Attribute Predicate Profile of SAML and XACML

Gregory Neven and Franz-Stefan Preiss
IBM Research – Zurich
Version 0.3, March 21, 2011

1 Introduction

This document describes a profile for SAML 2.0 [1] to allow SAML authorities to certify that a certain Boolean predicate holds over a subject's attribute values, without revealing the exact attribute values. The profile adheres to the defined extension points of SAML 2.0. It also defines a query format for attribute predicate assertions, and how such assertions are to be used within XACML [2].

SAML attribute assertions enable a SAML authority, sometimes also referred to as an issuer, to certify to a relying party that a subject is associated with a specified set of attribute values. There are many circumstances, however, where the relying party is not interested in the exact attribute values, but merely needs to be certain that a certain condition is satisfied. For example, a SAML authority may keep its subjects' dates of birth on record, but to control access to a teenage chat room, the relying party merely needs to ensure that the requesting subject belongs to the correct age group. Not only may users be reluctant to disclose such identifying information to the chat room host, but also the host itself may prefer not to know the exact date to avoid liability claims in case a data breach occurs.

For small numbers of predicates this problem can obviously be overcome by defining a dedicated Boolean attribute for each possible predicate, which the issuer authenticates by means of a SAML attribute statement. While this may work for common age restrictions such as being younger or older than 18, it is impractical when the space of relevant predicates is large. For example, it is rather unlikely that a dedicated attribute will be globally agreed upon if the minimum age to sign up for a theoretical driving test is 17 years and 9 months.

As another example where attribute predicates can be useful, consider an online opinion poll hosted on a city's website and a SAML authority that can certify subjects' ZIP codes. The city may want to make sure that only legal residents of the metropolitan area can participate in the poll, but for privacy reasons may not want to keep track of the voting statistics of separate neighborhoods. Other websites may want to make sure that the subject has an email address within a particular domain (e.g., youremployer.com), but to guarantee the users' privacy have no interest in the exact address.

Predicates could even involve relation among multiple attributes known to the SAML authority. For example, the signup form for an obesity summer camp may restrict access to people whose waist measure is greater than their inseam, but does not want to force participants to disclose these values.

1.1 Namespaces

Prefix	XML Namespace	Description
(none)	http://www.zurich.ibm.com/csc/security/SAMLAttributePredicatesProfile	this profile
samla	urn:oasis:names:tc:SAML:2.0:assertion	SAML Assertion
samlp	urn:oasis:names:tc:SAML:2.0:protocol	SAML Protocol
xacml	urn:oasis:names:tc:xacml:3.0:core:schema:wd-17	XACML
ds	http://www.w3.org/2000/09/xmldsig#	XML Signature

2 Overview

This profile defines how to query and respond attribute predicate assertions, as well as how such assertions can be used within an XACML implementation. In particular, it defines the following types and elements:

- `<AttributePredicateStatementType>`: A new SAML statement type, defined as a subtype of `<samla:StatementType>`, that contains predicates over attributes.
- `<AttributePredicate>`: A new element containing the description of an attribute predicate that is requested or certified.
- `<AttributePredicateQuery>`: A new SAML query type, defined as a subtype of `<samlp:SubjectQueryAbstractType>`, to transport queries for attribute predicate assertions.

Moreover, this profile describes how an XACML engine can trigger queries for required attribute predicates, and how the resulting responses can be used by the XACML Policy Decision Point (PDP).

3 Attribute Predicate Profile of SAML

3.1 Type `<AttributePredicateStatementType>`

The new SAML statement type `<AttributePredicateStatementType>` inherits from `<samla:StatementType>` and contains one or more `<AttributePredicate>` elements as defined in this profile. Multiple `<AttributePredicate>` elements occurring within one `<Statement>` element follow a conjunctive semantics, i.e., *all* of the contained predicates are asserted (when part of an assertion) or required (when part of a query) to be true.

```
<xs:complexType name="AttributePredicateStatementType">
  <xs:complexContent>
    <xs:extension base="samla:StatementAbstractType">
      <xs:sequence maxOccurs="unbounded">
        <xs:element ref="AttributePredicate" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

To express that an instance of a <saml:Statement> element is of type AttributePredicateStatementType, the xsi:type attribute is set to AttributePredicateStatementType. Such an instance of a <saml:Statement> is referred to as an "attribute predicate statement" in this profile.

3.2 Element <AttributePredicate>

The <AttributePredicate> element contains the description of a predicate (exactly one) over the subject's attributes that the issuer of the assertion certifies to hold. The predicate is expressed by means of an <xacml:Apply> element, which denotes application of a specified function (identified by a URI in the FunctionId attribute) to one or more arguments of the <xacml:Expression> element substitution group. Since the <xacml:Apply> element itself is also a member of this substitution group, arbitrarily complex predicates can be formulated by nesting multiple <xacml:Apply> elements.

```
<xs:element name="AttributePredicate" type="AttributePredicateType"/>
<xs:complexType name="AttributePredicateType">
  <xs:sequence>
    <xs:element ref="xacml:Apply"/>
  </xs:sequence>
  <xs:attribute name="FriendlyDescription" type="xs:string" use="optional"/>
</xs:complexType>
```

Of the members of the <xacml:Expression> element substitution group, the following can occur in the attribute predicate:

- <xacml:AttributeValue>
- <xacml:AttributeDesignator>
  - Any <xacml:AttributeDesignator> element used must be of Category urn:oasis:names:tc:xacml:1.0:subject-category:access-subject. Resource-, action-, and environment attributes are not allowed to occur in an attribute predicate query or assertion.
  - If no values are known for the specified AttributeId by the SAML authority with respect to the concerned subject, then the optional MustBePresent XML attribute governs whether during the evaluation of the predicate (see 3.4 Predicate Evaluation) the <xacml:AttributeDesginator> element is substituted with an empty bag (if MustBePresent is set to false), or whether the predicate evaluates to Indeterminate (if MustBePresent is set to true). In the latter case, a response with status code urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile MUST be returned by the SAML authority.
  - If the value of the Issuer attribute in the <xacml:AttributeDesignator> element is different from the value specified in the <samla:Issuer> element of the enclosing query or assertion, then a response with the newly introduced status code urn:com:ibm:zurich:SAML:2.0:status:InvalidPredicate MUST be returned.
- <xacml:Apply>
  - All function URIs marked as mandatory in XACML version 3.0 [2] MUST be supported as value for the FunctionId attribute by any implementation of this

Comment: The reason is that the SAML authority issues an attribute predicate statement over the attributes of a particular subject. In this context, resource-, action-, and environment attributes is meaningless.

Comment: Is this also the status code returned when attributes are missing for a normal attribute request?

profile. Functions marked as optional MAY be supported, as well as any additional self-defined functions, as long as the semantics are understood by all of the involved parties. The return data type of the outermost function MUST be `http://www.w3.org/2001/XMLSchema#boolean`.

- `<xacml:Function>`

The `<xacml:AttributeSelector>` and `<xacml:VariableReference>` elements are not allowed to occur in attribute predicate queries or assertions. If a SAML authority nevertheless encounters a query with such elements, then it MUST return a response with the (previously introduced) status code `urn:com:ibm:zurich:SAML:2.0:status:InvalidPredicate`.

Comment: For the <xacml:AttributeSelector> the reason is that there is no structure (such as the xacml:Request) that the selector's XPath expression could refer to.

For the <xacml:VariableReference> the reason is that the corresponding <xacml:VariableDefinition> elements cannot be specified within the <xacml:Apply>.

3.3 Element `<AttributePredicateQuery>`

The `<AttributePredicateQuery>` element used to carry (one ore more) requests for assertions on a specified attribute predicate.

```
<xs:element name="AttributePredicateQuery"
type="AttributePredicateQueryType"/>
<xs:complexType name="AttributePredicateQueryType">
  <xs:complexContent>
    <xs:extension base="samlp:SubjectQueryAbstractType">
      <xs:sequence>
        <xs:element ref="AttributePredicate" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

3.4 Predicate Evaluation

When a SAML authority receives an incoming attribute predicate query, then it evaluates the truth of the contained predicates in a way that is semantically equivalent to evaluating an XACML request containing all known attributes of the concerned subject against an XACML policy with a single rule that has the conjunction of the requested predicates as condition. Concrete implementations of this profile do not actually have to deploy an XACML PDP to evaluate predicates, but the returned response MUST be the same as what would have been obtained through the following procedure.

1. An XACML PDP is initialized with the policy below containing a single rule where the outermost `<xacml:Apply>` elements from the `n` queried attribute predicates are embedded in a conjunction in the `<xacml:Condition>` element.

```
<xacml:Policy
  RuleCombiningAlgId="identifier:rule-combining-algorithm:permit-overrides"
  PolicyId="urn:example:PredicatePolicy"
  Version="1.0">
  <xacml:Target/>
  <xacml:Rule RuleId="urn:example:PredicateRule" Effect="Permit">
    <xacml:Target/>
    <xacml:Condition>
      <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        (queried attribute predicate 1)
        (queried attribute predicate 2)
        ...
        (queried attribute predicate n)
      </xacml:Apply>
    </xacml:Condition>
```

```
  </xacml:Rule>
</xacml:Policy>
```

2. The SAML authority creates an XACML request context containing all known attributes for the concerned subject and submits it to the PDP. More precisely, if the concerned subject has a single-valued attribute `urn:example:AttributeId1` of type `xs:string` with value `"Value1"` and a multi-valued attribute `urn:example:AttributeId2` of type `xs:string` with values `"Value2"` and `"Value3"`, the request context submitted to the PDP is:

```
<xacml:Request ReturnPolicyIdList="false" CombinedDecision="false">
  <xacml:Attributes
    Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <xacml:Attribute IncludeInResult="false"
      AttributeId="urn:example:AttributeId1">
      <xacml:AttributeValue DataType="xs:string">
        Value1
      </xacml:AttributeValue>
    </xacml:Attribute>
    <xacml:Attribute IncludeInResult="false"
      AttributeId="urn:example:AttributeId2">
      <xacml:AttributeValue DataType="xs:string">
        Value2
      </xacml:AttributeValue>
      <xacml:AttributeValue DataType="xs:string">
        Value3
      </xacml:AttributeValue>
    </xacml:Attribute>
  </xacml:Attributes>
</xacml:Request>
```

3. If the XACML PDP returns decision `Permit`, then the SAML authority MAY return a SAML response with status code `urn:oasis:names:tc:SAML:2.0:status:Success` and an assertion containing an attribute predicate statement that contains `<AttributePredicate>` elements that are equal to the `<AttributePredicate>` elements contained in the query. Here, equality is defined either as string equality (i.e., including whitespace), or, if the assertion is signed, as equality under the XML canonicalization method used to compute the XML Signature. See [4] for more information on canonicalization methods. There may be situations where the SAML authority MAY not return an attribute predicate assertion, even though the XACML PDP does return Permit, for example, when additional policies apply, or when the subject does not give permission to return the queried assertion.

4. If the XACML PDP returns any decision other than Permit, then the SAML authority MUST return a response with a status code other than `urn:oasis:names:tc:SAML:2.0:status:Success`.

4 Attribute Predicate Profile of XACML

In the same way that the SAML profile of XACML [3] defines, among other things, how an XACML system can request SAML attribute assertions from an on-line attribute authority and use the returned SAML attribute assertions in the XACML system, this section defines how an XACML system can request and use SAML attribute predicate assertions.

The interactions between the different participants and components are depicted in Figure 1. The Subject requests access to a resource located at a remote Host. The Host has a policy in place requiring that the subject must be over 18 years of age to access the resource, as certified by an external Attribute Authority. Rather than requesting the exact date of birth from the Attribute

Authority, the Host merely requests confirmation of the fact that the requesting Subject is indeed over 18 years of age.

The mechanism to incorporate attribute predicates in an XACML system is that the Host specifies its policy in terms of dedicated Boolean attributes using identifiers that are only locally defined. For each of the dedicated local attributes, the Policy Enforcement Point (PEP) knows the translation to a predicate over globally meaningful attributes. When the Policy Decision Point (PDP) reports that one of these local attributes is missing, the PEP sends a query for the corresponding global attribute predicate to the Attribute Authority. The Attribute Authority evaluates the predicate, optionally authenticates the Subject or obtains her permission to disclose the predicate, and returns a signed assertion with the same predicate to the PEP. The PEP then resubmits a request to the PDP for the same resource, but with the value of the dedicated local attribute set to true.

Comment: Typically the PEP does not know which local attributes will be relevant for the PDP in its evaluation of the request. Thus, the PEP can not proactively query for the corresponding global attribute predicates and has to do so 'on demand' for the missing ones. Querying proactively for all global attributes is also not desirable.

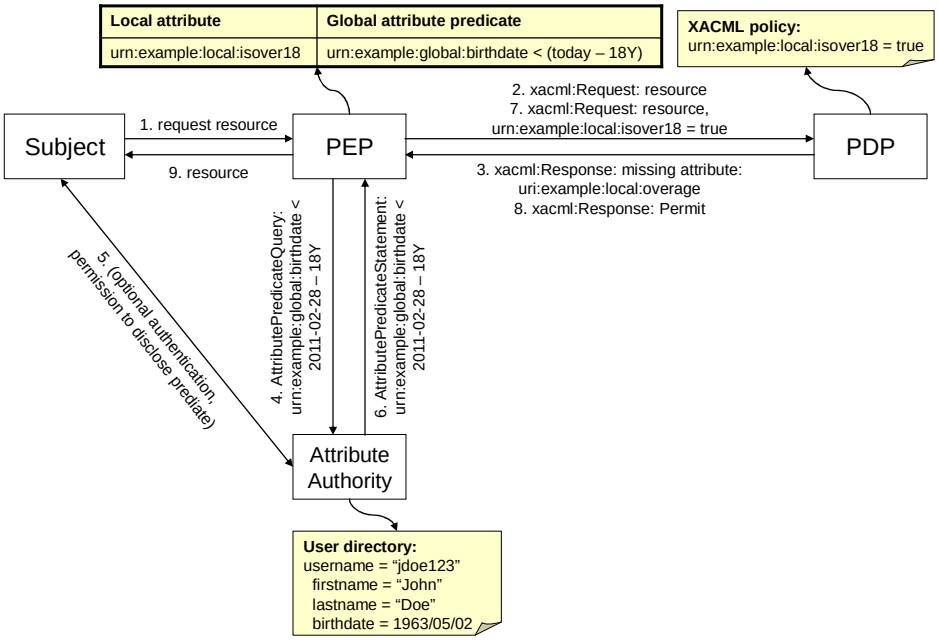

Figure 1: Interaction scenario when using attribute predicates in an XACML system

Details of the steps depicted in Figure 1 are as follows:

1. The Subject sends a request for access to a resource to the PEP of the Host.
2. The PEP constructs an XACML request context for the resource and sends it to the PDP.
3. The PDP evaluates the policy and finds the dedicated local attribute `urn:example:local:isover18` to be missing. It returns an `Indeterminate` response to the PEP stating `urn:example:local:isover18` as a missing attribute.

4. The PEP realizes that `urn:example:local:isover18` is a dedicated local attribute for which it knows the corresponding predicate over globally meaningful attributes. The PEP looks up the corresponding predicate and "partially" evaluates it by replacing all `<xacml:AttributeDesignator>` elements for known attributes with a bag of their value(s). The bag is created using an `<xacml:Apply>` element with the bag function of the appropriate data type. (See Section 5.2 for an example.) The PEP creates an `<AttributePredicateQuery>` for the resulting predicate and sends it to the Attribute Authority.
5. The Attribute Authority optionally authenticates the requesting Subject and obtains his permission to disclose the queried predicate.
6. The Attribute Authority evaluates the truth of the requested predicate based on the Subject's attributes according to the procedure defined in Section 3.4. If the Subject's stored attributes satisfy the predicate, it sends a response containing an assertion for the queried predicate to the PEP.
7. The PEP verifies the Attribute Authority's signature on the predicate and checks that the predicate in the response is equal (possibly modulo the canonicalization method) to the queried predicate. If so, then the PEP creates a new XACML request context for the resource with the value of the `urn:example:local:isover18` attribute set to `true` and sends it to the PDP.
8. The PDP evaluates the new request and sends the response to the PEP.
9. The PEP grants or denies access based on the decision embedded in the response.

Comment: Multiple issues here:
Should this 'partial' predicate evaluation process be defined more precisely?
Is the PEP the most appropriate component to do this partial evaluation, or would it better be the Context Handler, or even the PDP?
Does the PEP (or whoever performs the partial evaluation) have access to the relevant attributes (resource, action, environment, and possibly known subject attributes)?

5 Examples (non-normative)

This section contains a non-normative example of the use of attribute predicate assertions in an XACML system. The scenario is that of an on-line drink market that wants to restrict access to its hard liquor catalogue to users that are older than 18, the legal drinking age in the country where the drink market operates.

5.1 Example XACML Policy

The policy below protects the liquor catalogue with resource identifier `file://example/content/liquors` with a rule insisting that the dedicated local subject attribute `urn:example:local:isover18` is set to `true`.

```
<xacml:Policy
  RuleCombiningAlgId="identifier:rule-combining-algorithm:permit-overrides"
  PolicyId="urn:example:Policy1"
  Version="1.0">
  <xacml:Target/>
  <xacml:Rule RuleId="urn:example:Rule1" Effect="Permit">
    <xacml:Target>
      <xacml:AnyOf>
        <xacml:AllOf>
          <xacml:Match
            MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <xacml:AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#anyURI">
              file://example/content/liquors
            </xacml:AttributeValue>
            <xacml:AttributeDesignator
              DataType="http://www.w3.org/2001/XMLSchema#anyURI"
              MustBePresent="true"
```

```
              Category="urn:oasis:names:tc:xacml:3.0:attribute-
              category:resource"
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
              resource-id"/>
          </xacml:Match>
        </xacml:AllOf>
      </xacml:AnyOf>
    </xacml:Target>
    <xacml:Condition>
      <xacml:Apply
        FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
        one-and-only">
        <xacml:AttributeDesignator
          DataType="http://www.w3.org/2001/XMLSchema#boolean"
          MustBePresent="true"
          Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
          subject"
          AttributeId="urn:example:local:isover18"/>
      </xacml:Apply>
    </xacml:Condition>
  </xacml:Rule>
</xacml:Policy>
```

5.2 Example Translation Table

The PEP keeps track of the mapping between local attributes and predicates over global attributes. How it stores this mapping is outside the scope of this specification. In the example of the liquor store, the PEP may associate the local `urn:example:local:isover18` attribute with the following predicate.

```
<AttributePredicate
  FriendlyDescription="The subject is over 18 years of age.">
  <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-
    than-or-equal">
    <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-
      and-only">
      <xacml:AttributeDesignator
        DataType="http://www.w3.org/2001/XMLSchema#date"
        MustBePresent="true"
        Category="urn:oasis:names:tc:xacml:1.0:subject-category:
        access-subject"
        AttributeId="urn:example:global:birthdate"
        Issuer="idp.example.com"/>
    </xacml:Apply>
    <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:date-
      subtract-yearMonthDuration">
      <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-
        one-and-only">
        <xacml:AttributeDesignator
          DataType="http://www.w3.org/2001/XMLSchema#date"
          MustBePresent="true"
          Category="urn:oasis:names:tc:xacml:3.0:attribute-category:
          environment"
          AttributeId="urn:oasis:names:tc:xacml:1.0:environment:
          current-date"/>
      </xacml:Apply>
      <xacml:AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration">
        P18Y
      </xacml:AttributeValue>
    </xacml:Apply>
  </xacml:Apply>
```

```
</AttributePredicate>
```

5.3 Example Attribute Predicate Query

Note that the <AttributePredicate> above contains an <xacml:AttributeDesignator> for the environment attribute urn:oasis:names:tc:xacml:1.0:environment:current-date. This element first needs to be replaced by a bag of its values using an <xacml:Apply> element with the function urn:oasis:names:tc:xacml:1.0:function:date-bag. The resulting predicate is embedded in the following attribute predicate query that is sent to the Attribute Authority idp.example.com.

```
<AttributePredicateQuery
  Version="2.0"
  ID="query23a0821cf186ea0a22e3818750a809b6cb3b4cda"
  IssueInstant="2011-02-28T23:59:58">

  <samla:Issuer>idp.example.com</samla:Issuer>
  <samla:Subject>
    <samla:NameID
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
      pseudonym123456
    </samla:NameID>
  </samla:Subject>

  <AttributePredicate
    FriendlyDescription="The requestor is over 18 years of age.">
    <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-
      less-than-or-equal">
      <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-
        one-and-only">
        <xacml:AttributeDesignator
          DataType="http://www.w3.org/2001/XMLSchema#date"
          MustBePresent="true"
          Category="urn:oasis:names:tc:xacml:1.0:subject-category:
          access-subject"
          AttributeId="urn:example:global:birthdate"/>
        </xacml:Apply>
        <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:date-
          subtract-yearMonthDuration">
          <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
            date-one-and-only">
            <xacml:Apply
              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-bag">
              <xacml:AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#date">
                2011-02-28
              </xacml:AttributeValue>
            </xacml:Apply>
          </xacml:Apply>
          <xacml:AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration">
            P18Y
          </xacml:AttributeValue>
        </xacml:Apply>
      </xacml:Apply>
    </AttributePredicate>

</AttributePredicateQuery>
```

5.4 Example Attribute Response

The Attribute Authority verifies that the `urn:example:global:birthdate` attribute that he has on record for the user with transient ID `pseudonym123456` satisfies the queried predicate through the procedure described in Section 3.4. If it does, then it returns the following assertion as response to the PEP. (All cryptographic data is simulated, the signature doesn't verify correctly.)

```
<samlp:Response Version="2.0"
  ID="responsebd6996d271d23129dc2068c497ea3415f00b8b73"
  InResponseTo="query23a0821cf186ea0a22e3818750a809b6cb3b4cda"
  IssueInstant="2011-02-28T23:59:59">

  <samla:Issuer>idp.example.com</samla:Issuer>
  <samlp:Status>
    <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>

  <samla:Assertion IssueInstant="2011-02-28T23:59:59"
    ID="assertion116fcc68e6a3feb788e5c89520b5f4eadf5ebbcf" Version="2.0">
    <samla:Issuer>idp.example.com</samla:Issuer>
    <ds:Signature>
      <ds:SignedInfo>
        <ds:CanonicalizationMethod
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        <ds:SignatureMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
        <ds:Reference
          URI="#assertion116fcc68e6a3feb788e5c89520b5f4eadf5ebbcf">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#
            envelopedsignature"/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <InclusiveNamespaces PrefixList="#default xacml samla samlp ds xsi"
            xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>
          192d3b53f6c4fda041ff36899a91422e9e9a8a2b
        </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
       hq4zk+ZknjggCQgZm7ea8fI79gJEsRy3E8LHDpYXWQIgZpkJN9CMLG8ENR4Nrw+n
       7iyzixBvKXX8P53BTCT4VghPBWhFYSt9tHWu/AtJfOTh6qaAsNdeCyG86jmtp3TD
       MwuL/cBUj2OtBZOQMFn7jQ9YB7klIz3RqVL+wNmeWI4=
    </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>
          MIICyjCCAjOgAwIBAgICAnUwDQYJKoZIhvcNAQEEBQAwgakxCzAJBgNVBAYTAlVT
        </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>

  <samla:Subject>
    <samla:NameID
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
      pseudonym123456
```

```
    </samla:NameID>
  </samla:Subject>

  <samla:Statement xsi:type="AttributePredicateStatementType">
    <AttributePredicate
      FriendlyDescription="The requestor is over 18 years of age.">
      <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-
        less-than-or-equal">
        <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-
          one-and-only">
          <xacml:AttributeDesignator
            DataType="http://www.w3.org/2001/XMLSchema#date"
            MustBePresent="true"
            Category="urn:oasis:names:tc:xacml:1.0:subject-category:
            access-subject"
            AttributeId="urn:example:global:birthdate"/>
          </xacml:Apply>
          <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:
            date-subtract-yearMonthDuration">
            <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
              date-one-and-only">
              <xacml:Apply
                FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-bag">
                <xacml:AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#date">
                  2011-02-28
                </xacml:AttributeValue>
              </xacml:Apply>
            </xacml:Apply>
            <xacml:AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration">
              P18Y
            </xacml:AttributeValue>
          </xacml:Apply>
        </xacml:Apply>
      </AttributePredicate>
    </samla:Statement>
  </samla:Assertion>
</samlp:Response>
```

After receiving the response, the PEP verifies the correctness of the signature and checks that the predicate included in the statement is equal (possibly modulo the canonicalization method of the signature) to the queried predicate. If both tests succeed, the PEP creates a new XACML request context for resource `file://example/content/liquors` with the subject attribute `urn:example:local:isover18` set to `true`.

6 References

[1] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005.

[2] eXtensible access control markup language (XACML) Version 3.0, OASIS Committee Specification 01, August 2010. http://docs.oasis-open.org/xacml/3.0/xacml-194 3.0-core-spec-cs-01-en.doc

[3] SAML 2.0 Profile of XACML, Version 2.0, Committee Draft 1, 16 April 2009.

[4] XML-Signature Syntax and Processing, World Wide Web Consortium, February 2002. http://www.w3.org/TR/xmldsig-core/