ACCESS CONTROL

Fadi Mohsen, PhD

OUTLINE

- Security Principles
- Access Control
  - Introduction,
  - Models,
  - Languages/Technologies,
  - Analysis and Enforcement
- Federated Setting
  - Technologies (OAuth2, OpenID Connect)
  - Tokens

ACCESS CONTROL

- **Authentication:** Are you who you say you are?
  - A binary decision-access is granted or it is not.
  - Authenticate human to machine
  - Or, possibly, machine to machine
- **Authorization:** Are you allowed to do that?
  - Once you have access, what can you do?
  - A more fine-grained set of restrictions on access to various system resources
- Note: "access control" often used as synonym for authorization

ARE YOU WHO YOU SAY YOU ARE?

- Authenticate a human to a machine?
- Can be based on…
  - Something you **know**
    - For example, a password, PIN
  - Something you **have**
    - For example, a smartcard, ATM
  - Something you **are**
    - For example, your fingerprint, iris
- Why is "something you know" more popular than "something you have" and "something you are"?

Latest authentication news: [Click here]
Latest multi-factor authentication news: [Click here]

SECURITY (DESIGN) PRINCIPLES

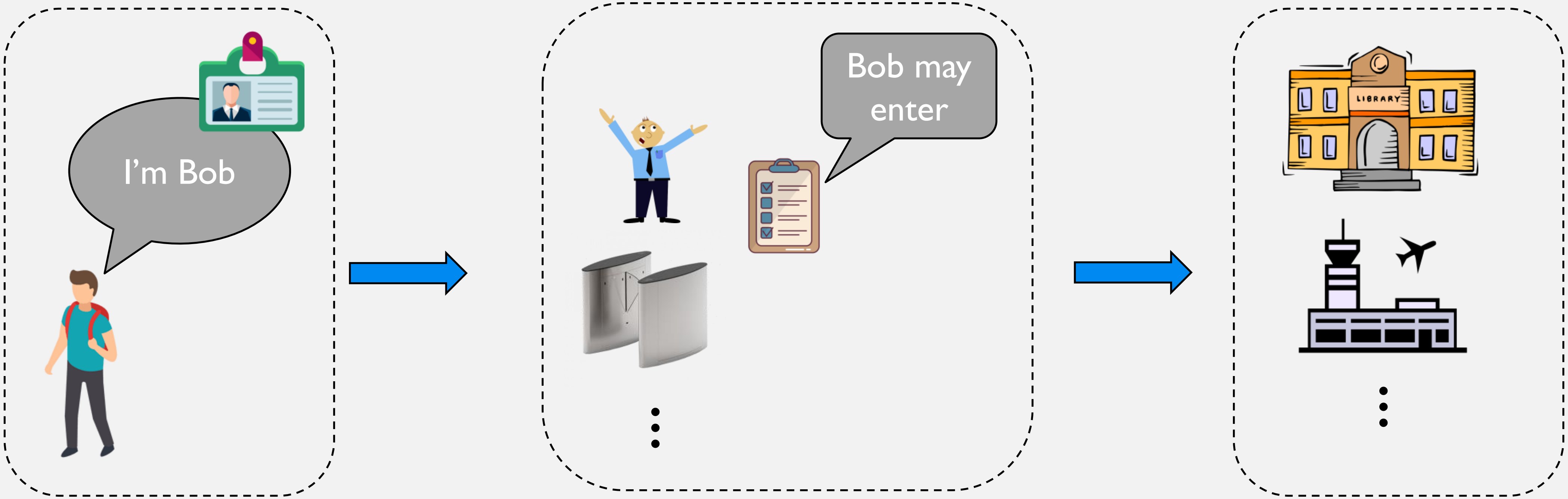
1. **Least Privilege → *allow only what is needed***
2. Fail-safe defaults. → *default permission is "deny"*
3. Economy of Mechanism → *as simple as possible*
4. **Complete Mediation → *every access should be checked***
5. Open Design → *secrecy (of design/implementation) should not be the source of security*
6. **Separation of Privilege/Duty → *more than one condition***
7. Least Common Mechanism → *access mechanisms should not be shared*
8. Least Astonishment → *understandable by the users*

See [1,2,3] for more details

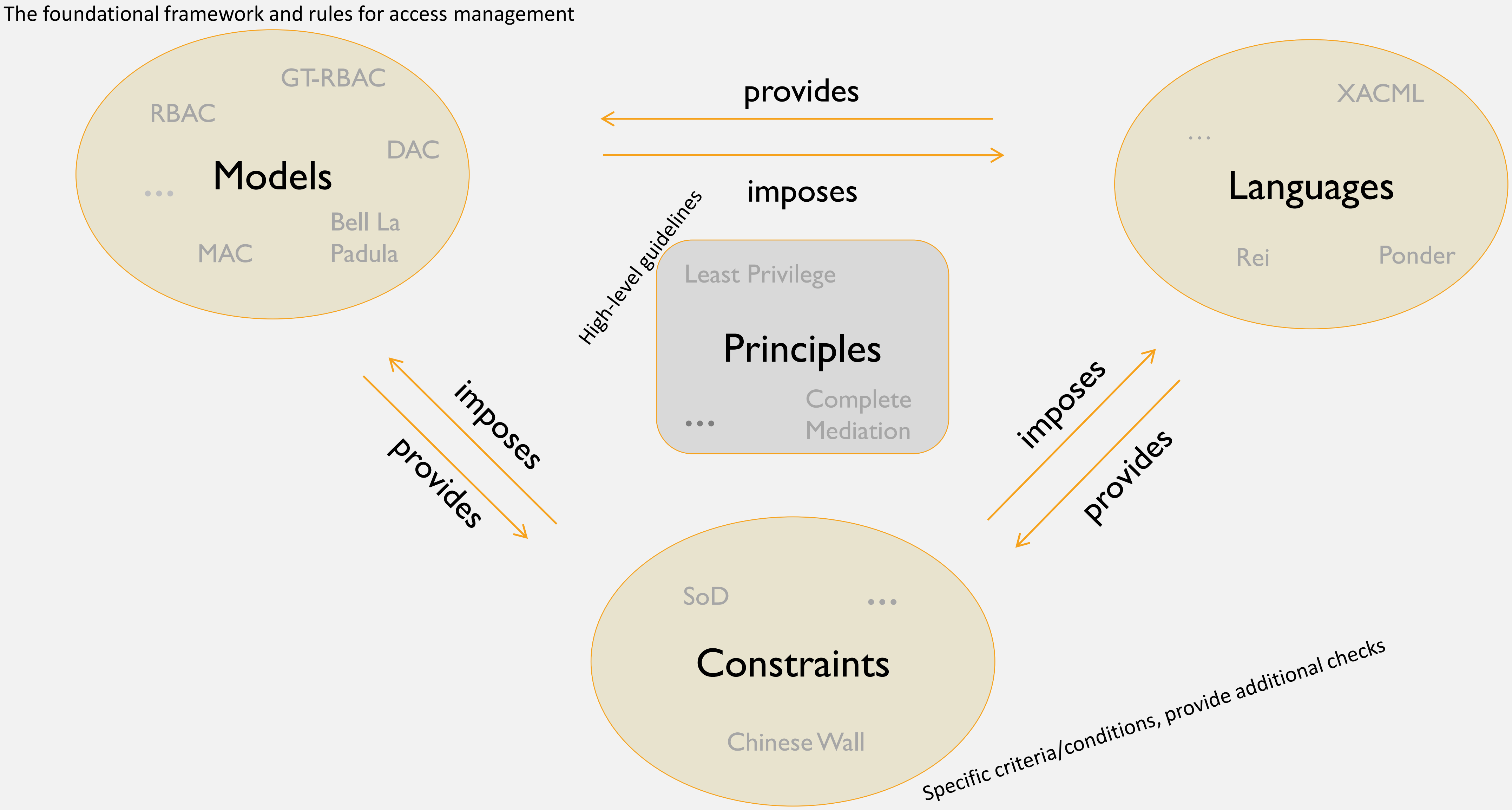
ACCESS CONTROL

Who can do **what** on **which resource**

Subject Action Object



ACCESS CONTROL SYSTEMS

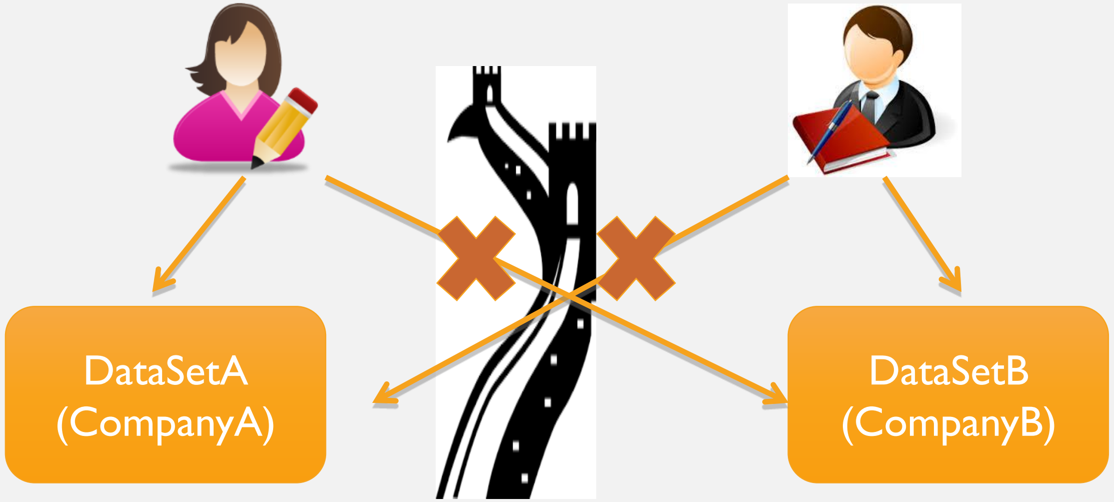


CONSTRAINTS: EXAMPLE

- L&G is a law-firm
  - Customers: CompanyA and CompanyB; competitors
  - Lawyers in L&G can access to case data

  according to an **authorization/access control policy**

- L&G applies Chinese Wall when dealing with the cases/data of **CompanyA** and **CompanyB**



ACCESS CONTROL OVERVIEW

- Authorization Components
  - Subjects: *Alicia*, *Will*, *Kerry*, …
  - Objects: *DataSetA* and *DataSetB*
  - Actions: Read, Write, …
- Access rules are encoded
  - Alicia may access to DataSetA and DataSetB
  - Will may access to DataSetA and DatasetB
  - Kerry may not access to DataSetA
  - …

Users

Resources

Rights

Policy

“A **policy** is written according to a **model** using a **language**”

ACCESS CONTROL MODELS

- Types of Access Control :
  - Discretionary Access Control (DAC): Owners can “pass” permissions. Examples: UNIX File system,…
  - Mandatory Access Control (MAC): Central authority imposes rules. Examples: Military systems…

- Access Control Matrix[1,2]

	File1	File2	File3
Alice	read, write	execute	read
Bob	own	read	read, write

- Access Control Lists

File1
Alice, {read, write}
Bob, {own}

File2
Alice, {execute}
Bob, {read}

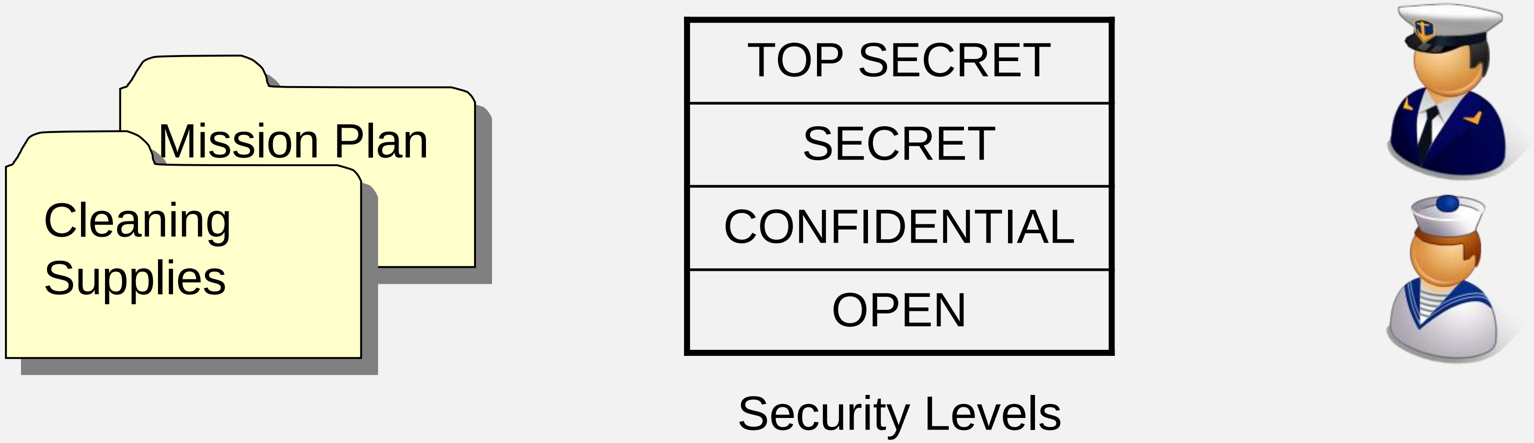
File3
Alice, {read}
Bob, {read, write}

- Capability list (C-list)

Alice
File1, {read, write}
File2, {execute}
File3, {read}

MULTI-LEVEL SECURITY (MLS)

Bell-LaPadula Model



Mission Plan

Cleaning Supplies

TOP SECRET

SECRET

CONFIDENTIAL

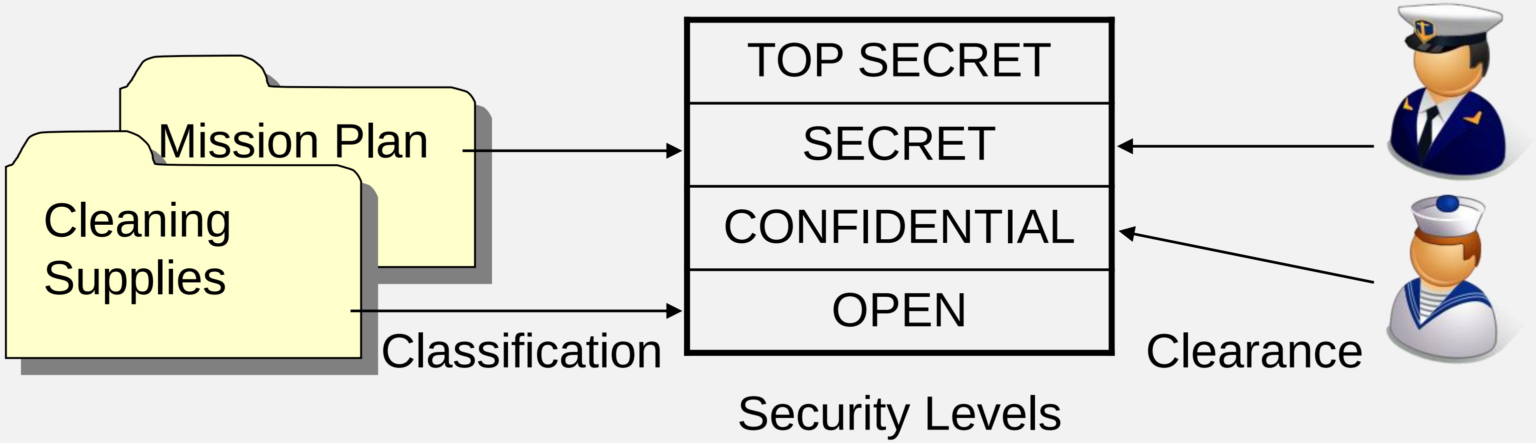
OPEN

Security Levels

Credits : Dr. Jerry den Hartog

MULTI-LEVEL SECURITY (MLS)

Bell-LaPadula

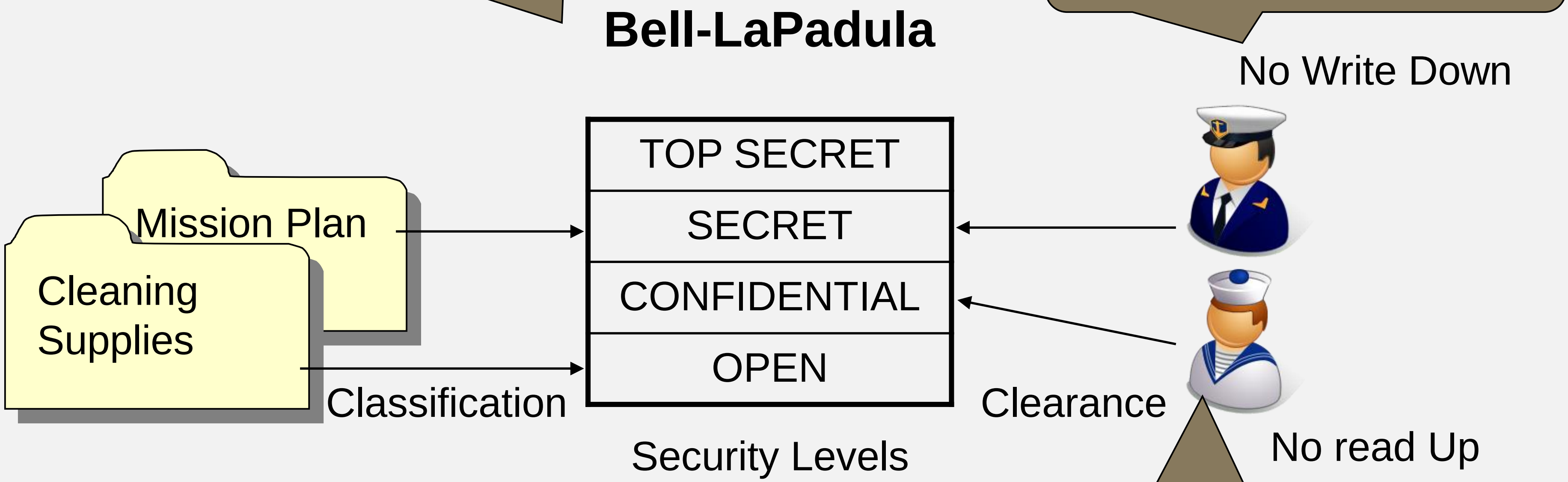


Credits : Dr. Jerry den Hartog

MULTI-LEVEL SECURITY (MLS)

Protects **confidentiality** but not **integrity**

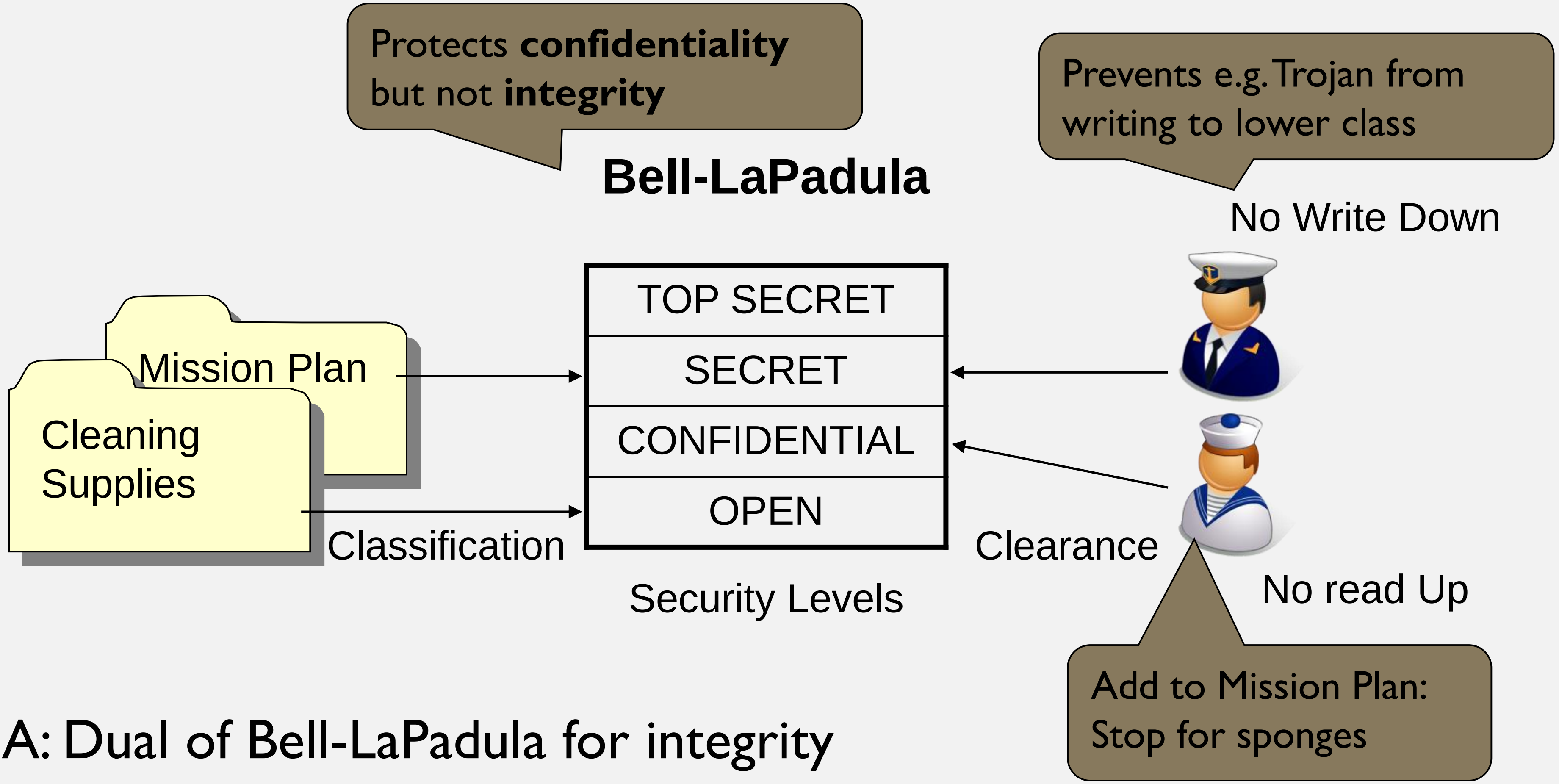
Prevents e.g. Trojan from writing to lower class



Add to Mission Plan: Stop for sponges

Credits : Dr. Jerry den Hartog

MULTI-LEVEL SECURITY (MLS)



Protects **confidentiality** but not **integrity**

Bell-LaPadula

Prevents e.g. Trojan from writing to lower class

No Write Down

Mission Plan

Cleaning Supplies

Classification

TOP SECRET
SECRET
CONFIDENTIAL
OPEN

Security Levels

Clearance

No read Up

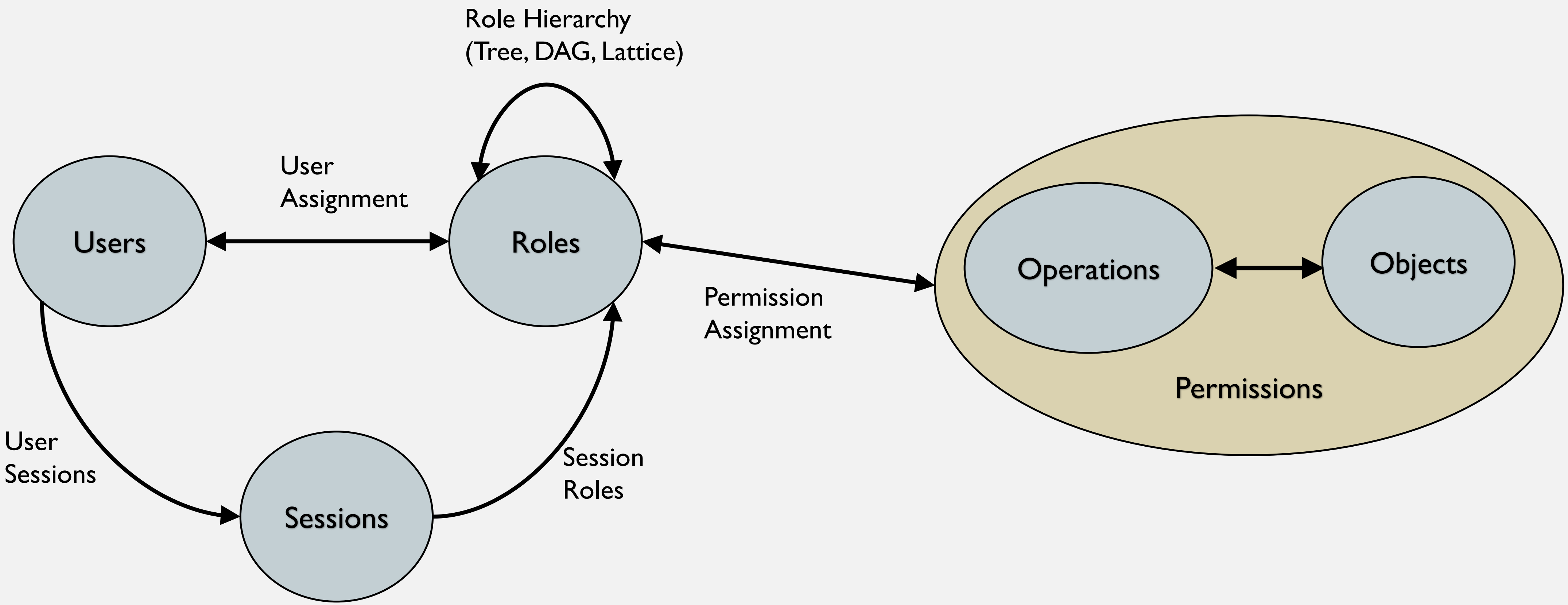
Add to Mission Plan: Stop for sponges

BIBA: Dual of Bell-LaPadula for integrity

Credits : Dr. Jerry den Hartog

ROLE-BASED ACCESS CONTROL (RBAC) MODEL

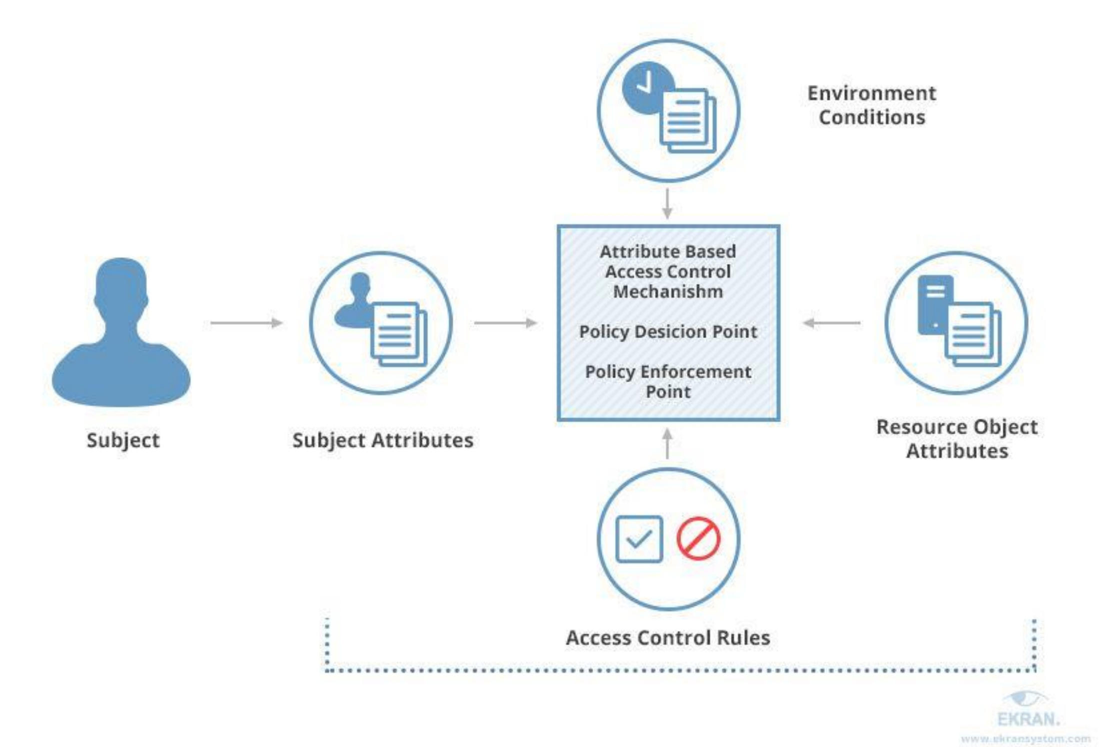
- Enjoys rich literature and a standard
- Naturally fits to the real world, i.e. roles → job functions …



1: Remake of http://www.si-g.com/HTML/2004-10-21_RBAC_at_Mazda.htm

ATTRIBUTE-BASED ACCESS CONTROL (ABAC) MODEL

- Attributes can describe users, resources, actions and the environment/s in which the action happens, e.g.
  - Identity, affiliation, role, clearance are all attributes of a user/**subject**
  - **resource** id, category are of a resource,
  - **actions** are operations to be performed
  - time-of-day is of an **environment**
- Express permission based on Attributes
  - Can be used to capture other models
- Instances: XACML, PTACL [5]



Ref: https://www.ekransystem.com/en/blog/rbac-vs-abac

RECALL THE EXAMPLE

- Role-based Access Control (RBAC)
  - A **Lawyer** can read/write {…}
  - An **Investigator** can read/write {…}
  - **Alicia** is a lawyer, **Will** is a lawyer
  - **Kalinda** is an Investigator

- Attribute-based Access Control (ABAC): Rules are specified by using attributes
  - Alicia.**role** = Lawyer
  - DataSetA.**COI** = ClassA

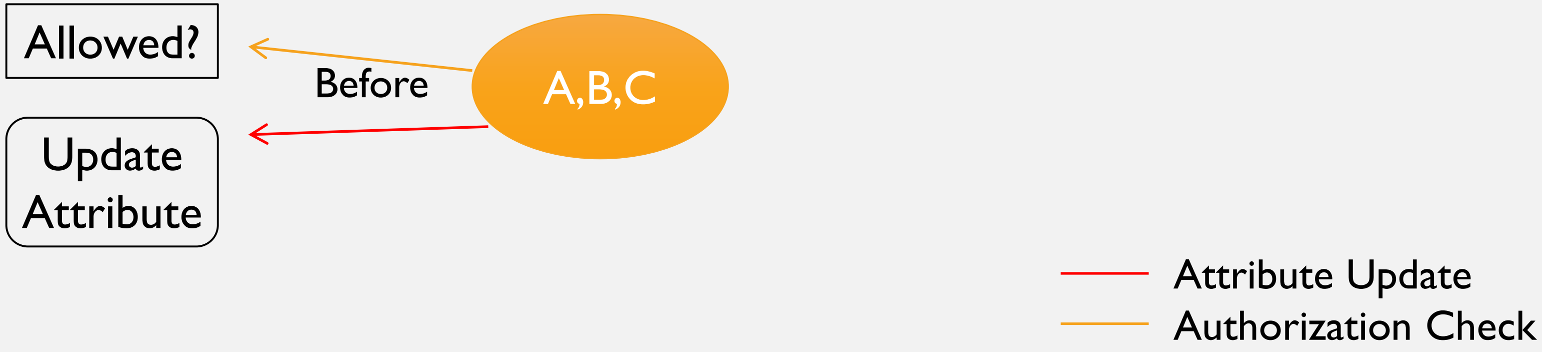
USAGE CONTROL (UCON)

- Attribute-based Access control with
  - Continuity of Enforcement
    - Skype call : Check credit continuously (every minute, second)
    - Youtube : Show an ad of 12 seconds at every new video
    - Digital Rights Management (DRM)
  - Mutability of Attributes
    - Modify the credit information after one minute of call
    - Attribute modifications may lead to permission revocation
  - Obligations
    - Make sure 12 seconds (video watching) passed

See [4] for more details

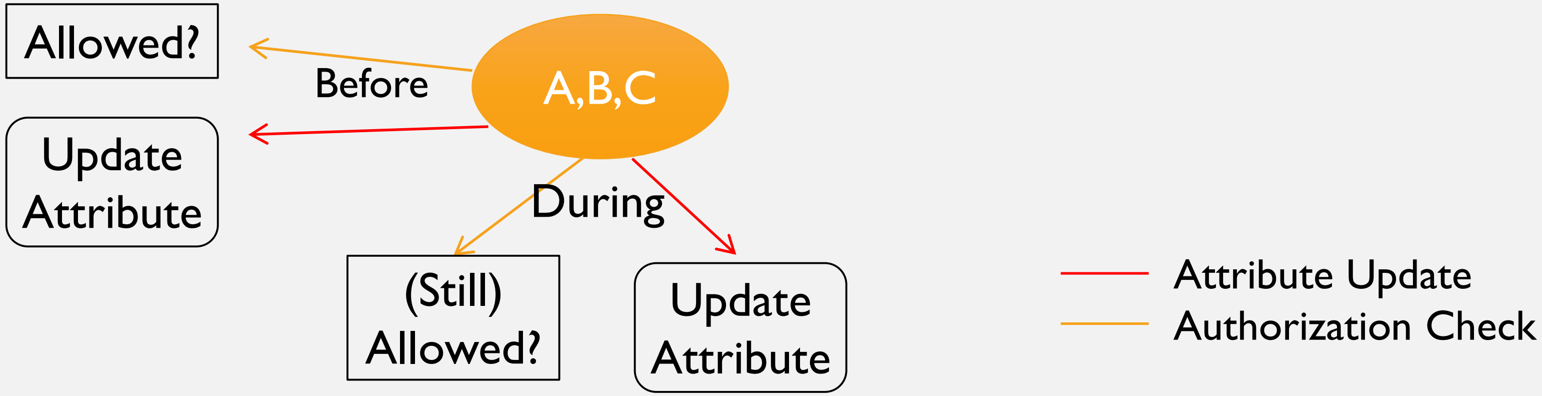
USAGE CONTROL

- Most-known usage control model: UCON_{ABC}
- Core Components: *Subjects*, *Objects*, *Their Attributes*, *Rights*, ***Authorizations***, ***Obligations***, ***Conditions***
  - **A**uthorizations: credit(alice) > cost(call, US)
  - **O**bligations: watchedAdvertisement(alice) → {true, false}
  - **C**onditions: time < 18:00



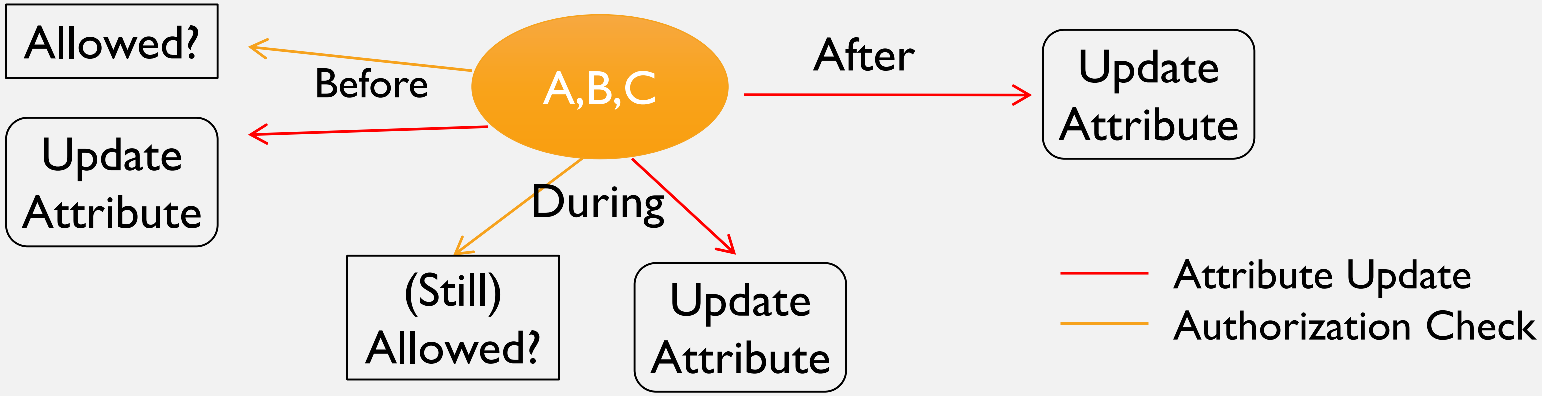
USAGE CONTROL

- Most-known usage control model: UCON_{ABC}
- Core Components: *Subjects*, *Objects*, *Their Attributes*, *Rights*, *Authorizations*, *Obligations*, *Conditions*
  - **A**uthorizations: credit(alice) > cost(call, US)
  - O**b**ligations: watchedAdvertisement(alice) → {true, false}
  - **C**onditions: time < 18:00



USAGE CONTROL

- Most-known usage control model: UCON_{ABC}
- Core Components: *Subjects*, *Objects*, *Their Attributes*, *Rights*, *Authorizations*, *Obligations*, *Conditions*
  - **A**uthorizations: credit(alice) > cost(call, US)
  - O**b**ligations: watchedAdvertisement(alice) → {true, false}
  - **C**onditions: time < 18:00

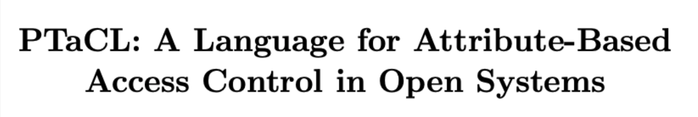


USAGE CONTROL

- Highlights
  - Next generation access control with continuity of enforcement, mutability of attributes, obligations
  - Extensive (can model many requirements)
  - Better suited for dealing with copyright infringement and managing consumable rights, e.g. Digital Rights Management (DRM)
- Complications
  - Lack of specification languages for policies
  - Enforcement (continuously) requires fine tuning

LANGUAGES

- The policies are specified by "using a language"
  - ***XACML* (Model: ABAC, Syntax: XML-based) and its extensions**
    - *RT* (Model: RBAC, Syntax: Logic-based)
    - *Rei* , *Ponder2, SecPAL, AIR*, …



PTaCL: A Language for Attribute-Based Access Control in Open Systems

XACML

PrimeLife

S4P

SecPAL for Privacy

EPAL

means

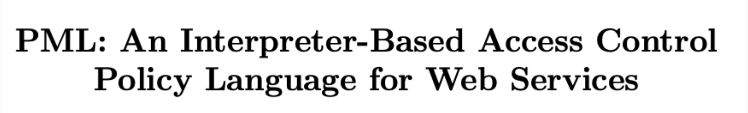
Enterprise Privacy Authorization Language

AIR Policy Language

SPL: An access control language

Rule Based Enforcement - Rei

JACPoL: A Simple but Expressive JSON-based Access Control Policy Language



PML: An Interpreter-Based Access Control Policy Language for Web Services

The Jeeves Language

XACML

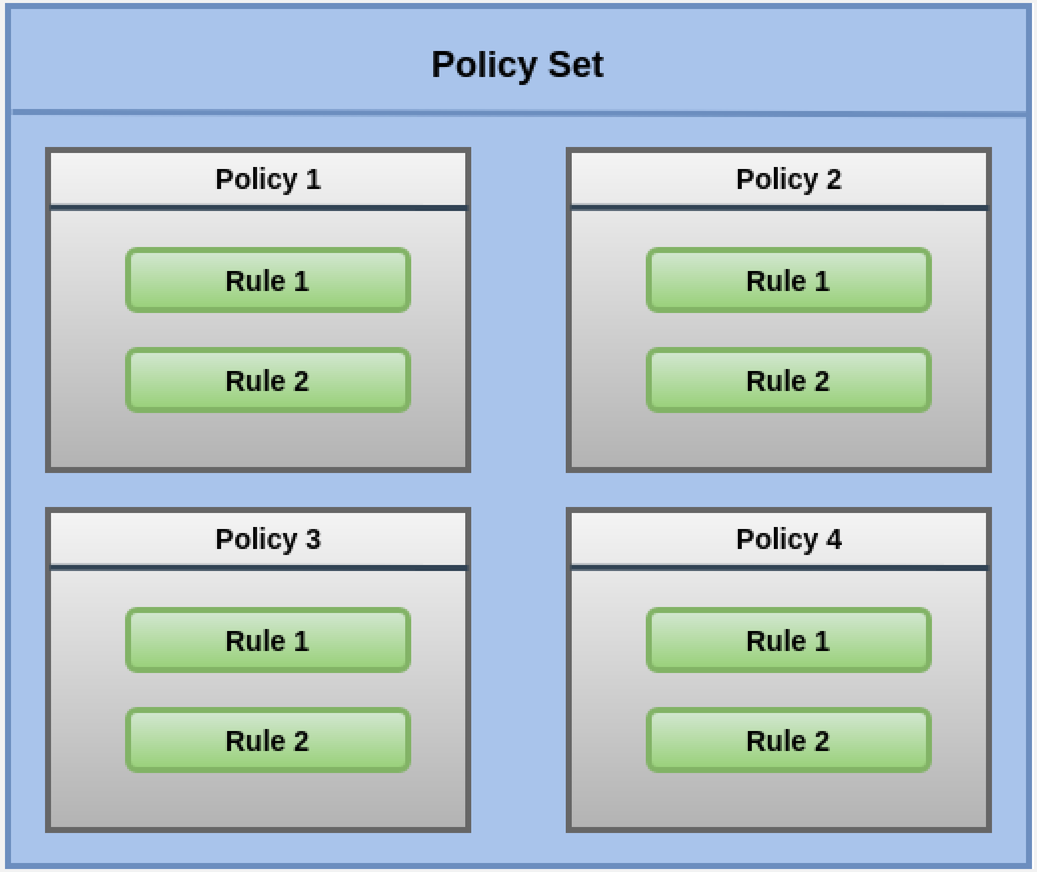
- e**X**tensible **A**ccess **C**ontrol **M**arkup **L**anguage (**XACML**) [6]

- (OASIS) Standard for specifying and enforcing ABAC policies
  - **15 data types**
  - **~250 functions**

- XML-based syntax with many profiles for RBAC, REST, JSON etc.

- Defines an enforcement architecture

XACML



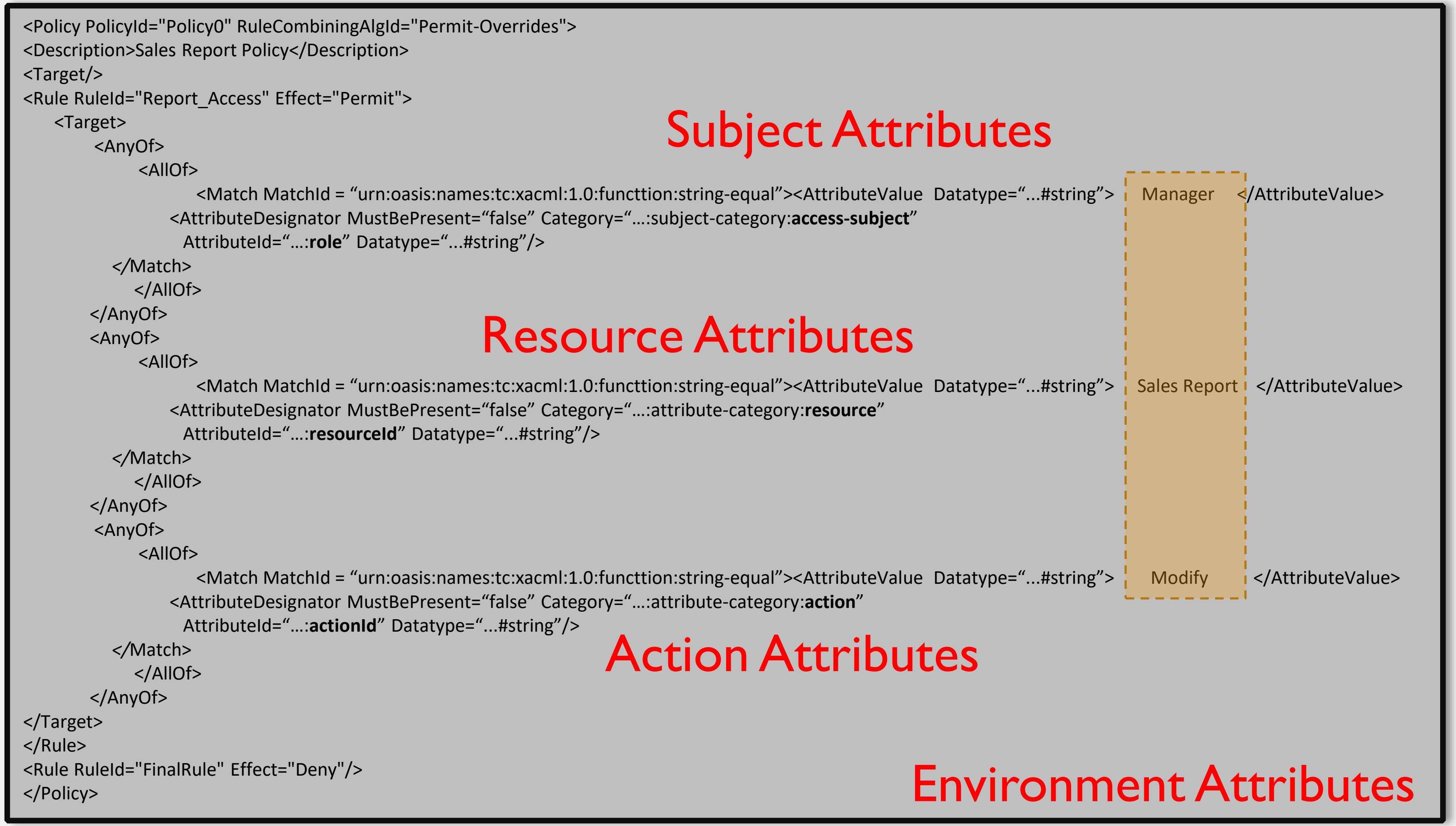
Top-level Policy elements. [10]

XACML

```
<Policy PolicyId="Policy0" RuleCombiningAlgId="Permit-Overrides">
<Description>Sales Report Policy</Description>
<Target/>
<Rule RuleId="Report_Access" Effect="Permit">
    <Target>
         <AnyOf>
              <AllOf>
                     <Match MatchId = “urn:oasis:names:tc:xacml:1.0:functtion:string-equal”><AttributeValue  Datatype=“…#string”>   Manager   </AttributeValue>
                  <AttributeDesignator MustBePresent=“false” Category=“…:subject-category:access-subject”
                    AttributeId=“…:role” Datatype=“…#string”/>
           </Match>
              </AllOf>
        </AnyOf>
        <AnyOf>
              <AllOf>
                     <Match MatchId = “urn:oasis:names:tc:xacml:1.0:functtion:string-equal”><AttributeValue  Datatype=“…#string”>  Sales Report  </AttributeValue>
                  <AttributeDesignator MustBePresent=“false” Category=“…:attribute-category:resource”
                    AttributeId=“…:resourceId” Datatype=“…#string”/>
           </Match>
              </AllOf>
        </AnyOf>
         <AnyOf>
              <AllOf>
                     <Match MatchId = “urn:oasis:names:tc:xacml:1.0:functtion:string-equal”><AttributeValue  Datatype=“…#string”>   Modify   </AttributeValue>
                  <AttributeDesignator MustBePresent=“false” Category=“…:attribute-category:action”
                    AttributeId=“…:actionId” Datatype=“…#string”/>
           </Match>
              </AllOf>
        </AnyOf>
</Target>
</Rule>
<Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

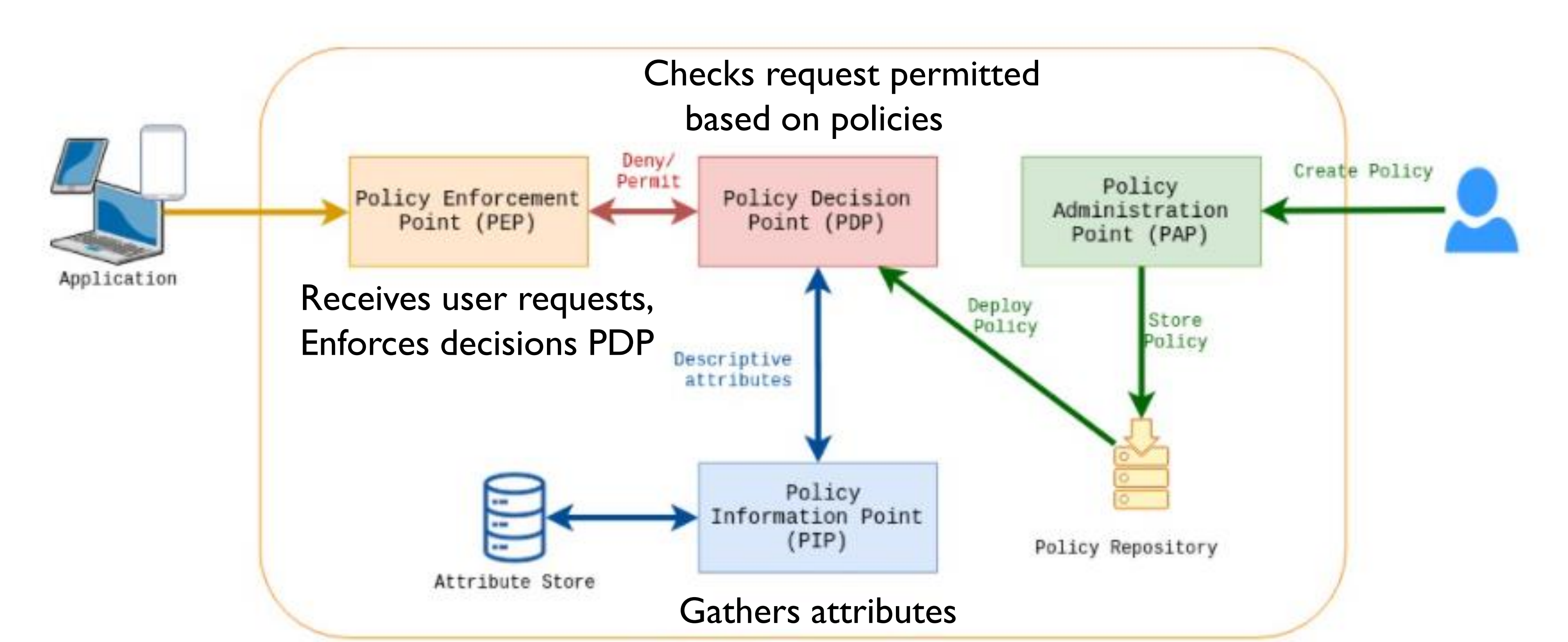
XACML

<Target> element specifies the set of requests to which it applies.



```
<Policy PolicyId="Policy0" RuleCombiningAlgId="Permit-Overrides">
<Description>Sales Report Policy</Description>
<Target/>
<Rule RuleId="Report_Access" Effect="Permit">
   <Target>
       <AnyOf>
           <AllOf>
                <Match MatchId = “urn:oasis:names:tc:xacml:1.0:functtion:string-equal”><AttributeValue  Datatype=“…#string”>  Manager  </AttributeValue>
             <AttributeDesignator MustBePresent=“false” Category=“…:subject-category:access-subject”
              AttributeId=“…:role” Datatype=“…#string”/>
         </Match>
           </AllOf>
      </AnyOf>
      <AnyOf>
           <AllOf>
                <Match MatchId = “urn:oasis:names:tc:xacml:1.0:functtion:string-equal”><AttributeValue  Datatype=“…#string”>  Sales Report  </AttributeValue>
             <AttributeDesignator MustBePresent=“false” Category=“…:attribute-category:resource”
              AttributeId=“…:resourceId” Datatype=“…#string”/>
         </Match>
           </AllOf>
      </AnyOf>
       <AnyOf>
           <AllOf>
                <Match MatchId = “urn:oasis:names:tc:xacml:1.0:functtion:string-equal”><AttributeValue  Datatype=“…#string”>  Modify  </AttributeValue>
             <AttributeDesignator MustBePresent=“false” Category=“…:attribute-category:action”
              AttributeId=“…:actionId” Datatype=“…#string”/>
         </Match>
           </AllOf>
      </AnyOf>
</Target>
</Rule>
<Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

XACML


Application
Policy Enforcement Point (PEP)
Deny/ Permit
Policy Decision Point (PDP)
Policy Administration Point (PAP)
Create Policy
Deploy Policy
Store Policy
Descriptive attributes
Policy Information Point (PIP)
Attribute Store
Policy Repository

Checks request permitted based on policies

Receives user requests, Enforces decisions PDP

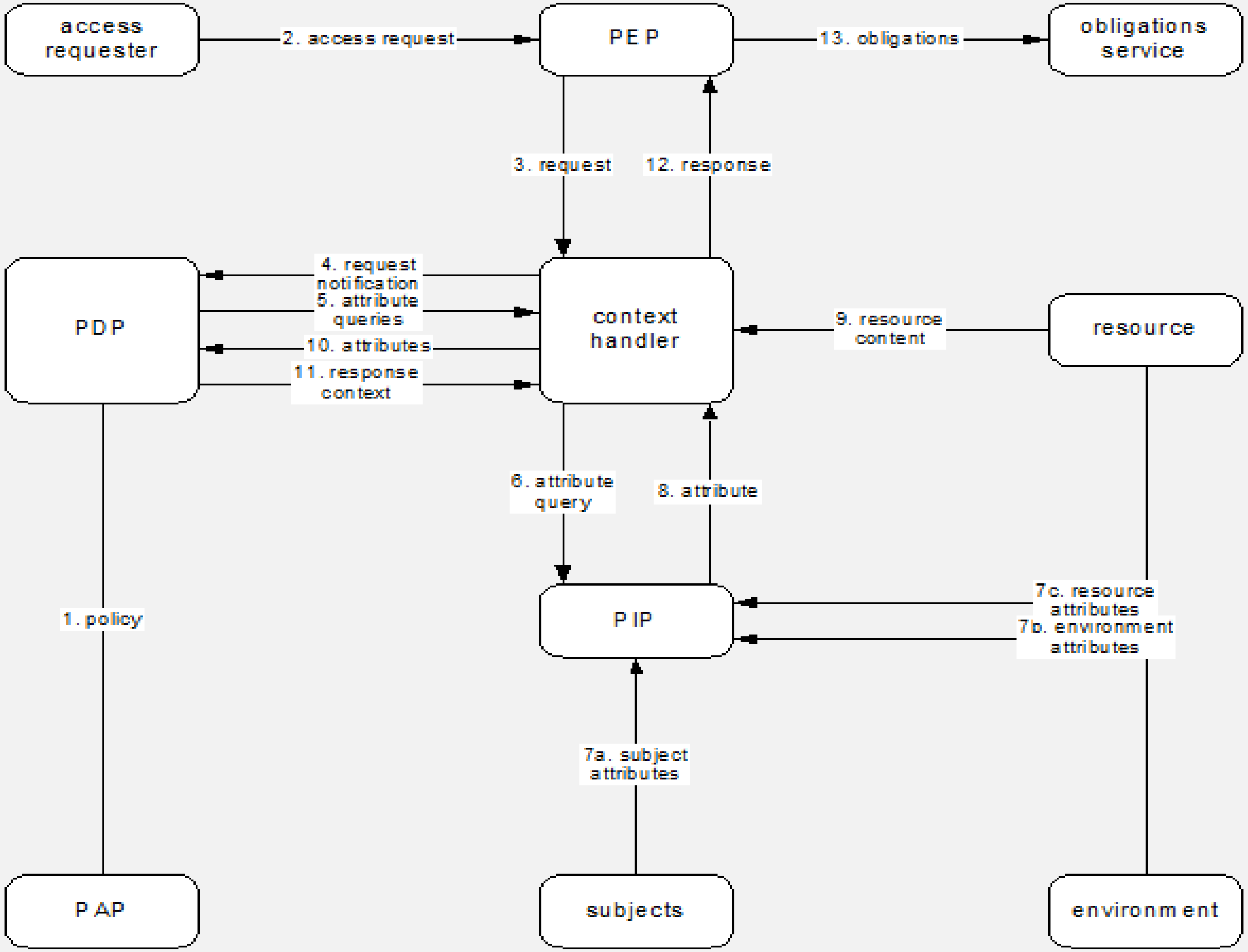
Gathers attributes

XACML Architecture [10]

Context Handler: Responsible for conversions between XACML canonical format and native formats

ENFORCEMENT ARCHITECTURE

Policies made available to PDP are evaluated against the access requests

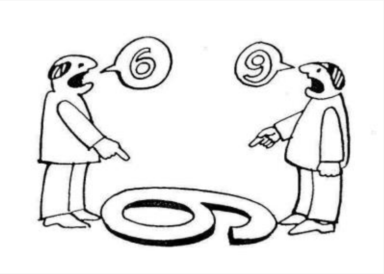


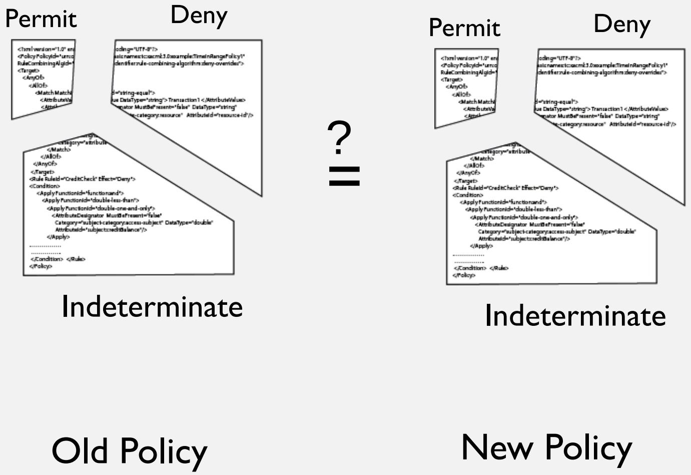
Ref: Data Flow in XACML, courtesy of XACML v3 Spec.

POLICY ANALYSIS

Automated Tools for assisting security administrators

- Are there conflicts? Permit or Deny
- Alice is permitted to print?
- My updated policy preserve the old authorizations?
- What are the differences between my old and new policies?





Permit Deny

Indeterminate

Old Policy

$\overset{?}{=}$

Permit Deny

Indeterminate

New Policy

See [7] for conflict resolutions

POLICY ANALYSIS

- Granularity in the analysis is important!
- Below is an example XACML policy for managing transactions in a bank's system.

P[dov] :	resource-type = "transaction" ∧
	action-id = "create"

r1[Deny] :	t.value + t.cost > u.credit
r2[Deny]:	current-day ∉ {Mo,Tu,We,Th,Fri} ∨ current-time < 08:00 ∨ current-time > 18:00
r3[Permit]:	true

Most available tools

- "value + cost > credit" is a single boolean variable **V**
- You get ***V* = *true*** or ***V* = *false***

No more information…

Recent Work

Satisfiability Modulo Theories (SMT) allows reasoning at a more granular level;
value = 50 euro, cost = 5 euro,
credit = 100 euro

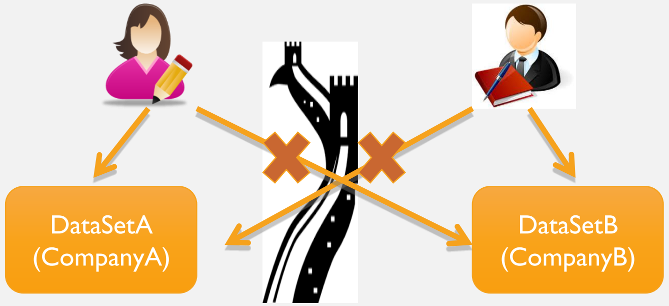
POLICY ANALYSIS

- Formal encoding of the policy
- Different properties can be checked
  - Policy refinement
  - Policy subsumption Change-impact analysis : Find differences between old and new policy
  - Attribute Hiding attacks: Hiding an attribute leads to a more favorable authorization decision?
    - Partial: Hide an attribute/value pair (att = value)
    - General: Hide an attribute as a whole (att)
  - Scenario finding: Give me a proof/request for a given scenario
  - ….

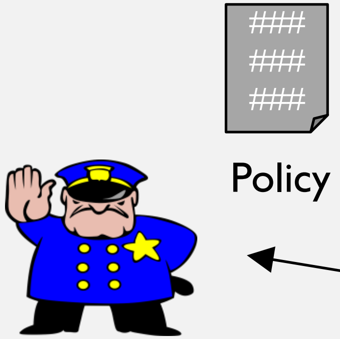
CONSTRAINTS AND MECHANISMS

- Chinese Wall

Recall: CompanyA and CompanyB



- Separation of Duty
- many others…



Enforcement

Mechanisms :

- Mutually exclusive roles
- Assignment of attributes
- …

Security Principles:

- Least Privilege
- Complete Mediation
- …

FEDERATED AUTH(N) AND AUTH(Z)

- Also known as single sign-on

- Decentralized setting
  - No central authority
  - Common entities are *Client*, *Service Provider (SP)*, *Identity Provider (IdP)*
  - There is a pre-established trust between SP and IdP

- Federations of organizations, e.g. companies, universities

- Tokens are frequently used!!

- Examples:
  - Eduroam
  - Microsoft
  - Google
  - Facebook
  - Linkedin
  - Github
  - …

FEDERATED AUTH(N) AND AUTH(Z)

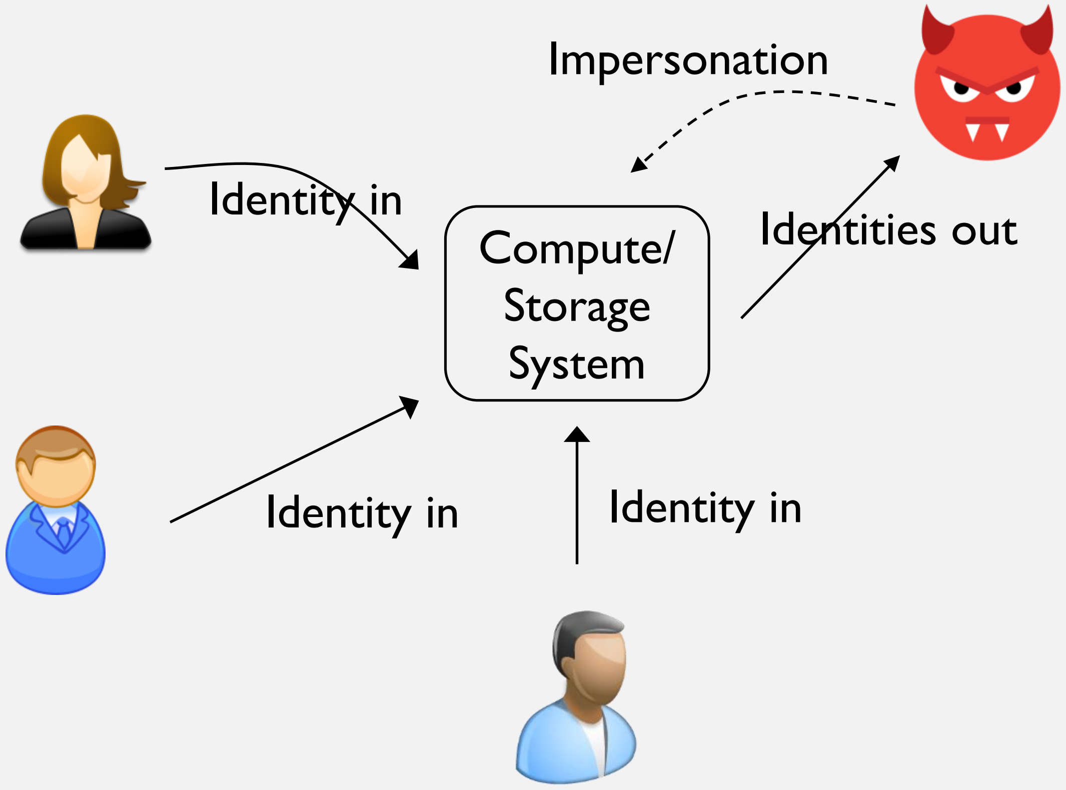
General Steps

- Client/user requests access to a resource or service
- Service Provider replies with a list of trusted IDPs
- Client authenticates (to the trusted IDP) and obtains an Authentication assertion (AuthNAssert) or a Secure Token (ST);
- Client presents AuthN Assert or ST/cookie to the Service Provider's Authorization service that validates presented credentials and evaluates the request against the access control policy.
  - Decision Permit or Deny
  - Authorisation service may issue an Authorisation assertion (AuthzAssertion)
- Client presents Authz Assertion to the Resource and gets access to it.

WHY TOKENS?

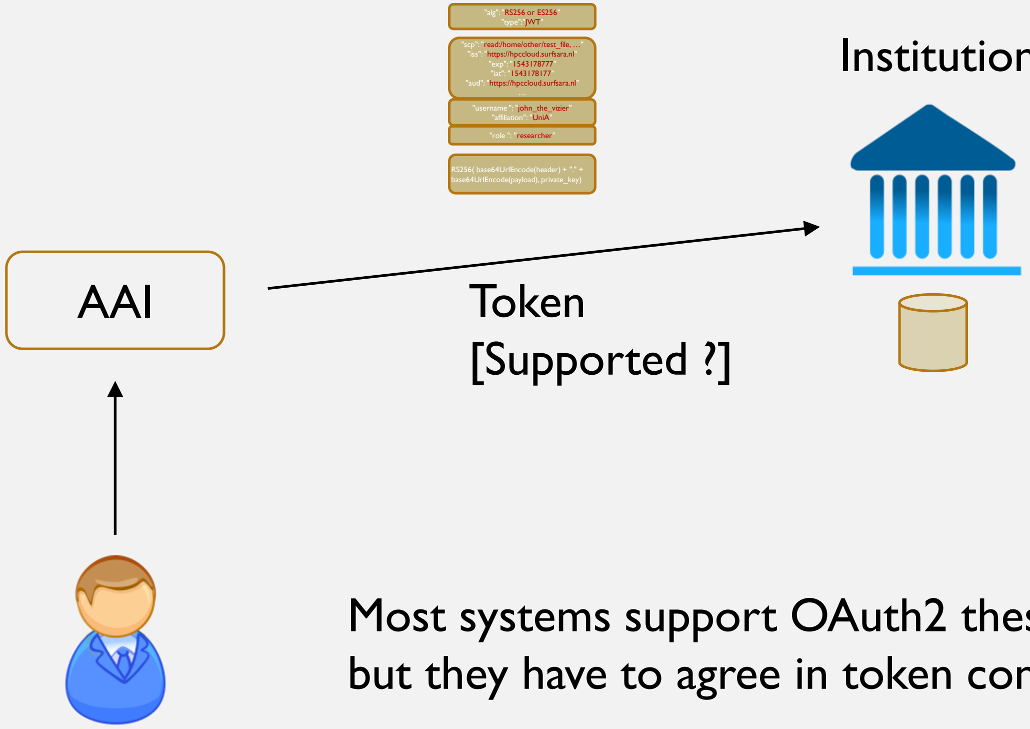
SCENARIO : PRIVACY

Cases, e.g. open systems, where privacy is important



SCENARIO : INTEGRATION

Integration with internal/external systems



```
"alg": "RS256 or ES256"
"type":"JWT"

"scp": "read:/home/other/test_file, …"
"iss": "https://hpccloud.surfsara.nl"
"exp": "1543178777"
"iat": "1543178177"
"aud": "https://hpccloud.surfsara.nl"
…
"username ": "john_the_vizier"
"affiliation": "UniA"
"role ": "researcher"

RS256( base64UrlEncode(header) + "." +
base64UrlEncode(payload), private_key)
```

Most systems support OAuth2 these days but they have to agree in token content.

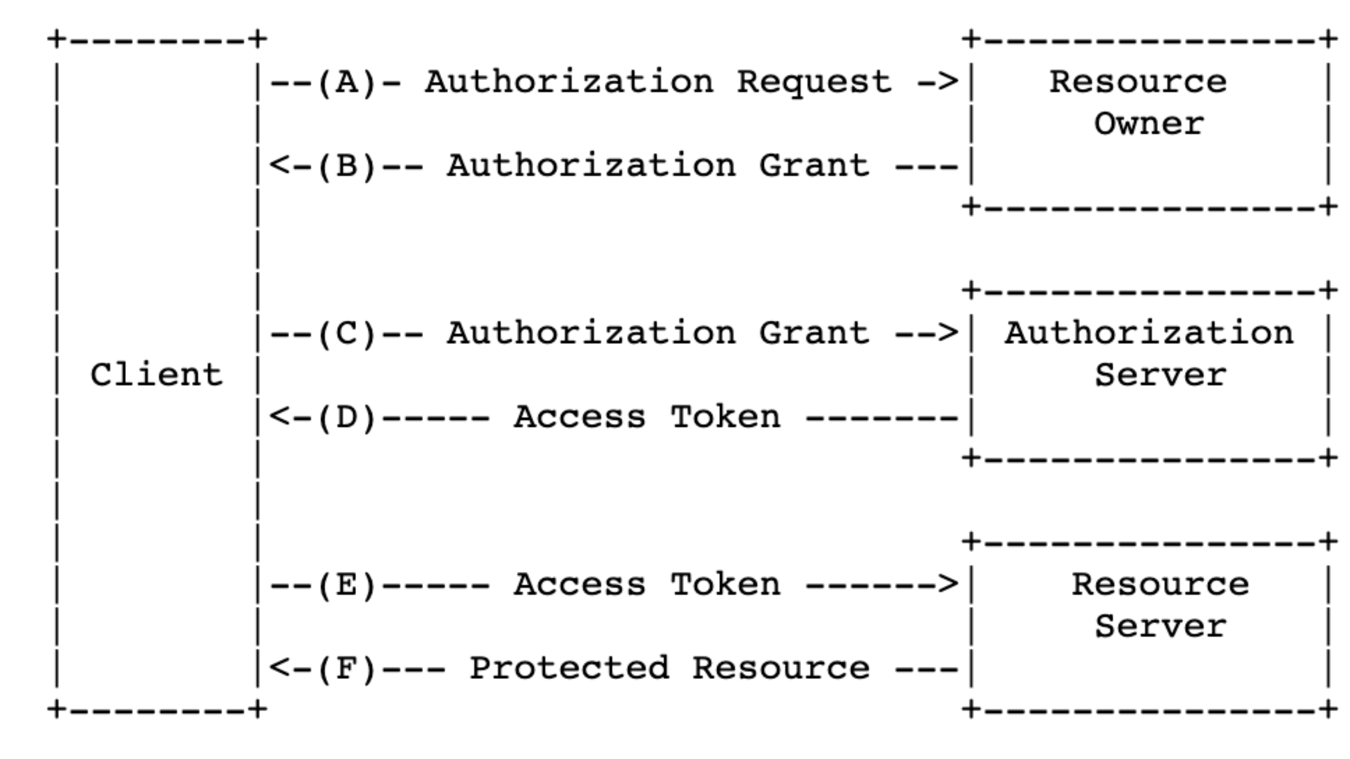
OAUTH2

Four roles:

- *Client → needs to register with AuthZ server*
- *Resource Owner*
- *Authorization Server*
- *Resource Server*

Authorization Grant : Required to obtain an **access token**. Four types:

- Authorization Code
- Implicit (get token directly)
- Resource Owner Password Credentials
- Client Credentials



```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Ref: OAuth2 Protocol, courtesy of OAuth2 Spec.

OAUTH2 - EXAMPLE

Scenario:

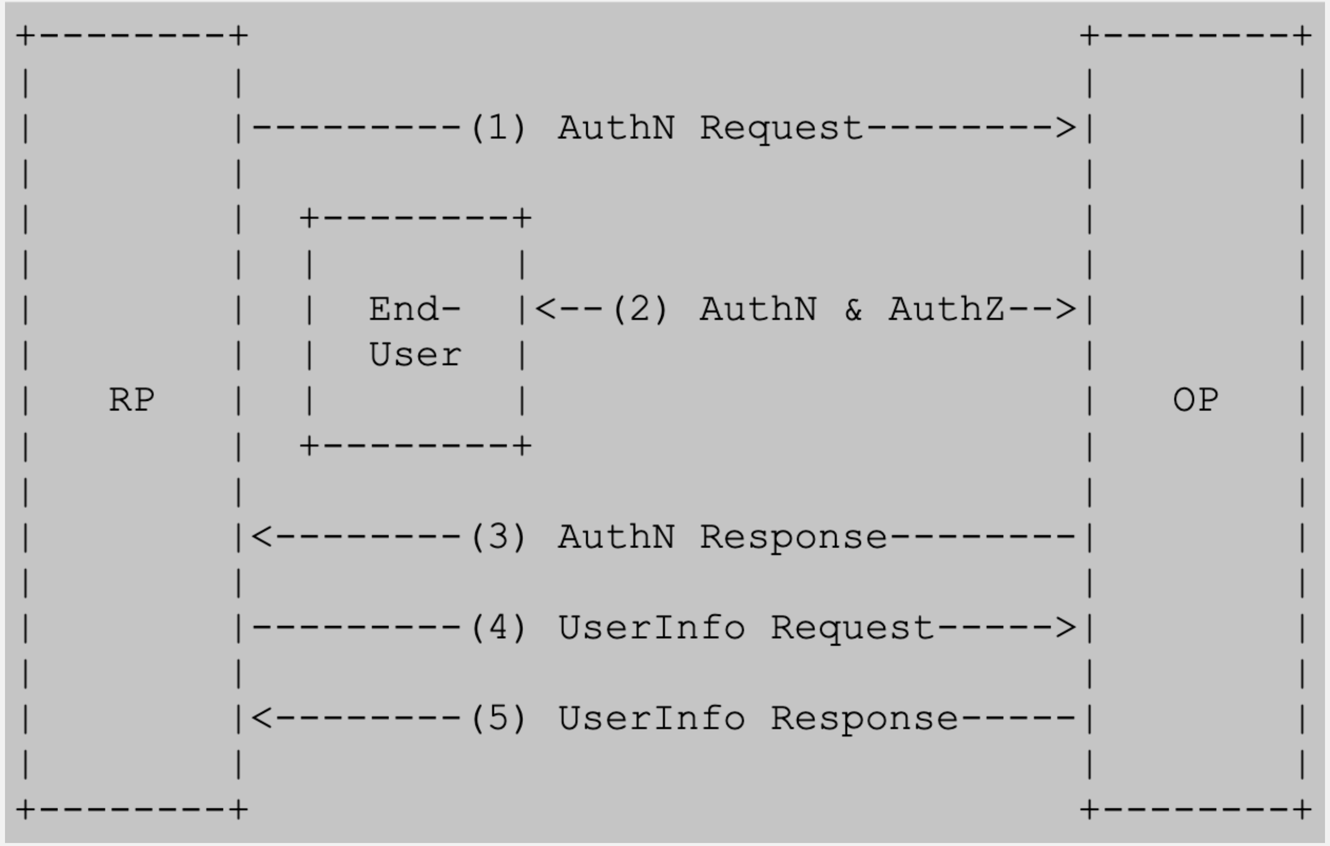
- A website wants to obtain information about your Google profile.
- You are redirected by the client (the website) to the authorization server (Google).
- If you authorize access, the authorization server sends an authorization code to the client (the website) in the callback response.
- Then, this code is exchanged against an access token between the client and the authorization server.
- The website is now able to use this access token to query the resource server (Google again) and retrieve your profile data.

OPENID CONNECT

- Authentication layer on top of OAuth2

- Besides OAuth2 roles
  - Relying Party
  - OpenID Provider
  - End-user

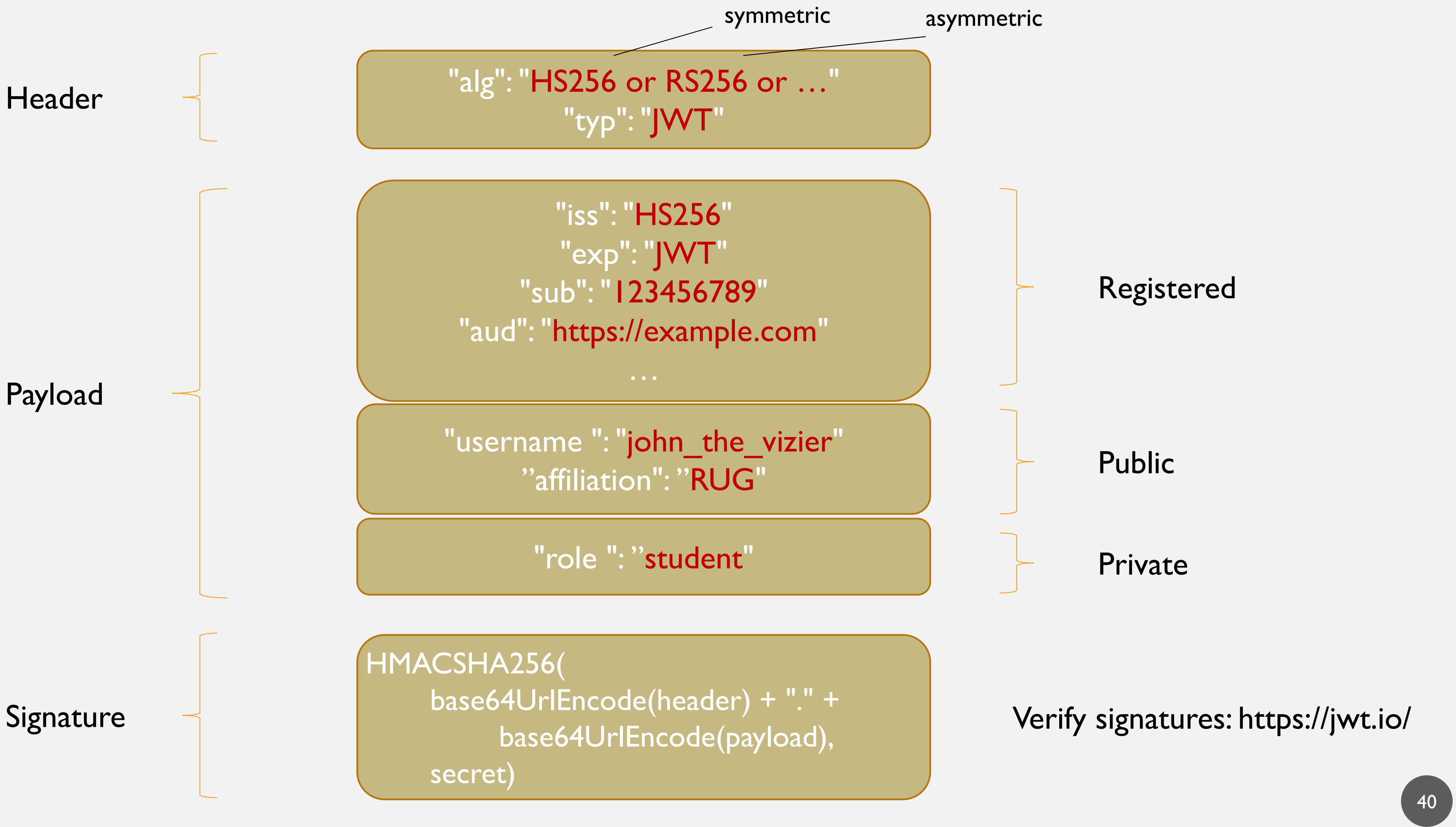
- Three authentication flows:
  - Authorization Code Flow
  - Implicit Flow
  - Hybrid Flow



```
+--------+                                   +--------+
|        |                                   |        |
|        |---------(1) AuthN Request-------->|        |
|        |                                   |        |
|        |  +--------+                       |        |
|        |  |        |                       |        |
|        |  |  End-  |<--(2) AuthN & AuthZ-->|        |
|        |  |  User  |                       |        |
|   RP   |  |        |                       |   OP   |
|        |  +--------+                       |        |
|        |                                   |        |
|        |<--------(3) AuthN Response--------|        |
|        |                                   |        |
|        |---------(4) UserInfo Request----->|        |
|        |                                   |        |
|        |<--------(5) UserInfo Response-----|        |
|        |                                   |        |
+--------+                                   +--------+
```

Clients includes the openid scope value in the Oauth Authorization Request. Information about the authentication performed is returned in a **JSON Web Token (JWT)** [JWT] called an ID Token

JSON WEB TOKEN



symmetric

asymmetric

Header

"alg": "HS256 or RS256 or …"
"typ": "JWT"

Payload

"iss": "HS256"
"exp": "JWT"
"sub": "123456789"
"aud": "https://example.com"
…

Registered

"username ": "john_the_vizier"
”affiliation": ”RUG"

Public

"role ": ”student"

Private

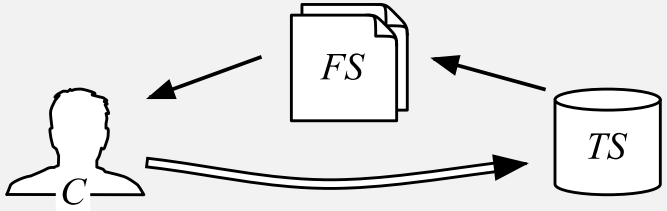
Signature

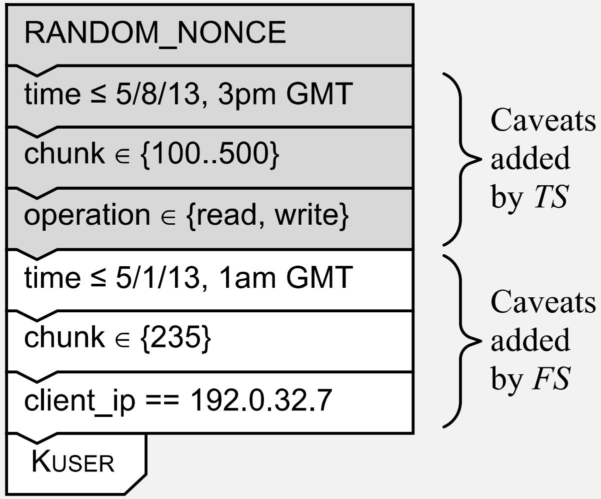
```
HMACSHA256(
        base64UrlEncode(header) + "." +
                base64UrlEncode(payload),
        secret)
```

Verify signatures: https://jwt.io/

MACAROONS

- Flexible authorization tokens, i.e. **bearer tokens** (the client presents the macaroon along with the request.)
- Google originated open source
- Features:
  - **Delegation** with attenuation and third-party restrictions/caveats
  - **Attenuation** on how, when and where the tokens can be exercised
  - **Proof-carrying**: The caveats/restrictions are constructed using chained HMAC
  - **Third-party caveats**: Restrictions/caveats can be enforced by third parties.





RANDOM_NONCE

time ≤ 5/8/13, 3pm GMT

chunk ∈ {100..500}

operation ∈ {read, write}

time ≤ 5/1/13, 1am GMT

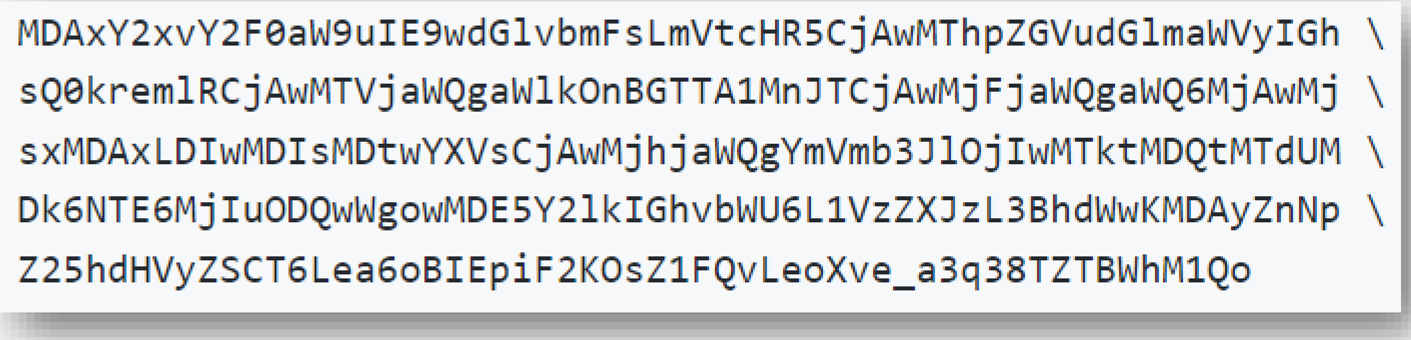
chunk ∈ {235}

client_ip == 192.0.32.7

KUSER

Caveats added by *TS*

Caveats added by *FS*



```
MDAxY2xvY2F0aW9uIE9wdGlvbmFsLmVtcHR5CjAwMThpZGVudGlmaWVyIGh \
sQ0kremlRCjAwMTVjaWQgaWlkOnBGTTA1MnJTCjAwMjFjaWQgaWQ6MjAwMj \
sxMDAxLDIwMDIsMDtwYXVsCjAwMjhjaWQgYmVmb3JlOjIwMTktMDQtMTdUM \
Dk6NTE6MjIuODQwWgowMDE5Y2lkIGhvbWU6L1VzZXJzL3BhdWwKMDAyZnNp \
Z25hdHVyZSCT6Lea6oBIEpiF2KOsZ1FQvLeoXve_a3q38TZTBWhM1Qo
```

A macaroon contains a location and an identifier and a list of caveats (key, value).

REFERENCES

[1] Lampson, Butler W. (1971). "Protection". Proceedings of the 5th Princeton Conference on Information Sciences and Systems. p. 437.

[2] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in Operating Systems. Communications of the ACM 19(8): 461-471. 1976

[3] Matt Bishop, Computer Security, Art and Science, Second Edition, Pearson

[4] Jaehong Park, Ravi S. Sandhu: The UCONABC usage control model. ACM Trans. Inf. Syst. Secur. 7(1): 128-174 (2004)

[5] Jason Crampton, Charles Morisset: PTaCL: A Language for Attribute-Based Access Control in Open Systems. POST 2012: 390-409

[6] eXtensible Access Control Markup Language (XACML) Version 3.0 OASIS Standard (Link : http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html)

REFERENCES

[7] Ninghui Li, Qihua Wang, Wahbeh Qardaji, Elisa Bertino, Prathima Rao, Jorge Lobo, and Dan Lin, "Access control policy combining: theory meets practice", In Proceedings of the 14th ACM symposium on Access control models and technologies (SACMAT '09), pages 135-144, ACM, 2009.

[8] Sun XACML Implementation (Link : http://sourceforge.net/projects/sunxacml/)

[9] UMU-XACML-Editor (Link : http://sourceforge.net/projects/umu-xacmleditor/)

[10] A Basic Introduction to XACML (link: https://dzone.com/articles/a-beginners-guide-to-xacml)