RZ 3818 (# Z1204-004) 04/20/2012
Computer Science 25 pages

Research Report

A Language Framework for Privacy-Preserving Attribute-Based Authentication

Jan Camenisch[1], Maria Dubovitskaya[1], Anja Lehmann[1], Gregory Neven[1], Christian Paquin[2], and Franz-Stefan Preiss[1]

[1]IBM Research Zurich
8803 Rüschlikon
Switzerland

[2]Microsoft Research

A Language Framework for Privacy-Preserving Attribute-Based Authentication

Jan Camenisch[1], Maria Dubovitskaya[1], Anja Lehmann[1],
Gregory Neven[1], Christian Paquin[2], and Franz-Stefan Preiss[1]

[1] IBM Research – Zurich
[2] Microsoft Research

Abstract. Existing cryptographic realizations of privacy-friendly authentication mechanisms such as anonymous credentials, minimal disclosure tokens, self-blindable credentials, and group signatures vary largely in the features they offer and in how these features are realized. Some features such as revocation or de-anonymization even require the combination of several cryptographic protocols. These differences and the complexity of the cryptographic protocols hinder the deployment of these mechanisms for practical applications and also make it almost impossible to switch the underlying cryptographic algorithms once the application has been designed. In this paper, we aim to bridge this gap and simplify the design and deployment of privacy-friendly authentication mechanisms. We unify the different concepts and features and define privacy-preserving attribute-based credentials (Privacy-ABCs), provide a language framework in XML schema, and give a semantics to describe the effect of the different transactions in a privacy-friendly authentication system using Privacy-ABCs. Our language framework enables application developers to use Privacy-ABCs with their different features without having to consider the specific cryptographic algorithms under the hood, similarly as they do today for digital signatures, where they do not need to worry about the particulars of the RSA and DSA algorithms either.

1 Introduction

More and more transactions in our daily life are performed electronically and the security of these transactions is an important concern. Strong authentication and according authorization based on certified attributes of the requester is paramount for protecting critical information and infrastructures online.

Most existing techniques for transfering trusted user attributes cause privacy concerns. In systems where an online identity provider creates access tokens on demand, such as SAML, OpenID, or WS-Federation, the identity provider can impersonate its users and can track a user's moves online. Systems with offline token creation, such as X.509 certificates and some WS-Trust profiles, force the user to reveal more attributes than strictly needed (as otherwise the issuer's signature cannot be verified) and make her online transactions linkable across different websites.

These drawbacks can be overcome with privacy-preserving authentication mechanisms based on advanced cryptographic primitives such as anonymous credentials, minimal disclosure tokens, self-blindable credentials, or group signatures [12, 9, 14, 17, 5, 33]. In these schemes, users obtain certified credentials for their attributes from

trusted issuers and later derive, without further assistance from any issuer, unlinkable tokens that reveal only the required attribute information yet remain verifiable under the issuer's public key. Well-known examples being Brands' scheme [9] and Camenisch-Lysyanskaya's scheme [14], which have been implemented in Microsoft's U-Prove [32] and IBM's Identity Mixer [22], respectively. Both implementations are freely available and efficient enough for practical use, yet the real-world adoption is slower than one may hope.

One possible reason for the slow adoption of privacy-preserving authentication technologies might be that the various schemes described in the literature have a large set of features where similar features are often called differently or are realized with different cryptographic mechanisms. Many of the features such as credential revocation, efficient attribute encoding, or anonymity lifting even require a combination of separate cryptographic protocols. This makes these technologies hard to understand and compare and, most importantly, very difficult to use.

To overcome this, in this paper we provide unified definitions of the concepts and features of the different privacy-preserving authentication mechanisms. We will refer to this unification as *privacy-preserving attribute-based credentials* or *Privacy-ABCs*. Our definitions abstract away from the concrete cryptographic realizations but are carefully crafted so that they can be instantiated with the different cryptographic protocols—or a combination of them. To enable the use and integration of Privacy-ABCs in authentication and authorization systems, we further present cryptography-agnostic definitions of all concepts as well as a language framework with data formats for, e.g., policies and claims. All languages are specified in XML schema and separate the abstract functionality expected from the underlying cryptographic mechanisms from the opaque containers for the cryptographic data itself. Thus, these languages allow application developers to employ Privacy-ABCs without having to think about their cryptographic realization, similarly to how digital signatures or encryption can be used today. The language described in this paper has been implemented in the ABC4Trust project (www.abc4trust.eu) and will be made available as part of a reference implementation of a Privacy-ABC system which will include a number of cryptographic solutions.

Finally, we present a formal semantics that precisely defines the meaning of our comprehensive language and their expressed features. Such a rigorous mathematical description, for instance, allows to determine whether a user can fulfill a given authentication policy with her credential portfolio or whether a derived access token satisfies a policy. As our language covers the entire Privacy-ABC system, we also provide semantics that describe the intended system behaviour, i.e., the effects of state transitions—which are steered by our language—on the different entities and their knowledge states.

2 Related work

Our work is based on the card-based access control requirements language (CARL) recently proposed Camenisch et al. [18]. CARL allows a service provider (verifier) to specify which attributes certified by whom a user needs to present in order to get access. Compared to our work, CARL defines only a small part of a Privacy-ABC system, namely the presentation policy, but does not consider how these attributes are transmit-

ted nor how credentials are issued or revoked. Bichsel et al. [6] have extended CARL to cover the transmission of certified attributes. The U-Prove token format [32] covers credential issuance and presentation but only supports selective attribute disclosure. It does not consider other features such as attribute predicates, inspection, key binding, (cryptographic) pseudonyms, or revocation.

Privacy-ABCs can be used to realize a privacy-respecting form of attribute-based access control. Traditional attribute-based access control [7, 36, 34], however, does not see attributes as grouped together in a credential or token. Thus our framework allows one to realize more specific and more precise access control policies. Also, role-based access control [20, 30] can be seen as a special case of our attribute-based setting by encoding a user's roles as attributes. Recent work [23] extended RBAC with privacy-preserving authentication for the particular case of role and location attributes.

Bonatti and Samarati [7] also propose a language for specifying access control rules based on "credentials". The language focuses on credential ownership and does not allow for more advanced requirements such as for example revealing of attributes, signing statements, or inspection. The same is true for the languages proposed by Ardagna et al. [2] and by Winsborough et al. [35]. However, the latter allows one to impose attribute properties on credentials and its extension by Li et al. [25] supports revealing of attributes. The Auth-SL language [28] focuses on multi-factor authentication and enables the policy author to specify restrictions on the properties of the authentication mechanisms themselves, but not on attributes of individual users.

The language by Ardagna et al. [1] can also be considered as a predecessor to our language in the sense that it focuses on anonymous credential systems and some of the advanced features. However, it considers only the presentation phase and is less expressive than ours, for instance, it cannot express statements involving attributes from different credentials.

VeryIDX [27] is a system to prevent identity theft by permitting the use of certain identity attributes only in combination with other identity attributes. So-called verification policies specify which attributes have to be presented together. However, these policies are introduced only conceptually without any details on exact expressivity, syntax, or semantics.

Several logic-based, technology-neutral approaches to distributed access control have been proposed [3, 4, 21, 24]. However, none of these have been designed with Privacy-ABCs in mind. In particular, they do not support selective disclosure of attributes, proving predicates over attributes, or attribute inspection.

So, our language framework is the first that covers the whole life-cycle of Privacy-ABCs and also the first one unifying the whole breadth of their features.

3 Concepts and Features

Figure 1 gives an overview of the entities involved in Privacy-ABC systems and the interactions between them. These entities are *users*, *issuers*, *verifiers*, *inspectors* and *revocation authorities*. Each issuer generates a secret issuance key and publishes the *issuer parameters* that include the corresponding public verification key. Similarly, each

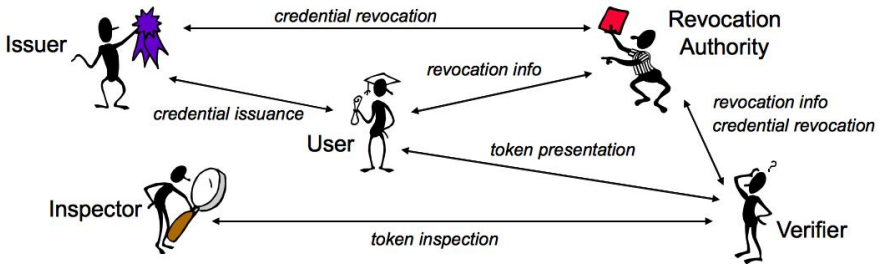

Fig. 1. Entities and interactions diagram.

inspector generates a private decryption key and a corresponding public encryption key, and each revocation authority generates and publishes its revocation parameters.

Pseudonyms. Each user generates a secret key. Unlike traditional public-key authentication schemes, however, there is no single public key corresponding to the secret key. Rather, the user can generate as many public keys as she wishes. These public keys are called *pseudonyms* in Privacy-ABCs. Pseudonyms are cryptographically unlinkable, meaning that given two different pseudonyms, one cannot tell whether they were generated from the same or from different secret keys. By generating a different pseudonym for every verifier, users can thus be known under different unlinkable pseudonyms to different sites, yet use the same secret key to authenticate to all of them.

While it is sufficient for users to generate a single secret key, they can also have multiple secret keys. A secret key can be generated by a piece of trusted hardware (e.g., a smart card) that stores and uses the key in computations (e.g., to generate pseudonyms), but that never reveals the key. The key is thereby "bound" to the hardware, in the sense that it can only be used in combination with the hardware.

There are situations, however, where the possibility to generate an unlimited number of unlinkable pseudonyms is undesirable. For example, in an online opinion poll, users should not be able to bias the result by entering multiple votes under different pseudonyms. In such situations, the verifier can request a special pseudonym called a *scope-exclusive pseudonym*, which is unique for the user's secret key and a given *scope string*. Scope-exclusive pseudonyms for different scope strings remain unlinkable, just like normal pseudonyms. By using the URL of the opinion poll as the scope string, for example, the verifier can ensure that each user can only register a single pseudonym to vote.

Credentials and Key Binding. A *credential* is a certified container of attributes issued by an issuer to a user. Formally, an *attribute* is described by the *attribute type* that determines the semantics of the attribute (e.g., first name) and the *attribute value* that determines its contents (e.g., "John"). By issuing a credential, the issuer vouches for the correctness of the contained attributes with respect to the user. The *credential specification* lists the attribute types that are encoded in a credential. A credential specification can be created by the issuer, or by an external authority so that multiple issuers can issue credentials according to the same specification. The credential specification must be published and distributed over a trusted channel. How exactly this is done goes

beyond the scope of our language framework; the specification could for example be digitally signed by its creator.

Optionally, a credential can be "bound" to a user's secret key, meaning that it cannot be used without knowing the secret key. We call this option *key binding*. It is somewhat analogous to traditional public-key certificates, where the certificate contains the CA's signature on the user's public key, but unlike traditional public-key certificates, a Privacy-ABC is not bound to a unique public key: it is only bound to a unique secret key. A user can derive as many pseudonyms as she wishes from this secret key and (optionally) show that they were derived from the same secret key that underlies the credential.

Presentation. To authenticate to a verifier, the user first obtains the *presentation policy* that describes which credentials the user must present and which information from these credentials she must reveal. If the user possesses the necessary credentials, she can derive from these credentials a *presentation token* that satisfies the presentation policy. The validity of a presentation token can be checked using the issuer parameters of all credentials underlying the presentation token.

Presentation tokens derived from Privacy-ABCs only reveal the attributes that were explicitly requested by the presentation policy – all the other attributes contained in the credentials remain hidden. Moreover, presentation tokens are cryptographically unlinkable (meaning no collusion of issuers and verifiers can tell whether two presentation tokens were generated by the same user or by different users) and untraceable (meaning that no such collusion can correlate a presentation token to the issuance of the underlying credentials). Of course, presentation tokens are only as unlinkable as the information they intentionally reveal.

Rather than requesting and revealing full attribute values, presentation policies and tokens can also request and reveal *predicates* over one or more issued attributes. For example, a token could reveal that the name on the user's credit card matches that on her driver's license, without revealing the name. As another example, a token could reveal that the user's birthdate is before January 1st, 1994, without revealing her exact birthdate.

Issuance. In the simplest setting, an issuer knows all attribute values to be issued and simply embeds them into a credential. Privacy-ABCs also support advanced issuance features where attributes are blindly "carried over" from existing credentials, without the issuer becoming privy to their values. Similarly, the issuer can blindly issue self-claimed attribute values (i.e., not certified by an existing credential), carry over the secret key to which a credential is bound, or assign a uniformly random value to an attribute such that the issuer cannot see it and the user cannot bias it.

Advanced issuance is an interactive protocol between the user and the issuer. In the first move, the issuer provides the user with an *issuance policy* that consists of a presentation policy specifying which pseudonyms and/or existing credentials the user must present, and of a *credential template* specifying which attributes or secret keys of the newly issued credential will be generated at random or carried over from credentials or pseudonyms in the presentation policy. In response, the user sends an *issuance token* containing a presentation token that satisfies the issuance policy. Then the (pos-

sibly multi-round) cryptographic issuance protocol ensues, at the end of which the user obtains the new credential.

Inspection. Absolute user anonymity in online services easily leads to abuses such as spam, harassment, or fraud. Privacy-ABCs provide the option to add accountability for misbehaving users through a feature called *inspection*. Here, a presentation token contains one or more credential attributes that are encrypted under the public key of a trusted *inspector*. The verifier can check that the correct attribute values were encrypted, but cannot see their actual values. The *inspection grounds* describe the circumstances under which the verifier may call upon the inspector to recover the actual attribute values. The inspector is trusted to collaborate only when the inspection grounds have been met; verifiers cannot change the inspection grounds after receiving a presentation token, as the grounds are cryptographically tied to the token.

The presentation policy specifies which attributes from which credentials have to be encrypted, together with the inspector public keys and inspection grounds they have to be encrypted.

Revocation. Credentials may need to be revoked for several reasons: the credential and the related user secrets may have been compromised, the user may have lost her right to carry a credential, or some of her attribute values may have changed. In such cases, credentials need to be revoked globally and we call this *issuer-driven revocation*. Sometimes credentials may be revoked only for specific contexts. For example, a hooligan may see his digital identity card revoked for accessing sport stadiums, but may still use it for all other purposes. We call this *verifier-driven revocation*.

Revocation for Privacy-ABCs is cryptographically more complicated than for classical certificates, but many efficient mechanisms exist [16, 29, 8, 13, 26]. Bar a few exceptions, all of them can be used for both issuer-driven and verifier-driven revocation.

We describe revocation in a generic mechanism-agnostic way and consider credentials to be revoked by dedicated *revocation authorities*. They are separate entities in general, but may be under the control of the issuer or verifier in particular settings. The revocation authority publishes static *revocation authority parameters* and periodically publishes the most recent *revocation information*. When creating presentation tokens, users prove that their credentials have not been revoked, possibly using *non-revocation evidence* that they fetch and update from the revocation authority. The revocation authority to be used is specified in the issuer parameters for issuer-driven revocation and in the presentation policy for verifier-driven revocation. When a credential is subject to issuer-driven revocation, a presentation token related to this credential must always contain a proof that the presented credential has not been revoked. Issuer-driven revocation is performed based on the *revocation handle*, which is a dedicated unique attribute embedded in a credential. Verifier-driven revocation can be performed based on any combination of attribute values, possibly even from different credentials. This allows the revocation authority for example to exclude certain combinations of first names and last names to be used in a presentation token.

4 Language Framework

Given the multitude of distributed entities involved in a full-fledged Privacy-ABC system, the communication formats that are used between these entities must be specified and standardized.

None of the existing format standards for identity management protocols such as SAML, WS-Trust, or OpenID support all Privacy-ABCs' features. Although, most of them can be extended to support a subset of these features, we define for the sake of simplicity and completeness a dedicated language framework which addresses all unique Privacy-ABC features. Our languages can be integrated into existing identity management systems.

In this section we introduce our framework covering the full life-cycle of Privacy-ABCs, including setup, issuance, presentation, revocation, and inspection. As the main purpose of our data artifacts is to be processed and generated by automated policy and credential handling mechanisms, we define all artifacts in XML schema notation, although one could also create a profile using a different encoding such as ASN.1 or JSON.

The XML artifacts formally describe and orchestrate the underlying cryptographic mechanisms and provide opaque containers for carrying the cryptographic data. Whenever appropriate, our formats also support user-friendly textual names or descriptions which allow to show a descriptive version of the XML artifacts to a user and to involve her in the issuance or presentation process if necessary.

For didactic purposes we describe the different artifacts realizing the concepts from Section 3 by means of examples. The full schema is available in [11]. In what follows, we explicitly distinguish between user attributes (as contained in a credential) and XML attributes (as defined by XML schema) whenever they could be confused.

4.1 Credential Specification

Recall that the credential specification describes the common structure and possible features of credentials. For example, suppose the Republic of Utopia issues electronic identity cards to its citizens containing their full name, state, and date of birth. Utopia may issue Privacy-ABCs according to the credential specification shown in Figure 2.

```
01 <CredentialSpecification KeyBinding="true" Revocable="true">
02   <SpecificationUID>urn:creds:id</SpecificationUID>
03   <AttributeDescriptions MaxLength="32">
04     <AttributeDescription Type="urn:creds:id:name" DataType="xs:string" Encoding="xenc:sha256">
05       <FriendlyAttributeName lang="EN"> Full Name </FriendlyAttributeName>
06     </AttributeDescription>
07     <AttributeDescription Type="urn:creds:id:state" DataType="xs:string" Encoding="xenc:sha256"/>
08     <AttributeDescription Type="urn:creds:id:bdate" DataType="xs:date" Encoding="ASN1:GeneralizedTime"/>
09   </AttributeDescriptions>
10 </CredentialSpecification>
```

Fig. 2. Credential specification of the electronic identity card.

The XML attribute **KeyBinding** indicates whether credentials adhering to this specification must be bound to a secret key. The XML attribute **Revocable** being set to *"true"* indicates that the credentials will be subject to issuer-driven revocation and hence have

a built-in revocation handle. The assigned revocation authority is specified in the issuer parameters.

To encode user attribute values in a Privacy-ABC, they must be mapped to integers of a limited length. The maximal length is indicated by the **MaxLength** XML attribute (Line 3), here 32 bytes. Electronic identity cards contain a person's full name, state, and date of birth. The XML attributes **Type**, **DataType**, and **Encoding** respectively contain the unique identifier for the user attribute type, for the data type, and for the encoding algorithm that specifies how the value is to be mapped to an integer of the correct size (Lines 4,7,8). Attributes that may have values longer than **MaxLength** have to be hashed, as is done here for the name using SHA-256. The specification can also define human-readable names for the user attributes in different languages (Line 5).

4.2 Issuer and Revocation Authority Parameters

The government of Utopia, that acts as issuer and revocation authority for the identity cards, generates an issuance key pair and publishes the issuer parameters, and generates and publishes the revocation authority parameters. They are illustrated in Figures 6 and 7 given in Appendix A. The issuer parameters contain a unique identifier, the cryptographic Privacy-ABC mechanism, the hash algorithm, the credential specification for the credentials that will be issued, and the identifier of the revocation authority parameters that will manage the issuer-driven revocation.

The revocation authority parameters can be used for both issuer- and verifier-driven revocation. They specify a unique identifier for the parameters, the cryptographic revocation mechanisms, and references to the network endpoints where the most recent revocation information and non-revocation evidence can be fetched.

4.3 Presentation Policy with Basic Features

Suppose that a user already possesses an identity card from the Republic of Utopia issued according to the above credential specification. Suppose further that all residents of Utopia can sign up for one free library card using an online issuance service. To get a library card the applicant must present her valid identity card and reveal only the state from it. This results in the presentation policy depicted in Figure 3.

```
01 <PresentationPolicy PolicyUID="libcard">
02   <Message>
03     <Nonce> bkQydHBQWDR4TUZzbXJKYUM= </Nonce>
04   </Message>
05   <Pseudonym Alias="nym"  Scope="urn:library:issuance" Exclusive="true"/>
06   <Credential Alias="id" SameKeyBindingAs="nym">
07     <CredentialSpecAlternatives>
08       <CredentialSpecUID>urn:creds:id</CredentialSpecUID>
09     </CredentialSpecAlternatives>
10     <IssuerAlternatives>
11       <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
12     </IssuerAlternatives>
13     <DisclosedAttribute AttributeType= "urn:creds:id:state"/>
14   </Credential>
15 </PresentationPolicy>
```

Fig. 3. Presentation policy for an identity card.

We now will go through the above presentation policy and describe how the different features of Privacy-ABCs can be realized with our language. We will first focus on the basic features and describe extended concepts such as inspection and revocation in our second example.

Signing messages. A presentation token can optionally sign a message. The message to be signed is specified in the policy (Fig. 3, Lines 2-4). It can include a nonce (to prevent replay attacks and to use for cryptographic evidence generation), any application-specific message, and a human-readable name and/or description of the policy.

Pseudonyms. The optional **Pseudonym** element (Fig. 3, Line 5) indicates that the presentation token must contain a pseudonym. A pseudonym can be presented by itself or in relation with a credential if key binding is used (which we discuss later).

The XML attribute **Exclusive** in the example above indicates that a scope-exclusive pseudonym must be created, with the scope string given by the XML attribute **Scope**. This ensures that each user can create only a single pseudonym satisfying this policy, so that the registration service can prevent the same user from obtaining multiple library cards. Setting **Exclusive** to *"false"* would allow an ordinary pseudonym to be presented. The **Pseudonym** element has an optional boolean XML attribute **Established**, not illustrated in the example, which, when set to *"true"*, requires the user to re-authenticate under a previously established pseudonym. The presentation policy can request multiple pseudonyms, e.g., in order to verify that different pseudonyms actually belong to the same user.

Selective Disclosure. For each credential that the user is requested to present, the policy contains a **Credential** element (Fig. 3, Lines 6-14). The XML attribute **Alias** assigns an alias by means of which the credential can be referred to from other places in the policy, e.g., from the attribute predicates. For each credential, a list of accepted credential specifications and issuer parameters can be specified, as well as the list of attributes that must be disclosed. The above policy only requests the presentation of an identity card and the exposure of the state by the **DisclosedAttribute** element (Fig. 3, Line 13).

Key Binding. If present, the **SameKeyBindingAs** attribute of a **Credential** or **Pseudonym** element (Fig. 3, Line 6), contains an alias referring either to another Pseudonym element within this policy, or to a Credential element for a credential with key binding. This indicates that the current pseudonym or credential and the referred pseudonym or credential have to be bound to the same key. In our example above, the policy requests that the identity card and the presented pseudonym must belong to the same secret key.

4.4 Issuance Policy

To support the advanced features described in Section 3, we propose a dedicated *issuance policy*. A library card contains the applicant's name and is bound to the same secret key as the identity card. So the identity card must not only be presented, but also used as a source to carry over the name and the secret key to the library card, and the library should learn neither of them during the issuance process. Altogether, to issue library cards the state library creates an issuance policy depicted in Figure 4. It contains the presentation policy from Figure 3 and the credential template that we describe in details below.

```
01 <IssuancePolicy>
02  <PresentationPolicy PolicyUID="libcard">... </PresentationPolicy>
03  <CredentialTemplate SameKeyBindingAs="id">
04    <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
05    <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
06    <UnknownAttributes>
07      <CarriedOverAttribute TargetAttributeType= "urn:utopia:lib:name">
08        <SourceCredentialInfo Alias="id" AttributeType="urn:creds:id:name"/>
09      </CarriedOverAttribute>
10    </UnknownAttributes>
11  </CredentialTemplate>
12 </IssuancePolicy>
```

Fig. 4. Issuance policy for a library card. The presentation policy on Line 2 is depicted in Figure 3.

Credential Template. A credential template describes the relation of the new credential to the existing credentials that were requested in the presentation policy. The credential template (Fig. 4, Lines 3-11) must first state the unique identifier of the credential specification and issuer parameters of the newly issued credential. The optional XML attribute **SameKeyBindingAs** further specifies that the new credential will be bound to the same secret key as a credential or pseudonym in the presentation policy, in this case the identity card.

Within the **UnknownAttributes** (Fig. 4, Lines 6-10) it further specifies which user attributes of the new credential will be carried over from existing credentials in the presentation token. The **SourceCredentialInfo** element (Fig. 4, Line 8) indicates the credential and the user attribute of which the value will be carried over.

Not illustrated in the above example, an attribute value can also be specified to be chosen jointly at random by the issuer and the user. This is achieved by setting the optional XML attribute **JointlyRandom** to *"true"*.

4.5 Presentation and Issuance Token

A *presentation token* consists of the *presentation token description*, containing the mechanism-agnostic description of the revealed information, and the *cryptographic evidence*, containing opaque values from the specific cryptography that "implements" the token description. The presentation token description roughly uses the same syntax as the presentation policy above. An *issuance token* is a special presentation token that satisfies the stated presentation policy, but that contains additional cryptographic information required by the credential template.

The main difference with the presentation and issuance policy is that in the returned token the **Pseudonym** and the **DisclosedAttribute** elements (if requested in the policy) now also contain the concrete values **PseudonymValue** and **AttributeValue** as child elements. Finally, all data from the cryptographic implementation of the presentation token and the advanced issuance features are grouped together in the **CryptoEvidence** element. The issuance token that would be generated in response to the issuance policy depicted in Figure 4 is given in Figure 8 in the Appendix A.

4.6 Presentation Policy with Extended Features

Suppose that the state library has a privacy-friendly online interface for borrowing digital and paper books. Books can be browsed and borrowed anonymously using the digital

library cards based on Privacy-ABCs. Paper books can be delivered in anonymous numbered mailboxes at the post office. However, when books are returned late or damaged, the library must be able to identify the reader to impose an appropriate fine. Recidivist negligence may even lead to exclusion from borrowing further paper books, but borrowing digital books remains possible.

Moreover, assume that the library has special conditions for young readers that can be used by anyone below the age of twenty-six. As library cards do not contain a date of birth, a user must prove to be below that age by combining her library card with her identity card. Altogether, for borrowing books under the "young-reader"-conditions, users have to satisfy the presentation policy depicted in Figure 5.

```
01 <PresentationPolicyAlternatives>
02   <PresentationPolicy PolicyUID= "young-reader">
03     <Message>...</Message>
04     <Credential Alias="libcard" SameKeyBindingAs="id">
05       <CredentialSpecAlternatives>
06         <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
07       </CredentialSpecAlternatives>
08       <IssuerAlternatives>
09         <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
10       </IssuerAlternatives>
11       <DisclosedAttribute  AttributeType= "urn:utopia:lib:name">
12         <InspectorAlternatives>
13           <InspectorPublicKeyUID> urn:lib:arbitrator </InspectorPublicKeyUID>
14         </InspectorAlternatives>
15         <InspectionGrounds> Late return or damage. </InspectionGrounds>
16       </DisclosedAttribute>
17     </Credential>
18     <Credential Alias="id">
19       <CredentialSpecAlternatives>
20         <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
21       </CredentialSpecAlternatives>
22       <IssuerAlternatives>
23         <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
24       </IssuerAlternatives>
25     </Credential>
26     <VerifierDrivenRevocation>
27       <RevocationParametersUID> urn:lib:blacklist </RevocationParametersUID>
28       <Attribute CredentialAlias ="libcard" AttributeType= "urn:utopia:lib:name"/>
29     </VerifierDrivenRevocation>
30     <AttributePredicate Function= "...:date-greater-than" >
31       <Attribute CredentialAlias ="id" AttributeType= "urn:creds:id:bdate"/>
32       <ConstantValue>1986-04-10 </ConstantValue>
33     </AttributePredicate>
34   </PresentationPolicy>
35 <PresentationPolicyAlternatives>
```

Fig. 5. Presentation policy for borrowing books.

The presentation policy, when used for plain presentation (i.e., not within an issuance policy) can consist of multiple policy alternatives, each wrapped in a separate **PresentationPolicy** element (Fig. 5, Lines 2-34). The returned presentation token must satisfy (at least) one of the specified policies.

The example presentation policy contains two **Credential** elements, for the library and for the identity card, which must belong to the same secret key as indicated by the XML attribute **SameKeyBindingAs**.

Attribute Predicates. No user attributes of the identity card have to be revealed, but the **AttributePredicate** element (Fig. 5, Lines 30-33) specifies that the date of birth must

be after April 10th, 1986, i.e., that the reader is younger than twenty-six. Supported predicate functions include equality, inequality, greater-than and less-than tests for most basic data types, as well as membership of a list of values. The arguments of the predicate function may be credential attributes (referred to by the credential alias and the attribute type) or constant values.

Inspection. To be able to nevertheless reveal the name of an anonymous borrower and to impose a fine when a book is returned late or damaged, the library can make use of inspection. The **DisclosedAttribute** element for the user attribute *"...:name"* contains **InspectorPublicKeyUID** and **InspectionGrounds** child elements, indicating that the attribute value must not be disclosed to the verifier, but to the specified inspector with the specified inspection grounds. The former child element specifies the inspector's public key under which the value must be encrypted, in this case belonging to a designated arbiter within the library. The latter element specifies the circumstances under which the attribute value may be revealed by the arbiter. Our language also provides a data artifact for inspection public keys, which we omit here for space reasons.

Issuer-Driven Revocation. When the presentation policy requests a credential that is subject to issuer-driven revocation, the credential must be proven to be valid with respect to the most recent revocation information. However, a policy can also specify a particular version of the revocation information to use. In the latter case, the element **IssuerParametersUID** has an extra XML attribute **RevocationInformationUID** containing the identifier of the specific revocation information. The specification of the referenced **RevocationInformation** is given in [11].

Verifier-Driven Revocation. If customers return borrowed books late or damaged, they will be excluded from borrowing further paper books, but they are still allowed to use the library's online services. In our example above, this is handled by the **VerifierDrivenRevocation** element (Fig. 5, Lines 26-29), which specifies that the user attribute *"...:name"* of the library card must be checked against the most recent revocation information from the revocation authority *"urn:lib:blacklist"*. Revocation can also be based on a combination of user attributes from different credentials, in which case there will be multiple **Attribute** child elements per **VerifierDrivenRevocation**. The presentation policy can also contain multiple **VerifierDrivenRevocation** elements for one or several credentials, the returned presentation token must then prove its non-revoked status for *all* of them.

5 Semantics

To precisely specify the meaning of the XML language that we propose, we provide a formal semantics which mathematically defines the various language features and thus the intended system behavior. The semantics that we propose is based on the one shown by Camenisch et al. [18] for the CARL language. Similar to their approach, we also define our semantics in the context of a state transition system, that assumes state transitions for credential issuance, credential revocation, token presentation and token inspection. However, rather than elaborating only on one of the transitions while disregarding the aspects of the other ones, our semantics precisely describes the meaning

for all the possible transitions including the concepts of pseudonyms and key binding. In our model, users maintain credential portfolios containing the credentials that were issued to them. Issuers and verifiers maintain knowledge states about their communication partners in the form of logical predicates. These predicates express that a certain logical expression over the communication partner's credentials has been successfully verified at the time of the transition. For modelling our system, we assume to have a global view on the credential portfolios and knowledge states of the parties in the system. We define the semantics in multiple steps. First, we introduce a set of basic dedicated functions and attributes which are necessary for modeling revocation, pseudonyms, and inspection. Then, we define the meaning of attribute predicates as they form the basis of our knowledge states. After doing so, the semantics of credential revocation and basic credential issuance is defined. On the basis of those concepts, we elaborate on the meaning of token presentation and policy verification. Having done so, we close our semantical definition by describing token inspection transitions as well as advanced issuance scenarios. Similar to the proposal of Camenisch et al. [18], the semantics is given in terms of the effects that the transitions have on the knowledge states and credential portfolios of the parties involved in the transition, rather than describing *how* they are performed. Whenever possible we preserved the notation introduced for CARL [18].

5.1 Types and Functions

We assume as given an ontology $\mathfrak{T}$ that defines *data types*, *attributes*, *credential types*, and functions on the data types, which correspond to the data types, attribute types, credential specifications and predicate functions from the XML notation. The parties involved in a certain transaction are assumed to use the same ontology for performing this transaction. First, the ontology defines a set of data types $\beta_1, \ldots, \beta_{n_\beta}$. These include, e.g., $String$, $Bool$, Int, $Date$ and URI. We denote with $[\![\beta]\!]$ the extension of a data type β, i.e., the set of constants of type β. Further, the ontology specifies a set of credential types $\tau_1, \ldots, \tau_{n_\tau}$. Every credential type τ is defined by a set of attributes with their types $A_\tau = \{a_1 :: \beta_1, \ldots, a_l :: \beta_l\}$, where we denote with $a :: \beta$ an attribute a that has data type β. Subsequently, a credential of type τ is represented as a function from A_τ to values of the respective data type. For example, a credential with attributes $\{a_1 :: \beta_1, a_2 :: \beta_2\}$ is a function f with domain $\{a_1, a_2\}$ such that $f(a_1) \in [\![\beta_1]\!]$ and $f(a_2) \in [\![\beta_2]\!]$. We denote with $f :: \tau$ that a credential f has data type τ. Finally, the ontology defines a set of functions on the data types. We assume to have at least the equivalence relation $==_\beta$ on every data type β. We also assume a total order on both types Int and $Date$.

5.2 Dedicated Functions and Attributes

In the following, we describe a set of dedicated functions, which are used for specifying the semantics associated with pseudonyms and inspection, as well as a set of dedicated attributes used for modeling pseudonyms and revocation. To do so, we assume every party (issuers, verifiers, etc.) is associated with a private-public key pair, and we assume a non-deterministic function $k \leftarrow$ *genKey*, which generates a new unique secret key k

every time it is called. We denote the set of keys that user U generated with $\mathfrak{S}_U$ and assume all user's key sets are disjoint.

Further, we assume a non-deterministic function $p \leftarrow \mathit{nym}(k)$, which derives a new unique *pseudonym* value p from secret key k every time it is called, a boolean function $\mathit{verifyNym}(p, k)$ that is $true$ iff p was derived from k. Pseudonyms derived from the same secret key are in principle unlinkable, i.e., for two pseudonyms $p_1 = \mathit{nym}(k)$ and $p_2 = \mathit{nym}(k')$, it cannot be determined whether $k = k'$. Also, we assume a function $d \leftarrow \mathit{seNym}(k, w)$, which derives a unique scope-exclusive pseudonym d from a secret key k for scope w, as well as a boolean function $\mathit{verifySeNym}(d, w, k)$ that is $true$ iff d was derived from k for w. Scope-exclusive pseudonyms are also unlinkable, i.e. for two pseudonyms $d_1 \leftarrow \mathit{seNym}(k, w)$ and $d_2 \leftarrow \mathit{seNym}(k', w')$ with $w \neq w'$, it cannot be determined whether $k = k'$. We denote with $\mathfrak{N}_U$ and $\mathfrak{D}_U$ the pseudonyms and scope-exclusive pseudonyms that were derived from a key $k \in \mathfrak{S}_U$. Additionally, we assume a boolean function $\mathit{verifyNyms}(n, n')$ that takes two—scope-exclusive or non-scope-exclusive—pseudonyms n and n' as parameters and that is $true$ iff both pseudonyms are derived from the same secret key, i.e., iff $\exists U, w, w' \cdot \exists k \in \mathfrak{S}_U \cdot \big(\mathit{verifyNym}(n,k) \wedge \mathit{verifyNym}(n',k)\big) \vee \big(\mathit{verifyNym}(n,k) \wedge \mathit{verifySeNym}(n',w',k)\big) \vee \big(\mathit{verifySeNym}(n,w,k) \wedge \mathit{verifyNym}(n',k)\big) \vee \big(\mathit{verifySeNym}(n,w,k) \wedge \mathit{verifySeNym}(n',w',k)\big)$.

To formalize token inspection, we further assume an encryption function $e \leftarrow \mathit{vEncrypt}(S, g, v_1, v_2, \ldots)$, which encrypts a concatenation of values $v_1, v_2, \ldots \in [\![\beta_1]\!] \cup \cdots \cup [\![\beta_{n_\beta}]\!]$ under the public key of party S and inspection grounds g, and a function $\langle v_1, v_2, \ldots \rangle \leftarrow \mathit{vDecrypt}(e, sk, g')$, which decrypts e with the secret key sk on grounds g' to the originally encrypted concatenation $\langle v_1, v_2, \ldots \rangle$ iff sk is the secret key of S and $g = g'$. Further, we assume a boolean function $\mathit{verifyEnc}(e, S, g, v_1, v_2, \ldots)$, which evaluates to $true$ iff $e = \mathit{vEncrypt}(S, g, v_1, v_2, \ldots)$.

All credentials have the dedicated attributes $type$, $issuer$, and $kBound$, where the latter is a boolean attribute that is $true$ for credentials with key binding, i.e., for which the credential specification has XML attribute **KeyBinding**="*true*", and $false$ for credentials without key binding. Credentials with key binding additionally have an attribute $uKey$ of type $String$ containing the secret key of its owner. Revocable credentials have a dedicated attribute $rHandle$ of type $String$ representing the revocation handle.

5.3 Attribute Predicate Semantics

An **AttributePredicate** element represents a logical predicate $\phi = f(\ldots)$, where f is a boolean-valued function over *attribute variables* and constants, which are expressed by **Attribute** and **ConstantValue** XML elements, respectively. We denote attribute variables with $C.a$ for a credential alias C and an attribute a, which are reflected by the **CredentialAlias** and **AttributeType** XML attributes of the **Attribute** XML element.

We define the meaning of attribute predicates with respect to a given ontology $\mathfrak{T}$ and an interpretation $\mathcal{I}$. An interpretation assigns values to the symbols (e.g., f, $C.a$, etc.) in a formula so that the formula can be evaluated. In particular, for a function symbol f, we denote by $f^{\mathfrak{T}}$ the ontology-defined meaning of f, i.e., the function defined for this symbol by ontology $\mathfrak{T}$. For terms $t_1, \ldots, t_n$, where a term is an attribute variable or a constant, we define $f(t_1, \ldots, t_n)^{\mathcal{I}} = f^{\mathfrak{T}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$ and $(C.a)^{\mathcal{I}} = (C^{\mathcal{I}})(a)$. See

Section 5.6 for more details on $C^{\mathcal{I}}$. If an interpretation $\mathcal{I}$ assigns the value *true* to a predicate ϕ, it is called a model of that predicate, which we denote as $\mathcal{I} \models \phi$. Multiple predicates $\phi_1, \ldots, \phi_n$ within the same **PresentationPolicy** element are conjunctively combined to $\phi = \phi_1 \wedge \ldots \wedge \phi_n$.With respect to logical connectives (e.g., $\wedge$, $\vee$, $\neg$, $\rightarrow$), the satisfaction relation $\models$ is defined as:

$$\mathcal{I} \models \phi_1 \wedge \phi_2 \text{ iff } \mathcal{I} \models \phi_1 \text{ and } \mathcal{I} \models \phi_2 \;\; ; \;\; \mathcal{I} \models \neg\phi \text{ iff } \mathcal{I} \not\models \phi$$

Other constructs of the predicate are the usual abbreviations, e.g., $\phi_1 \vee \phi_2$ is short for $\neg(\neg\phi_1 \wedge \neg\phi_2)$.

5.4 Credential Issuance

We first consider simple issuance policies whose credential template merely specifies a credential type τ and an issuer I, i.e., issuance policies without a **PresentationPolicy** part and without an **UnknownAttributes** part. After introducing the semantics for token presentation (cf. Section 5.6), we extend the issuance semantics for unrestricted issuance policies in Section 5.9. An issuance transaction between a user U and an issuer I has effects on the user's credential portfolio as well as the issuer's knowledge state.

Let $\mathfrak{P}_U$ be the credential portfolio of U, i.e., the set of credentials that U owns, and $\mathfrak{P}'_U$ the augmented portfolio of U after the issuance transition. The user's portfolio is augmented with a new credential N such that $\mathfrak{P}'_U = \mathfrak{P}_U \cup \{N\}$ where N has the following properties: $N(type) = \tau$, $N(issuer) = I$, and $N(a_i) = A_i$ for $1 \leq i \leq |A_\tau|$ where the A_i are the attribute values that I certifies for U. If τ is revocable, then $N(rHandle) = r$, where r is a globally unique issuer-chosen revocation handle.

Let $\mathfrak{K}_I(X)$ be an *attribute predicate* expressing the knowledge of issuer I about the credential portfolio of party X, where X is some issuer-chosen identifier under which U is currently known to I, and $\mathfrak{K}'_I(X)$ be the knowledge state after the transition. Note that the same user U may be known to the issuer under multiple different identifiers. The knowledge of I increases such that: $\mathfrak{K}'_I(X) = \mathfrak{K}_I(X) \wedge N.type == \tau \wedge N.issuer == I \wedge \wedge_{i=1}^{|A_\tau|} N.a_i == A_i$. If τ is revocable, then $\mathfrak{K}'_I(X)$ has the additional conjunct $N.rHandle == r$. This models that for simple issuance policies the issuer learns the values of *all* the attributes that are certified in the newly issued credential N. For advanced issuance policies (which we cover in 5.9), however, this is different as then the issuer's knowledge after the issuance transition may contain only a subset of the attributes certified in the new credentials.

5.5 Credential Revocation

As we will specify later, presentation tokens can only be proven if the involved credentials are not revoked. In order to define the semantics of revocation, we first introduce revocation epochs and revocation logs. Note that these concepts are only part of our ideal world that is modeled, i.e., typically they do not have counterparts in a real implementation. We assume revocation authorities maintain revocation epochs in the form of sequence numbers, where a function $h \leftarrow \mathit{currEpoch}(Y)$ returns the current epoch $h \in \mathbb{Z}$ for authority Y. Revocation authorities advance their current epoch at their own

discretion, e.g., at regular time intervals or whenever a new credential is revoked, from $h \in \mathbb{Z}$ to $h' \in \mathbb{Z}$ such that $h' > h$.

In our model, issuers act as revocation authorities of the credentials they issued. Every issuer I maintains a local *revocation log file* $\mathfrak{R}_I$. Each log file is a (possibly empty) set of pairs (h, r) of an epoch $h \in \mathbb{Z}$ and a revocation handle r, representing that the credential with handle r was revoked in epoch h. An issuer adding an entry to his revocation log models an issuer-revocation transition in our transition system. A *credential C is* not issuer-revoked *for epoch h* iff $\nexists\big(h', C(rHandle)\big) \in \mathfrak{R}_{C(issuer)} \cdot h' \leq h$. A boolean predicate *notIssRevoked*(I, h, r) reflects this fact by stating whether issuer I considers a credential with handle r as valid, i.e., not revoked, in epoch h.

Similar to issuer-driven revocation, for modelling verifier-driven revocation we assume every revocation authority V maintains a local revocation log file $\mathfrak{L}_V$, which is a (possibly empty) set of tuples $(h, v_1, v_2, \ldots)$ of an epoch $h \in \mathbb{Z}$ and an arbitrary length of values $v_1, v_2, \ldots \in [\![\beta_1]\!] \cup \cdots \cup [\![\beta_{n_\beta}]\!]$. The predicate *notVerRevoked*$(V, h, v_1, v_2, \ldots)$ reflects whether the tuple $(v_1, v_2, \ldots)$ is not verifier-revoked by revocation authority V for epoch h, and is $true$ iff $\nexists(h', v_1, v_2 \ldots) \in \mathfrak{L}_V \cdot h' \leq h$.

5.6 Token Presentation

In this section we define the semantics of a token presentation transition, which models that a user U proves to a verifier S that she fulfills the statements made in a given presentation token. We first formalize the requirements on the user's portfolio such that it can be used to prove a token, and then we specify the knowledge increase of the verifier to whom the token is proved.

Let T be a presentation token that, from an abstract point of view, i.e. without considering its form in XML notation, contains the following information[3]:

- $\underline{k}$ pseudonym values $\underline{p}_1, \ldots, \underline{p}_{\underline{k}}$
- $\underline{\ell}$ scope-excl. pseudonym values $\underline{d}_1, \ldots, \underline{d}_{\underline{\ell}}$ for scopes $\underline{w}_{d_1}, \ldots, \underline{w}_{d_\ell}$
- $\underline{n}$ credentials $\underline{C}_1 :: \underline{\tau}_1, \ldots, \underline{C}_{\underline{n}} :: \underline{\tau}_{\underline{n}}$ of issuers $\underline{I}_{C_1}, \ldots, \underline{I}_{C_n}$ with reference revocation epochs $\underline{h}_{C_1}, \ldots, \underline{h}_{C_n}$
- Attribute predicate $\underline{\phi}$
- Disclosed values $\underline{v}_{(i,S)}$ for the attributes $\underline{C.a}_{(i,S)}$, for $1 \leq i \leq \underline{n}_S$ where $\underline{n}_S$ is the number of attributes disclosed to S
- Attributes $\underline{C.a}_{(i,S_j)}$ that are verifiably encrypted for inspector $\underline{S}_j$ in ciphertext $\underline{e}_{S_j}$ on grounds $\underline{g}_{S_j}$ for $1 \leq i \leq \underline{n}_{S_j}$ and $1 \leq j \leq \underline{q}$, where $\underline{n}_{S_j}$ is the number of attributes disclosed to $\underline{S}_j$ and $\underline{q}$ is the number of different inspectors
- Non-revoked attributes $\underline{C.a}_{(i,V_j)}$ at revocation authority $\underline{V}_j$ with reference revocation epochs $\underline{h}_{V_j}$ for $1 \leq i \leq \underline{n}_{V_j}$ and $1 \leq j \leq \underline{o}$ where $\underline{n}_{V_j}$ is the number of revocation attributes of $\underline{V}_j$ and $\underline{o}$ is the number of revocation authorities
- Message $\underline{m}$
- Symmetric key binding relation $\underline{K} \subseteq \underline{R} \times \underline{R}$, with $\underline{R} = \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \cup \{\underline{p}_1, \ldots, \underline{p}_{\underline{k}}\} \cup \{\underline{d}_1, \ldots, \underline{d}_{\underline{\ell}}\}$, where $x \mapsto y \in \underline{K}$ states that x is key-bound to y.

[3] Note that we underline variables to express that they are associated with a presentation token.

As a credential's type and issuer are treated as dedicated attributes, for each credential $C :: \tau \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\}$ issued by I we let the predicate $\underline{\phi}$ contain an additional conjunct $C.type == \tau \wedge C.issuer == I$. Further, for all pairs of credentials $C, C' \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\}$ that are bound to the same key, i.e., $C \mapsto C' \in \underline{K}$, we let the predicate $\underline{\phi}$ contains a conjunct $C.uKey == C'.uKey$. Note that there is only one inspection ground per inspector in our model, while in the XML notation there is one ground per attribute to disclose. This is because in our model the attributes disclosed to one inspector are *packaged* into one ciphertext to capture their semantic correlation. Without preserving this correlation, dishonest verifiers might mix attributes disclosed within different ciphertexts or across different token presentations.

We say that U *can prove the presentation token* T iff there exists a credential assignment b and an interpretation $\mathcal{I}$ such that

1. b is a total mapping from $\{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\}$ to $\mathfrak{P}_U$,
2. $\mathcal{I} \models \underline{\phi}$, with $\underline{C}_i^{\mathcal{I}} = b(\underline{C}_i)$ for $1 \leq i \leq n$,
3. $\forall C \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \cdot \mathit{notIssRevoked}\big(\underline{I}_C, \underline{h}_C, (C.rHandle)^{\mathcal{I}}\big)$,
4. $\forall V \in \{\underline{V}_1, \ldots, \underline{V}_{\underline{o}}\} \cdot \mathit{notVerRevoked}(V, \underline{h}_V, (\underline{C.a}_{(1,V)})^{\mathcal{I}}, \ldots, (\underline{C.a}_{(\underline{n}_V,V)})^{\mathcal{I}})$,
5. $\forall i \in \{1, \ldots, \underline{n}_S\} \cdot (\underline{C.a}_{(i,S)})^{\mathcal{I}} = \underline{v}_{(i,S)}$, i.e., the sent values are correct w.r.t. the credential assignment b,
6. $\forall x \in \{\underline{S}_1, \ldots, \underline{S}_{\underline{q}}\} \cdot \mathit{verifyEnc}\big(\underline{e}_x, x, \underline{g}_x, (\underline{C.a}_{(1,x)})^{\mathcal{I}}, \ldots, (\underline{C.a}_{(\underline{n}_x,x)})^{\mathcal{I}}\big)$,
7. $\forall i \in \{1, \ldots, \underline{k}\} \cdot \underline{p}_i \in \mathfrak{N}_U \wedge \big(e(\underline{p}_i) \rightarrow \mathfrak{K}_S(\underline{p}_i) \neq \epsilon\big)^4 \wedge \big(\forall C \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \cdot \underline{p}_i \mapsto C \in \underline{K} \rightarrow \mathit{verifyNym}(\underline{p}_i, (C.uKey)^{\mathcal{I}})\big) \wedge \big(\forall j \in \{1, \ldots, i-1, i+1, \ldots, \underline{k}\} \cdot \underline{p}_i \mapsto \underline{p}_j \in \underline{K} \rightarrow \mathit{verifyNyms}(\underline{p}_i, \underline{p}_j)\big) \wedge \big(\forall j \in \{1, \ldots, \underline{\ell}\} \cdot \underline{p}_i \mapsto \underline{d}_j \in \underline{K} \rightarrow \mathit{verifyNyms}(\underline{p}_i, \underline{d}_j)\big)$, where the predicate $e(x)$ denotes that pseudonym x is claimed to be established and ϵ denotes an empty predicate, i.e., U knows the secret keys of the pseudonyms, the pseudonyms are correctly established and all stated key-bindings are correct, and
8. $\forall i \in \{1, \ldots, \underline{\ell}\} \cdot \underline{d}_i \in \mathfrak{D}_U \wedge \big(e(\underline{d}_i) \rightarrow \mathfrak{K}_S(\underline{d}_i) \neq \epsilon\big) \wedge \big(\forall C \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \cdot \underline{d}_i \mapsto C \in \underline{K} \rightarrow \mathit{verifySeNym}(\underline{d}_i, \underline{w}_{\underline{d}_i}, (C.uKey)^{\mathcal{I}})\big) \wedge \big(\forall j \in \{1, \ldots, i-1, i+1, \ldots, \underline{\ell}\} \cdot \underline{d}_i \mapsto \underline{d}_j \in \underline{K} \rightarrow \mathit{verifyNyms}(\underline{d}_i, \underline{d}_j)\big) \wedge \big(\forall j \in \{1, \ldots, \underline{k}\} \cdot \underline{d}_i \mapsto \underline{p}_j \in \underline{K} \rightarrow \mathit{verifyNyms}(\underline{d}_i, \underline{p}_j)\big)$.

When a user *can* prove a token and convinces the verifier S of this fact by engaging in a presentation proof protocol, the effect of the token presentation transition is twofold. First, there is an increase of knowledge of S about the token's pseudonyms in the form of a logical predicate. This models that S learns that the corresponding predicate was $true$ at the time of the transition. Second, the transcript of the proof protocol acts as transferable *signature* σ_T on message $\underline{m}$, i.e., S may subsequently use σ_T to convince third parties about the fact that the user who proved token T (i.e., who fulfilled all the properties stated in T) consented to the statement $\underline{m}$.

Concerning the knowledge increase, we require that the credential variables in the verifier's knowledge state are disjoint with the credential variables $\underline{C}_1, \ldots, \underline{C}_n$ of T. If this is not the case, the variables of the verifier's state have to be renamed. Let $\underline{K}_x^C =$

[4] If pseudonym $\underline{p}_i$ is claimed to be established, then $\underline{p}_i$ must have been used in a previous token presentation transition with verifier S

$\{y \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \mid x \mapsto y \in \underline{K}\}$ be the set containing all credentials that x is key-bound to, and let $\underline{K}_x^{\mathfrak{N}}$ and $\underline{K}_x^{\mathfrak{D}}$ be analog sets for pseudonyms and scope-exclusive pseudonyms, respectively. The following specifies the knowledge of S after the token presentation transition about the $1 \leq i \leq \underline{k}$ pseudonyms:

$$
\begin{aligned}
\mathfrak{K}'_S(\underline{p}_i) = \mathfrak{K}_S(\underline{p}_i) \wedge \underline{\phi} \wedge \wedge_{j=1}^{\underline{n}_S} \underline{C.a}_{(j,S)} == \underline{v}_{(j,S)} \wedge \\
\wedge_{j=1}^{\underline{n}} \textit{notIssRevoked}(\underline{I}_{C_j}, \underline{h}_{C_j}, \underline{C}_j.rHandle) \wedge \\
\wedge_{j=1}^{\underline{o}} \textit{notVerRevoked}(\underline{V}_j, \underline{h}_{V_j}, \underline{C.a}_{(1,V_j)}, \ldots, \underline{C.a}_{(\underline{n}_{V_j}, V_j)}) \wedge \\
\wedge_{j=1}^{\underline{q}} \textit{verifyEnc}(\underline{e}_{S_j}, \underline{S}_j, \underline{g}_{S_j}, \underline{C.a}_{(1,S_j)}, \ldots, \underline{C.a}_{(\underline{n}_{S_j}, S_j)}) \wedge \\
\wedge_{x \in \underline{K}_{\underline{p}_i}^C} \textit{verifyNym}(\underline{p}_i, x.uKey) \wedge \\
\wedge_{x \in \underline{K}_{\underline{p}_i}^{\mathfrak{N}} \cup \underline{K}_{\underline{p}_i}^{\mathfrak{D}}} \textit{verifyNyms}(\underline{p}_i, x) \wedge \\
\wedge_{j=1}^{i-1} \overrightarrow{\mathfrak{K}'_S(\underline{p}_j)} \wedge \wedge_{j=i+1}^{\underline{k}} \overrightarrow{\mathfrak{K}'_S(\underline{p}_j)} \wedge \wedge_{j=1}^{\underline{\ell}} \overrightarrow{\mathfrak{K}'_S(\underline{d}_j)}
\end{aligned}
$$

The analog we require for knowledge states $\mathfrak{K}'_S(\underline{d}_i)$ for $1 \leq i \leq \underline{\ell}$, where the last three lines of the formula are adapted accordingly. The above formula models that S learns the values that U discloses, and that the predicate $\underline{\phi}$ over the contained attribute variables holds, however, S does not learn the values of these attributes. Further, S learns that the credentials used to prove the token are neither issuer-revoked nor verifier-revoked, however, S neither learns the revocation handle nor the values of the non-revoked attributes, respectively. Also, S learns that the ciphertext $\underline{e}_x$ for inspector x contains the values that correspond to the attributes $\underline{C.a}_{(1,x)}, \ldots, \underline{C.a}_{(1,\underline{n}_x)}$, however, he does not know what the values are. The verifier also learns which credentials and pseudonyms have the same underlying key, however, he does not learn the key itself. The last line of above formula models that the knowledge states of all pseudonyms used within the presentation token are merged. We use the notation $\overrightarrow{\cdot}$ to express that a predicate is not used by value, but rather by reference. This models that a verifier can transitively link all pseudonyms that were jointly contained in presentation tokens proved to him. For example, consider two consecutive token presentations where in the first one the predicate ψ_1 for pseudonyms p_1 and p_2 is proved, and in the second one ψ_2 with p_1. With the 'by reference' semantics, $\mathfrak{K}(p_2)$ is then $\psi_1 \wedge \psi_2$, rather than just ψ_1 with the 'by value' semantics. For the knowledge states that are merged, we require that the states' variables are disjoint, as otherwise new (and possibly wrong) knowledge about credentials is created by the merge operation itself. In case a token contains no pseudonyms at all, i.e., $\underline{k} + \underline{\ell} = 0$, a new unique identifier X is used to refer to the new knowledge $\mathfrak{K}'_S(X) = \underline{\phi} \wedge \cdots$ (same as above, except the last three formula lines as they contain references to pseudonyms). Unlinkability of credentials is achieved due to the renaming of the verifier's variables that was mentioned above, i.e., a verifier does not learn whether a user used the same or different credentials to prove a particular token.

5.7 Policy Verification

For a server to make an access control decision after a presentation token has been proved, he verifies that token w.r.t. a presentation policy. In the following, we describe

how such verification is performed in detail. Let P be the presentation policy of S that U needs to satisfy. Each alternative contains from an abstract point of view, i.e. without considering its form in XML notation, the following information:

- k pseudonym aliases $p_1, \ldots, p_k$, optionally established
- ℓ scope-excl. pseudonym aliases $d_1, \ldots, d_\ell$ for scopes $w_{d_1}, \ldots, w_{d_\ell}$, optionally established
- Message m
- n credentials $C_1, \ldots, C_n$ with type- and issuer-alternatives, and revocation epochs $h_{I_{(i,C_j)}}$ for each issuer alternative $I_{(i,C_j)}$ for $1 \leq i \leq n_{C_j}$ and $1 \leq j \leq n$, where n_{C_j} is the number of issuer alternatives for C_j
- Attribute predicate ϕ
- Attributes $C.a_{(i,S)}$ to disclose for $1 \leq i \leq n_S$ where n_S is the number of attributes to disclose to S
- Attributes $C.a_{(i,S_j)}$ to disclose to inspector S_j on grounds g_{S_j} for $1 \leq i \leq n_{S_j}$ and $1 \leq j \leq q$ where n_{S_j} is the number of attributes to disclose to S_j and q is the number of different inspectors
- Non-revoked attributes $C.a_{(i,V_j)}$ at revocation authority V_j with reference revocation epochs h_{V_j} for $1 \leq i \leq n_{V_j}$ and $1 \leq j \leq o$ where n_{V_j} is the number of revocation attributes of V_j and o is the number of revocation authorities
- Symmetric key binding relation $K \subseteq R \times R$, with $R = \{C_1, \ldots, C_n\} \cup \{p_1, \ldots, p_k\} \cup \{d_1, \ldots, d_\ell\}$, where $x \mapsto y \in K$ states that x is key-bound to y.

For simplicity, we do not consider inspector alternatives. Again, a credential's type and issuer are treated as dedicated attributes, and thus are specified as part of the predicate ϕ. In particular, specifying alternatives $\tau_1, \ldots, \tau_n$ for a credential C is an abbreviation for specifying $C.type == \tau_1 \vee \ldots \vee C.type == \tau_n$ as additional conjunct of ϕ. Issuer alternatives are treated analog. We further assume that for all pairs of credentials $C, C' \in \{C_1, \ldots, C_n\}$ that are bound to the same key, i.e., $C \mapsto C' \in K$, ϕ contains a conjunct $C.uKey == C'.uKey$. Note that the revocation epochs are optional in the XML notation. For our model we assume $h_x = \mathit{currEpoch}(x)$ for revocation authority x in case h_x is not given.

We say that *the presentation token T fulfills the presentation policy P* iff:

1. $\underline{k} \geq k$ and $\forall x \in \{1, \ldots, k\} \cdot e(x) \rightarrow \big(\mathfrak{K}_S(\underline{p}_x) \neq \epsilon\big)$, where the predicate $e(x)$ denotes that x-th pseudonym in the policy is required to be established and ϵ denotes an empty predicate, i.e., at least k pseudonyms are provided and established, respectively,
2. $\underline{\ell} \geq \ell$ and $\forall x \in \{1, \ldots, \ell\} \cdot \underline{w}_{d_x} = w_{d_x} \wedge \big(e(x) \rightarrow (\mathfrak{K}_S(\underline{d}_x) \neq \epsilon)\big)$, i.e., at least ℓ scope-exclusive pseudonyms for the requested scopes are provided and established, respectively,
3. $m = \underline{m}$, i.e., the correct message is signed,
4. $\{C_1, \ldots, C_n\} \subseteq \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\}$, i.e., the same credential variables are used in the policy and the token,

5. $\underline{\phi}$ implies ϕ, i.e., the proved attribute predicate implies the one requested, which incorporates verifying the permitted type- and issuer-alternatives,
6. $\forall x \in \{C_1, \ldots, C_n\} \cdot \underline{h}_x \geq h_{\underline{I}_x}$, i.e., all requested credentials were valid w.r.t. the corresponding epoch required by the policy or a later one,
7. $\{C.a_{(1,S)}, \ldots, C.a_{(n_S,S)}\} \subseteq \{\underline{C.a}_{(1,S)}, \ldots, \underline{C.a}_{(\underline{n}_S,S)}\}$, i.e., at least the requested attributes to disclose are disclosed,
8. $\forall x \in \{S_1, \ldots, S_q\} \cdot g_x = \underline{g}_x \wedge \{C.a_{(1,x)}, \ldots, C.a_{(n_x,S)}\} \subseteq \{\underline{C.a}_{(1,x)}, \ldots, \underline{C.a}_{(\underline{n}_x,x)}\}$, i.e., at least the requested attributes to disclose to the respective inspectors are disclosed,
9. $\forall x \in \{V_1, \ldots, V_n\} \cdot \underline{h}_x \geq h_x \wedge \forall i \in \{1, \ldots, n_x\} \cdot C.a_{(i,x)} = \underline{C.a}_{(i,x)}$, i.e., the exact requested combination of attribute values is not revoked at the respective requested revocation authorities, and
10. there exists a total injective mapping r from R to $\underline{R}$ that maps (a) the policy's credential aliases to token aliases and (b) the policy's pseudonym aliases to token pseudonym values, such that $\forall x \mapsto y \in K \cdot \exists x' \mapsto y' \in \underline{K} \cdot x' = r(x) \wedge y' = r(y)$, i.e., for all key-bindings required by the policy there exists a corresponding key-binding in the token.

A token T fulfills a presentation policy $P = \{P_1, \ldots, P_{n_P}\}$ with alternatives $P_1, \ldots, P_{n_P}$ iff $\exists p \in P$ such that T fulfills p.

5.8 Token Inspection

During a token presentation transition, a server S may obtain verifiably encrypted values for attributes $\underline{C.a}_{(i,I)}$ for $1 \leq i \leq \underline{n}_I$ in the form of a ciphertext $\underline{e}_I$ that is only decryptable by the designated inspector I on grounds $\underline{g}_I$. To obtain the plaintext values, S forwards the ciphertext and the decryption grounds to I who first checks whether these grounds are indeed fulfilled and (only) then decrypts $\underline{e}_I$ with his secret key sk to the originally encrypted values $\langle v_1, \ldots, v_{\underline{n}_I} \rangle \leftarrow$ *vDecrypt*$(\underline{e}_I, sk, \underline{g}_I)$. Assuming that I shares these values with S and that the user who initially provided the ciphertext was known under identifier X, the knowledge state of S after this transition is: $\mathfrak{K}'_S(X) = \mathfrak{K}_S(X) \wedge \underline{C.a}_{(1,I)} == v_1 \wedge \cdots \wedge \underline{C.a}_{(\underline{n}_I,I)} == v_{\underline{n}_I}$. This models that S learns the plaintext values of the verifiably encrypted attributes.

5.9 Advanced Credential Issuance

Now we look at issuance transaction between a user and an issuer that are based on more advanced issuance policies than the one described in Section 5.4. In particular, we now consider credential templates that may also contain key-binding information and carried-over attributes.

Let $Q = \langle P, M \rangle$ be an issuance policy that consists of a presentation policy P and credential template M, where P contains the information as specified in Section 5.7, and M contains, from an abstract point of view, i.e. without considering its form in XML notation, the following information:

– Type τ to be issued

- Issuer I
- Attributes q_i to carry over from $C.a_{(i,I)}$ for $1 \leq i \leq n_I$ where n_I is the number of attributes to carry over
- Attributes $r_1, \ldots, r_{n_r}$ to establish with joint random values
- Optional key-binding either (1) to credential B, or (2) to the i-th pseudonym for $1 \leq i \leq k$, or (3) to the i-th scope-exclusive pseudonym for $1 \leq i \leq \ell$

Further, let T be a presentation token containing the information as specified in Section 5.6, that can be fulfilled by U under interpretation $\mathcal{I}$. An advanced issuance transition between a user U and an issuer I is performed when U proves the presentation token T to I under interpretation $\mathcal{I}$ such that the proved T fulfills the presentation policy P that is contained in Q. As for basic issuance, the transition has effects on the user's credential portfolio as well as the issuer's knowledge state.

The set of credentials that U owns is augmented in the same way as this is the case for basic issuance, where the newly issued credential N has the following *additional* properties:

- $N(q_i) = \left(C.a_{(i,I)}\right)^{\mathcal{I}}$ for $1 \leq i \leq n_I$
- $N(r_i)$ is a jointly generated random number for $1 \leq i \leq n_r$
- In case of a key-binding with a credential:
  $N(uKey) = (B.uKey)^{\mathcal{I}}$
- In case of a key-binding with a (scope-exclusive) pseudonym:
  $N(uKey) = s$, where s is the user key that underlies $\underline{p}_i$ or $\underline{d}_i$ from the token T, respectively

The issuer's knowledge increase of the advanced issuance transaction is a conjunction of:

1. the knowledge increase that we specified for token presentation transitions in Section 5.6,
2. the knowledge increase that we specified for basic issuance in Section 5.4,
3. the predicate $\bigwedge_{i=1}^{n_I} N.q_i == C.a_{(i,I)}$, which models that I knows that the carried-over attributes are equal to its source attributes, however, I does not learn the values of those attributes,
4. the predicate $\bigwedge_{i=1}^{n_r}$ *jointlyRandom*$(N.r_i)$, which models that I knows which attributes have jointly random generated values (expressed by a dedicated boolean function *jointlyRandom*$(C.a)$ for attribute variable $C.a$), but without I learning these values,
5. in case of a key-binding with a credential: the predicate
   $N.uKey == B.uKey$, which models that I learns that the user keys of the new and the source credentials are equal, however, I does not learn the key itself,
6. in case of a key-binding with a pseudonym: the predicate
   verifyNym$(\underline{p}_i, N.uKey)$, which models that I learns that the new credential is bound to the same user key that underlies $\underline{p}_i$, however, I does not learn the key itself,
7. in case of a key-binding with a scope-exclusive pseudonym: the predicate
   verifySeNym$(\underline{d}_i, \underline{w}_{d_i}, N.uKey)$, which models that I learns that the new credential

is bound to the same user key that underlies $\underline{d}_i$ for scope $\underline{w}_{d_i}$, however, I does not learn the key itself.

5.10 Knowledge-Based Policy Verification

In Section 5.7 we show how a verifier determines whether a given presentation policy is fulfilled by the information proved in a given presentation token. Rather than considering merely the information provided in one single token for verifying the policy, a verifier S may also consider all the accumulated knowledge $\mathfrak{K}_S$ about all known pseudonyms for the verification. This flexible approach allows a user to prove just parts of a policy while still being granted access due to prior token presentations. As knowledge states are indirectly *time stamped* with epochs, verifiers can choose to disregard—or forget—knowledge before a certain epoch. The presentation policy language can accordingly be extended with a *switch* stating whether all of the verifier's accumulated knowledge is considered for policy verification or only the information provided in the current token. However, we leave the formalization of such sophisticated policy verification approach for future work.

6 Conclusion

We presented a language framework enabling a unified deployment of Privacy-ABC technologies, in particular, of U-Prove and Identity Mixer. Our framework improves upon the state of the art [32, 18] by covering the entire life-cycle of Privacy-ABCs, including issuance, presentation, inspection, and revocation, and by supporting advanced features such as pseudonyms and key binding. The framework offers a set of abstract concepts that make it possible for application developers to set up a Privacy-ABC infrastructure and to author policies without having to deal with the intricacies of the underlying cryptography. We demonstrate the soundness of our languages by providing a formal semantics that specifies the effects of issuing, presenting, verifying, inspecting, and revoking credentials on the user's credential portfolio and on the knowledge states of the involved parties.

The proposed language framework has been implemented as part of the ABC4Trust project, where it will be rolled out in two pilot projects. Preliminary tests indicate that our language framework adds a noticeable but reasonable overhead to the cryptographic routines, comparable to the overhead incurred by, for example, XML Signature [31] with respect to the underlying signing algorithm.

Our language framework supports a number of different authentication mechanisms including the mentioned privacy-preserving ones but also standard mechanisms such as X.509. However, most of them will not support the full set of features but we are currently working on a protocol framework that allows the combination of different cryptographic mechanisms to address this.

Acknowledgements

The authors thank Bart De Decker for his valuable comments on the formalization of the language semantics, and Lan Nguyen for his comments on the design of the revocation

framework. The authors further thank Robert Enderlein for his feedback on the XML language that also led to simplifications of the implementation. The research leading to these results was supported by the European Commission under the grant for the ABC4Trust project.

References

1. 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. *J. of Comput. Secur.*, 18(1), 2010.
2. C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *J. Comput. Secur.*, 16(4), 2008.
3. A. W. Appel and E. W. Felten. Proof-carrying authentication. *ACM CCS 1999*.
4. K. D. Bowers, L. Bauer, D. Garg, F. Pfenning, and M. K. Reiter. Consumable credentials in linear-logic-based access-control systems. *NDSS 2007*.
5. M.Belenkiy, J.Camenisch, M. Chase, M.Kohlweiss, A.Lysyanskaya, and H.Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO 2009*, vol. 5677 of *LNCS*.
6. P.Bichsel, J.Camenisch, and F.-S.Preiss. A comprehensive framework enabling data-minimizing authentication. In *ACM DIM 2011*.
7. P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *J. Comput. Secur.*, 10(3), 2002.
8. S.Brands, L.Demuynck, and B. De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *ACISP 07*, vol. 4586 of *LNCS*.
9. S.Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. MIT Press, 2000.
10. J.Camenisch, M.Dubovitskaya, A.Lehmann, G.Neven, C.Paquin, and F.-S.Preiss. A language framework for privacy-preserving attribute-based authentication. IBM Research Technical Report RZ 3818, 2012.
11. J.Camenisch, I. Krontiris, A.Lehmann, G.Neven, C.Paquin, K. Rannenberg, and H. Zwingelberg. Architecture for attribute-based credential technologies. ABC4Trust deliverable D2.1, 2011.
12. D.Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 24(2):84–88, 1981.
13. J.Camenisch, M.Kohlweiss, and C.Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC 2009*, vol. 5443 of *LNCS*.
14. J.Camenisch and A.Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, vol. 2045 of *LNCS*.
15. J.Camenisch and A.Lysyanskaya. An identity escrow scheme with appointed verifiers. In *CRYPTO 2001*, vol. 2139 of *LNCS*.
16. J.Camenisch and A.Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, vol. 2442 of *LNCS*.
17. J.Camenisch and A.Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, vol. 3152 of *LNCS*.
18. J.Camenisch, S.Mödersheim, G.Neven, F.-S.Preiss, and D.Sommer. A card requirements language enabling privacy-preserving access control. In *SACMAT 2010*.
19. J. R. Douceur. The Sybil attack. In *IPTPS 2002*, vol. 2429 of *LNCS*.
20. D. Ferraiolo and R. Kuhn. Role-based access control. *NIST-NCSC 1992*.

21. D. Garg, L. Bauer, K. D. Bowers, F. Pfenning, and M. K. Reiter. A linear logic of authorization and knowledge. *ESORICS 2006*.
22. Identity Mixer. `http://idemix.wordpress.com/`.
23. M.Kirkpatrick, G.Ghinita, and E.Bertino. Privacy-preserving enforcement of spatially aware RBAC. In *IEEE Transactions on Dependable and Secure Computing*, 99(PrePrints), 2011.
24. N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM TISSEC*, 6(1), 2003.
25. J. Li, N. Li, and W. Winsborough. Automated trust negotiation using cryptographic credentials. *ACM CCS 2005*.
26. T.Nakanishi, H.Fujii, Y.Hira, and N.Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *PKC 2009*, volume 5443 of *LNCS*.
27. F.Paci, N.Shang, K.Steuer Jr., R.Fernando, E.Bertino. VeryIDX - A privacy preserving digital identity management system for mobile devices. *Mobile Data Management 2009*.
28. A.Squicciarini, A.Bhargav-Spantzel, E.Bertino, and A.Czeksis. Auth-SL – A system for the specification and enforcement of quality-based authentication policies. In *ICICS 2007*.
29. Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, vol. 3376 of *LNCS*.
30. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Comput.*, 29(2), 1996.
31. S.Shirasuna, A.Slominski, L.Fang, and D.Gannon. Performance comparison of security mechanisms for grid services. GRID 2004.
32. Microsoft U-Prove Community Technology Preview R2. `http://www.microsoft.com/uprove`.
33. E.R. Verheul. Self-Blindable Credential Certificates from the Weil Pairing. ASIACRYPT 2001.
34. L. Wang, D. Wijesekera, and S. Jajodia. A logic-based framework for attribute based access control. *ACM FMSE 2004*.
35. W. Winsborough, K. Seamons, and V. Jones. Automated trust negotiation. *DISCEX 2000*.
36. OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0, 2005.

A XML Examples

This section shows XML artifacts for our example described in Section 4. Figures 6 and 7 depicts the issuer and revocation authority parameters for the identity card of the Republic of Utopia.

```
<IssuerParameters>
 <ParametersUID>urn:utopia:id:issuer</ParametersUID>
 <AlgorithmID>urn:com:microsoft:uprove</AlgorithmID>
 <SystemParameters>...</SystemParameters>
 <CredentialSpecUID>urn:creds:id</CredentialSpecUID>
 <HashAlgorithm>xenc:sha256</HashAlgorithm>
 <CryptoParams>...</CryptoParams>
 <KeyBindingInfo>...</KeyBindingInfo>
 <RevocationParametersUID>
   urn:utopia:id:ra
 </RevocationParametersUID>
</IssuerParameters>
```

Fig. 6. Issuer parameters.

```
<RevocationAuthorityParameters>
 <ParametersUID>urn:utopia:id:ra</ParametersUID>
 <RevocationMechanism>
   urn:abc4trust:accumulators:cl
 </RevocationMechanism>
 <RevocationInfoReference ReferenceType="url">
   https:utopia.gov/id/revauth/revinfo
 </RevocationInfoReference>
 <NonRevocationEvidenceReference ReferenceType="url">
   https:utopia.gov/id/revauth/nrevevidence
 </NonRevocationEvidenceReference>
 <CryptoParams>...</CryptoParams>
</RevocationAuthorityParameters>
```

Fig. 7. Revocation authority parameters.

The issuance token that would be generated in response to the issuance policy for the library card as depicted in Figure 4, is shown in the Figure 8 below.

```
<IssuanceToken>
  <IssuanceTokenDescription>
    <PresentationTokenDescription PolicyUID="libcard">
      <Message>...</Message>
      <Pseudonym Alias="nym"  Scope="urn:library:issuance" Exclusive="true"/>
        <PseudonymValue>MER2VXISHI=</PseudonymValue>
      </Pseudonym>
      <Credential Alias="id" SameKeyBindingAs="nym">
        ...
        <DisclosedAttribute  AttributeType="urn:creds:id:state">
          <AttributeValue> Nirvana </AttributeValue>
        </DisclosedAttribute>
      </Credential>
    </PresentationTokenDescription>
    <CredentialTemplate SameKeyBindingAs="id">...</CredentialTemplate>
  </IssuanceTokenDescription>
  <CryptoEvidence>...</CryptoEvidence>
</IssuanceToken>
```

Fig. 8. Issuance token for obtaining the library card.