D8.13 Reference Implementation for Standardization V2

Fatbardh Veseli, Jan Camenisch, Jonas Lindstrøm Jensen

Editor:	*Fatbardh Veseli (Goethe University Frankfurt)*
Reviewers:	*Jimm Lerch (Eurodocs), Norbert Goetze (Nokia)*
Identifier:	*D8.13*
Type:	*Deliverable*
Version:	*2.0*
Date:	*2014-11-06*
Status:	*Final*
Class:	*Public*

Abstract

Standardization is considered of particular importance for the dissemination of ABC4Trust results. In this deliverable, we present a brief overview of the Reference Implementation of ABC4Trust and identify its standardization-relevant parts, including the implemented protocols and XML formats, externally available APIs, policy language, and cryptographic libraries. We also consider interoperability with de-facto standard identity management schemes as an alternative to standardization, which could further ease the adoption of the Reference Implementation. Furthermore, we identify some infrastructure challenges in porting the Reference Implementation to mobile phones caused by the lack of native support for certain components of the Reference Implementation in some mobile operating systems. For all of the above, we provide a motivation for the presented discussion points, providing not only lessons based on our experience in the design and development of the Reference Implementation, but also presenting insights on some of the challenges towards the adoption of the Reference Implementation including standardization and beyond.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257782 for the project Attribute-based Credentials for Trust (ABC4Trust) as part of the "ICT Trust and Security Research" theme.

Members of the ABC4Trust Consortium

1.	Alexandra Institute AS	ALX	Denmark
2.	CryptoExperts SAS	CRX	France
3.	Eurodocs AB	EDOC	Sweden
4.	IBM Research – Zurich	IBM	Switzerland
5.	Johann Wolfgang Goethe-Universität Frankfurt	GUF	Germany
6.	Microsoft Belgium NV	MS	Belgium
7.	Miracle A/S	MCL	Denmark
8.	Nokia	NSN	Germany
9.	Computer Technology Institute and Press “Diophantus”	CTI	Greece
10.	Söderhamn Kommun	SK	Sweden
11.	Technische Universität Darmstadt	TUD	Germany
12.	Unabhängiges Landeszentrum für Datenschutz	ULD	Germany

List of Contributors

Chapter	Author(s)
Executive Summary	Fatbardh Veseli (GUF)
First Chapter	Fatbardh Veseli (GUF)
Second Chapter	Fatbardh Veseli (GUF)
Third Chapter	Fatbardh Veseli (GUF)
Fourth Chapter	Jan Camenisch (IBM), Fatbardh Veseli (GUF)
Fifth Chapter	Jan Camenisch (IBM)
Sixth Chapter	Jonas Lindstrøm Jensen (ALX)
Conclusion	Fatbardh Veseli (GUF)

Executive Summary

Having identified standardization as an important outreach activity with a strategic impact potential, ABC4Trust has allocated resources to actively contribute to relevant standardization organisations and bodies. In this regard, there are two important components developed inside the project, for which the impact for standardization is considered to be higher, namely 1) the Architecture for attribute-based credentials, and 2) its Reference Implementation.

The final version of the Reference Implementation [D4.2] faithfully implements the ABC4Trust architecture, its policy language, as well as the defined APIs for inter-component communication. It not only implements a number of cryptographic mechanisms that support Privacy-ABC features, but also provides a flexible opportunity to potentially plug in new cryptographic mechanisms. Most importantly, it provides the possibility for developers to easier integrate the Reference Implementation into their respective identity management schemes and applications without having to worry about the underlying layers, including the crypto layers. Therefore, it is considered to be one of the key outcomes of ABC4Trust.

This deliverable provides an overview of the main standardization areas for the Reference Implementation. It identifies the components of the Reference Implementation that have strong standardization value and potential, such as policy language, protocol specification, and cryptographic mechanisms realizing Privacy-ABC features. Further, it identifies other important opportunities that could push forward the adoption of Privacy-ABC technologies, such as achieving interoperability with successful technologies in current identity ecosystems. Finally, it discusses a number of challenges in deploying the Reference Implementation into de-facto standard mobile operating systems, namely lack of native support for certain libraries in current mobile and browser platforms, which currently makes porting the Reference Implementation to mobile platforms more troublesome.

Table of Contents

Index of Figures

Index of Tables

1 Introduction

Following the unified architecture design for Privacy-ABC technologies [D2.2], the Reference Implementation (RI) [D4.2] of ABC4Trust is one of the most valuable deliverables of ABC4Trust project. It faithfully implements the application programming interfaces (APIs) and the language framework defined in the architecture, fulfilling the goals of the projects for an implementation that enables federation and interchangeability of different underlying Privacy-ABC technologies.

The final version of the RI [D4.2] has provided further improvements, including updates to certain parts of the code and, most importantly, an introduction of a new cryptographic architecture that is more modular and efficient. The code and the documentation of the Reference Implementation is now available as an open source repository in GitHub [P2ABC], and includes both the Java source code and JavaScript implementation of the browser plugin.

In this deliverable, we analyse the relation of the reference implementation and standardization. We investigate the parts of the RI that would be more important to standardize and list a number of challenges in the existing de-facto "standard technologies" when deploying the reference implementation. The second chapter provides a brief summary of the architecture of Privacy-ABCs. Third chapter then provides a brief overview of the Reference Implementation of this architecture, as well as a description of the protocol specifications and data exchange formats that are implemented in the RI. The fourth chapter takes a look at some of the prominent identity management technologies that are successful in the market and discusses on the potential of providing interoperability of the Privacy-ABC- with these technologies. The fifth chapter presents a list of what we consider to be the more important components of the RI that could be standardized in international standardization projects. The sixth chapter brings an overview of some of the challenges in implementing the RI into existing de-facto standard mobile platforms. Finally, Chapter 7 concludes the deliverable with an outlook.

2 ABC4Trust Architecture – Entities and their Interactions[1]

The ABC4Trust architecture provides definitions of concepts such as *pseudonyms*, *credentials*, *(key-) binding,* and the processes of *issuance*, *presentation*, *revocation* and *inspection*, presenting a complete overview of the lifecycle of Privacy-ABCs. Related concepts, such as presentation tokens, issuance tokens, presentation policies and issuance policy, are also described.

A detailed description of all these concepts and features has already been included in Chapter 2 "Features and Concepts of Privacy-ABCs" of [D2.2]. This chapter provides a general overview on the different involved entities and the type of interactions that they engage in, as presented in Figure 2-1:

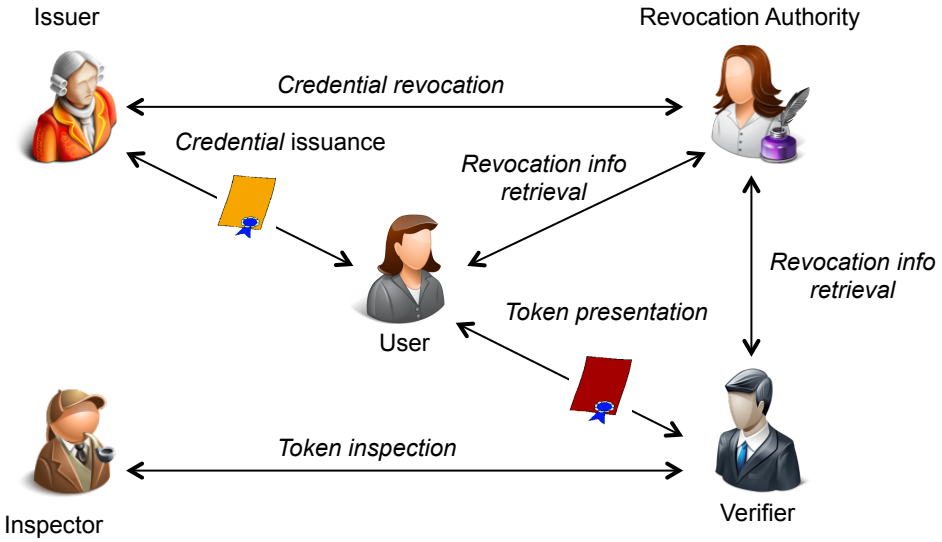

Figure 2-1: ABC4Trust Entities and their main interactions

- The *User* is at the centre of the picture, collecting credentials from various Issuers and controlling which information from which credentials she presents to which verifiers. The human User is represented by her User Agent, a software component running either on a local device (e.g. on the User's computer or mobile phone) or remotely on a trusted cloud service. The User may own special hardware tokens to which credentials can be bound to improve security. In identity management literature, the User is sometimes referred to as the requestor or the subject.

- An *Issuer* issues *credentials* to Users, thereby vouching for the correctness of the information contained in the credential with respect to the User to whom the credential is issued. Before issuing a credential, the Issuer may have to authenticate the User, which it may do using Privacy-ABCs, using a different online mechanism (e.g. username and password) or using out-of-band communication (e.g. by requiring the User to physically present herself at the Issuer's office). In the identity management literature, the Issuer is sometimes referred to as the identity provider or attributes authority.

- A *Verifier* protects access to a resource or service that it offers by imposing restrictions on the credentials that Users must own and the information from these credentials that Users must present in order to access the service. The Verifier's restrictions are described in its *presentation policy*. The

[1] The text for this chapter is an abbreviated version of "Chapter 2 - Features and Concepts of Privacy-ABCs", which is published in "Deliverable 2.2 – Architecture for Privacy-ABC technologies – Final Version" [D2.2].

User generates from her credentials a *presentation token* that contains the required information and the supporting cryptographic evidence. In the identity management literature, the Verifier is sometimes also referred to as the relying party, the server or the service provider.

- A *Revocation Authority* is responsible for *revoking* issued *credentials*, so that these credentials can no longer be used to generate a presentation token. Both the User and the Verifier must obtain the most recent revocation information from the Revocation Authority to generate and, respectively, verify presentation tokens.

- An *Inspector* is a trusted entity, which has the technical capability to "decrypt" some particular attribute used in the presentation token in order to avoid potential misuse of the privacy of partially anonymous users of Privacy-ABC technologies. To make use of this feature, the Verifier must specify in the *presentation policy* the entity which can perform such "inspection" should any of the specified *inspection grounds* materialize, e.g. when there is a threat or misuse of Privacy-ABC technologies. Therefore, the User is made aware of the de-anonymization possibility during the presentation and actively participates to make this possible, so that the User can make an informed decision based on her trust in the Inspector.

In an actual deployment, some of the above roles may actually be fulfilled by the same entity or split among many. For example, an Issuer can at the same time play the roles of Revocation Authority and Inspector. Besides, an Issuer could later also be the Verifier of tokens derived from credentials that it issued." [D2.2]

3 Reference Implementation and Standardization[2]

One major goal of the ABC4Trust project is to provide an open source Reference Implementation (RI) so that developers can easily integrate Privacy-ABC technologies in their applications. This way, programmers will be able to enhance their applications with strong users' authentication while concurrently protecting users' privacy.

The Reference Implementation demonstrates the practical feasibility of Privacy-ABC technologies. Its current version implements and realizes most of the concepts and features of the ABC4Trust Architecture together with the protocols and API defined in the architecture of Privacy-ABC technologies [D2.2]. The ABC Engine enables abstraction of the underlying cryptographic mechanism used. It is responsible for invoking the Crypto-Engine, which in turn generates all the necessary cryptographic information. The Reference Implementation has currently incorporated Microsoft's U-Prove and IBM's Idemix as underlying cryptographic engines. However, it is possible to incorporate additional cryptographic engines that support the desired subset of Privacy-ABC features in the future [D4.2].

3.1 Overview of the Reference Implementation (RI)

The basic building block of the Reference Implementation, from the perspective of an application developer, is the ABC-Engine. This block provides the methods that are required in order to implement the functionality of specific ABC roles. For example, it provides to an Issuer a method for issuing credentials, to a User a method for creating presentation tokens based on their credentials and to a Verifier a method for verifying presentation tokens. Through a well-defined API, the functionality of the ABC-Engine is exposed to the application layer as a RESTful web service that uses the XML for the data exchange. This way, the ABC-Engine can be easily integrated into different kinds of web applications [KP12].

For illustration purposes, Figure 3-1 presents a typical scenario of a user authenticating towards a Service Provider (Verifier). In particular, this figure represents an overview of the system components on the User's and the Verifier's sides, and it describes the communication flow between the two entities. It also shows how through a well-defined API the functionality of the ABC-Engine is exposed to the application layer as web services.

The User's browser plugin is invoked whenever a User wants to access a service offered by the Verifier. The plugin communicates with the ABC-Engine of the Verifier and gets a presentation policy that states what personal information (attribute values) the User must reveal or which of their personal characteristics she must prove in order to have access to this specific service. Then, it communicates with the User's ABC-Engine to find out if she has the necessary credentials to satisfy the respective policy. In case she does, it presents to the user an interface that explains to her what information is required and prompts her to select one of the possible combinations of credentials that can satisfy the presentation policy.

After the user has selected the credentials to be used, the plugin communicates again with the local (User) ABC-Engine, in order to create a presentation token that fulfils the requirements of the presentation policy. At that point, the plugin sends the presentation token to the Verifier's ABC-

[2] The material presented in this chapter has been originally contributed in different parts of the ABC4Trust deliverable "D4.1 Initial Reference Implementation" [D4.1] and an ABC4Trust contribution to the "PHP Magazine" [KP12]. For a more detailed description, please refer to the original sources.

Engine, which checks whether the token satisfies the policy. If the policy is satisfied, the User is allowed to access the requested resource (service).

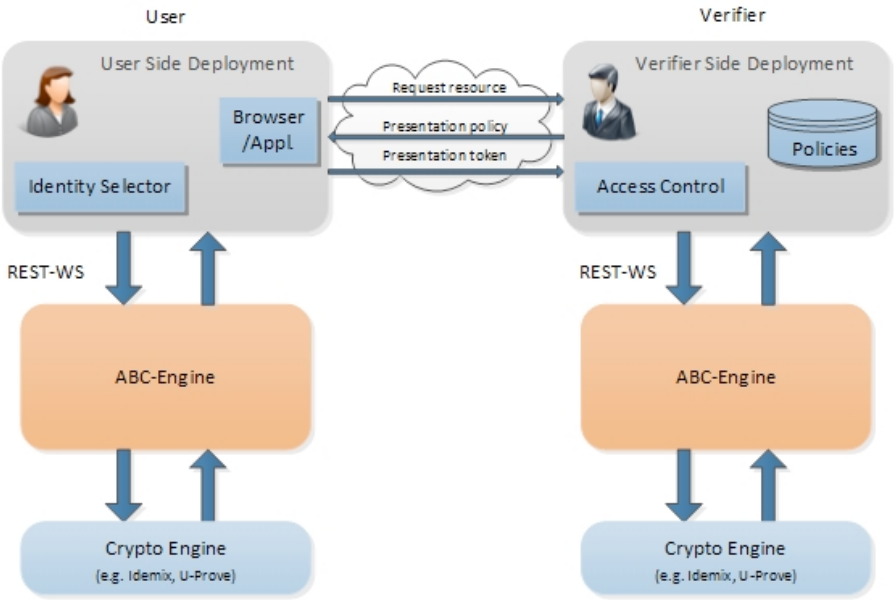

Figure 3-1: Overview of the components and steps involved during presentation between the User and the Verifier

3.2 Runtime Requirements

The Reference Implementation has some very simple requirements with respect to the runtime environment. It is written in Java and so it requires the Java Runtime Environment (JRE). The Reference Implementation supports any operating system, which has Java installed. However, the current deployment of the Reference Implementation requires the use of a special browser plugin to communicate with the application of the User, which is currently only available for two browsers, namely Mozilla's Firefox and Microsoft's Internet Explorer.

A more detailed description of the current Reference Implementation, its supported features, examples and requirements can be found in the ABC4Trust deliverable D4.2 "Final Reference Implementation" [D4.2].

3.3 Protocol specification and data artefacts[3]

The data artefacts and the API defined in WP2 of ABC4Trust are good candidates to be included in the discussion of standardization. The architecture of ABC4Trust is described in details in [D2.2] and it follows a layered approach, where all Privacy-ABC related functionalities are grouped together in a layer called ABCE (ABC Engine). Then the project has defined the API this layer offers to the application layer, thereby abstracting the internal design and structure for the application developers.

[3] This chapter presents a collection materials previously published in the Chapter 3, 4, 5 and 7 of the architecture of Privacy-ABC technologies [D2.2], as well as relevant sections from the Reference Implementation [D4.2].

Equally important to the API is the specification of the data artefacts exchanged between the implicated actors, introduced earlier in Chapter 2, in such a way that the underlying differences of concrete Privacy-ABCs are abstracted away through the definition of formats that can convey information independently from the mechanism-specific cryptographic data. Instead of using formats typically met in identity management protocols like SAML, WS-Trust, or OpenID, the project chose to define a dedicated format. This was because the unique features of Privacy-ABCs, like pseudonyms, inspections or revocation, can be expressed and supported in this way. Deliverable "D2.2 - Architecture for Attribute-based Credential Technologies – Final Version" [D2.2] emphasized on the XML based specification of the corresponding messages exchanged during the whole lifecycle of Privacy-ABCs, i.e. issuance, presentation, inspection, and revocation.

3.4 Example of the XML data formats and policy language

The presentation protocol starts by the Verifier sending a presentation policy to the User stating, among other things, what kind of credentials are accepted and which attributes must be disclosed (the `AttributeType` of the `DisclosedAttribute` tag). To illustrate our protocol specification with an example, let us suppose that a user already possesses an electronic library card, where the secret related to her credential (possibly together with the credential itself) is stored. [4]

An example of the presentation policy for this scenario is presented in Table 3-1. In this case, the Verifier requires the use of a *scope-exclusive pseudonym* and a credential issued by `urn:sweden:id`, both bound to the same secret key. In addition, the User must *disclose* the value of the attribute `city` contained in her *id* and *prove* that she was born before the given date (proving that she is older than 18 years old without disclosing her birthday).

As a response, the User computes the necessary presentation token fulfilling the given presentation policy and sends it back to the Verifier, as shown in Table 3-2. As we can see, the User discloses the city attribute from her Swedish id, which in this example is *Soderhamn,* while also performing a predicate over the value contained in the attribute date of birth from her id with the `date-less-than` function. In this case, the Verifier can only verify that the User is older than 18 years old, namely whether the date of birth of the User is before the given date (1996-11-05[5]).

[4] The example presented here and parts of the illustrating XML code presented in this section are adapted from a previously published technical report in [CDL+12]. For a more detailed description on the materials in this chapter, you can also refer to the original contributions, as referenced here.

[5] Assuming that the date of today is November 6th, 2014. Otherwise, this date should then be updated automatically in the presentation policy, depending on the day of access.

Table 3-1: An example of a presentation policy sent from the Verifier to the User, adapted from [CDL+12]

```
…
<PresentationPolicy PolicyUID="policy1" KeyBinding="true">
    …
<Pseudonym Alias="nym" Scope="http://sweden.gov/poll0105" Exclusive="true"/>
    <Credential Alias="id" SameKeyBindingAs=”nym”>
        <CredentialSpecAlternatives>
            <CredentialSpecUID>urn:sweden:id</CredentialSpecUID>
        </CredentialSpecAlternatives>
        <IssuerAlternatives>
            <IssuerParametersUID>urn:sweden:id:issuer</IssuerParametersUID>
        </IssuerAlternatives>
        <DisclosedAttribute AttributeType="urn:sweden:id:city"/>
    </Credential>
    <AttributePredicate Function="urn:oasis:names:tc:xacml:1.0:function:date-
less-than">
        <Attribute CredentialAlias="id" AttributeType="urn:sweden:id:bdate"/>
        <ConstantValue>1996-11-06</ConstantValue>
    </AttributePredicate>
    …
</PresentationPolicy>
…
```

Table 3-2: An example of a presentation token generated by the User as a response to the presentation policy from Table 4.1, adapted from [CDL+12]

```
…
<PresentationToken xmlns="http://abc4trust.eu/wp2/abcschemav1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc"
    xsi:schemaLocation="http://abc4trust.eu/wp2/abcschemav1.0 schema.xsd"
    Version="1.0">

    <PresentationTokenDescription PolicyUID="policy1" KeyBinding="true">

        <Pseudonym Alias="nym" Scope="http://sweden.gov/poll0105" Exclusive="true">
            <PseudonymValue>MER2VXpyR0VaOW51YXdVNHRISHI=</PseudonymValue>
        </Pseudonym>
        <Credential Alias="id">
            <CredentialSpecUID>urn:sweden:id</CredentialSpecUID>
            <IssuerParametersUID>urn:sweden:id:issuer</IssuerParametersUID>
            <DisclosedAttribute AttributeType="urn:sweden:id:city">
                <AttributeValue>Soderhamn</AttributeValue>
            </DisclosedAttribute>
        </Credential>
        <AttributePredicate Function="urn:oasis:names:tc:xacml:1.0:function:date-
less-than">
            <Attribute CredentialAlias="id" AttributeType="urn:sweden:id:bdate"/>
            <ConstantValue>1996-11-06</ConstantValue>
        </AttributePredicate>
    </PresentationTokenDescription>

    <CryptoEvidence>   </CryptoEvidence>

</PresentationToken>
…
```

4 Integration with existing identity management schemes

It is important to note that the definition of ABC4Trust's own XML schema does not prevent Privacy-ABCs to be used in conjunction with existing IdM frameworks, like WS-*, SAML, OpenID, OAuth and X.509, so it won't require medications in the existing standards. Chapter 8 of Deliverable D2.2 [D2.2] is dedicated to this topic, which describes that "most of the privacy concerns occur in cross-domain data sharing, i.e., when information travels from one domain to another. Therefore, an ABC "proxy" can be used as a privacy filter between domains using well-known federated token transformer pattern (such as the WS-Trust STS). This is useful to avoid modifying legacy applications and infrastructure, and still benefit from the security and privacy properties of Privacy-ABC technologies."

Privacy-ABCs offer a plethora of features, some of which are complex to use. While it is these features that make Privacy-ABCs so powerful and allow for an effective protection of privacy, this feature richness makes them more complex, making them harder to use and to adopt. One means to make their adoption easier is make them interoperable with existing infrastructures such as for instance OpenID, OpenID Connect, OAuth, SAML and LDAP based authentication system. More precisely, it should be easy to consume Privacy-ABCs using existing formats and protocol interfaces (OpenID, OpenID Connect, OAuth, SAML) on the one hand, and also to issue Privacy-ABCs using existing attribute and identity sources (such as LDAP), on the other hand. In other words, translations from and to exiting standards and quasi standards are needed to help the adoption of Privacy-ABCs. We discuss these aspects in more detail in the following sections, but it must be pointed out that both of these issues are rather meant to identify software components that need to be made available in order to achieve the desired interoperability than standardization contribution in the strict sense.

4.1 Integration for consumption of user attributes (Relying Parties)

The decision of whether to grant users access to resources is taken based on a large variety of criteria, including the users' usernames, attributes, etc. These attributes are often requested using protocols and formats such as OpenID, OpenID Connect, OAuth or SAML or by querying LDAP databases.

Thus, to foster adoption of Privacy-ABCs, translations need to be made to such formats and protocols to the formats and protocols that ABC4Trust has defined. This involves the translation of the attribute request to presentation policies and then from the presentation policies back to attribute answers. This translation will have to be done by some software components that speak to existing protocols and formats on one end and the ABC4Trust protocols and formats on the other end. Depending on the protocols, some of this translation can be done automatically; for other protocols additional information might be required that an application developer will have to specify upon setting-up the translation components.

4.2 Integration for issuance of attributes (Identity Service Providers)

User attributes and authentication information are typically stored in some database such as LDAP that is consulted in authentication and authorization processes. Further, attribute and identity service providers currently use different protocols and formats to provide attribute information about the user, such as OpenID, OpenID Connect, OAuth or SAML. In order for these providers to be able to issue Privacy-ABCs, integration with such protocols and systems is necessary. Ideally, it should be sufficient to point the issuance service to an LDAP server in setup or to an OpenID provider, and then the system should be up and running. That is, an ABC4Trust credential specification that will specify the attribute types in the attribute source should be derived automatically from the LDAP server of

OpenID provider. Furthermore, to issue the credential to users, the standards authentication process towards the LDAP database or the OpenID provider should be considered automatically before the issuance protocol of ABC4Trust is run.

5 Standardization of the RI Components

Reference Implementation contains a large amount of code that has been used and improved in real applications so far. Two logically separate components of the Reference Implementation make more sense to standardize in international standards, one being the data formats and the APIs, and the other one being the implementations of the cryptographic constructs.

5.1 Standardization of the policy language and data formats

The ABC4Trust architecture [D2.2] describes in details the available APIs that can be used. The input and return types used in the API refer to the XML artefacts defined in Chapter 4 of [D2.2], unless standard data types like *boolean* or *string* are used.

Indeed, the XML data formats and the API definitions implemented in the Reference Implementation (RI) represent the contribution with the highest potential for standardization. The first step would be finding the relevant standardization projects, but this has been an issue until now. There seem to be some standardization efforts inside OASIS or W3C that might have some relevance, but none has resulted in being reasonably satisfactory for our topic. We consider such efforts to be deliberate decisions with a clear impact rather than a goal in and of itself.

As mentioned before, all the formats that ABC4Trust specified would have to be standardized so that higher-level application can make use of the ABC4Trust technology. Because the language to express presentation policies is rich and powerful, it has lots of different options and requires many things to be specified. For simple applications such a powerful language is not completely necessary and may actually hinder its adoption, because it needs to be properly understood before it can be used. Instead, it would make sense to standardize a number of simple default policies for the most typical uses cases such as age verification, where it is just verified that users are above a certain age (e.g., over 12, over 16, over 18, over 21, etc.).

5.2 Standardization of the cryptographic libraries

The cryptographic realization of Privacy-ABC technologies consists of a number of different cryptographic primitives including *signature, encryption, revocation,* and *commitment* schemes that are then composed with so-called *zero-knowledge proof* protocols. For such a composition to work, the cryptographic primitives need to have additional properties that ordinary signature and encryption scheme miss. Nevertheless, for most of the cryptographic primitives appearing in the ABC4Trust crypto architecture, there exist different implementations, e.g., for the signature scheme one can use Brands signatures, RSA-based Camenisch-Lysyanskaya signatures, or ECC-based Camenisch-Lysyanskaya signatures.

Thus, it seems natural that all these primitives be standardized, both on a high-level in terms of what properties they need to provide as well as on a specific level defining the implementation. Additionally, one will have to standardize how these components are then combined to realize a given presentation policy. This will end up in a quite complex framework of standards.

Such standardization will be useful if there were crypto implementations by many different providers. However, as the ABC4Trust Reference Implementation (including the Cryptographic Engine) is available in source code and can be used even commercially free of charge, this standardization seems not to be urgently needed.

6 Standardization for browsers and mobile phones

During the development of the ABC4Trust Reference Implementation, the intended platform of the user has been a laptop or desktop computer along with a smart card. However, during recent years, users have adopted smartphones and tablets on a massive scale and expect the same functionality on these devices as they have on their laptops and desktops.

Mobile devices do not offer the same hardware and software features as on laptop and desktop platforms and this presented some issues when the ABC4Trust reference implementation was ported to mobile devices (see deliverable "*D4.4 Smartphone feasibility analysis"* [D4.4]) – not only that certain libraries were not available, but also that the use of certain hardware, e.g. smart card readers, is not possible to the same extent as on laptop and desktop platforms.

In this chapter we will discuss some of the software and hardware features that are needed in order to implement the reference implementation on mobile platforms, and to what extent they are currently available on mobile devices and how we would preferably see that they are made available in the future.

6.1 Availability of cryptographic libraries

One important feature is the availability of cryptographic libraries and libraries for arbitrary precision integer arithmetic. On all platforms, both JavaScript and native platforms, there are such libraries. However, for iOS and OSX this relies on third-party libraries and for JavaScript the W3C Cryptographic API[6] only provides some cryptographic functions and this API is, except for a random number generator, not widely implemented. Third party JavaScript cryptographic libraries do exist - most notably Google's End-to-End[7] and Microsoft research's JavaScript Cryptography Library[8], and both of these libraries also offer integer arithmetic with arbitrary precision.

Ideally it would be preferable to have such libraries implemented in the browser, e.g. as a part of the W3C Cryptographic API, because it would provide better performance and might also be more secure due to improved memory handling.

6.2 Smart cards

In the ABC4Trust pilot projects the user interface was a website that communicated with the reference implementation and a smart card reader through a browser plugin. Such plugins are not available on browsers for mobile platforms, so such an architecture, where the user's credentials are bound to a smart card, is not possible to implement for mobile devices. While it is possible to implement the Reference Implementation in JavaScript, it will *not* be possible to use smart cards.

In native mobile apps it is possible to use the NFC-chip (available on some Windows and Android mobile devices) to communicate with smart cards. Some devices also have a so-called secure element, which is a smart card embedded in the device that act as a secure storage for the user's private information. The API of the secure element is typically limited (if even available for app developers),

[6] http://www.w3.org/TR/WebCryptoAPI/
[7] https://code.google.com/p/end-to-end/
[8] http://research.microsoft.com/en-us/downloads/29f9385d-da4c-479a-b2ea-2a7bb335d727/

and cannot be used for custom cryptographic protocols, such as the Privacy-ABC technologies used in ABC4Trust.

In order to maintain the same security model and usability for a JavaScript implementation of the ABC4Trust reference implementation, we would need smart card support through the browser. This has been discussed in the W3C consortium[9], but is currently not standardized.

6.3 Conclusion

We conclude that for a native implementation of the reference implementation on mobile platforms, the needed features are already there. A better API for accessing the secure element would be preferable, but workarounds, e.g. using an NFC-chip as a smart card reader, are possible.

If we were to implement the reference implementation as a web application in JavaScript, we would need more features. The most critical is that there is no support for smart cards on this platform and we would, hence, have to use other less secure options that are not hardware-backed, such as storing sensitive data in the browser's storage, or maybe use a key derived from a secret password only known by the user. Another issue is the availability of libraries for cryptographic operations and arbitrary precision integer arithmetic, where third party software is available, but for performance and security reasons it would be preferable if these were implemented as part of the browser.

[9] http://www.w3.org/2012/webcrypto/webcrypto-next-workshop/.

7 Summary and outlook

ABC4Trust is a research project bringing together industry, academia and others to address the federation and interchangeability of technologies that support trustworthy yet privacy-preserving Attribute-based Credentials (ABC). Attribute-based Credentials allow a holder to reveal a minimal set of personal information to Relying Parties, helping to provide a higher degree of user privacy in the digital world. Thus, these credentials facilitate the simultaneous implementation of a trustworthy and privacy-protecting mechanism for a safer digital society.

Standardization is an important consideration for ABC4Trust, to which we have paid particular attention during the project. We have identified some of the components of the Reference Implementation with high standardization potential, including the policy language, data formats and the externally available APIs, and realizations of different cryptographic libraries.

Besides standardizing components of the Reference Implementation, achieving interoperability with successful identity management schemes can also ease the adoption of the Reference Implementation. We discuss in a bit more detail the approach to achieving interoperability for issuing attributes, as well as consuming presentation tokens produced by the Reference Implementation.

On top of that, we have described additional factors that could enable an easier integration of the RI in existing browsers and mobile platforms. These have been based on the challenges we have faced when studying the feasibility of this approach, which include a lack of native browsers support for cryptographic operations used by the RI. Finally, we also give insights on the secure storage of the Privacy-ABCs when used in mobile platforms instead of smart cards.

8 References

[CDL+12] Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. *A Language Framework for Privacy-Preserving Attribute-Based Authentication.* Research Report, RZ3818, 2012.

[D2.2] Ahmad Sabouri (ed.). *D2.2 Architecture for Attribute-based Credential Technologies – Final Version*, 2014 (ABC4Trust deliverable).

[D4.1] Hans Guldage and Janus Dam Nielsen. *D4.1 Initial Reference Implementation*, 2012 (ABC4Trust deliverable).

[D4.2] Thomas Baignères, Patrik Bichsel, Robert R. Enderlein, Hans Knudsen, Kasper Damgård, Jonas Jensen, Gregory Neven, Janus Nielsen, Pascal Paillier, Michael Stausholm. *D4.2 Final Reference Implementation*, 2014 (ABC4Trust deliverable).

[D4.4] Jonas Lindstrøm Jensen. *D4.4 Smartphone feasibility analysis*, 2014 (ABC4Trust deliverable).

[KP12] Ioannis Krontiris, Apostolos Pyrgelis. *Menschen vergessen – Systeme nicht, Mehr Datenschutz für den Einzelnen mit ABC4Trust*. PHP Magazine, Issue 5, July 2012.

[P2ABC] Online repository for the code and documentation of the reference implementation. Available in GitHub at https://github.com/p2abcengine/p2abcengine.