Concepts Around Privacy-Preserving Attribute-Based Credentials

Making Authentication with Anonymous Credentials Practical

Jan Camenisch

IBM Research, Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland

Abstract. This article provides a short overview of the concepts around privacy-preserving attribute-based authentication. It then briefly discusses the cryptographic realisation of these concepts and describes an architecture implementing them.

1 Introduction

The Internet has transformed our environment and how we interact with each other dramatically. Soon all things surrounding us will be part of the Internet, producing, processing, providing, and consuming enormous amounts of data. It seems impossible to protect all devices and systems, virtually or physically. Therefore secure authorisation and communication and protecting stored data are of vital importance. To this end, it is necessary to authenticate and encrypt every single bit as well as explicitly define who is allowed to do what with the bit, i.e., to attach a data-usage policy to each bit. However, authenticating every bit communicated will most probably decrease users' privacy substantially and, more generally, the security of the overall system.

To alleviate this, privacy-preserving authentication mechanisms, in particular attribute-based authentication together with anonymous credentials, should be applied. Unfortunately, today they are not employed in practice. One reason for this might be the complexity of privacy-preserving authentication mechanisms due to the large numbers of features they provide. Also, the security properties, such as unlinkability, of privacy-preserving authentication mechanisms are often counterintuitive. To address this, the ABC4Trust project [1] has put forth a number of concepts that capture, simplify, and unify the properties of privacy-preserving authentication technologies. These concepts aim to make these authentication technologies easier to understand, deploy, and use. The ABC4Trust project also runs two pilots to show the applicability of these technologies in real-world scenarios [1,15].

In this article we summarise and explain these concepts. To this end, we first discuss the basic authentication scenario and the entities involved. Then we discuss each of the concepts. Next, we describe the realisation of these concept in the ABC4Trust architecture and the reference implementation. Finally, we

M. Hansen et al. (Eds.): Privacy and Identity 2014, IFIP AICT 421, pp. 53–63, 2014.

conclude with a discussion on obstacles that need to be overcome in order for privacy-preserving authentication to be used practice.

2 Authentication Scenario and Entities

The basic entities of a privacy-protecting authentication system with attribute-based credentials (privacy-ABC system) include a user, an issuer (often called identity provider), and a verifier (often called relying party). These entities essentially occur in any authentication scenario, in particular also if authentication is performed with X.509 certificates or the OpenID protocol. The remaining two entities are the revocation authority and the inspector. The former also occurs in traditional certificate-based authentication systems, whereas the latter is a specialty of privacy-preserving authentication. All these entities are depicted in Figure 1. In a real deployment, there are further entities involved, for instance the providers of the computing platforms, different software components, and a public key infrastructure, etc. In this article, however, we do not consider such entities as they are not particular to privacy-ABC systems.

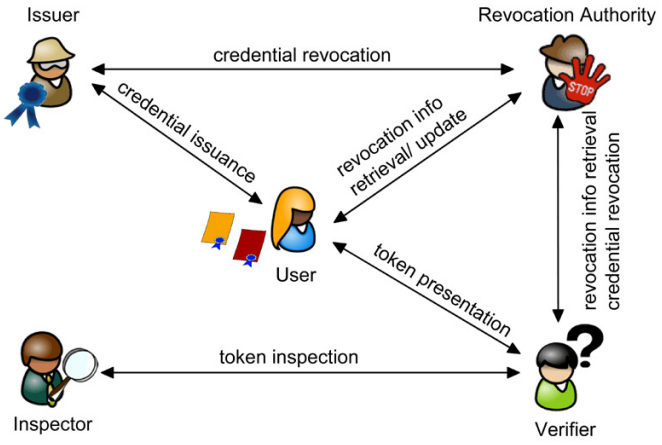

Fig. 1. Entities and the interactions between them [4]

These parties each perform a number of tasks and interact with each other to make the authentication system work. First, to initiate the overall system, 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 authority parameters. It is assumed that all entities have a means to retrieve the public keys of the issuers, revocation authorities, inspectors, and verifiers, e.g., by using some form of public key infrastructure.

Most parties interact with each other, as can be seen from Figure 1, where these interactions are depicted and named according to their purpose. Now, depending on which (privacy-preserving) authentication technology is used, these interactions might be realised differently and consist of only a single flow or an interactive protocol. Also the time at which they occur depends on the technology used.

The first interaction, the *credential issuance* protocol, allows a user to obtain credentials from an issuer. A credential contains attributes that its issuer vouches for with respect to 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 authority prior to 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 does not require the user to interact with the issuer! However, the user might need to retrieve information from the revocation authority, depending on the specific revocation scheme used. 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 which conditions the inspector should reveal these attributes. Upon receipt of a presentation token from a user, a verifier checks whether the presentation token is valid with respect to the relevant issuers' public keys, the inspectors' 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 forward the token to the inspector who will then follows the instructions defined in the inspection grounds, e.g., to reveal the attribute to a designated party.

Informally, a secure realisation of a privacy-ABC system guarantees (1) that 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.

3 Concepts

In this section we provide a brief explanation of the main features of privacy-ABCs. We start by explaining what we mean by an identity as all the concepts are based on this view of identity. We then discuss the concepts underlying the authentication and authorisation with a privacy-ABC system.

3.1 Attributes and Identities

For the purpose of this exposition, we consider an *identity* to consist of the *attributes* that another party knows linkable a user, say Alice. Figure 2 shows some identities that Alice has with some parties. For instance, she has an identity with her employer comprising her full name, salary, address, education, and the languages she speaks. Her identity with an on-line shop may comprise her name, address, credit card number, and purchase history. Her identity with an on-line forum may comprise a nickname and her hobbies. There might of course be many other people who have a similar or the same identity as Alice with some party, i.e., the set of attributes that the party knows about Alice is the same as the set of attributes the party knows of another person, say Jim. Thus, the party can per se not distinguish whether it is communicating with Jim or Alice. Often parties ensure that this does not happen by requiring some attributes to be unique, e.g., by requiring users to provide different nick names. Nevertheless, we do not want to rule out that different users can have the same identity with the same party or with different parties.

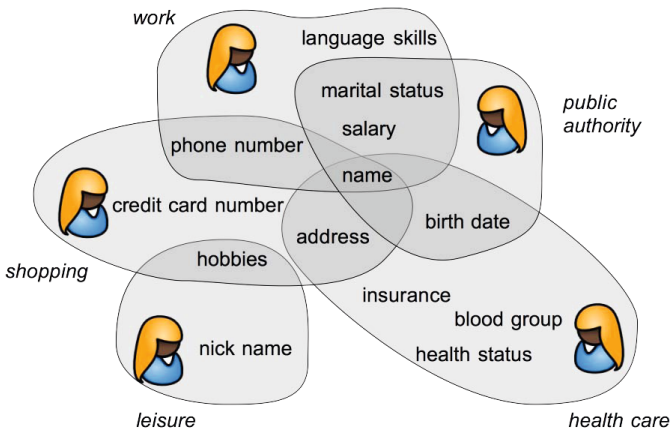

Fig. 2. An identity is a collection of attributes someone knows about. Shown here are some of Alice's identities [7,14]

Now, for this concept of identities is to be useful, we require two basic mechanisms: one that allows a user to authenticate as the owner of an identity and one that allows a user to transfer an attribute from one identity to another identity, i.e., to let the latter learn an attribute that the former entity knows and vouches for.

These mechanisms shall protect the privacy of the user, i.e., the parties with which the user has established these identities shall not be able to link them – unless this is implied by the uniqueness of an attribute value or a set of attribute values.

While we consider an identity to be a set of attributes that someone knows about a user, the same set of attributes can be different identities. That is, a

party might know the same set of attribute values about two different users and indeed should consider these to be different entities. Similarly, two parties who know a single or two users by the same set of attribute values, should consider these as different identities. Here the concept of the authentication mechanism becomes especially useful. If the specific authentication parameters (e.g., public key or pseudonym value) associated with different sets of attribute values are identical, then the identity should considered to be the same. Also, a user can link different identities of hers together using the authentication mechanism by either using the same authentication parameters with the different identities or, preferably, by using specific properties of the authentication mechanism to prove that she is the holder of both identities. We discuss such mechanisms next.

3.2 Pseudonyms

To authenticate as the owner of an identity, a user can establish a (cryptographic) *pseudonym* with a party. Technically, a pseudonyms is essentially a public key of a cryptographic identification scheme. However, different from a traditional cryptographic identification scheme where there is a single secret key and public key pair, here an unlimited number of pseudonyms (public keys) can be derived from the same secret key. Pseudonyms are unlinkable, i.e., a party cannot tell whether two given pseudonyms correspond to different secret keys. While technically, a user can also have different secret keys, it is instructive to consider each secret key to define a separate user – very much like a real-world passport defines a person's identity.

In some situations, however, the possibility to generate an unlimited number of unlinkable pseudonyms is undesirable. For instance, a verifier might want to allow only one account per user. To still support privacy in such cases, ABC technologies allow *scope-exclusive* pseudonyms. Such pseudonyms are generate from the secret key and a scope identifier (string) and are unique per scope for a given user secret key. Nevertheless, scope-exclusive pseudonyms for different scopes remain unlinkable.

3.3 Credentials

A credential is a set of attributes that is (digitally) signed by an issuer. While in principle an issuer could first generate a credential and then send it to the user, e.g., by email, privacy-ABCs typically require an interactive issuance protocols to realise the enhanced privacy properties to ensure that the issuer does not learn the secret key of the user and, for some special applications, to allow the user to keep some of the attributes hidden from the issuer. An example of such a special application is one-show credential: here the user encodes a random attribute in the credential that is hidden during issuance but is require to be revealed during presentation.

The validity of a credential can be verified with regard to the issuer parameters published by the issuer and a credential specification. The latter defines the semantic of a credential such as what attribute types the credential contains.

While the issuer parameters are specific for each issuer, a credential specification can be shared by many issuers. The credential specification further defines whether a credential is revocable and/or bound to a key of the user. We discuss these two features next.

3.4 Binding Credentials to Keys

When a credential specification requires that a credential be key bound, the issuance protocol will require the user to input a secret key to which the credential will be issued without the issuer learning any information about the secret key. A key-bound credential can only be presented with knowledge of the secret key. Per se, the secret key can be any secret of the user's choice. However, for some applications, it might make sense to restrict the user in this choice. To this end, the issuer can specify in the *issuance policy* that the credential be bound to the same key as some other credential(s) or to the secret key underlying some particular pseudonym that the user has sent to the issuer earlier. For instance, to enforce that all of a user's credentials be bound to the same secret key, one could require each user to register with some root authority from which a user would get a key-bound (root) credential. Later, whenever some issuer issues a credential to a user, he would specify that the credentials issued be key-bound to the same key as a root credential.

3.5 Revocation of Credentials

There might be many reasons why a credential should be revoked. A user might have lost the right that the credential attested to her or her secret key and all her credentials were compromised. In this case, the issuer(s) of the credential(s) concerned must be able to invalidate them. For ordinary credentials, this is typically done by some whitelist or blacklist containing the serial numbers to the valid or invalid credentials, respectively, or by letting credentials be valid for only a short time. The latter works for privacy-ABC as well, whereas the former does not as it would require users to reveal the serial number of their credential and thus all privacy would be lost again. Fortunately, the cryptographic literature provides several mechanisms that allow users to convince a verifier that the serial number of their credential is on a whitelist or not on a blacklist. These mechanisms all have in common that the issuer publishes some public revocation information which both users and verifiers should consult. Some mechanisms further require that users be able to retrieve specific information related to their credential so that they will be able to perform the proof of validity of their credential. We refer to this as issuer-driven revocation.

Sometimes, however, a credential might not need to be revoked globally but rather some verifier might want to stop accepting a credential from a specific user. For example, a hooligan may see his digital identity card revoked for accessing sport stadiums, but may still use it for all other purposes. The specification therefore also allows for verifier-driven revocation Here, verifiers can specify their own lists and have users prove to them whether or not they figure on such a list.

The ABC4Trust specifications define revocation very generically and just define a revocation authority who manages and publishes the revocation information. The only difference between issuer-driven and verifier-driven revocation is that the former is performed based on the revocation handle (which is treated as a dedicated attribute), whereas the latter can be performed based on any attribute value or even a combination of values, possibly even from different credentials.

3.6 Presentation Policy and Presentation Token

The probably most important difference between privacy-ABCs and ordinary credentials is how a user authenticates to a verifier with them. In an ordinary PKI, the user sends her credentials (thereby revealing all attribute values!) to the verifier and authenticates as the holder of the credentials. The verifier then verifies the validity of the credentials and whether or not the attributes contained therein match his access control requirements.

A privacy-ABC system is much more privacy friendly than such traditional approaches. It first requires the verifier to specify which attributes certified by whom it requires from the user. We call this statement the *presentation policy*. In fact, the policy allows an even more privacy-friendly option. Instead of requiring a user to reveal an attribute, the verifier could request only a predicate over an attribute such as "over 18" instead of "reveal birthdate" or just ask that the last name in one credential be the same as in another credential without having to reveal the value of the last name. It can further be specified that a user also send a pseudonym and authenticate with regard to it. A presentation policy further allows one to restrict verifiers in what attributes they learn, e.g., by requiring that policies be signed by a data protection authority.

In a second step, once the user has received the presentation policy from the verifier, the user can decide whether she wants to reveal this information to the verifier, and if so, which credentials she wants to use (in case multiple credentials apply) provided she possesses all necessary credentials. If not, the user will have to somehow get the missing credentials issued. To support the user in these tasks, ABC4Trust provides a graphical user interface (identity selector) showing the user the different choices. Once the user has made her choices, she creates a *presentation token* from the credentials she selected. A presentation token can be seen as a transformed (set of) credential(s) that contains only the attributes from the original credential(s) that the user wishes to reveal. Cryptographically, a presentation token verifies with regard to the signature verification keys of original credentials' issuers, just like the original credentials themselves.

3.7 Inspectable Presentation Tokens

In some situation, the information required from a user changes depending on the behaviour of the user. So, while it might be fine that a user can access some service by convincing the service provider that she has obtained a subscription credential, it might be necessary that additional information be available in

case of abuse. To address such scenarios, a presentation policy can state that certain attributes need to be provided in encrypted form, encrypted under the public key of a designated party – the inspector. That is, the verifier will not be privy to inspectable attributes unless he forwards the presentation token to the designated inspector and the *inspection grounds* stated in the presentation policy are met. An inspection ground is a free format text that cannot be modified and that the inspector is expected read and comply with before handing the decrypted attributes to the verifier.

3.8 Issuance of Credentials

As mentioned, a credential is issued in a protocol between the user and the issuer. Issuing a credential is in some sense just a special case of providing a service and so an issuer might require a user to present a number of credentials issued by other parties or to authenticate with regard to a pseudonym beforehand. While in many cases, the issuer might know all the attributes values of the issued credential, that is not always necessary and sometimes even undesirable. Therefore, ABC4Trust provides an advanced issuance protocol that allows the issuer to "carry over" attribute values from credentials that a user already possesses into the credential that gets issued without the values getting revealed. To enable this technically, the presentation policy is extended into an *issuance policy*, and accordingly the user will generate an *issuance token*.

4 Realisation

The concepts discussed in the preceding section can be implemented with a number of cryptographic schemes and algorithms, such as credential systems, group signatures, blind signatures, and verifiable encryption. More precisely, a full-fledged privacy-ABC system that realises all the concepts can be built from signature schemes with special properties (e.g., Camenisch-Lyskanskaya signatures [8,9,10] and Brands signatures [3]), commitment schemes, verifiable encryption schemes [11], and generalised Schnorr proofs [5].

As the combined cryptographic protocols get quite complex and therefore hard to use, the ABC4Trust project developed a policy language to deal with these concepts and to orchestrate the cryptographic protocols. The ABC4Trust privacy-ABC system thus can be used merely by understanding the concepts and not having to worry about the underlying cryptography. More precisely, the architecture provides XML schemas for all artefacts, including issuer parameters, revocation authority parameters, inspector parameters, credential specification, issuance policies and tokens, presentation policies and tokens, and pseudonyms [6]. The architecture further defines the components of a privacy-ABC system and their interfaces. The (main) components of the user and the verifier are depicted in Figure 3.

The architecture has three layers: an application layer, a policy layer and a crypto engine layer.

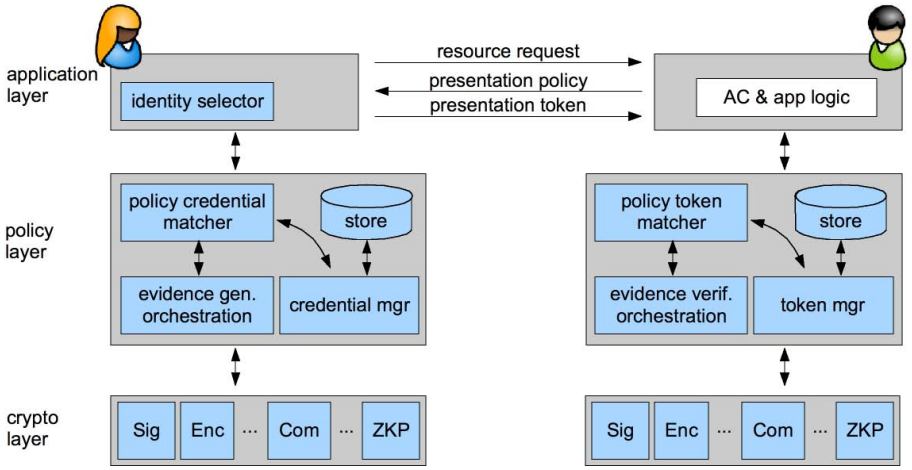

Fig. 3. ABC4Trust architecture, partial view on user and verifier [4,6]

The application layer is the consumer of the privacy-ABC system, it could be a browser on the user's side and a web service on the verifier's side. The application layer interacts with the policy layer by means of the APIs provided and is responsible for exchanging policies and tokens between the parties. How this is done is outside of the scope of the ABC4Trust architecture. On the user's side the application layer is also responsible for the presenting the presentation (and issuance) policy to the user and for allowing her to make her choices.

The policy layer processes presentation and issuance policies, matches credentials against policies and tokens against polices (policy credential matcher and policy token matcher), and orchestrates the generation and verification of presentation and issuance tokens (evidence generation orchestration and evidence verification orchestration). The policy layer is also responsible for storing a user's credential and tokens that a verifier receives (store) and for managing the credentials and tokens (credential manager and token manager). The latter includes dealing with revocation information and updating the credentials accordingly.

The crypto engine layer implements the cryptographic operation of privacy-ABCs. It contains the u-Prove and the IBM identity mixer (idemix) signature schemes (Sig), the idemix verifiable encryption (Enc) and revocation schemes, pseudonyms, commitment schemes (Com), and various cryptographic mechanisms to prove and verify attribute predicates, including zero-knowledge proof protocols (ZKP). The forthcoming ABC4Trust crypto architecture will describe this in detail. In the mean time, the reader is referred to the identity mixer specifications [2,13], which can be seen as a preliminary versions of this architecture.

The reference implementation of ABC4Trust encompasses all components of the policy and crypto layer, the identity selector, as well as an example application. The reference implementation is available from the GitHub repository [12] (and the sites linked from there).

5 Conclusion

We believe that privacy-ABCs should be the default technology to be used to implement any form of access control and that they will be as essential for a secure Internet just as there would be no e-commerce without SSL (Secure Socket Layer). The technology per se is ready for this. Indeed, the ABC4Trust project is currently conducting two pilots that, on the one hand, validate the architecture and the reference implantation and, on the other hand, show that privacy-ABC technology is ready to be used in practice. Also, a number of other research groups have successfully run pilots and demonstrator showing that the technology is ready for deployment. A number of obstacles, however, remain to be overcome before the privacy-ABCs will be in widespread use.

First of all, privacy-ABCs are more complex than ordinary attribute-based credentials, and their features are somewhat counterintuitive. This makes them challenging to deploy and use. To address this, the complexity of these technologies needs to be reduced and their possibilities communicated to the various stakeholders.

Furthermore, to enable the use of privacy-ABCs, the different cryptographic mechanisms and the policy languages need to be standardised. The architecture of ABC4Trust is a step towards this goal.

The obstacle that is probably the most difficult to overcome is the design of intuitive user interfaces for privacy-ABCs. A few approaches have been proposed, but for all of them it seems that users were not able to clearly understand what information they will reveal to verifiers (see, e.g., [16]).

Finally, the speed with which the Internet evolves and new applications are introduced and embraced makes it very challenging to address the emerging security and privacy problem, in particular because privacy and security too often are not taken into consideration by design. Thus, without privacy becoming a mandatory design principle, privacy technology will not be used as it is always easier to build applications without addressing security and privacy. Fortunately, the general public is becoming increasingly aware of the need of proper security and privacy protection and we have reason hope that future applications will only succeed when taking this into account.

Acknowledgements. The author enjoyed many discussions with his IBM colleagues and the people participating in the ABC4Trust project. The paper benefitted a lot from the comments by the reviewers. Thank you! This work was supported by the EC-funded project ABC4Trust.

References

1. ABC4Trust – Attribute-based Credentials for Trust, `http://www.abc4trust.eu`
2. Bichsel, P., Camenisch, J.: Mixing Identities with Ease. In: de Leeuw, E., Fischer-Hübner, S., Fritsch, L. (eds.) IDMAN 2010. IFIP AICT, vol. 343, pp. 1–17. Springer, Heidelberg (2010)

3. Brands, S.: Rethinking Public Key Infrastructure and Digital Certificates – Building in Privacy. PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands (1999)
4. Camenisch, J., Dubovitskaya, M., Lehmann, A., Neven, G., Paquin, C., Preiss, F.-S.: Concepts and Languages for Privacy-Preserving Attribute-Based Authentication. In: Fischer-Hübner, S., de Leeuw, E., Mitchell, C. (eds.) IDMAN 2013. IFIP AICT, vol. 396, pp. 34–52. Springer, Heidelberg (2013)
5. Camenisch, J., Kiayias, A., Yung, M.: On the Portability of Generalized Schnorr Proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009)
6. Camenisch, J., Krontiris, I., Lehmann, A., Neven, G., Paquin, C., Rannenberg, K., Zwingelberg, H.: D2.1 Architecture for Attribute-based Credential Technologies – Version 1, `http://abc4trust.eu/download/ABC4Trust-D2.1-Architecture-V1.2.pdf`
7. Camenisch, J., Lehmann, A., Neven, G.: Electronic Identities need Private Credentials. IEEE Security & Privacy 10(1), 80–83 (2012)
8. Camenisch, J.L., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
9. Camenisch, J.L., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
10. Camenisch, J.L., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
11. Camenisch, J.L., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
12. Github Repository. Privacy-preserving attribute-based credential engine (p2abcengine), `https://github.com/p2abcengine/p2abcengine/`.
13. IBM Research. Specification of the Identity Mixer Cryptographic Library – Version 2.3.0. IBM Technical Report RZ3730
14. Leenes, R., Schallaböck, J., Hansen, M.: PRIME White Paper – third and final version (2008), `https://www.prime-project.eu/prime_products/whitepaper/`
15. Stamatiou, Y.: Privacy Respecting ICT Innovation in Education: Electronic Course Evaluation in Higher Education and Beyond. In: IFIP Summer School 2013 on Privacy and Identity Management for Emerging Services and Technologies. IFIP AICT. Springer (2014)
16. Wästlund, E., Fischer-Hübner, S.: The Users' Mental Models' Effect on their Comprehension of Anonymous Credentials. In: Privacy and Identity Management for Life. Springer (2011)