Seminar 4: SECURITY 1

INFO-H-511 : Web Services

V. Baele, M. Elhouderi, P. Dagnely

Université libre de Bruxelles

Thusday, April 18

Introduction

Internet :

1. public network
2. insecure channel
3. we have to care about web services for security

Introduction

Web services create new security problems :

1. Accessibility adds more risks
2. WSDL and SOAP structure messages are known
3. UDDI listing service

Introduction

It exists two family of level security

1. network level-security
   - Firewall
   - IDS
   - Symmetric and Asymmetric encryption
   - Digital certificates and signatures
   - ...
2. application level-security
   - SSL
   - IPsec
   - ...

Introduction

So we should be aware of :

- who the requesters are
- which informations are being requested
- which specifics services are being requested
- ...

Definition

Authentication is the mechanism by which clients and service providers prove to one another that tey are acting on behalf of specifics users or systems.
[Monzillo, 2002][Singh, 2004]

Mutual authentication

When the authentication is bi-directional.

We can use several techniques :

- user name and password
- encrypted communications
- digital signatures
- ...

Definition

Authorization mechanisms allow only authentic caller identities to access resources, such as hosts, files, web pages, ...

Typical autorization policies permit access to different resources for distinct collections of authenticated clients on the basis of roles, groups or privileges. It's often parametrised with the **permissions**.

Definition

Integrity comprise two requirements :

1. The data received must be the same as the data send (**not modified**)
2. Who modified the data (**authenticity**)
3. At any time in the future, it's possible to prove whether different copies of the same documents are in fact identical

We can use several techniques :

- digital signature
  see XML signature below.
- hash algorithms
- ...

Definition

Confidentiality is the ability to ensure that messages and data are available only to those who are authorized to view them

We can use several techniques :

- encrypted communication
  see XML encryption below.
- hash algorithms
- ...

Infrastructure placed between networks

- logicaly separation
- IP-based access control
- protect privacy and integrity
- first line of defense

Firewall examines all messages - traffic coming into and leaving

- block non-authorized messages
- indetification by :
  - name
  - IP (authentication)
  - application
  - ...
- rules programmed into the firewall system
- can block ports

- encrypt and decrypt message
- from original text = plain text
  to coded message = cypher text
- need a key : public or private
- 3 main techniques
  - symmetric
  - asymmetric
  - certificates and signatures

Symmetric : One same key for encryption and decryption

- 1 shared key
- large amount of data
- easily and quickly implemented
- only use it a few time
- BUT every pair of group needs his own key

Asymmetric : One for encryption and one for decryption

- 1 pair of keys
  - 1 public : encrypt
  - 1 private : decrypt
- Bad performance
- Used to send symmetric key

Exchange during the communication

- Certificates : document that uniquely identifies the party
- time period of validity
- Signature for the verification

Not possible on the network layer [OSI Model]

- SSL (Secure Socket Layer)
- IPSec (Layer 3)
- Kerberos

- Authentication
- Authorisation
- Integrity and confidentiality
- Non-repudiation
- Auditing

Based on Public Key + Digital Certificate

- Server Authenticate to client
- Client & Server agrees on ciphering
- Communication channel encrypted

Network level, no application requirement.

- Uses Transport/Tunnel mode
- Authentication Header
- ESP — Encapsulating Security Payload

Security in an insecure environment.

- Ticketing mechanism
- Third Party authentication
- Time dependant (protection from replay/timing attack)

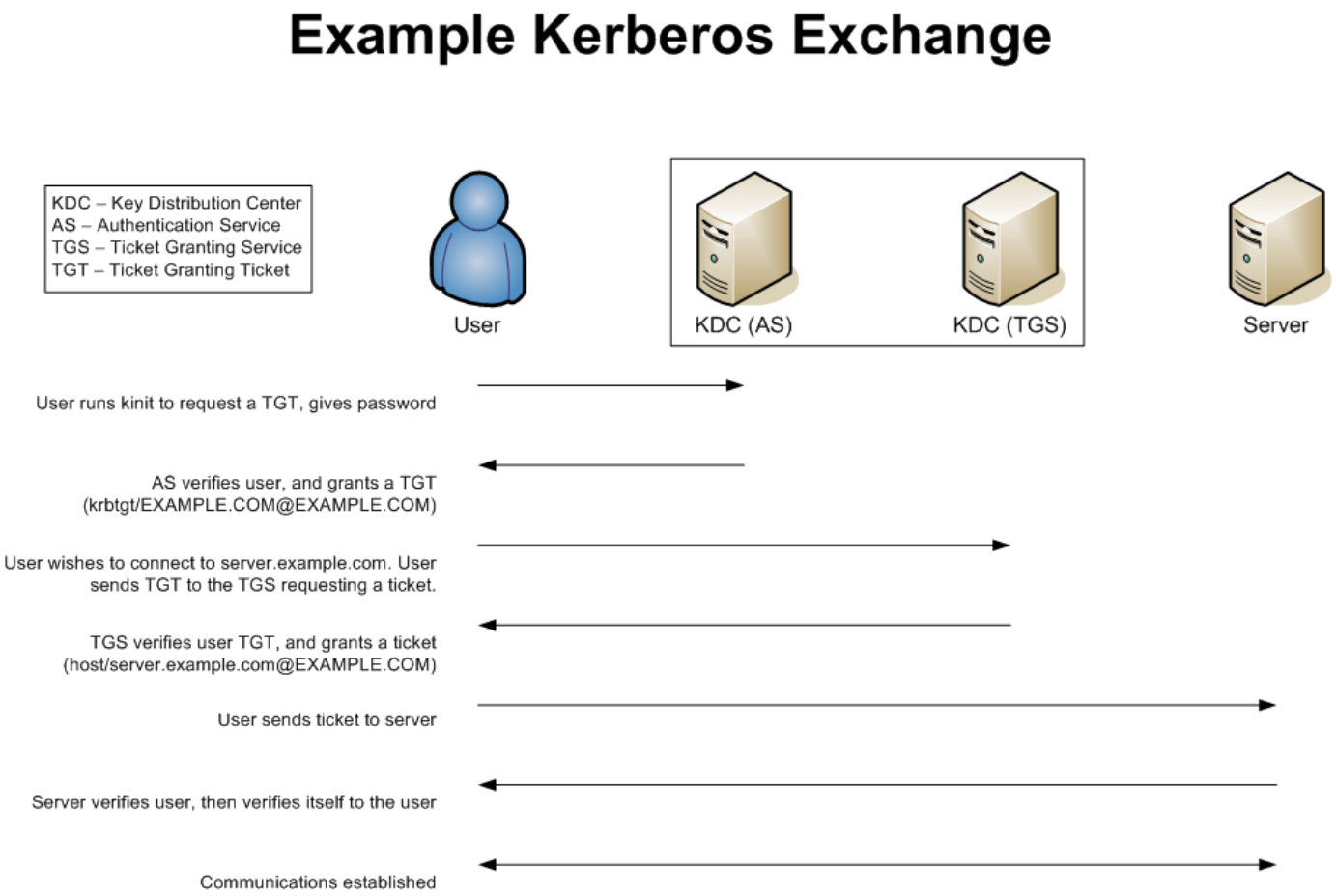


FIGURE: Keroberos Message Exchange

Application HTTP

- Basic authentication
- Form based authentication
- Mutual HTTPS based Authentication

Application Level Security Mechanisms

Spring Security Example

```
1  <http pattern="/css/**" security="none"/>
2  <http pattern="/login.jsp*" security="none"/>
3
4  <http auto-config='true'>
5    <intercept-url pattern="/**" access="ROLE_USER" />
6    <form-login login-page='/login.jsp'/>
7  </http>
```

- Roles
- Security policy
- Container context

- User trusting service provider
- SSL
- Preventing eavesdropping

- Associating action & identity
- Kerberos

- Tracking security events
- Safeguard for incoming attacks
- Log systems are used

Provides high level of security

- Certification Authority
- User registration/authentication
- Generate/Renew/Revoke certificate
- Publishing certificates
- Archive & restoring certificate

Provides high level of security

- Lists PKI accepted & revoked
- Uses LDAP protocol

Security Topologies

Protecting assets

Isolating the valuable resource from network entry point, requires strategic hierarchy often using DMZ (demilitarized zone).

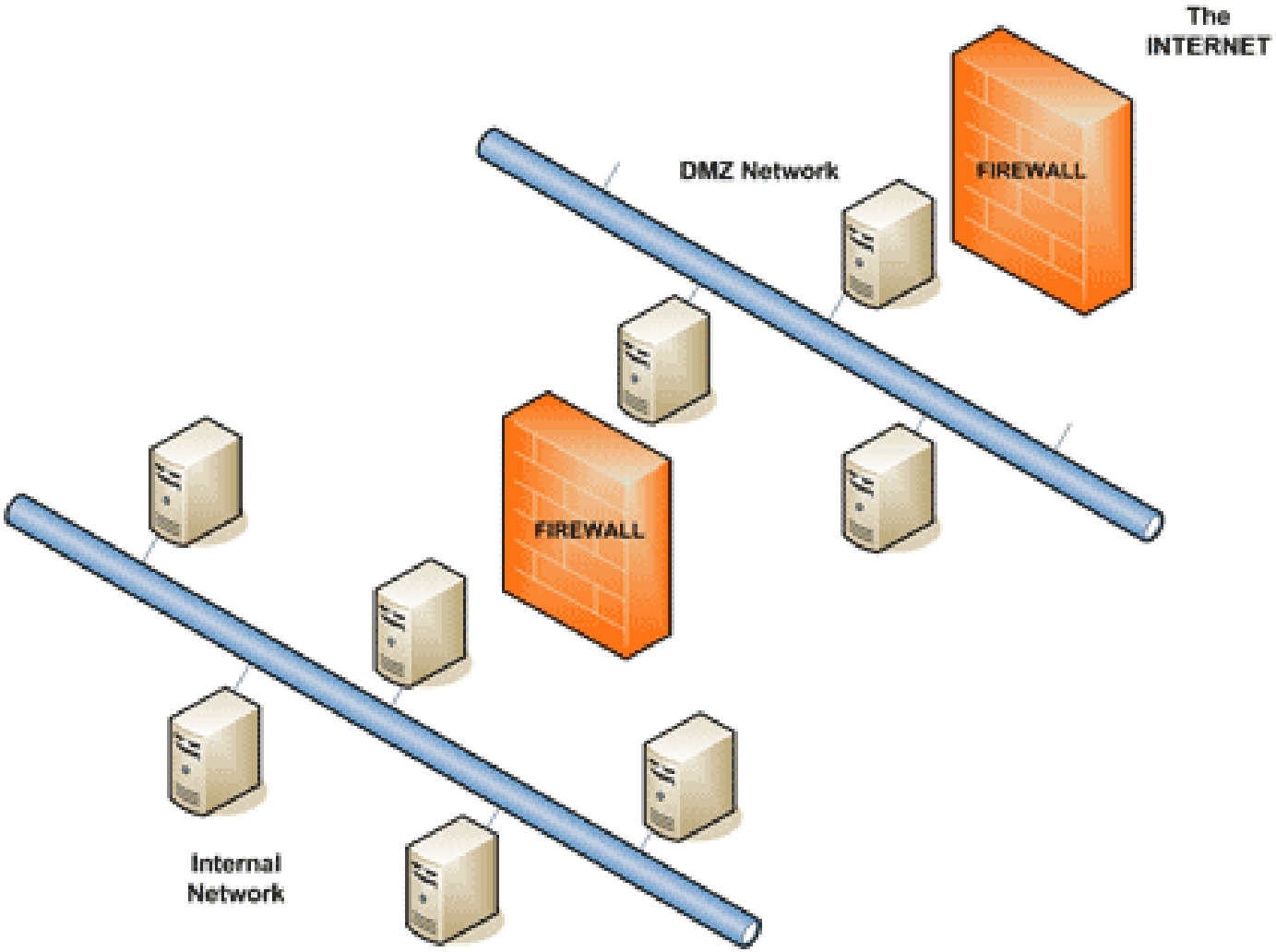

The
INTERNET
FIREWALL
DMZ Network
FIREWALL
Internal
Network

FIGURE: DMZ

XML trust services

Definition

XML trust services is a suite of open XML specifications developed to make it easier to integrate a broad range of XML security services into integrated business applications over the web.

XML trust services

The main technologies of XML trust services are :

- XML signature
  for the authentication of data's
- XML encryption
  for the encryption of data's
- XML key management specification (XKMS)
  for managing key registration and authentication.
- Security assertions markup language (SAML)
  for specifying entitlement and identity.
- XML Access control markup language (XACML)
  for specifying access rights.

XML signature

Definition

The goal of XML signature is to ensure data integrity, message authentication and non repudiation services.

XML signature allow to sign several resources : XML, HTML, Jpeg, ...
XML signature allow to sign only a **part** of a document

XML signature

3 types

- enveloping signatures : the signature envelop the entire document.
- enveloped signature : the signature concern a part of the document.
- detached signature : signature and document are independent (often the document is referenced by an URI).

XML signature

2 things to prove

To have a secure signature we have to keep track of :

1. The identity of the person who sign
2. The identity of the resources signed

XML signature

Structure

```
1  <Signature>
2    <SignedInfo>
3      <CanonicalizationMethod />
4      <SignatureMethod />
5      <Reference>
6          <Transforms>
7          <DigestMethod>
8          <DigestValue>
9      </Reference>
10     <Reference /> etc.
11   </SignedInfo>
12   <SignatureValue />
13   <KeyInfo />
14   <Object />
15 </Signature>
```

XML signature
canonical form

A XML document can have more than one legal serialization representation.

- <Elem> is equal to <Elem >
- different order of the namespace declarations
- use relative or absolute URL
- add white space, line ending, ...
- ...

So the same XML document can have two different digest.

definition

This algorithm guarantee that logically-identical document product exactly identical serialization representation.

In practice it's just a set of instruction wich transform every time the XML file into the same "essence" file
But it's **expensive** and complex.

definition

Contains the list of transformation which have to be applied to the signed resource before the computation of the digest

It exist Five transformations supported by the W3C, but you can create your own.

- Canonicalization
- Enveloped signature transform
- Base64
- Xpath filtering
- XSLT

XML signature

signedInfo : example

```
1  <SignedInfo>
2    <CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
3    <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
4    <Reference URI="config.xml">
5      <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
6      <DigestValue>DEADBEEF</DigestValue>
7    </Reference>
8    <Reference URI="index.html">
9      <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
10     <DigestValue>DEADBEEF</DigestValue>
11   </Reference>
12   <Reference URI="icon.png">
13     <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
14     <DigestValue>DEADBEEF</DigestValue>
15   </Reference>
16   <Reference URI="#prop">
17     <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
18     <DigestValue>DEADBEEF</DigestValue>
19   </Reference>
20 </SignedInfo>
```

Definition

Optional element which contains the informations about the key (public key, name, certificates, ...)

It exists containers for RSA, DSA, X509, openPGP, ...

XML signature

keyInfo : example

```
1  <KeyInfo>
2    <X509Data>
3      <X509Certificate>DEADBEEF</X509Certificate>
4    </X509Data>
5  </KeyInfo>
```

Definition

Optional element which contains additional informations about the signature

Two majors use cases :

- The signatureProperties element : can contains date, timestamps, serial number of cryptographic hardware, ...
- Contains the original data, in case of an enveloping signature

XML signature

Object : example

```
1  <Object>
2    <SignatureProperties>
3      <SignatureProperty Id="aMadeUpTimeStamp"
4        Target="#2ndDecemberNewsItem">
5        <timestamp xmlns="http://www.ietf.org/rfcxxxx.txt"/>
6          <date>2004122</date>
7          <time>18:30<time>
8        </timestamp>
9      </SignatureProperty>
10   </SignatureProperties>
11 </Object>
```

XML signature use two validations :

1. Reference validation : verify the digest value of every resource signed.
   Verify if we talk about the same document.
2. Signature validation : verify the signedValue.
   Verify if we talk to the right person.

XML signature

enveloped signature : example

```
1  <Letter>
2     <Return-address>address</Return-address>
3     <To>You</To>
4     <Message>msg body</Message>
5     <From>
6        <ds:Signature xmlns:ds="&ds;">
7           <ds:SignedInfo>
8              <ds:CanonicalizationMethod Algorithm=
9                   "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
10             <ds:SignatureMethod Algorithm=
11                  "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
12             <ds:Reference URI="">
13                <ds:Transforms>
14                   <ds:Transform Algorithm="&enveloped;">
15                   </ds:Transform>
16                </ds:Transforms>
17                <ds:DigestMethod Algorithm="&digest;"/>
18                <ds:DigestValue></ds:DigestValue>
19             </ds:Reference>
20          </ds:SignedInfo>
21          <ds:SignatureValue/>
22       </ds:Signature>
23    </From>
24    <Attach>attachement</Attach>
25 </Letter>
```

XML signature

enveloping signature : example

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3    <ds:SignedInfo>
4      <ds:CanonicalizationMethod
5            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
6      <ds:SignatureMethod
7            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
8      <ds:Reference URI="#obj">
9        <ds:DigestMethod
10           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
11       <ds:DigestValue/>
12     </ds:Reference>
13   </ds:SignedInfo>
14   <ds:SignatureValue/>
15   <ds:Object Id="obj">Hello, World!</ds:Object>
16 </ds:Signature>
```

XML signature

detached signature : example

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
3    <ds:SignedInfo>
4      <ds:CanonicalizationMethod
5           Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
6      <ds:SignatureMethod
7           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
8      <ds:Reference URI="http://www.w3.org/TR/xml-stylesheet">
9        <ds:DigestMethod
10            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
11       <ds:DigestValue/>
12     </ds:Reference>
13   </ds:SignedInfo>
14   <ds:SignatureValue/>
15 </ds:Signature>
```

XML encryption

Definition

XML encryption is a W3C specification to encrypt XML entities.

We can encrypt arbitrary data, XML document, XML element, XML element content or reference to a resource outside an XML document.

XML encryption

5 steps

There is 5 steps to encrypt/decrypt data :

1. Selecting the XML document to be encrypted (in whole or in part)
2. Converting the XML document to a canonical form (if necessary)
3. Encrypting the document with the public key of the receiver
4. Sending the encrypted XML document
5. A XML (web service) firewall translate the content in a decrypted form and then forward if to a SOAP server, ...

XML encryption

2 elements

To do that we use two XML element :

- **encryptedData** element : contains the encrypted data, and replace the original data in the document
- **encryptedKey** : provides informations about the key used

XML encryption

plain text

```
1 <PaymentInfo xmlns='http://example.org/paymentv2'>
2   <Name>Alfred</Name>
3   <CreditCard Limit='5,000' Currency='USD'>
4     <Number>4019 2445 0277 5567</Number>
5     <Issuer>Example Bank</Issuer>
6     <Expiration>04/02</Expiration>
7   </CreditCard>
8 </PaymentInfo>
```

XML encryption

cipher text

```
1  <PaymentInfo xmlns='http://example.org/paymentv2'>
2    <Name>Alfred</Name>
3    <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
4        xmlns='http://www.w3.org/2001/04/xmlenc#'>
5      <EncryptionMethod
6          Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>
7          <KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>
8            <KeyName>Alfred</KeyName>
9          </KeyInfo>
10         <CipherData>
11           <CipherValue>ydUNqHkMrD...</CipherValue>
12         </CipherData>
13   </EncryptedData>
14 </PaymentInfo>
```

XML encryption

cipher text

```
1  <PaymentInfo xmlns='http://example.org/paymentv2'>
2    <Name>Alfred</Name>
3    <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
4        xmlns='http://www.w3.org/2001/04/xmlenc#'>
5      <EncryptionMethod
6          Algorithm='http://www.w3.org/2001/04/xmlenc#tripledes-cbc'/>
7          <KeyInfo xmlns='http://www.w3.org/2000/09/xmldsig#'>
8            <KeyName>Alfred</KeyName>
9          </KeyInfo>
10         <CipherData>
11           <CipherReference URI="http://www.example.com/CipherValues.xml"></CipherReference>
12         </CipherData>
13   </EncryptedData>
14 </PaymentInfo>
```

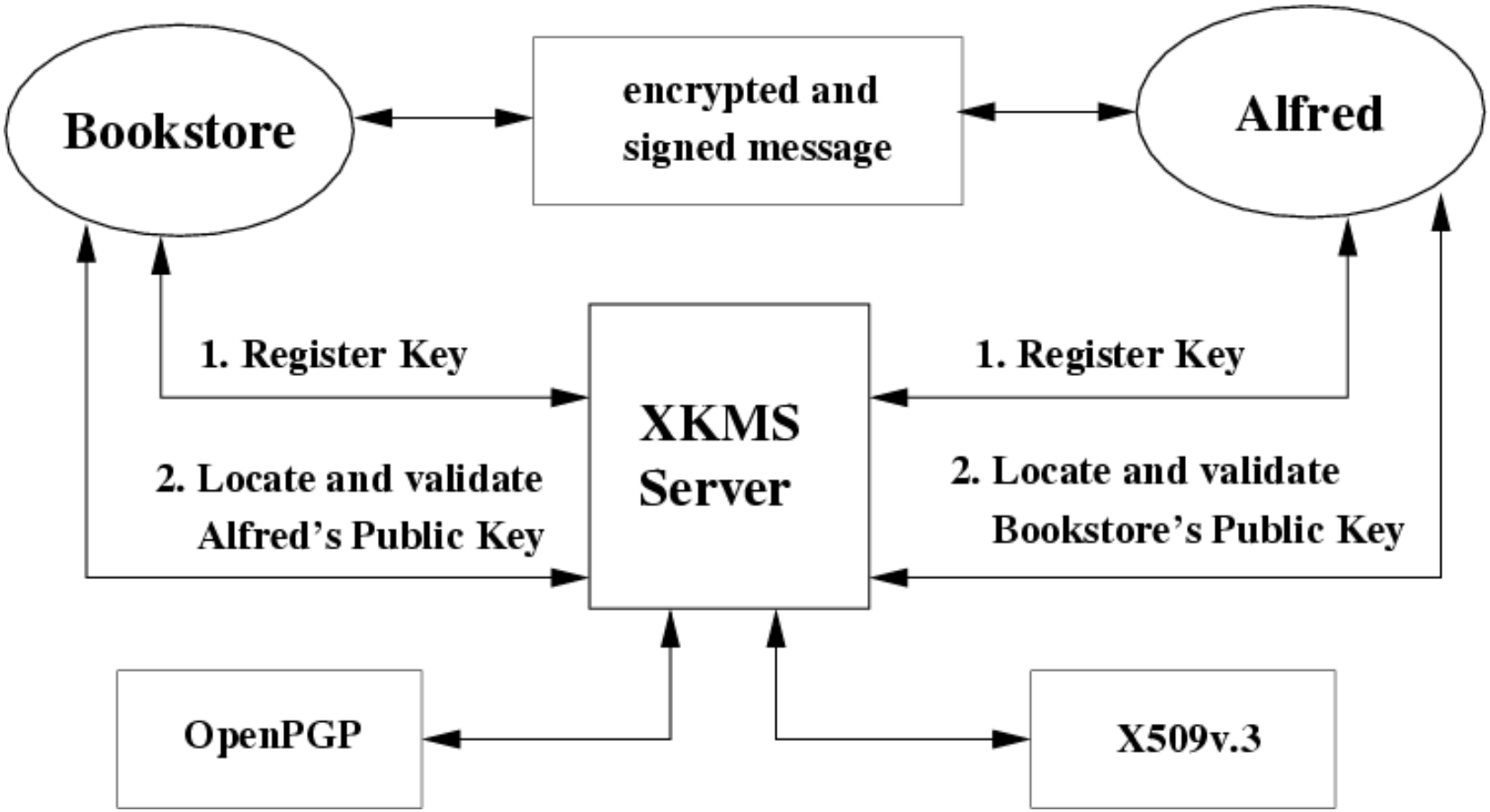
XKMS

Definition

XML key management specification is a protocol created to making easier for XML-based applications to incorporate security mechanism based on PKI (Public Key Infrastructure).

In practice it's a web-based interface to existing PKI which allow to obtaining a valid key associated with someone.

1. XKISS : XML Key Information Service Specification
   - Locate service : retrieve the public key of a service
   - validate service : retrieve and validate the public key of a service
2. XKRSS : XML Key Registration Service Specification
   - Register service : register key and related informations.
     It allow :
     - key registration
     - key revocation
     - key recovery
     - key reissuing


Bookstore
encrypted and
signed message
Alfred
1. Register Key
1. Register Key
XKMS
Server
2. Locate and validate
Alfred's Public Key
2. Locate and validate
Bookstore's Public Key
OpenPGP
X509v.3

XKMS

locate request : example

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LocateRequest xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3       xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
4       Id="I4593b8d4b6bd9ae7262560b5de1016bc"
5       Service="http://test.xmltrustcenter.org/XKMS"
6       xmlns="http://www.w3.org/2002/03/xkms#">
7    <RespondWith>KeyValue</RespondWith>
8    <QueryKeyBinding>
9     <ds:KeyInfo>
10     <ds:X509Data>
11      <ds:X509Certificate>MIICAjC...aFAE=</ds:X509Certificate>
12     </ds:X509Data>
13    </ds:KeyInfo>
14    <KeyUsage>Signature</KeyUsage>
15   </QueryKeyBinding>
16  </LocateRequest>
```

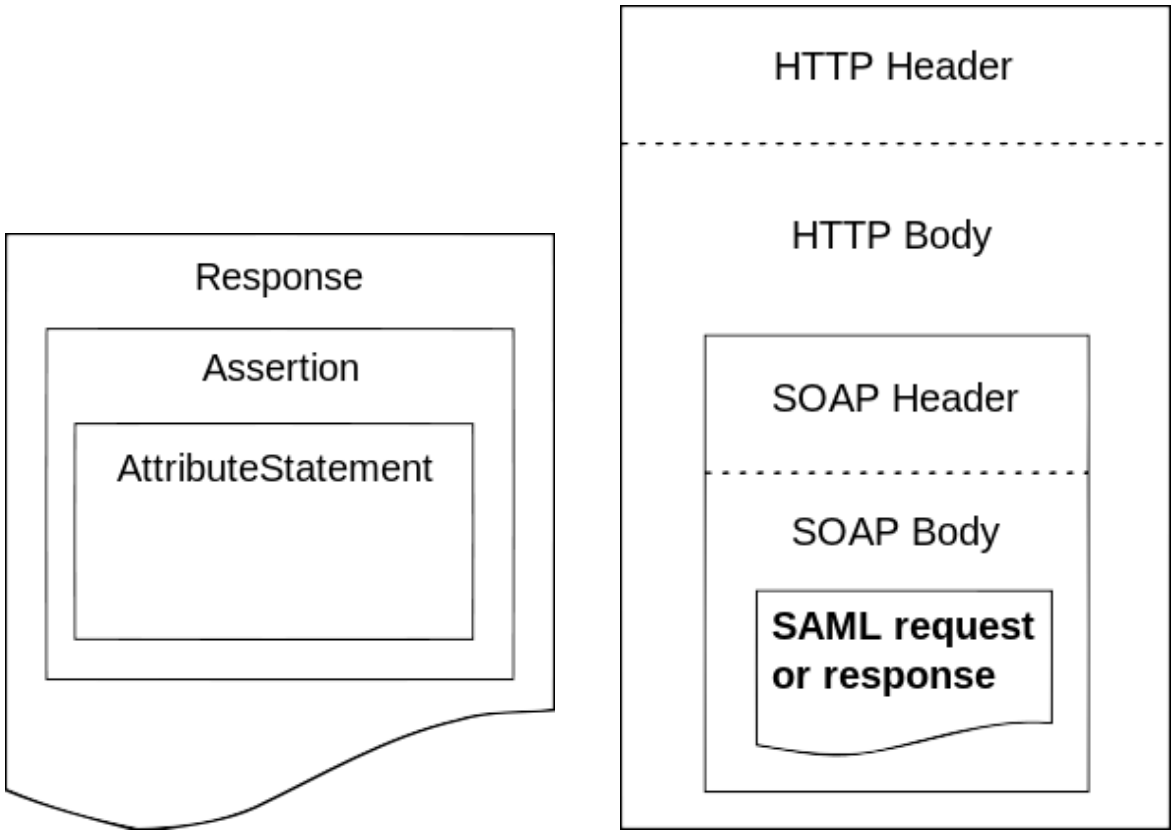
XKMS

locate response : example

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LocateResult xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
3      xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
4      Id="I46ee58f131435361d1e51545de10a9aa"
5      Service="http://test.xmltrustcenter.org/XKMS" ResultMajor="Success"
6      RequestId="#I4593b8d4b6bd9ae7262560b5de1016bc"
7      xmlns="http://www.w3.org/2002/03/xkms#">
8   <UnverifiedKeyBinding Id="I36b45b969a9020dbe1da2cb793016117">
9    <ds:KeyInfo>
10    <ds:KeyValue>
11     <ds:RSAKeyValue>
12       <ds:Modulus>zvbTdKsT...Hw8=</ds:Modulus>
13       <ds:Exponent>AQAB</ds:Exponent>
14     </ds:RSAKeyValue>
15    </ds:KeyValue>
16   </ds:KeyInfo>
17   <KeyUsage>Signature</KeyUsage>
18   <KeyUsage>Encryption</KeyUsage>
19   <KeyUsage>Exchange</KeyUsage>
20  </UnverifiedKeyBinding>
21 </LocateResult>
```

Security Assertion Markup Language

- XML Based
- Authorization & Authentification
- Internet single sign-on
- Small identity informations are exchanged


Response
Assertion
AttributeStatement
HTTP Header
HTTP Body
SOAP Header
SOAP Body
SAML request
or response

SAML relies on several components built using XML

- XML Schema
- XML Signature
- XML Encryption
- SOAP

SAML defines XML assertions, bindings, profiles

- Authentication statements
- Attribute statements
- Authorization decision statements

SAML rules regarding packaging and exchanging queries, responses.

- Authentication query
- Attribute query
- Authorization decision query

In SAML 2.0 attribute query can return "a assertion as an attribute"

SAML

Query Example 1

```
<samlp:AttributeQuery
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  ID="aaf23196-1773-2113-474a-fe114412ab72"
  Version="2.0"
  IssueInstant="2006-07-17T20:31:40">
  <saml:Issuer
    Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
    CN=trscavo@uiuc.edu,OU=User,O=NCSA-TEST,C=US
  </saml:Issuer>
  <saml:Subject>
    <saml:NameID
      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName">
      CN=trscavo@uiuc.edu,OU=User,O=NCSA-TEST,C=US
    </saml:NameID>
  </saml:Subject>
  <saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:2.5.4.42"
    FriendlyName="givenName">
  </saml:Attribute>
  <saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:1.3.6.1.4.1.1466.115.121.1.26"
    FriendlyName="mail">
  </saml:Attribute>
</samlp:AttributeQuery>
```

SAML 1.1 Only SOAP binding, Internet SSO (HTTP POST)
SAML 2.0 associate more bindings, such as artefacts, HTTP Redirect, SAML URI.

- Authentication query
- Attribute query
- Authorization decision query

A SAML profile specifies the way the assertions, protocols, bindings are made to support the use-case. (Internet SSO is a profile).

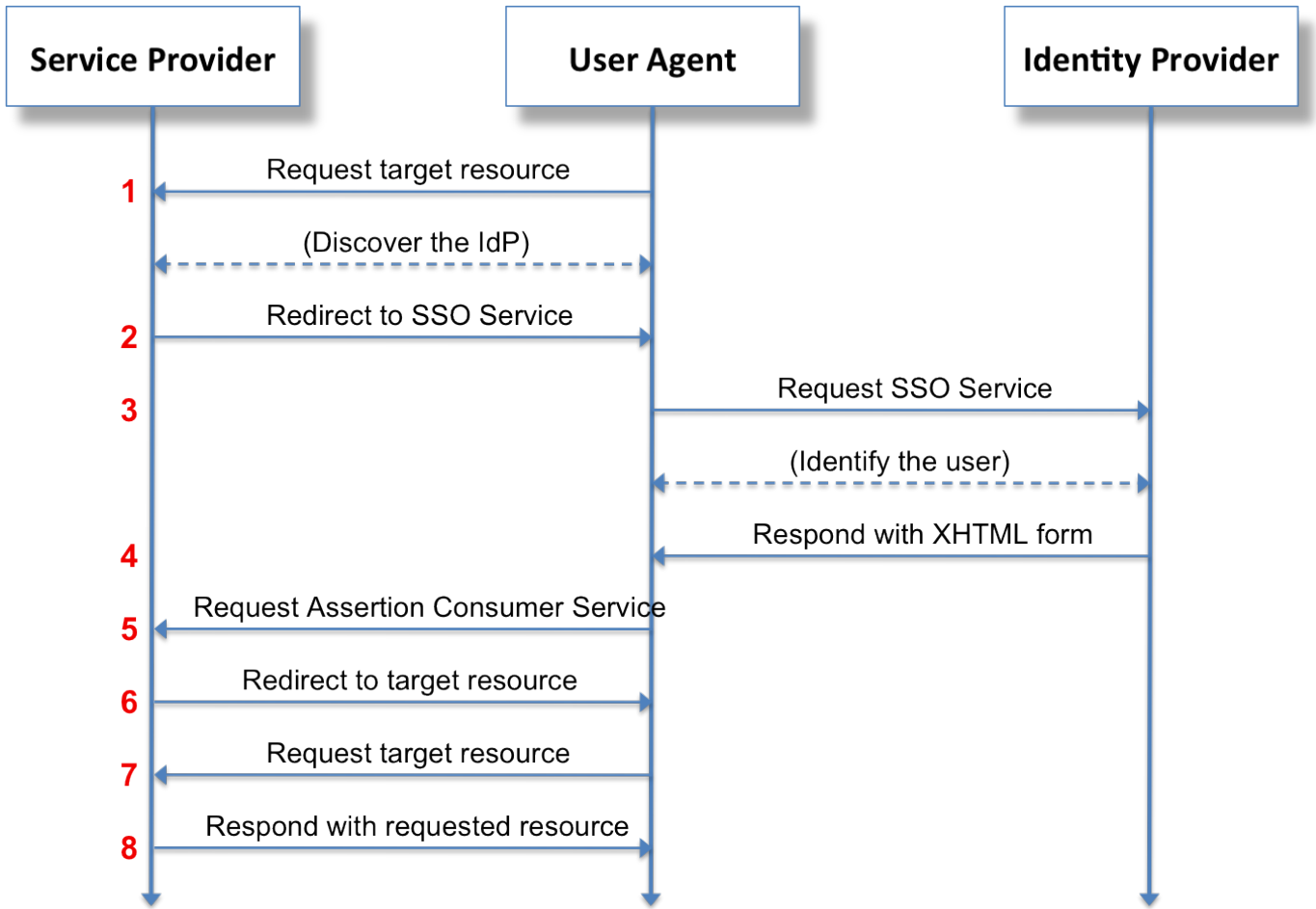


FIGURE: SAML 2.0 Internet SSO

XACML

XML Access CONTROL Markup Language

- SAML only define How the exchange of access information is done
- XACML define how to use this information
- Language : same definition as SAML
- Offers vocabulary for rules -> authorization

2 Basics components

- An access control policy

  Specify rules

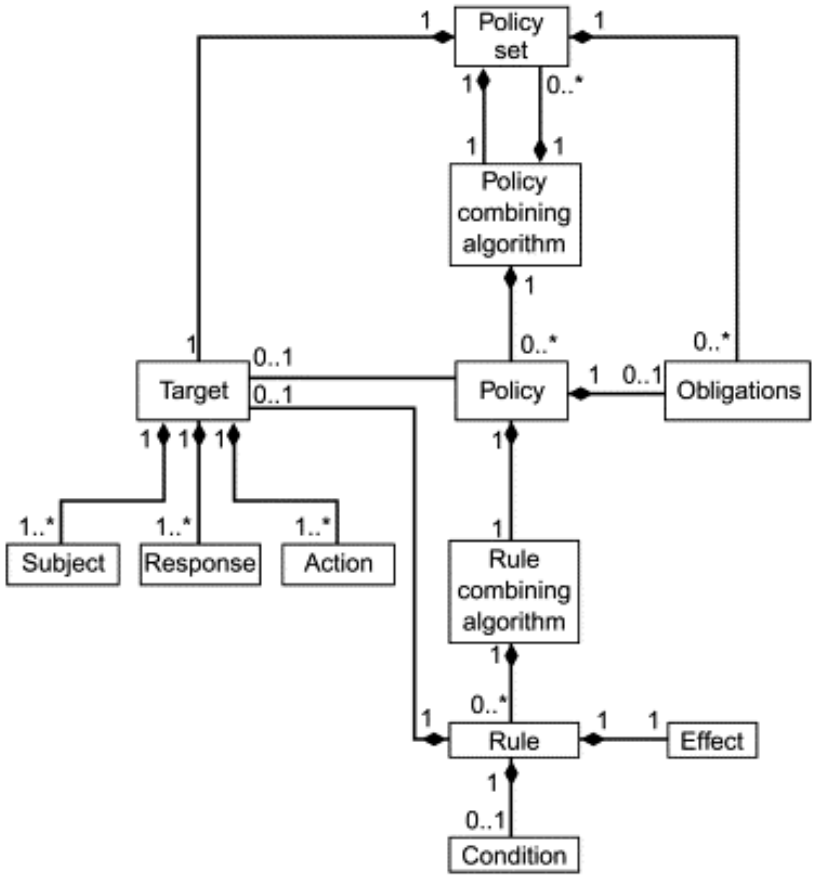
  Describe access control requirements
- Request / Response language

  Describe request and answer to queries

  Action allowed
  - Permit
  - Deny
  - Indeterminate
  - Not applicable

XACML policy

Grammar



XACML example

Action : login

```
1  <Policy PolicyId="SamplePolicy"
2            RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
                 overrides">
3
4      <!-- This Policy only applies to requests on the SampleServer -->
5      <Target>
6        <Subjects>
7          <AnySubject/>
8        </Subjects>
9        <Resources>
10         <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
11           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer</
                   AttributeValue>
12           <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
13                                        AttributeId="urn:oasis:names:tc:xacml:1.0
                                                 :resource:resource-id"/>
14         </ResourceMatch>
15       </Resources>
16       <Actions>
17         <AnyAction/>
18       </Actions>
19     </Target>
```

XACML example

Action : login

```
1   <!-- Rule to see if we should allow the Subject to login -->
2   <Rule RuleId="LoginRule" Effect="Permit">
3
4     <!-- Only use this Rule if the action is login -->
5     <Target>
6       <Subjects>
7         <AnySubject/>
8       </Subjects>
9       <Resources>
10        <AnyResource/>
11      </Resources>
12      <Actions>
13        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
14          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">login</
                AttributeValue>
15          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
16                                     AttributeId="ServerAction"/>
17        </ActionMatch>
18      </Actions>
19    </Target>
```

XACML example

Action : login

```
1       <!-- Only allow logins from 9am to 5pm -->
2       <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
3         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"
4           <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
5             <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
6                                           AttributeId="urn:oasis:names:tc:xacml:1.0
                                                  :environment:current-time"/>
7           </Apply>
8           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</
                AttributeValue>
9         </Apply>
10        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"
11          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
12            <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
13                                          AttributeId="urn:oasis:names:tc:xacml:1.0
                                                  :environment:current-time"/>
14          </Apply>
15          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</
                AttributeValue>
16        </Apply>
17      </Condition>
18
19    </Rule>
20    <!-- We could include other Rules for different actions here -->
21    <!-- A final, "fall-through" Rule that always Denies -->
22    <Rule RuleId="FinalRule" Effect="Deny"/>
23  </Policy>
```

XACML Response

2 Tags

- <Result>
- <Decision>
  Permit
  Deny