Enabling Privacy-Preserving Credential-Based Access Control with XACML and SAML

Claudio A. Ardagna*, Sabrina De Capitani di Vimercati*, Gregory Neven†, Stefano Paraboschi‡,
Franz-Stefan Preiss†, Pierangela Samarati*, and Mario Verdicchio‡
*Università degli Studi di Milano, Italy
†IBM Research - Zürich, Switzerland
‡Università degli Studi di Bergamo, Italy

***Abstract*—In this paper we describe extensions to the access control industry standards XACML and SAML to enable privacy-preserving and credential-based access control. Rather than assuming that an enforcement point knows all the requester's attributes, our extensions allow the requester to learn which attributes have to be revealed and which conditions must be satisfied, thereby enabling to leverage the advantages of privacy-preserving technologies such as anonymous credentials. Moreover, our extensions follow a credential-based approach, i.e., attributes are regarded as being bundled together in credentials, and the policy can refer to attributes within specific credentials. In addition to defining language extensions, we also show how the XACML architecture and model of evaluating policies can be adapted to the credential-based setting, and we discuss the problems that such extensions entail.**

***Keywords*-Access control, privacy, anonymous credentials, XACML.**

I. Introduction

An appropriate access protection of valuable online services and resources is fundamental to support the ongoing digitalization of businesses, institutions, and governments. In particular, enterprises are moving away from an assumption of an *a priori* knowledge of all authorized users. Rather, they are increasingly opening up their services to — possibly new and unknown — users in possession of credentials issued by trusted third-party identity providers.

In such a scenario, classic access control approaches are not adequate, and alternative approaches such as attribute-based access control (ABAC) [1] are more and more deployed, allowing for the expression of access control restrictions in terms of conditions over the attributes of the requester and of the protected resource. Moreover, standards such as the eXtensible Access Control Markup Language (XACML) and the Security Assertion Markup Language (SAML) were proposed to specify ABAC policies and to exchange authenticated attribute values, respectively.

A number of recent works [2], [3], [4], [5] have proposed a shift from ABAC to *credential-based access control* (CBAC, also called "card-based access control"), in which the attributes, which are collectively attached by a requester in ABAC, are instead grouped in credentials (or "cards") owned by such requester. The issuer of a credential vouches for the correctness of the attribute values with respect to the credential owner.

Not only does this abstract view on credentials intuitively mirror the real-world authentication cards found in every citizen's wallet today, but it also acts as an excellent model to unify authentication technologies as diverse as SAML, OpenID, X.509 certificates, trusted LDAP servers, and anonymous credentials [6], [7], [8]. In particular, we see anonymous credentials as particularly interesting, because of the strong privacy advantages they offer. More specifically, they enable the requester to selectively reveal subsets of attributes from a credential, and even to merely prove that the attributes contained in a credential satisfy a certain condition without revealing the exact attribute values, and all of this while preserving unlinkability between different uses of the same credential.

The goal of this work is to bring privacy-preserving credential-based access control to the real world by leveraging the status of XACML as *de facto* standard in access control languages. To do so, however, a number of issues need to be addressed. First, XACML does not manage attributes bundled in credentials, and, thus, does not allow to distinguish whether two attributes are contained in the same credential or in different ones. This feature is needed to avoid abuses like, for instance, the possibility for the owner of two university diploma credentials to use the course of study of one diploma and the grades of the other. Also, the type of the containing credential may be important, e.g., to distinguish the name as it appears on a passport and on a driver's license, even if both are issued by the state. To achieve such a result, it is fundamental to have a description of credentials and their attributes and that such description is shared among all the actors involved in the system, i.e., there is a need for an ontology of credential types.

Second, XACML prescribes that the requester communicates all of her attributes to the server for the evaluation of the access control policy, which is problematic from a privacy perspective. Some technologies such as SAML, OpenID, and anonymous credentials, offer the possibility to reveal only a subset of the attributes contained in a credential. Such features can be exploited by first communicating

the policy to the requester, so that she can disclose only the information necessary for the access.

Third, XACML and SAML merely allow requesters to reveal concrete attribute values, rather than allowing them to prove that certain conditions over the attributes hold. This further privacy-preserving feature can be obtained by leveraging the cryptographic power of anonymous credentials.

This work tackles above issues and is organized as follows: Sections II and III provide an overview of the literature and the scenario we refer to, respectively; Section IV presents the basics of XACML, which are then extended in Section V to deal with credentials; Section VI illustrates how we leverage existing Semantic Web standards to organize credentials and their attributes into hierarchies; Section VII provides the details of the extension of SAML to enable users to prove conditions over their attributes, and Section VIII shows the necessary modifications of the current XACML architecture to support the proposed extensions; finally, Section IX summarizes our efforts and discusses our future work.

II. Related Work

The problem of performing access control without prior knowledge of the access-requesters is typically overcome by basing the access control decisions on, possibly certified, properties the requester has. Most of the proposed solutions, e.g., [1], [9], [10], are based on different forms of logic. Although such approaches are highly expressive and powerful, they are difficult to apply in practice where simplicity and easy of use are required. In addition, all such proposals lack support for anonymous credentials.

The eXtensible Access Control Markup Language (XACML) [11] is the de facto standard for expressing access control policies and permits to express access control rules on the basis of a requester's properties. Although XACML enjoys large adoption in industry due to its simplicity and its powerful extension mechanism, it has several limitations. In particular, there is no support for certified credentials nor does it allow for dealing with unknown requesters. Rather, it is assumed that requesters provide all their properties together with the access request. However, this poses significant privacy risks for the requesters.

Some recent proposals have investigated credential-based access control with anonymous credentials. Ardagna et al. [3] introduce a language that is explicitly targeted to anonymous credentials. However, unlike the solution presented in this paper, their proposal does not extend to other technologies. Camenisch et al. [5] propose a language for technology-independent credential-based access control, but the language follows a proprietary syntax, which makes deployment in real-world access control scenarios hard. The recent work by Ardagna et al. [4] defines credential-based access control extensions for XACML, but cannot express the advanced functionalities of anonymous credentials. Neither of the above languages specifies wire formats necessary for the server to communicate the applicable access control requirements to the requester, and for the requester to send a description of her claimed credential properties back to the server. In this work we solve this issue by extending SAML with the necessary expressivity.

III. Scenario

The setting that we envisage is the following. The requester owns a set of credentials obtained from various issuers, possibly implemented in different credential technologies. A credential is a list of attribute-value pairs with technology-specific information, called *pre-evidence*, that the requester will need to substantiate the claims that she will make about the credential. A credential is always of a certain *type* that defines which attributes are contained in the credential. We assume that every credential has the dedicated attributes *type* and *issuer*.

Servers host resources and protect them with policies expressed in an extended version of XACML. Users requesting access to a resource receive the relevant policy, which describes the requirements on the requester's credentials in order to be granted access. The policy may include requirements on multiple credentials at the same time, meaning that multiple credentials have to be presented in order to obtain access, and may include *provisional actions*, i.e., actions that the requester needs to fulfill prior to being granted access.

Subsequently, the requester inspects the policy and, if she has the necessary credentials to satisfy it, she creates a *claim* over a suitable subset of her credentials, which can describe (1) values of attributes contained in these credentials, (2) conditions over non-disclosed attributes, and (3) the fulfilled provisional actions. From the pre-evidence contained in the credentials, the requester derives (technology-specific) *evidence* for the claim to convince the server of its correctness, of the integrity of the attribute values, of her ownership of the credentials, and, possibly, of the freshness of the claim.

Afterwards, the requester makes a new request for the resource, but this time she includes the created claim and evidence. The server verifies the validity of the evidence w.r.t. the claim and evaluates whether the policy is fulfilled by the claim. Acces is granted or denied accordingly.

Note that there is a clear distinction between a credential and a claim: while the former is a typically long-lived bundle of attributes, the latter is a short-lived description of properties that these attributes enjoy. The requester keeps her credentials and the associated pre-evidence until they expire or are revoked; a claim and the corresponding evidence are usually only relevant within a single session with a server.

IV. Basics of XACML

XACML defines an XML-based access control policy language as well as a processing model for evaluating the

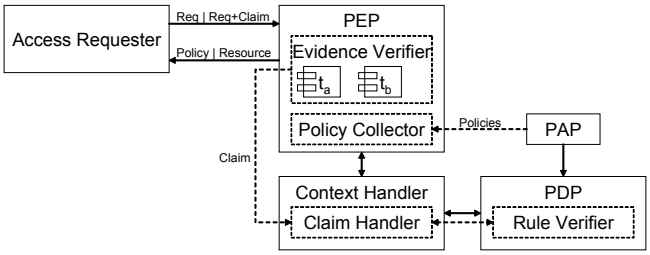

Figure 1. XACML architecture with extensions. Standard XACML components are depicted with *solid* lines. Extensions are depicted with *dotted* lines.

policies on the basis of a given XACML access request. Such request specifies by means of *attributes* which *subject* (i.e., who) wants to perform which *action* (i.e., do what) on which *resource* (i.e., on what).

An XACML policy has a *PolicySet* root element that further contains *Policy* or *PolicySet* elements. A *Policy* contains a set of *Rule*s that define positive or negative authorizations (*Permit* or *Deny* rules). The *Rule*, *Policy* and *PolicySet* elements may contain a *Target* that determines their applicability, i.e., to which access requests the respective elements and their children apply. The *Target* is expressed in terms of simple combinations of attributes describing applicable subjects, actions and resources. The applicability of a *Rule* is further determined by a boolean *Condition* that allows for more complex restrictions by means of functions over such attributes. Functions are stated by means of *Apply* elements that specify a respective *FunctionID* XML-attribute (e.g., 'string-equal') and that contain child elements representing the appropriate function parameters. Such parameters may be: (1) *AttributeDesignator* elements referring to attributes given in the request, (2) concrete attribute values, or (3) further *Apply* elements. Each *PolicySet* and *Policy* element also specifies a *combining algorithm* defining how to combine the different outcomes of the contained child elements (i.e., *Policy*/*PolicySet* and *Rule* elements, respectively) when a request is evaluated w.r.t. an XACML policy.

An XACML system consists at least of a policy enforcement point (PEP), a policy decision point (PDP), a policy administration point (PAP) and a context handler (cf. Figure 1). Access requesters issue their requests to the PEP who is responsible for enforcing the access control decisions that are rendered by the PDP on the basis of the request. The PDP makes decisions w.r.t. policies that are created and maintained by the PAP. The context handler is an intermediate component between the PEP and the PDP that buffers the attributes that were given to the PEP in the request and provides them to the PDP on demand.

V. Credential-based XACML

In the following we use the namespace prefix `xacml` to refer to the XACML 3.0 [11] namespace. Our extensions are defined in namespace `http://www.primelife.eu`, denoted by prefix `pl` or without prefix.

The language extensions that we propose to XACML go beyond the standard extension points. All proposed extensions are in line with the semantics of existing XACML language constructs though, i.e., we do not alter the semantics of existing elements or attributes. To facilitate the adoption of our approach in existing XACML code bases, we designed our extensions for minimal impact on the language and the XACML evaluation mechanism.

XACML rules that contain credential requirements can only have effect `Permit`. Rules with effect `Deny` are pointless as they essentially require that the requester *does not* have a certain credential. Assuming that the requester's goal is to obtain access, she can always pretend not to have the specified credentials.

Our extensions enable policy authors to express conditions on the credentials that a requester must own and the actions that she must perform to be granted access. To this end we augment the `<xacml:Rule>` element with optional `<CredentialRequirements>` and `<ProvisionalActions>` child elements. The former describes the credentials that the requester needs to own and the conditions these credentials have to satisfy. The latter describes the actions that she has to perform. We now discuss both elements in more detail.

A. Credential Requirements

To express credential-based access control policies, the language needs a way to refer to the credentials that bundle several attributes together. For example, it must be possible to refer to the requester's name as it appears on her passport, not on her credit card. Cross-credential conditions are another important use case: for example, the policy language must allow to express that the names on a credit card and on a passport must match, or that the expiration date of an entry visa is before the expiration date of a passport.

To this end, `<CredentialRequirements>` contains a `<Credential>` child element for each credential involved in the rule, which is assigned a (rule-wide unique) identifier `CredId` as an attribute. The `<Credential>` can contain `<AttributeMatchAnyOf>` child elements that allow to compare an attribute of that credential to a list of values. The `<CredentialRequirements>` also contain a `<Condition>` where conditions on the credentials' attributes can be expressed. Inside a condition, one can refer to an attribute `AttrId` within a particular credential by means of `<CredAttributeDesignator>` which takes both `CredId` and `AttrId` as attributes. We purposely did not add an optional `CredId` attribute to `<xacml:AttributeDesignator>`, as the `Issuer` attribute of the latter would conflict with the credential attribute `pl:Issuer`. An example rule is given in Figure 2.

```
<Rule Effect="Permit" RuleId="rule2">
 <xacml:Condition>
       <!-- XACML condition relevant for the rule's applicability -->
 </xacml:Condition>
 <CredentialRequirements>
  <Credential CredentialId='pp'>
   <AttributeMatchAnyOf AttributeId="pl:CredentialType">
    <MatchValue MatchId="pl:subtype-of">un:PhotoID</MatchValue>
   </AttributeMatchAnyOf>
   <AttributeMatchAnyOf AttributeId="pl:Issuer">
    <MatchValue MatchId="xacml:anyURI-equal">http://www.usa.gov</MatchValue>
   </AttributeMatchAnyOf>
  </Credential>
  <Condition>
   <xacml:Apply FunctionId='xacml:date-less-than-or-equal'>
    <CredentialAttributeDesignator CredId="pp" AttributeId="un:DateOfBirth"/>
    <xacml:Apply FunctionId="xacml:date-subtract-yearMonthDuration">
     <xacml:EnvironmentAttributeDesignator AttributeId="xacml:current-date"/>
     <xacml:AttributeValue DataType="xs:duration">P21Y</xacml:AttributeValue>
    </xacml:Apply>
   </xacml:Apply>
  </Condition>
 </CredentialRequirements>
 <ProvisionalActions>
  <ProvisionalAction ActionId="pl:Reveal">
   <xacml:AttributeValue DataType="xs:anyURI">un:Sex</xacml:AttributeValue>
   <xacml:AttributeValue DataType="xs:anyURI">pp</xacml:AttributeValue>
  </ProvisionalAction>
 </ProvisionalActions>
</Rule>
```

Figure 2. Example rule stating that access is granted to users who are at least twenty-one years old according to a piece of PhotoID issued by the US government, but only after revealing the gender mentioned on the same piece of PhotoID. Namespace prefix `xacml` refers to the XACML 3.0 namespace, `xs` to XML Schema, `pl` to `http://www.primelife.eu`, and `un` to `http://www.un.org`.

Conditions on credential attributes are expressed using the same schema as the `<xacml:Condition>` element (extended by the mentioned `<CredAttributeDesignator>`), but are contained in a separate `<Condition>` child element of a `<Credential>` element. The reason for this separation is that whenever no adequate claim is attached to a resource request, the applicable policy must be returned. Deciding applicability of a policy entails evaluating the `<xacml:Condition>`, which is impossible when it involves credential attributes that have not been revealed yet. To avoid this issue, we keep the credential requirements separate from elements relevant to the rule's applicability.

Conditions can contain any combination of restrictions on any credential attributes, including the issuer and the type of the credential. For matching credential types, we introduce a new function `pl:subtype-of` that checks whether the presented credential is of a subtype of a specified credential type as per the ontology that we will discuss in Section VI.

B. Provisional Actions

The `<ProvisionalActions>` element contains the actions that have to be performed by a requester prior to being granted access. The types of actions that we model are:

– *Consent:* The requester has to explicitly consent to a given statement, e.g., the terms of service. How consent is given could depend on the underlying technology: it could for example involve a cryptographic signature, or simply a click on a button in the user interface.

– *Attribute disclosure:* Rather than assuming that all attributes of a credential are revealed by default, the policy explicitly lists which attributes of which credential need to be revealed. Moreover, in order to fully leverage the power of privacy-enhancing technologies such as anonymous credentials that allow the requester to prove conditions without revealing the attribute values, this list does *not* (necessarily) include the attributes that occur in the conditions. Apart from the attribute and credential identifiers, the requirement to reveal an attribute can optionally specify a data handling policy describing how the attribute value will be treated.
In some cases, it is not the PEP who needs the attribute value, but some third party. For instance, an online bookshop may require the requester to reveal her address to a shipping company, not to the bookstore itself. When using anonymous credentials, such requirements can be efficiently fulfilled by means of verifiable encryption [12]. We model this with an optional fourth argument describing the entity to whom the attribute must be disclosed.

– *Consumption control:* Consumption control allows the policy author to impose limitations on how often the same credential can be used to obtain access. For example, one could impose that each ID card can only be used once to vote in an online opinion poll. A consumption control statement has to specify the credential to be consumed, the number of units to spend, the limit of units that can be spent in total, and a "consumption scope". We refer to [5] for details on their exact semantics.

Rather than restricting our extensions to the above action types, we leave open the possibility to add new types of provisional actions later. We do so by a mechanism similar to the one used for defining functions in XACML. Namely, each provisional action is contained in a `<ProvisionalAction>` element that includes an action identifier as an attribute `ActionId`. We define action identifiers for the action types above, but allow users to add more identifiers later.

Provisional actions can be parameterized with arguments, e.g., the statement to be consented to in a `Consent` action. We use an approach similar to how in XACML arguments are passed to functions in the `<xacml:Apply>` element. Namely, a `<ProvisionalAction>` can take any number of `<xacml:Expression>` child elements, in which the action arguments are encoded. For example, the policy in Figure 4 contains the requirement to reveal the gender as specified on the identity card.

This is, however, slightly problematic for actions with optional arguments: since arguments are passed as an unnamed sequence of `<xacml:Expression>` elements, parsing them becomes ambiguous if more than one argument is optional. For example, the requirement to reveal an attribute takes one mandatory argument (the attribute to be revealed) and three optional arguments (the recipient, the data handling policy, and the credential identifier). We solve this problem by introducing separate action identifiers for each possible combination of specified attributes. For example, for attribute disclosure we define the action iden-

tifiers pl:Reveal, pl:RevealTo, pl:RevealUnderDHP, and pl:RevealToUnderDHP.

VI. Credential Type Ontologies

For an effective extension of XACML that bundles attributes into credentials, such organization of information, which we call a *credential type ontology*, must be shared by all involved parties. The best way to achieve this result is to express the structures of the credentials in a widespread standard language. Moreover, this language should allow for the extension of existing types, so that credential issuers can introduce new types within the context of the ontologies already in use. We rely on the Web Ontology Language (OWL) [13], which is XML-based, thus allowing for ontologies that are easily extended and exchanged by applications, regardless of the platform.

In our model, each credential belongs to a specific type, which defines the list of its attributes. The type of a credential is itself an attribute. Types and attributes are defined in ontologies that can be referred to in the policy language by means of URIs, possibly mapped onto prefixes (e.g. http://www.un.org/ is mapped onto un:). For example, the United Nations could design an ontology that defines the attribute un:nationality as specifying the nationality of a credential bearer, encoded as a two-character ISO 3166 country code, and the credential type un:Passport as a digital template for real-world passports, comprised of a list of attributes including un:nationality. Attributes defined in an ontology can be reused in other ontologies: a government may define a credential type for national ID cards, including the un:nationality attribute. Credential types are organized in a hierarchy with supertypes and subtypes. If credential type A has a subtype B, then B contains all the attributes of A with possible restrictions on their values, and it may include additional attributes. Multiple inheritance is allowed, meaning that a subtype contains all the attributes of all its supertypes.

Our use of OWL for credential ontologies is based on the following idea: a credential type is modeled as an OWL class. In particular, the URI of the OWL class can be used as the URI of the credential type, and we are enabled to exploit the OWL class definition mechanism to define which attributes a certain credential type includes. Moreover, we can make use of OWL's subclassing mechanism (rdfs:subClassOf) to model inheritance among credential types in a very natural way, with OWL subclasses corresponding to credential subtypes.

We define a root credential type with URI pl:Credential (cf. Figure 3), which is the supertype of all other credential types and contains the credential's issuer (pl:issuer) as an attribute.

All other credentials can be defined as a subclass of the pl:Credential class. For instance, the United Nations' un:Passport credential type can be defined as a subclass of pl:Credential by extending its attribute list with attributes un:firstName, un:lastName, un:dateOfBirth, un:nationality, and un:photo.

```
<owl:Class rdf:about="Credential">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="issuer"/>
   <owl:qualifiedCardinality
       rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:qualifiedCardinality>
   <owl:onDataRange rdf:resource="&xsd;anyURI"/>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Figure 3. An OWL class for credentials

In our model, when a policy requires the presentation of a credential of a certain type, a credential of any subtype is accepted, as it contains all the information included in the supertype, and some more. Function pl:subtype-of is introduced to be applied to two URIs to check whether the first argument refers to a subclass of the class referred to by the second argument. Let us suppose a credential of type un:Passport is presented by an access requester for the evaluation of a policy that requires an instance of pl:Credential. By exploiting the capabilities of ontology reasoners, function pl:subtype-of states that un:Passport is indeed a subclass of pl:Credential, so that the requester can be granted access.

VII. SAML as Claims Language

Here we describe how we extend SAML for transporting the claims defined in Section III. SAML is a standard allowing for the exchange of certified attributes bundled together into *assertions*, which are similarly structured as credentials. The standard, however, allows only for the exchange of attribute values but not conditions on such values nor notifications of provisional action fulfillment. To address these issues, we use the standard's extension points to embed our <Condition> and <ProvisionalAction> elements into SAML assertions. This allows for the expression of conditions on attributes from one or more credentials as well as action fulfillments, together with the necessary evidence.

VIII. Architectural Extensions

In the following we sketch how we adapt the XACML architecture such that (1) the credential-based XACML policy applicable to a request is communicated to the requester, and (2) the policy can be evaluated on the basis of the provided SAML claim. The modified architecture maintains all standard XACML functionality, i.e., the modifications are *extensions* that do not substitute existing functionality and that are usable in combination with standard features.

We adapt the XACML communication model for allowing the following two-round pattern. In the first round, the requester specifies a resource and obtains the relevant policy from the PEP; in the second round the requester sends the same request with an additional SAML claim. Resending

the request is necessary because the XACML architecture is stateless, meaning that the individual components do not maintain information across multiple rounds. A PEP's response in the first round is embedded in an *XACMLPolicy Assertion* element (cf. SAML profile of XACML [14]), to which the requester is supposed to reply with an appropriate SAML claim. The PEP grants or denies access depending on the claim's validity and the decision of the PDP.

We need to modify the PEP such that it obtains all policies applicable to a user's request and it sends them in a pre-evaluated version to the user. The pre-evaluation substitutes known attributes, e.g., environment attributes such as time and date, with concrete values. One way to obtain all applicable policies is to exploit the *ReturnPolicyIdList* feature of XACML 3.0, which enables a PEP to learn from a PDP all the relevant policy identifiers. The PEP can then obtain the policies directly from the policy administration point (PAP). Alternatively, the PDP could be modified to return not only a list of `PolicyId`s to the PEP, but rather the policies themselves.

When the PEP receives a request with an attached SAML claim, it has to verify the validity of the claim and make it available to the PDP. To verify the validity of the claim evidence, we extend the PEP with an *evidence verifier* component (cf. Figure 1). For every supported credential technology t, this component has a plug-in that can verify evidence specific to this technology. To make the claim available to the PDP, we introduce a *claim handler* component within XACML's context handler. If the claim is valid, the PEP forwards it to the claim handler, which buffers it so that it can be retrieved by the PDP. The PEP then forwards the request (without attached claim) to the PDP.

A PDP evaluates a request from the PEP as usual w.r.t. the rules in the policy. However, rules with credential requirements or provisional actions are treated specially. For such rules to yield a `Permit` decision, not only its applicability (specified by its *target* and *condition*) is relevant, but also the fulfillment of the credential-requirements and provisional actions if any are specified. If so, the PDP fetches the claim from the claim handler. We extend the PDP with a *rule verifier* component that, for given credential requirements, given provisional actions, and a given claim, decides whether the claim implies the requirements and fulfills all the provisional actions. If so, then the rule evaluates to `Permit`, otherwise it evaluates to `Indeterminate`.

IX. Conclusion

In this paper, we proposed extensions to the industry standards XACML and SAML to add support for credential-based access control that fully leverage the power of privacy-preserving technologies such as anonymous credentials. A prototype engine for the language and architecture extensions that we defined is currently being developed under the auspices of the European Commission's PrimeLife project.

Our current architecture requires a new access control process with new claims to take place for each resource request with associated credential requirements. In future work, we will investigate to which extent this can be avoided by comparing the relevant policy to previously made claims.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483 for the project PrimeLife.

References

[1] P. Bonatti and P. Samarati, "A unified framework for regulating access and information release on the web," *JCS*, vol. 10, no. 3, 2002.

[2] J. Li, N. Li, and W. Winsborough, "Automated trust negotiation using cryptographic credentials," in *ACM CCS 2005*.

[3] C. A. Ardagna, J. Camenisch, M. Kohlweiss, R. Leenes, G. Neven, B. Priem, P. Samarati, D. Sommer, and M. Verdicchio, "Exploiting cryptography for privacy-enhanced access control," *JCS*, vol. 18, no. 1, 2010.

[4] C. A. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, and M. Verdicchio, "Expressive and deployable access control in open web service applications," *IEEE Transaction on Services Computing*, 2010, to appear.

[5] J. Camenisch, S. Moedersheim, G. Neven, F.-S. Preiss, and D. Sommer, " A card requirements language enabling privacy-preserving access control," in *ACM SACMAT 2010*, to appear.

[6] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *CACM*, vol. 28, no. 10, 1985.

[7] S. Brands, "Rethinking public key infrastructure and digital certificates— building in privacy," Ph.D. dissertation, Eindhoven Institute of Technology, The Netherlands, 1999.

[8] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *EUROCRYPT 2001*, ser. LNCS, vol. 2045. Springer, 2001.

[9] S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmanian, "Flexible support for multiple access control policies," *ACM TODS*, vol. 26, no. 2, 2001.

[10] M. Winslett, N. Ching, V. Jones, and I. Slepchin, "Assuring security and privacy for digital library transactions on the web: Client and server security policies," in *ADL 1997*.

[11] OASIS, "eXtensible Access Control Markup Language (XACML) Version 3.0," 2009.

[12] J. Camenisch and V. Shoup, "Practical verifiable encryption and decryption of discrete logarithms," in *CRYPTO 2003*, ser. LNCS, vol. 2729. Springer, 2003.

[13] W3C, "OWL 2 Web Ontology Language," 2007.

[14] OASIS, "SAML 2.0 profile of XACML v2.0," 2005.

```
<pl:Policy xmlns:cr="http://www.primelife.eu/ppl/credential"
 xmlns:pl="http://www.primelife.eu/ppl"
 xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.primelife.eu/ppl PrimeLifeSchema.xsd
  urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
 PolicyId="policy1" RuleCombiningAlgId="">

 <xacml:Target/>

 <pl:Rule Effect="Permit" RuleId="rule1">

  <xacml:Target>
   <xacml:Resources>
    <xacml:Resource>
     <xacml:ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
      <xacml:AttributeValue DataType="xs:anyURI">http://www.store.com/subscribe.html</xacml:AttributeValue>
      <xacml:ResourceAttributeDesignator DataType="xs:anyURI" AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
     </xacml:ResourceMatch>
    </xacml:Resource>
   </xacml:Resources>
  </xacml:Target>

  <pl:CredentialRequirements>
   <pl:Credential CredId="#eid">
    <pl:AttributeMatchAnyOf AttributeId="pl:Issuer">
     <pl:MatchValue DataType="xs:anyURI" MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">http://www.fgov.be</pl:MatchValue>
    </pl:AttributeMatchAnyOf>
    <pl:AttributeMatchAnyOf AttributeId="pl:CredentialType">
     <pl:MatchValue DataType="xs:anyURI" MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">http://www.fgov.be/eID</pl:MatchValue>
    </pl:AttributeMatchAnyOf>
   </pl:Credential>

   <pl:Credential CredId="#creditcard">
    <pl:AttributeMatchAnyOf AttributeId="pl:Issuer">
     <pl:MatchValue DataType="xs:anyURI" MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">http://www.visa.com</pl:MatchValue>
     <pl:MatchValue DataType="xs:anyURI" MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">http://www.amex.com</pl:MatchValue>
    </pl:AttributeMatchAnyOf>
    <pl:AttributeMatchAnyOf AttributeId="pl:CredentialType">
     <pl:MatchValue DataType="xs:anyURI" MatchId="pl:subtype-of">http://www.banking.org/CreditCard</pl:MatchValue>
    </pl:AttributeMatchAnyOf>
   </pl:Credential>

   <pl:Condition>
    <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
     <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal">
      <pl:CredentialAttributeDesignator CredId="#eid" DataType="xs:date" AttributeId="http://www.fgov.be/eID/birthdate"/>
      <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration">
       <xacml:EnvironmentAttributeDesignator DataType="xs:date" AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"/>
       <xacml:AttributeValue DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration">P18Y</xacml:AttributeValue>
      </xacml:Apply>
     </xacml:Apply>
     <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-greater-than">
      <pl:CredentialAttributeDesignator CredId="#creditcard" DataType="xs:date" AttributeId="http://www.banking.org/CreditCard/expirationdate"/>
      <xacml:EnvironmentAttributeDesignator DataType="xs:date" AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"/>
     </xacml:Apply>
     <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <pl:CredentialAttributeDesignator CredId="#eid" DataType="xs:string" AttributeId="http://www.fgov.be/eId/firstname"/>
      <pl:CredentialAttributeDesignator CredId="#creditcard" DataType="xs:string" AttributeId="http://www.banking.org/CreditCard/name"/>
     </xacml:Apply>
     <xacml:Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <pl:CredentialAttributeDesignator CredId="#eid" DataType="xs:string" AttributeId="http://www.fgov.be/eId/lastname"/>
      <pl:CredentialAttributeDesignator CredId="#creditcard" DataType="xs:string" AttributeId="http://www.banking.org/CreditCard/surname"/>
     </xacml:Apply>
    </xacml:Apply>
   </pl:Condition>

  </pl:CredentialRequirements>

  <pl:ProvisionalActions>
   <pl:ProvisionalAction ActionId="http://www.primelife.eu/ppl/RevealUnderDHP">
    <xacml:AttributeValue DataType="xs:anyURI">http://www.fgov.be/eID/address</xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:string">
     May be used for shipping, administration, statistics, and marketing purposes.
     Will be deleted within one year.
    </xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:anyURI">#eid</xacml:AttributeValue>
   </pl:ProvisionalAction>

   <pl:ProvisionalAction ActionId="http://www.primelife.eu/ppl/RevealToUnderDHP">
    <xacml:AttributeValue DataType="xs:anyURI">http://www.banking.org/CreditCard/cardnumber</xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:anyURI">http://www.ogone.com</xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:string">
     May be used for payment purposes.
     Will be deleted within one month.
    </xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:anyURI">#creditcard</xacml:AttributeValue>
   </pl:ProvisionalAction>

   <pl:ProvisionalAction ActionId="http://www.primelife.eu/ppl/RevealToUnderDHP">
    <xacml:AttributeValue DataType="xs:anyURI">http://www.banking.org/CreditCard/expirationdate</xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:anyURI">http://www.ogone.com</xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:string">
     May be used for payment purposes.
     Will be deleted within one month.
    </xacml:AttributeValue>
    <xacml:AttributeValue DataType="xs:anyURI">#creditcard</xacml:AttributeValue>
   </pl:ProvisionalAction>
  </pl:ProvisionalActions>

 </pl:Rule>

</pl:Policy>
```

Figure 4. Full example policy.

APPENDIX

In Figure 4, we give a more elaborate example of a full XACML policy enhanced with our credential-based extensions. The policy specifies the restrictions that apply to create an account at store.com. It contains a single rule that protects access to the web page `http://www.store.com/subscribe.html`. The policy states that in order to create an account, the user has to prove that she owns an electronic identity (eID) card from the Belgian government and a credit card issued by either Visa or American Express. The conditions applying to these credentials are

- the birth date on the eID card must prove that the holder is currently older than 18 years;
- the credit card must not be expired, i.e., the expiration date must be in the future;
- and the first and last names on the eID card and the credit card have to match.

Apart from satisfying those conditions, the user must reveal her address to store.com under a data handling policy that says that it will be used for the purposes of shipping, administration, statistics, and marketing, and that it will be deleted within one year. She also has to reveal her credit card number and expiration date to the payment processor ogone.com, who will use the information for payment purposes only and delete it within one month.