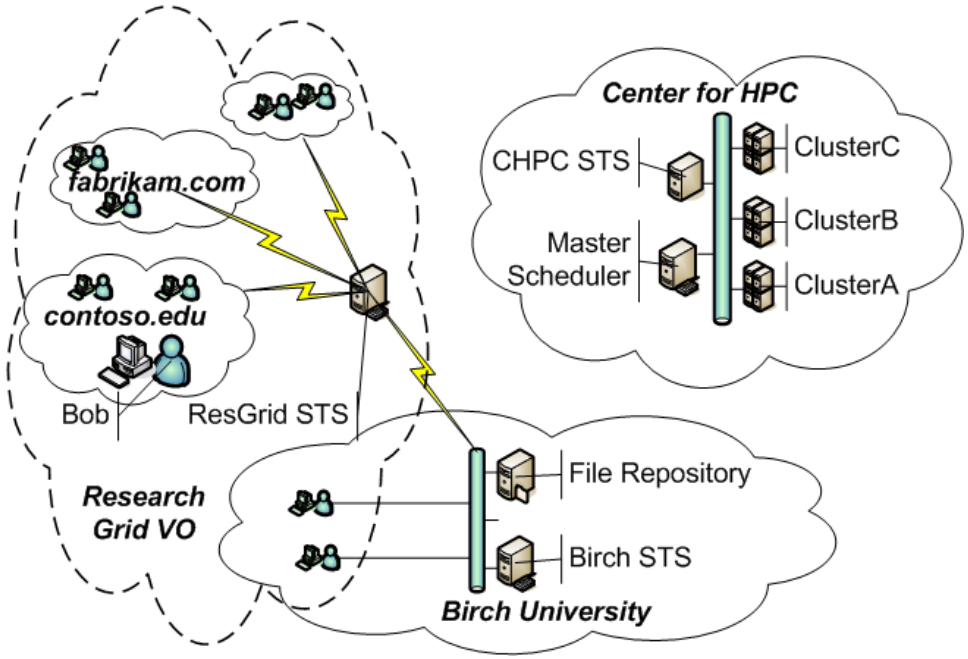
# Authorization

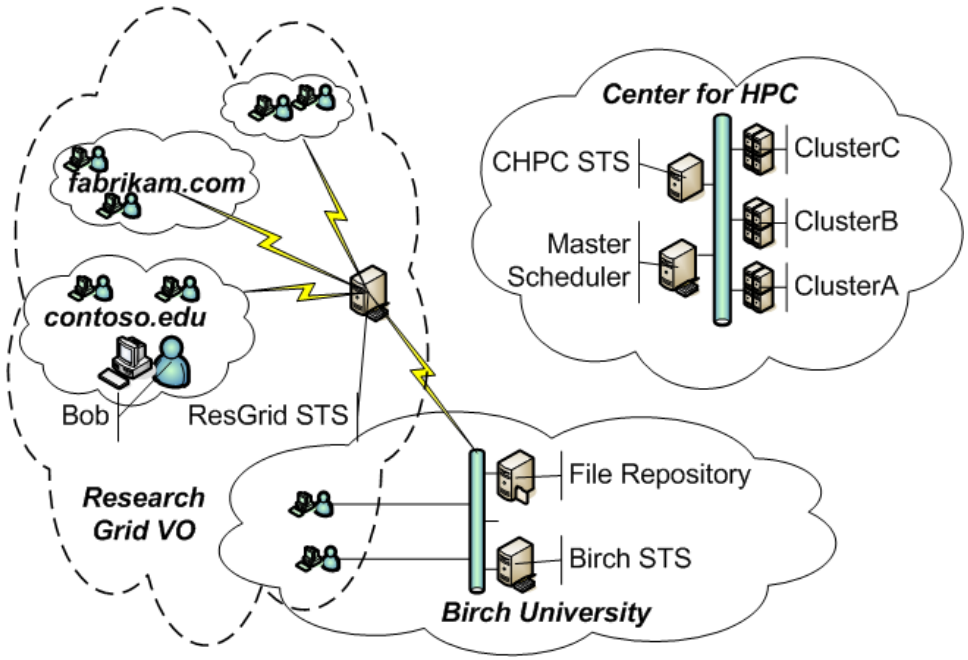
## Security Policy Assertion Language

# The Grid



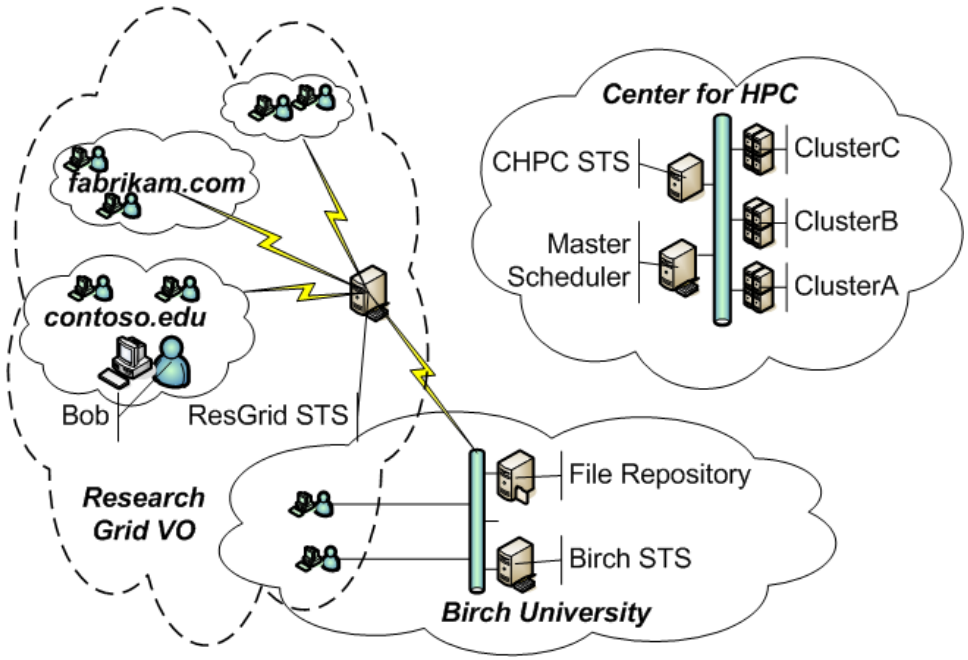

- Resources and user belong to a variety of different independent organizations
- Resources and users are connected via communication networks
- A virtual organization (VO) is a set of independent collaborating (real) organizations who establish a trust relationship for the purpose of sharing resources and skills to achieve a common objective

# The Problems




- users are identified by their (local) organization and are unknown to other organizations in the VO
- resources are controlled by policies defined by their controlling organizations
- a user may want to combine the use of resources from different organizations for which the user has been separately authorized

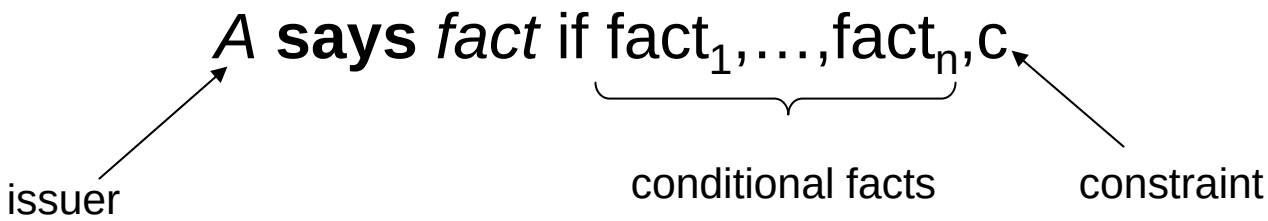
# The Goals




- Describe explicit trust relationships
- Express security token issuance policies
- Provide security tokens that contain identities, capabilities, and/or delegation policies
- Express resource authorization and delegation policies

# Elements

- Security tokens - digitally signed statements relevant to the authorization process (e.g., identities, capabilities, delegations)
- Security Token Server (STS) – a server that issues security tokens on behalf of a security principal
- Security Principal –
  - an entity capable of issuing authoritative statements (may be a person, organization, or service)
  - identified by a cryptographic key (e.g. K-ResGrid is the public key for the principal ResGrid)
- Assertion – a statement that a security principal believes to be valid possibly depending on other facts and constraints



# Types of Assertions

- Attribute
  - Expressing a binding between a principal and one or more attributes

```
STS says Alice is a researcher
```

- Capability
  - Expressing the right of a principal to exercise one or more actions on a resource

```
FileServer says Alice can read /project
```

- Delegation
  - Expressing the granting of a capability possessed by one principal to a second principal

```
Alice says Cluster can read /project/data
     If currentTime() <= 07/09/2006
```

- Trust
  - Expressing the willingness of one principal to believe certain types of assertions made by a second principal

```
Cluster says STS can say x is a researcher
FileSys says Univ can say x can say y can read /project
```

# Variables

- An assertion may contain variables (see previous examples).
- Variables
  - are strongly typed
  - can be unrestricted (bind to any concrete value of the correct type)
  - can be restricted to a subset of concrete values based on a specific pattern
- A phrase is “ground” when it has no variables
- Examples

```
Cluster says x can execute dbgrep if x is a researcher

FileServer says x can say y can read file if
x can read dir, file in dir, markedConfidential(file)=no

(The later is a constrained delegation rule)
```

# Constraints, Flat

- Constraints
  - Equality and inequality
  - Path constraints (hierarchical resources like file systems)
  - Regular expressions (patterns)

- Flat
  - A fact is “flat” if it does not include “can say” and nested otherwise
  - “Bob **can read** *f*” is flat
  - “Charlie **can say** Bob **can read** *f*” is nested

# Patterns

- The SecPAL prototype uses the pattern-matching symbols shown in the table

| Pattern | Matches |
|---|---|
| ^ | beginning of line |
| $ | end of line |
| . | any single character |
| [ … ] | any character in … |
| *x*-*y* | any character in the range *x* to *y* |
| *x*+ | one or more occurrences of *x* |
| (*x*?) | character *x* if it occurs |
| \ | escape |
| \w | single character in a-zA-Z0-9 |
| *character* | itself |

- Examples:

K-CHPC says K-ResGrid can say x possess rfc822Name=^[-_a-zA-Z0-9]+@[-_a-zA-Z0-9]+$

K-CHPC says K-Birch can say x possess serviceName=^http(s?):\w+\.birch\.edu/\w$

# Deduction Rules

$$\text{(cond)}\ \frac{(A \text{ says } fact \text{ if } fact_1, ..., fact_k, c) \in AC \quad AC, D \models A \text{ says } fact_i\theta \text{ for all } i \in \{1..k\} \qquad \models c\theta \qquad vars(fact\theta) = \emptyset}{AC, D \models A \text{ says } fact\theta}$$

$$\text{(can say)}\ \frac{AC, \infty \models A \text{ says } B \text{ can say}_D \ fact \qquad AC, D \models B \text{ says } fact}{AC, \infty \models A \text{ says } fact}$$

$$\text{(can act as)}\ \frac{AC, D \models A \text{ says } B \text{ can act as } C \qquad AC, D \models A \text{ says } C \ verbphrase}{AC, D \models A \text{ says } B \ verbphrase}$$

- AC is the assertion context
- D is the delegation flag (0=no delegation, infinity is unbounded delegation)
- $\theta$ is a binding of variables to constants and variables
- *vars(f)* is the set of free variables in *f*

# Using the deduction rules

Assertions:

STS says Alice is a researcher (1)

Cluster says STS can say x is a researcher (2)

Cluster says x can execute dbgrep if x is a researcher (3)

Proof of "Cluster says Alice can execute dbgrep":

Cluster says STS can say x is a researcher (2)

STS says Alice is a researcher (1)

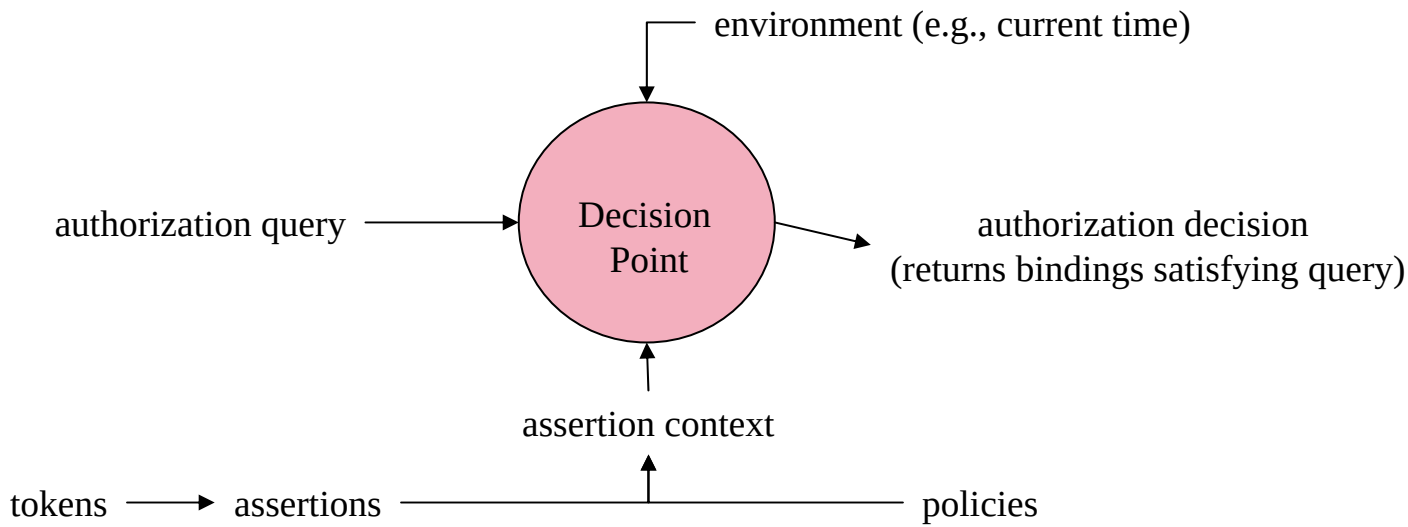
Cluster says Alice is a researcher (can say)(4)

Cluster says x can execute dbgrep if x is a researcher (3)

Cluster says Alice is a researcher (4)

Cluster says Alice can execute dbgrep (cond) (5)

# Authorization Queries




- Authorization query:
  K-ResGrid says *x* possess rfc822Name=*e*
- Authorization decision:
  K-ResGrid says K-Bob posess rfc822Name=bob@contoso.edu

# Authorization Query Table

- Provided by a local assertion context
- Maps parameterized operation names to predefined queries
- Resource guard invokes parameterized operation
- Example (containing deny-overrides):

`check-access-permission`(*x*):

`FileServer` **says** *x* has access from $t_1$ till $t_2$

$t_1$ <= currentTime() <= $t_2$,

not exists $t_3$,$t_4$ (

`FileServer` **says** *x* has no access from $t_3$ till $t_4$,

$t_3$ <= currentTime() <= $t_4$)

# Policy Idioms

- Mandatory Access Control (MAC)

`FileServer` **says** *x* can read *f* if

*x* is a user, *f* is a file, level(x) >= level(f)

`FileServer` **says** *x* can write *f* if

*x* is a user, *f* is a file, level(x) <= level(f)

- Roles

`NHS` **says** `FoundationTrainee` can read `/docs/`

`NHS` **says** `SpecialistTrainee` can act as `FoundationTrainee`

`NHS` **says** `SeniorMD` can act as `SpecialistTrainee`

`NHS` **says** `Alice` can act as `SeniorMD`

# Policy Idioms

- Attribute-based delegation: assigns permissions based on attributes rather than identity
- Example:

`Shop` **says** *x* is entitled to discount if

*x* is a student till *date*,
currentTime() <= *date*, currentDay() = `Friday`

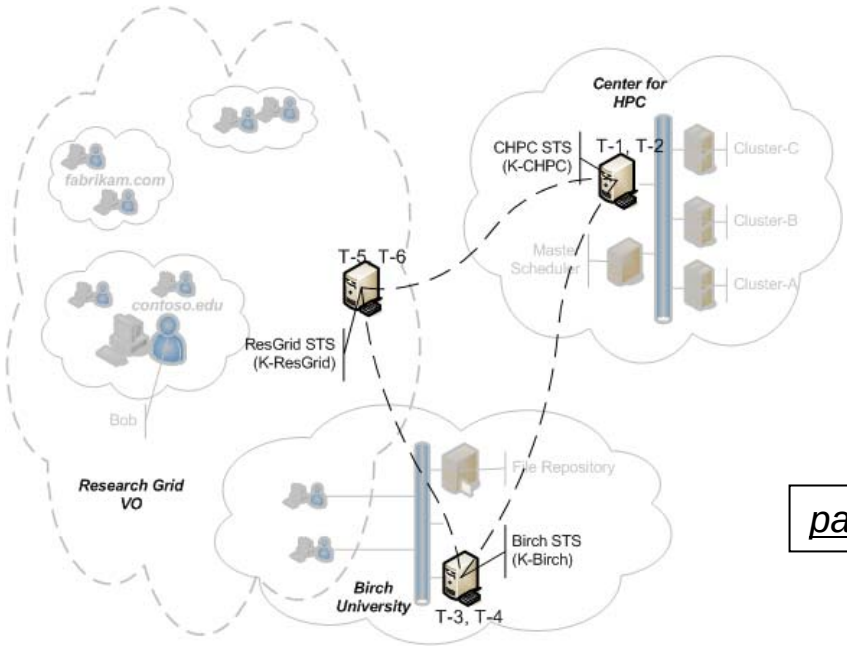
`Shop` **says** *univ* **can say** *x* is a student till date if

*univ* is a university,

`Shop` **says** *CommonwealthOfVirginia* **can say**

*univ* is a university

# Federated Trust

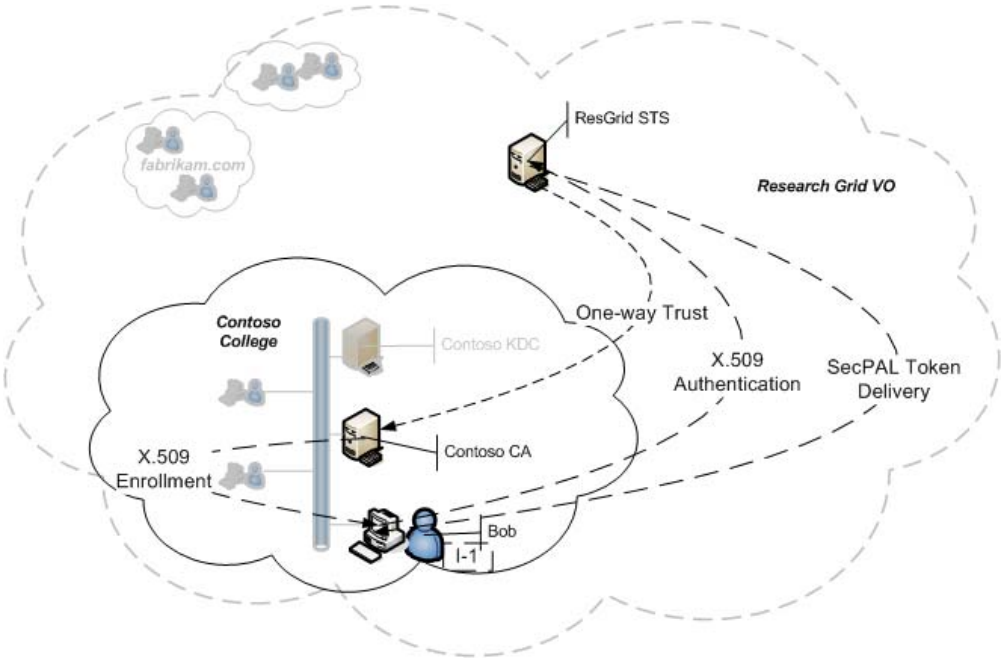



*pattern* denotes a pattern

## Trust Policies

T-1: K-CHPC says K-ResGrid can say *x* possess rfc822Name=*name*, groupName=ResGrid/*group*
T-2: K-CHPC says K-Birch can say x possess serviceName=http(s?)://*server*.birch.edu/*service*
T-3: K-Birch says K-ResGrid can say *x* possess rfc822Name=*name*, groupName=ResGrid/*group*
T-4: K-Birch says K-CHPC can say x possess appName=*app*,dnsName=*name*.chpc.com
T-5: K-ResGrid says K-Birch can say *x* possess serviceName=http(s?)://*service*.birch.edu
T-6: K-ResGrid says K-CHPC can say *x* possess serviceName=http(s?)://*server*.c-hpc.com/*service*

# Identity Token Acquisition




## Steps

1. Bob receives X.509 identity certificate from Contoso CA
2. ResGrid trusts Contoso CA to issue X.509 identity certificates
3. Bob passes certificate to ResGrid STS
4. ResGrid STS issues SecPAL token

## Assertions

ResGrid STS trust policy: K-ResGrid says K-Contoso can say *x* possess rfc822Name=*name@contoso.edu*
ResGrid from X.509 cert.: K-Contoso says K-Bob possess rfc822Name=bob@contoso.edu
ResGrid evaluates/issues: K-ResGrid says K-Bob possess rfc822Name=bob@contoso.edu