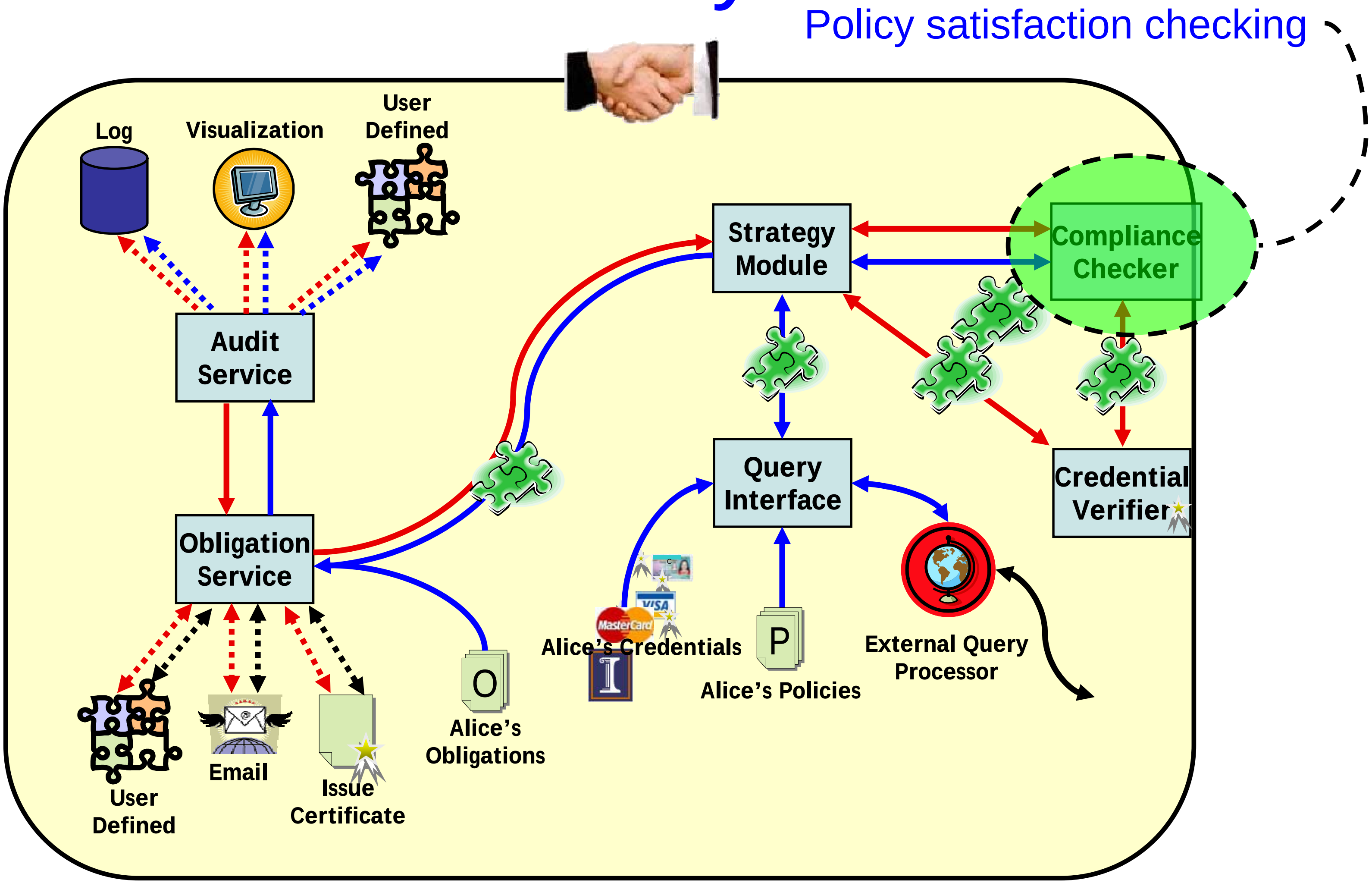
Delegation and replication can improve availability, performance of decentralized ABAC



[§] Y. Amir, R. Caudy, A. Munjal, T. Schlossnagle, C. Tutu, "N-Way Fail-Over Infrastructure for Reliable Servers and Routers," IEEE International Conference on Dependable Systems and Networks (DSN '03), June 2003.

How to integrate strategic decisions with other functionality?

I have no idea



Five other cool problems

1. How to implement sticky policies?
2. Can TN research give insights into distributed proof construction?
3. Theoretical ABAC / TN issues (pick one)
4. How to build a reputation system in a world without global identities?
5. Can programming languages use TN?

How to implement sticky policies?

I have no idea.

TN has close ties to distributed proof construction

Trust negotiation

- Bonatti and Samarati (CCS 2000)
- Yu, Winslett, and Seamons (TISSEC 2003)
- Li and Mitchell (DISCEX 2003)
- Becker and Sewell (POLICY 2004)
- Bertino, Ferrari, Squincciarini (IEEE TKDE 2004)
- Li, Li, and Winsborough (CCS 2005)
- And many others…

Distributed proof construction

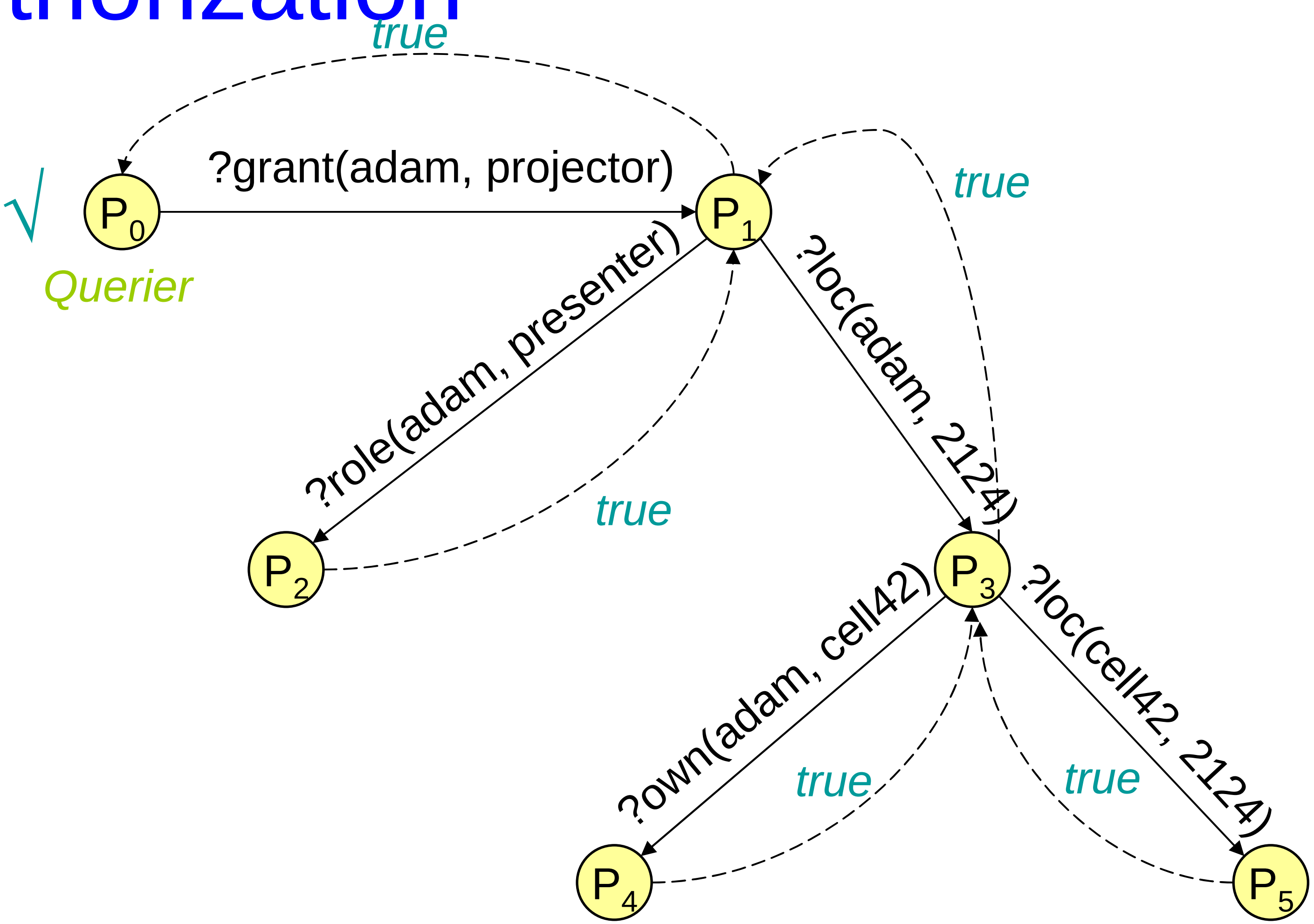
- Bauer, Gariss, and Reiter (Oakland 2005)
- Winslett, Zhang, and Bonatti (CCS 2005)
- Minami and Kotz (JPMC 2005, Pervasive 2006)

Logics

Policy languages

Strategies and proof tactics

Example distributed proof of authorization

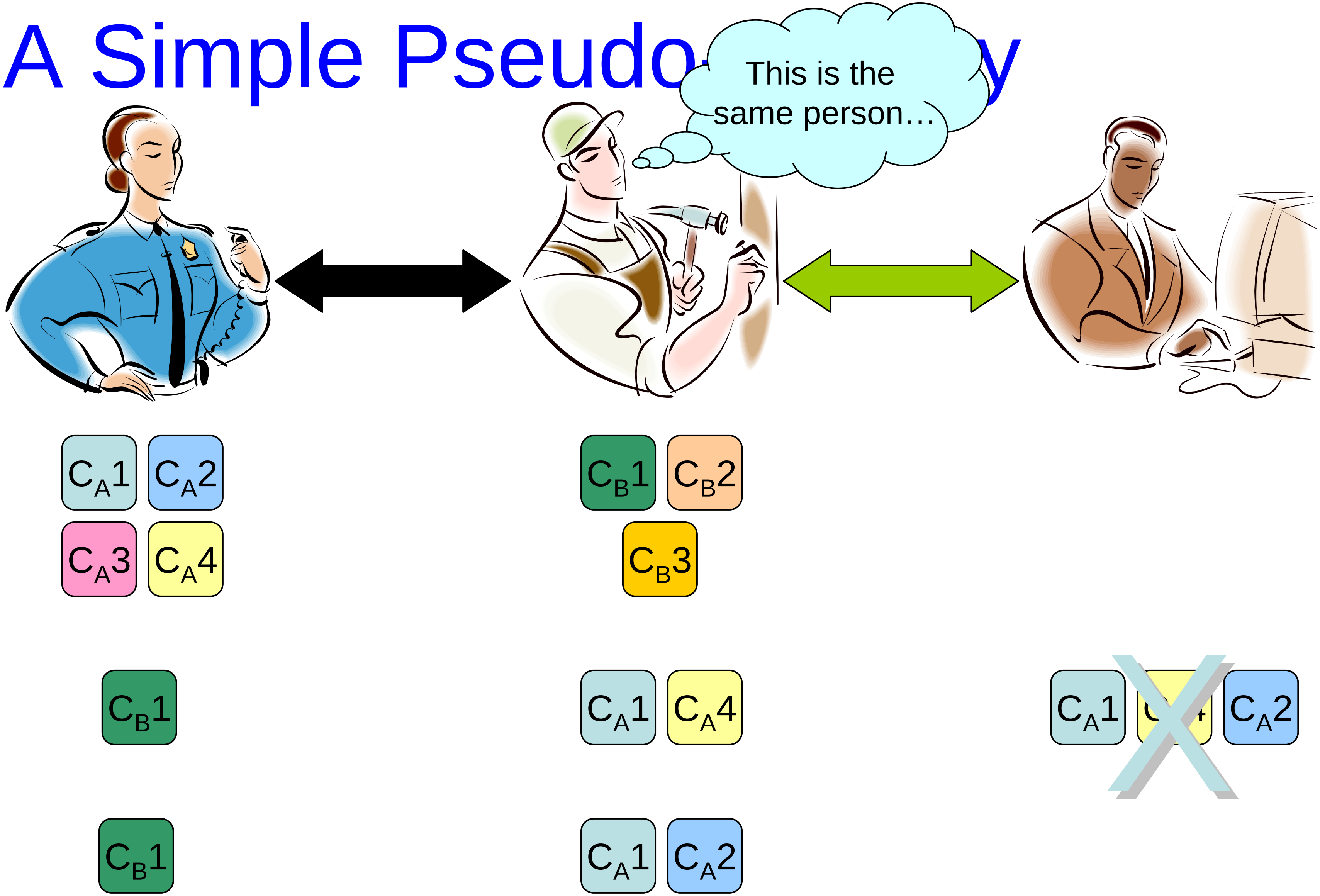


Without concrete user identities, how can we build support services?

- E.g., Reputation, audit, collusion detection
- Attribute certificates need not be bound to a particular *identity*

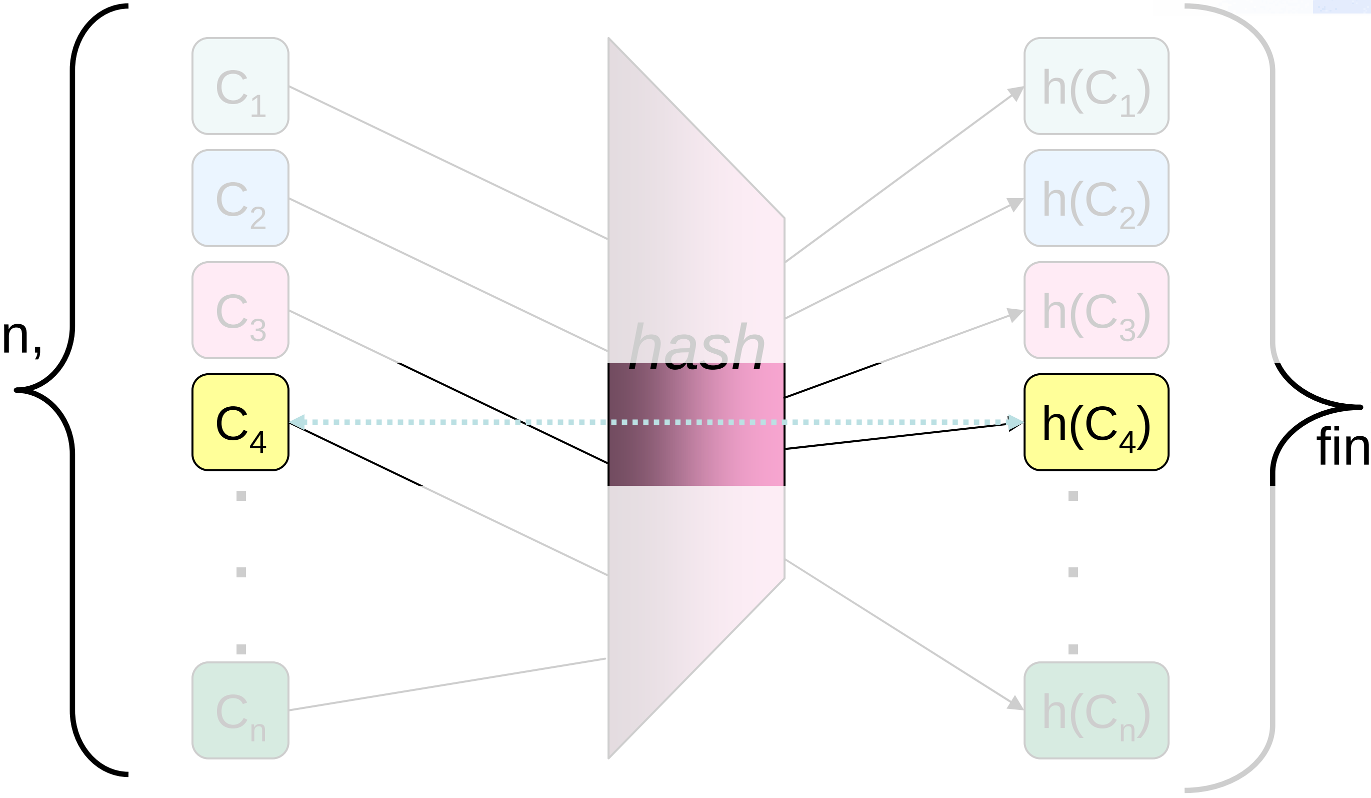
Observation: Each entity is described uniquely by the collection of credentials that she possesses

A Simple Pseudo y



Virtual fingerprints are privacy-preserving pseudonyms

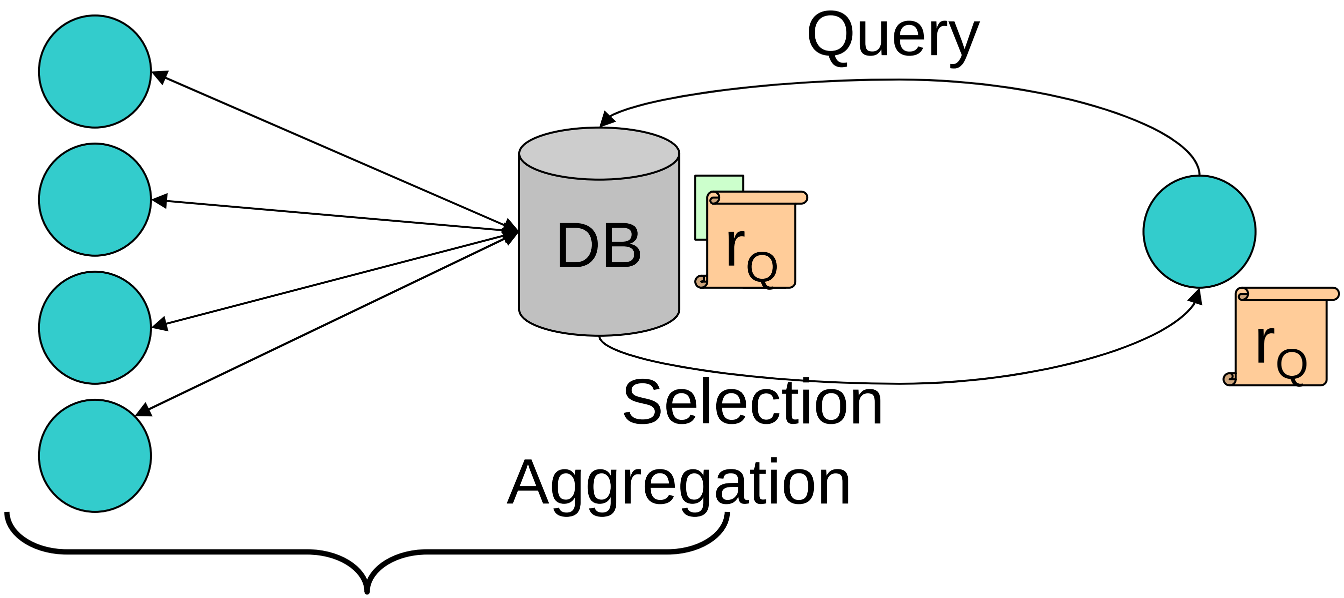
A description, *d*, …



… a virtual fingerprint, *f*

… maps to …

We can query reputation information associated with virtual fingerprints

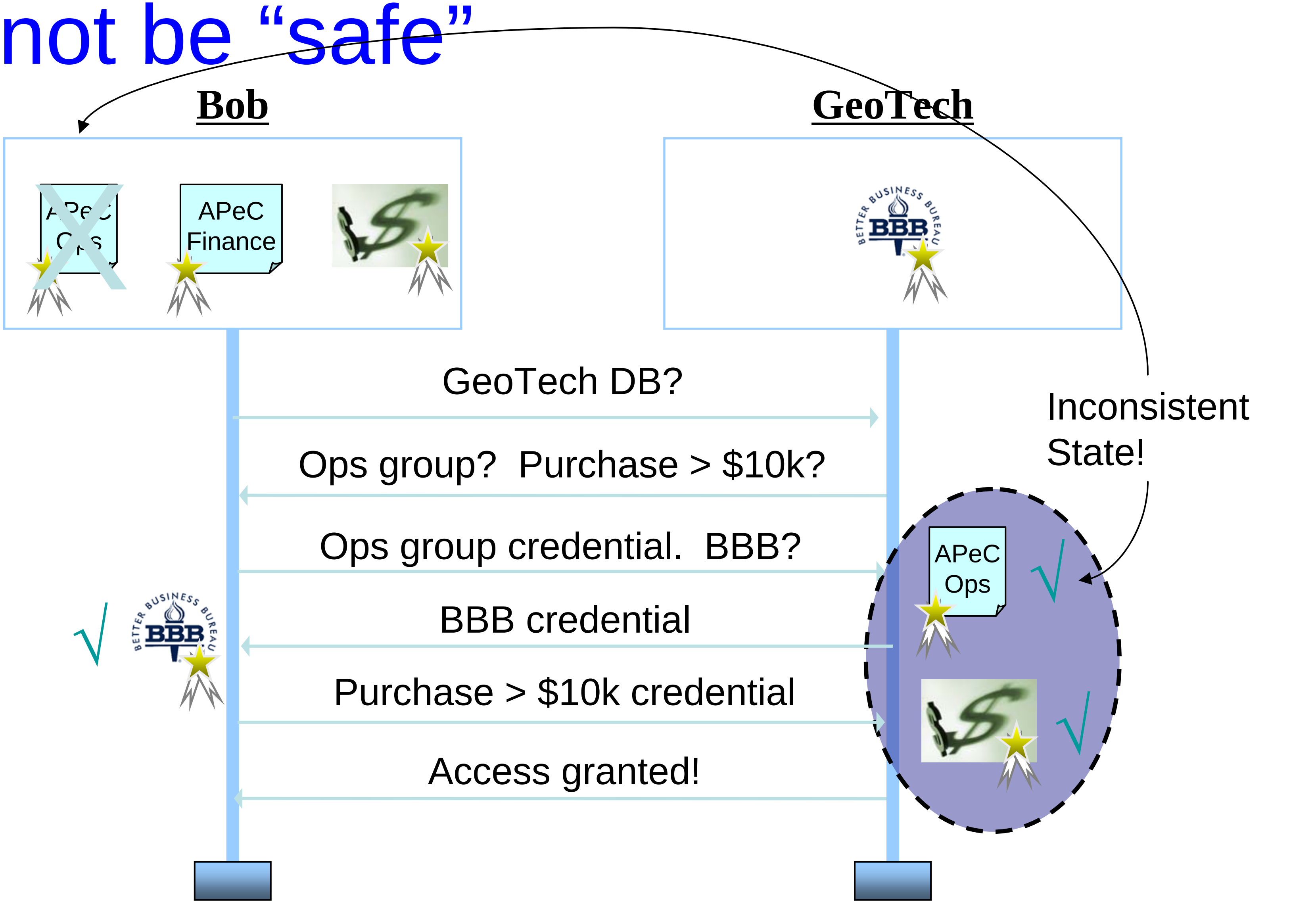


Rating collection and update

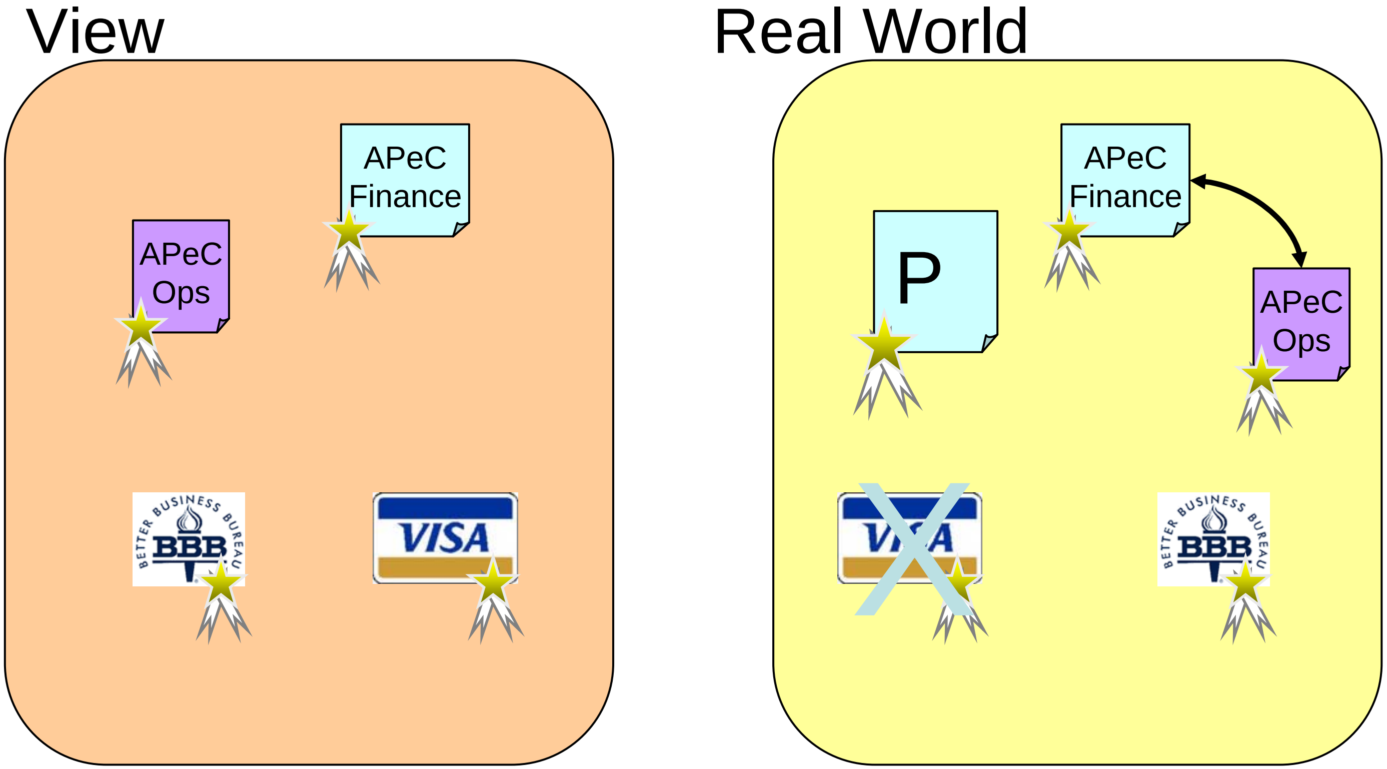
Collection, update, and selection independent of aggregation

- Improved reputation functions can be incorporated
- Existing reputation models can now be used in ABAC systems

A theory problem: access decisions may not be “safe”

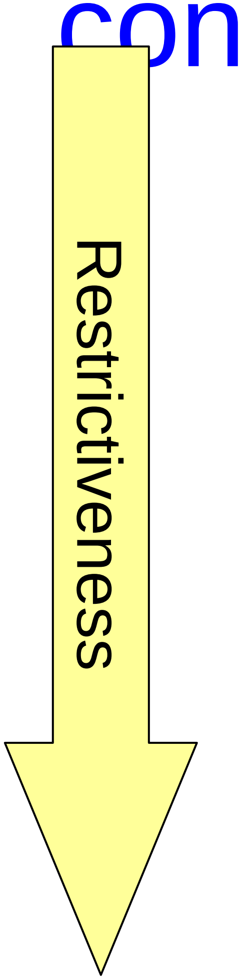


Incremental evaluation of credential validity may not be enough



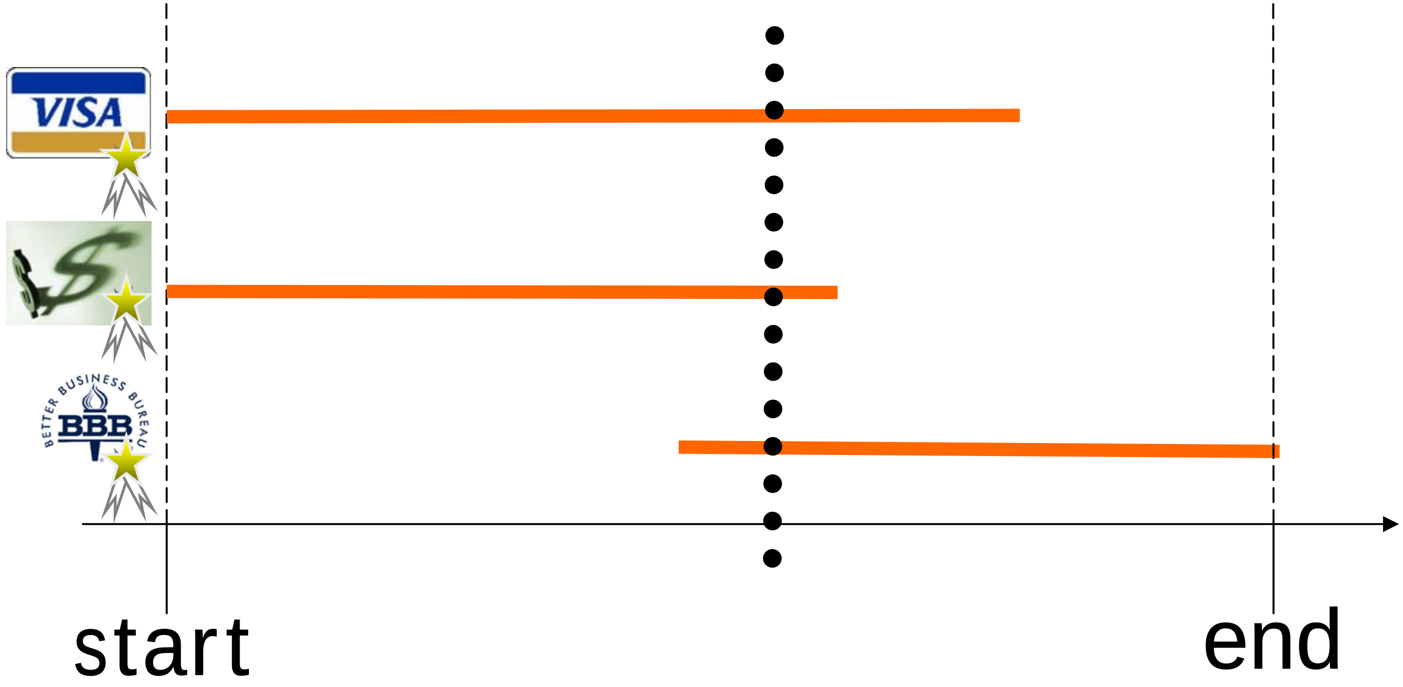
Similar consistency problems arise in other domains

Several possible levels of consistency


Restrictiveness

- Incremental
  - Credentials validated as they are received
- Internal
  - Credentials valid simultaneously at *some* time during protocol
- Endpoint
  - Credentials valid simultaneously at decision point
- Interval
  - Credentials valid from time received until decision point

Internal consistency = transactional semantics



Parties have no incentive to cooperate in the traditional transactional manner, but new implementation approaches can be used

In sum: my top 10 open problems for policy-based authorization

1. Policy HCI
2. Need for real-world feedback
3. Policy analysis
4. Vulnerability to attack
5. Systems issues (especially integration of strategic decisions with the rest of the system)
6. 7. 8. 9. 10. Other fun stuff