Concepts and Languages 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 overcome this issue and simplify both the design and deployment of privacy-friendly authentication mechanisms. We define and unify the concepts and features of privacy-preserving attribute-based credentials (Privacy-ABCs) and provide a language framework in XML schema. Our language framework enables application developers to use Privacy-ABCs with all their features without having to consider the specifics of the underlying cryptographic algorithms—similar to as they do today for digital signatures, where they do not need to worry about the particulars of the RSA and DSA algorithms either.

Keywords: Authentication, privacy, data-minimization, anonymous credentials, digital credentials.

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 transferring trusted user attributes cause privacy issues. 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 [16, 11, 21, 25, 6, 53]. 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 [11] and Camenisch-Lysyanskaya's scheme [21], which have been implemented in Microsoft's U-Prove [52] and IBM's Identity Mixer [36], 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, 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. The full language description and schema are available as a project deliverable [15].

2 Related work

Our work builds on the credential-based authentication requirements language (CARL) recently proposed by Camenisch et al. [26]. 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 transmitted nor how credentials are issued or revoked. Bichsel et al. [7] have extended CARL to cover the transmission of certified attributes. Version 1 of the U-Prove protocols [52] 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 [8, 56, 54], 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 [32, 50] can be seen as a special case of our attribute-based setting by encoding the user's roles as attributes. Recent work [38] extended RBAC with privacy-preserving authentication for the particular case of role and location attributes.

Bonatti and Samarati [8] 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. [55]. However, the latter allows one to impose attribute properties on credentials and its extension by Li et al. [41] supports revealing of attributes. The Auth-SL language [48] 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 [45] 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 and technology-neutral approaches to distributed access control have been proposed [3, 5, 34, 40]. 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.

Summarized, our language framework is the first that covers the whole life-cycle of Privacy-ABCs and also the first one unifying the full spectrum 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. The interactions are named according to their purpose. Depending on the technical realizations, these interactions will be realized differently and might occur multiple times using different protocols (we consider sending a single message also a protocol). 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 inspector generates a private decryption key and a corresponding public encryption key, and each revocation authority generates and publishes its revocation parameters. We assume that all entities have means to retrieve the public keys of the issuers, revoca-

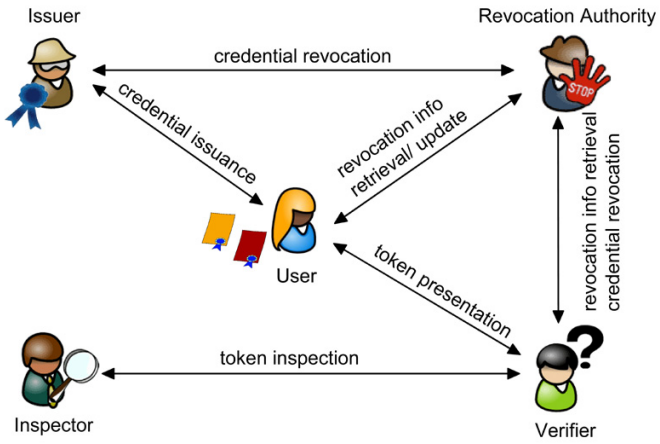

Fig. 1. Entities and the interactions between them.

tion authorities, inspectors, and verifiers. Users get issued credentials by issuers via the *credential issuance* protocol. A credential contains attributes that its issuer vouches for w.r.t. the user. A credential can also specify one or more revocation authorities who are able to revoke the credential if necessary for some reason. To issue a credential that is revocable, the user and/or the issuer might need to interact with the revocation authorize prior or during the issuance protocol. Using her credentials, a user can form a presentation token that contains a subset of the certified attributes, provided that the corresponding credentials have not been revoked. This process might require the user to retrieve information from the revocation authority. Additionally, some of the attributes can be encoded in the presentation token so that they can only be retrieved by an inspector. The user can attach inspection grounds specifying under this condition the inspector should reveal these attributes. Receiving a presentation token from a user, a verifier checks whether the presentation token is valid w.r.t. the relevant issuers' public keys and inspector public keys and the latest revocation information (thus, the verifier will interact with the revocation authority). If the verification succeeds, the verifier will be convinced that the attributes contained in the presentation token are vouched for by the corresponding issuers. Finally, if a presentation token contains attributes that can only be retrieved by an inspector and the inspection grounds are met, the verifier can interact with the inspector to learn these attributes.

Informally, a secure realization of a Privacy-ABC system guarantees that (1) users can only generate a valid presentation token if they were indeed issued the corresponding credentials that have not been revoked, (2) that attributes encoded in the presentation token for an inspector can indeed be retrieved by that inspector, and (3) that the presentation tokens do not reveal any further information about the users other than the attributes contained in them.

We now provide a brief explanation of the main features supported by Privacy-ABCs, with a focus on the ones that were not modeled so far in existing identity frameworks.

3.1 Pseudonyms

Each user can generate a secret key. However, unlike traditional public-key authentication schemes, 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 [16, 42] 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* [35]. Scope-exclusive pseudonyms for different scope strings remain unlinkable. 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.

3.2 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, i.e., it cannot be used without knowing the secret key [42]. 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.

3.3 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 presentation token can be verified 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.

3.4 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 [11, 24].

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

3.5 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* [12, 27]. 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 under which they have to be encrypted.

3.6 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 mechanisms with varying efficiency [39] exist [23, 49, 10, 18, 43]. 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.

3.7 Cryptographic Realization

Among the most prominent instantiations of Privacy-ABC systems are IBM's Identity Mixer [36] and Microsoft's U-Prove [52]. Both systems currently support only a subset of the features presented here, but will be extended to support the full feature set as part of the ABC4Trust project. Here, we sketch how the different features can be realized cryptographically and give pointers to relevant related literature; a full security analysis of the combined system is beyond the scope of this paper.

At the core of a Privacy-ABC system is a signature scheme with efficient protocols to prove possession of signatures. Identity Mixer and U-Prove build on the Camenisch-Lysyanskaya [24] and the Brands [11] signature schemes, respectively, but also other

schemes exist [25, 4]. An issued credential is a signature (or, for single-show schemes such as [11], a batch of renewable signatures) under such a scheme on the user's attributes. Some schemes inherently support key binding [24, 25, 4], others can be extended by using a randomly chosen attribute as secret key [11].

Ordinary pseudonyms are cryptographic commitments [46, 29] to the secret key; scope-exclusive pseudonyms can be realized as the output of a verifiable random function [30, 17] applied to the secret key as seed and the scope string as input. Inspection can be obtained through verifiable encryption [27] of the inspectable attributes. Revocation of Privacy-ABCs can be done through signed revocation lists [43], through dynamic accumulators [23, 49, 18], or through efficient updates of short-lived credentials [19].

The *glue* binding all primitives together in a presentation token is provided by generalized zero-knowledge proofs of knowledge of discrete logarithms [47, 20] made non-interactive through the Fiat-Shamir transform [33]. These proofs are also used for equality predicates over attributes; inequality predicates are proved with range proofs [9, 13].

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 Abstract Syntax Notation One (ASN.1) [37] or JavaScript Object Notation (JSON) [28].

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. For the sake of space and readability, these examples do not illustrate all features described in the previous section; we refer the reader to [15] for the full specification. 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, assume 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.

```
1  <CredentialSpecification KeyBinding="true" Revocable="true">
2    <SpecificationUID> urn:creds:id </SpecificationUID>
3    <AttributeDescriptions MaxLength="32">
4      <AttributeDescription Type="urn:creds:id:name" DataType="xs:string" Encoding="xenc:sha256">
5        <FriendlyAttributeName lang="EN"> Full Name </FriendlyAttributeName>
6      </AttributeDescription>
7      <AttributeDescription Type="urn:creds:id:state" DataType="xs:string" Encoding="xenc:sha256"/>
8      <AttributeDescription Type="urn:creds:id:bdate" DataType="xs:date" Encoding="date:unix:unsigned"/>
9    </AttributeDescriptions>
10 </CredentialSpecification>
```

Fig. 2. Credential specification of the 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 depends on the security parameter (basically, it is the bit length of exponents in the group) and is indicated by the **MaxLength** XML attribute (Line 3), here 32 bytes. In our example, 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 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, which are illustrated in Figure 3.

The **ParametersUID** element assigns unique identifiers for the issuer and revocation authority parameters. The issuer parameters additionally specify the chosen cryptographic Privacy-ABC and hash algorithm, the credential specification that credentials issued under these issuer parameters will follow, and the parameters identifier of the revocation authority that will manage the issuer-driven revocation. The **SystemParameters**, **CryptoParams**, and **KeyBindingInfo** contain cryptographic algorithm-specific information about the public key.

```
<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>
```

```
<RevocationAuthorityParameters>
  <ParametersUID> urn:utopia:id:ra </ParametersUID>
  <RevocationMechanism> urn:privacy−abc: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. 3. Issuer and revocation authority parameters.

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

Assume that a user already possesses an identity card from the Republic of Utopia issued according to the credential specification depicted in Figure 2. Further, assume 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 attribute certified by the card. This results in the presentation policy depicted in Figure 4.

```
<PresentationPolicy PolicyUID="libcard">
  <Message>
    <Nonce> bkQydHBQWDR4TUZzbXJKYUM= </Nonce>
  </Message>
  <Pseudonym Alias="nym" Scope="urn:library:issuance" Exclusive="true"/>
  <Credential Alias="id" SameKeyBindingAs="nym">
    <CredentialSpecAlternatives>
      <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
    </CredentialSpecAlternatives>
    <IssuerAlternatives>
      <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
    </IssuerAlternatives>
    <DisclosedAttribute AttributeType= "urn:creds:id:state"/>
  </Credential>
</PresentationPolicy>
```

Fig. 4. Presentation policy for an identity card.

We now go through the preceding presentation policy and describe how the different features of Privacy-ABCs can be realized with our language. We 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. 4, Lines 2–4). It can include a nonce, any application-specific message, and a human-readable name and/or description of the policy. The nonce will be used to prevent replay attacks, i.e. to ensure freshness of the presentation token, and for cryptographic evidence generation. Thus, when making use of the nonce, the presentation policy is not static anymore, but needs to be completed with a fresh nonce element for every request.

Pseudonyms. The optional **Pseudonym** element (Fig. 4, 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 associated XML attribute **Exclusive** 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., to verify that different pseudonyms actually belong to the same user.

Credentials and Selective Disclosure. For each credential that the user is requested to present, the policy contains a **Credential** element (Fig. 4, Lines 6–14), which describes the credential to present in detail. In particular, disjunctive lists of the accepted credential specifications and issuer parameters can be specified via **CredentialSpecAlternatives** and **IssuerAlternatives** elements, respectively (Fig. 4, Lines 7-9 and 10–12). The credential element also indicates all attributes that must be disclosed by the user via **DisclosedAttribute** elements (Fig. 4, Line 13). The XML attribute **Alias** assigns the credential an alias so that it can be referred to from other places in the policy, e.g., from the attribute predicates.

Key Binding. If present, the **SameKeyBindingAs** attribute of a **Credential** or **Pseudonym** element (Fig. 4, 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 preceding example, the policy requests that the identity card and the presented pseudonym must belong to the same secret key.

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 the issuance policy depicted in Figure 5. It contains the presentation policy from Figure 4 and the credential template that is described in detail below.

```
1  <IssuancePolicy>
2    <PresentationPolicy PolicyUID="libcard"> ... </PresentationPolicy>
3    <CredentialTemplate SameKeyBindingAs="id">
4      <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
5      <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
6      <UnknownAttributes>
7        <CarriedOverAttribute TargetAttributeType= "urn:utopia:lib:name">
8          <SourceCredentialInfo Alias="id" AttributeType="urn:creds:id:name"/>
9        </CarriedOverAttribute>
10     </UnknownAttributes>
11   </CredentialTemplate>
12 </IssuancePolicy>
```

Fig. 5. Issuance policy for a library card. The presentation policy on Line 2 is depicted in Figure 4.

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. 5, 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** element (Fig. 5, Lines 6–10) it is specified which user attributes of the new credential will be carried over from existing credentials in the presentation token. The **SourceCredentialInfo** element (Fig. 5, Line 8) indicates the credential and the user attribute of which the value will be carried over.

Although this is not illustrated in our 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"*.

4.4 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 a presentation policy. 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 to the presentation and issuance policy is that in the returned token a **Pseudonym** (if requested in the policy) now also contains a **PseudonymValue** (Fig. 6, Line 6). Similarly, the **DisclosedAttribute** elements (Fig. 6, Lines 10–12) in a token now also contain the actual user attribute values. Finally, all data from the cryptographic implementation of the presentation token and the advanced issuance features

are grouped together in the **CryptoEvidence** element (Fig. 6, Line 17). This data includes, e.g., proof that the contained identity card is not revoked by the issuer and that it is bound bound to the same secret key as the pseudonym.

```
1  <IssuanceToken>
2    <IssuanceTokenDescription>
3      <PresentationTokenDescription PolicyUID ="libcard" >
4        <Message> ... </Message>
5        <Pseudonym Alias="nym" Scope="urn:library:issuance" Exclusive="true" />
6          <PseudonymValue> MER2VXISHI=</PseudonymValue>
7        </Pseudonym>
8        <Credential Alias="id" SameKeyBindingAs="nym" >
9              ...
10         <DisclosedAttribute AttributeType="urn:creds:id:state" >
11           <AttributeValue> Nirvana </AttributeValue>
12         </DisclosedAttribute>
13       </Credential>
14     </PresentationTokenDescription>
15     <CredentialTemplate SameKeyBindingAs="id" > ... </CredentialTemplate>
16   </IssuanceTokenDescription>
17   <CryptoEvidence> ... </CryptoEvidence>
18 </IssuanceToken>
```

Fig. 6. Issuance token for obtaining the library card.

4.5 Presentation Policy with Extended Features

Assume 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. Repeated negligence may even lead to exclusion from borrowing further paper books while borrowing digital books remains possible.

Moreover, assume that the library offers special conditions for young readers that can be used by anyone below the age of twenty-six years. 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 7.

A presentation policy that is 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. 7, Lines 2–34). The returned presentation token must satisfy (at least) one of the specified policies.

The example presentation policy requires 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. 7, Lines 30–33) specifies that the date of birth must be after July 16th, 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

```
<PresentationPolicyAlternatives>
  <PresentationPolicy PolicyUID= "young−reader" >
    <Message> ... </Message>
    <Credential Alias="libcard" SameKeyBindingAs="id" >
      <CredentialSpecAlternatives>
        <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
      </CredentialSpecAlternatives>
      <IssuerAlternatives>
        <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
      </IssuerAlternatives>
      <DisclosedAttribute AttributeType= "urn:utopia:lib:name" >
        <InspectorAlternatives>
          <InspectorPublicKeyUID> urn:lib:arbitrator </InspectorPublicKeyUID>
        </InspectorAlternatives>
        <InspectionGrounds> Late return or damage. </InspectionGrounds>
      </DisclosedAttribute>
    </Credential>
    <Credential Alias="id" >
      <CredentialSpecAlternatives>
        <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
      </CredentialSpecAlternatives>
      <IssuerAlternatives>
        <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
      </IssuerAlternatives>
    </Credential>
    <VerifierDrivenRevocation>
      <RevocationParametersUID> urn:lib:blacklist </RevocationParametersUID>
      <Attribute CredentialAlias ="libcard" AttributeType="urn:utopia:lib:name" />
    </VerifierDrivenRevocation>
    <AttributePredicate Function= "...:date−greater−than" >
      <Attribute CredentialAlias ="id" AttributeType= "urn:creds:id:bdate" />
      <ConstantValue> 1986−07−16 </ConstantValue>
    </AttributePredicate>
  </PresentationPolicy>
</PresentationPolicyAlternatives>
```

Fig. 7. Presentation policy for borrowing books.

function may be credential attributes (referred to by the credential alias and the attribute type) or constant values. See [15] for an exhaustive list of supported predicates and data types and note that an attribute's encoding as defined in the credential specification has implications on which predicates can be used for it and whether it is inspectable [15, Sec. 4.2.1].

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 (as defined in the credential specification), the cre-

dential must be proved to be valid with respect to the most recent revocation information. However, a policy can also require the use of a particular version of the revocation information. In the latter case, the element **IssuerParametersUID** has an extra XML attribute **RevocationInformationUID** specifying the identifier of the specific revocation information. The specification of the referenced **RevocationInformation** is given in [15]. Presentation tokens can accordingly state the validity of credentials w.r.t. a particular version by using a **RevocationInformationUID** XML element in the corresponding Credential element.

Verifier-Driven Revocation. If customers return borrowed books late or damaged, they are excluded from borrowing further paper books, but they are still allowed to use the library's online services. In our example, this is handled by a **VerifierDrivenRevocation** element (Fig. 7, 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 Security Discussion

For our framework to be useful, the various parties need to be given security guarantees: a verifier wants to be sure that if a presentation token verifies then all the statements made in it are indeed supported by the issuer and have not been revoked. Furthermore, the verifier wants to be sure that inspection will succeed when needed. The issuer wants to have similar guarantees w.r.t. issuance tokens. The user wants assurance that her privacy is maintained, i.e., that no more information is leaked than what she willingly released in a presentation token.

Formally proving that such guarantees are fulfilled, provided the underlying cryptography is sound, is far beyond the scope of this paper and is the topic of future work. Likewise, we do not go into detail here on the complex trust relations between the different participants in a Privacy-ABC system, or on which particular privacy or security threats can arise when some of these participants collude. Since presentation policies can always ask to reveal much more information than strictly necessary, one could also consider adding yet another authority to the system to approve "reasonable" policies. Finally, in order to deploy a Privacy-ABC system, one also needs a public-key infrastructure (PKI) to certify issuer parameters, revocation authority parameters, and inspectors' public keys.

For the individual cryptographic building blocks, security proofs are given in the cryptographic literature, and Camenisch and Lysyanskaya give a proof for their credential system [21]. However, proving the security for the different combinations of the building blocks has not been done, although there is no reason to believe that such combinations would not be secure. Thus, the first step in proving security of our language framework is to provide precise security notions that capture the high-level security properties stated above and to prove that the composition of the cryptographic building

blocks as imposed by our languages achieves those. Next, one would have to show that the mapping of our languages to the concrete cryptographic realizations is sound. Only then one could attempt to formally prove that the security guarantees as stated above are fulfilled.

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 [52, 26] 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.

In an upcoming companion paper, 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. A complete description of our framework including the language description as well as the formal semantics is available as a technical report [14].

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 [51] 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, Robert Enderlein, Lan Nguyen, and the anonymous referees for their valuable comments. The research leading to these results was supported by the European Commission under Grant Agreement 257782 ABC4Trust.

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. M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. In *SCN 2006*, vol. 4116 of *LNCS*.

5. K. D. Bowers, L. Bauer, D. Garg, F. Pfenning, and M. K. Reiter. Consumable credentials in linear-logic-based access-control systems. *NDSS 2007*.
6. 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*.
7. P. Bichsel, J. Camenisch, and F.-S. Preiss. A comprehensive framework enabling data-minimizing authentication. In *ACM DIM 2011*.
8. P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *J. Comput. Secur.*, 10(3), 2002.
9. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*.
10. 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*.
11. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. MIT Press, 2000.
12. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT '91*, vol. 547 of *LNCS*.
13. J. Camenisch, R. Chaabouni, and A. Shelat. Efficient protocols for set membership and range proofs. In *ASIACRYPT 2008*, vol. 5350 of *LNCS*.
14. J. Camenisch, M. Dubovitskaya, A. Lehmann, G. Neven, C. Paquin, and F.-S. Preiss. A language framework for privacy-preserving attribute-based authentication. Technical Report RZ3818, IBM, 2012.
15. J. Camenisch, I. Krontiris, A. Lehmann, G. Neven, C. Paquin, K. Rannenberg, and H. Zwingelberg. H2.1 – ABC4Trust Architecture for Developers. ABC4Trust heartbeat H2.1, 2011.
16. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 24(2):84–88, 1981.
17. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Balancing accountability and privacy using e-cash. In *SCN 06*, vol. 4116 of *LNCS*.
18. 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*.
19. J. Camenisch, M. Kohlweiss, and C. Soriente. Solving revocation with efficient update of anonymous credentials. In *SCN 10*, vol. 6280 of *LNCS*.
20. J. Camenisch, A. Kiayias, and M. Yung. On the portability of generalized Schnorr proofs. In Antoine Joux, editor, *EUROCRYPT 2009*, vol. 5479 of *LNCS*.
21. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, vol. 2045 of *LNCS*.
22. J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. In *CRYPTO 2001*, vol. 2139 of *LNCS*.
23. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, vol. 2442 of *LNCS*.
24. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 02*, vol. 2576 of *LNCS*.
25. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, vol. 3152 of *LNCS*.
26. J. Camenisch, S. Mödersheim, G. Neven, F.-S. Preiss, and D. Sommer. A card requirements language enabling privacy-preserving access control. In *SACMAT 2010*.
27. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, vol. 2729 of *LNCS*.
28. D. Crockford. The application/json media type for JavaScript Object Notation (JSON). Internet Engineering Taskforce (IETF) RFC 4627, 2006.

29. I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, vol. 2501 of *LNCS*.
30. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC 2005*, vol. 3386 of *LNCS*.
31. J. R. Douceur. The Sybil attack. In *IPTPS 2002*, vol. 2429 of *LNCS*.
32. D. Ferraiolo and R. Kuhn. Role-based access control. *NIST-NCSC 1992*.
33. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, vol. 263 of *LNCS*.
34. D. Garg, L. Bauer, K. D. Bowers, F. Pfenning, and M. K. Reiter. A linear logic of authorization and knowledge. *ESORICS 2006*.
35. IBM Research Zurich Security Team. Specification of the identity mixer cryptographic library. Technical Report RZ3730, IBM, 2010.
36. Identity Mixer. `http://idemix.wordpress.com/`.
37. International Telecommunication Union. Abstract syntax notation one (ASN.1). ITU-T recommendation X.680, 2008.
38. M. Kirkpatrick, G. Ghinita, and E. Bertino. Privacy-preserving enforcement of spatially aware RBAC. In *IEEE Trans. on Dependable and Secure Computing*, 99(PrePrints), 2011.
39. J. Lapon, M. Kohlweiss, B. De Decker, and V. Naessens. Analysis of Revocation Strategies for Anonymous *Idemix* Credentials. *IFIP CMS 2011*.
40. N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM TISSEC*, 6(1), 2003.
41. J. Li, N. Li, and W. Winsborough. Automated trust negotiation using cryptographic credentials. *ACM CCS 2005*.
42. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *SAC 1999*, vol. 1758 of *LNCS*.
43. T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *PKC 2009*, vol. 5443 of *LNCS*.
44. L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, vol. 3376 of *LNCS*.
45. 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*.
46. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, vol. 576 of *LNCS*.
47. C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
48. 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*.
49. L. Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, vol. 3376 of *LNCS*.
50. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Comput.*, 29(2), 1996.
51. S. Shirasuna, A. Slominski, L. Fang, and D. Gannon. Performance comparison of security mechanisms for grid services. GRID 2004.
52. Microsoft U-Prove. `http://www.microsoft.com/uprove`.
53. E. R. Verheul. Self-Blindable Credential Certificates from the Weil Pairing. ASIACRYPT 2001.
54. L. Wang, D. Wijesekera, and S. Jajodia. A logic-based framework for attribute based access control. *ACM FMSE 2004*.
55. W. Winsborough, K. Seamons, and V. Jones. Automated trust negotiation. *DISCEX 2000*.
56. OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0, 2005.