Formal Treatment of Privacy-Enhancing Credential Systems

Jan Camenisch[1], Stephan Krenn[2(✉)], Anja Lehmann[1], Gert Læssøe Mikkelsen[3], Gregory Neven[1], and Michael Østergaard Pedersen[4]

[1] IBM Research – Zurich, Rüschlikon, Switzerland
{jca,anj,nev}@zurich.ibm.com
[2] AIT Austrian Institute of Technology GmbH, Vienna, Austria
stephan.krenn@ait.ac.at
[3] Alexandra Institute, Aarhus, Denmark
gert.l.mikkelsen@alexandra.dk
[4] Miracle A/S, Aarhus, Denmark
mop@miracleas.dk

Abstract. Privacy-enhancing attribute-based credentials (PABCs) are the core ingredients to privacy-friendly authentication systems. They allow users to obtain credentials on attributes and prove possession of these credentials in an unlinkable fashion while revealing only a subset of the attributes. In practice, PABCs typically need additional features like revocation, pseudonyms as privacy-friendly user public keys, or advanced issuance where attributes can be "blindly" carried over into new credentials. For many such features, provably secure solutions exist in isolation, but it is unclear how to securely combined them into a full-fledged PABC system, or even which properties such a system should fulfill.

We provide a formal treatment of PABCs supporting a variety of features by defining their syntax and security properties, resulting in the most comprehensive definitional framework for PABCs so far. Unlike previous efforts, our definitions are not targeted at one specific use-case; rather, we try to capture generic properties that can be useful in a variety of scenarios. We believe that our definitions can also be used as a starting point for diverse application-dependent extensions and variations of PABCs. We present and prove secure a generic and modular construction of a PABC system from simpler building blocks, allowing for a "plug-and-play" composition based on different instantiations of the building blocks. Finally, we give secure instantiations for each of the building blocks.

Keywords: Privacy · Anonymous credentials · Provable security

This work was supported by the Horizon 2020 project PRISMACLOUD under grant agreement no. 644962, and the FP7 projects FutureID and ABC4Trust under grant agreement nos. 318424 and 257782. Parts of this work were done while the second author was at IBM Research – Zurich. The full version is available online [1].

O. Dunkelman and L. Keliher (Eds.): SAC 2015, LNCS 9566, pp. 3–24, 2016.
DOI: 10.1007/978-3-319-31301-6_1

1 Introduction

Privacy-enhancing attribute-based credentials systems (aka PABCs, *anonymous credentials*, or *pseudonym systems*) allow for cryptographically strong user authentication while preserving the users' privacy by giving users full control over the information they reveal. There are three types of parties in a PABC system. *Issuers* assign sets of attribute values to *users* by issuing credentials for these sets. Users can present (i.e., prove possession of) their credentials to *verifiers* by revealing a subset of the attributes from one or more credentials. The verifiers can then check the validity of such presentations using the issuers' public keys, but they do not learn any information about the hidden attributes and cannot link different presentations by the same user. This basic functionality of a PABC system can be extended in a large number of ways, including pseudonyms, revocation of credentials, inspection, proving relations among attributes hidden in presented credentials, and key binding [2,3].

The importance of privacy and data minimization has been emphasized, e.g., by the European Commission in the European privacy standards [4,5] and by the US government in the National Strategy for Trusted Identities in Cyberspace (NSTIC) [6]. With IBM's Identity Mixer based on CL-signatures [7–10] and Microsoft's U-Prove based on Brands' signatures [11,12], practical solutions for PABCs exist and are currently deployed in several pilot projects [2,13–15]. In fact, numerous anonymous credential schemes as well as special cases thereof such as group signatures, direct anonymous attestation, or identity escrow have been proposed, offering a large variety of different features [8,10–12,16–24].

Despite this large body of work, a unified definitional framework for the security properties of a full-fledged PABC system is still missing. Existing schemes either have targeted security definitions for specific use cases [8,18–20] or do not provide provable security guarantees at all [12,21,22]. One possible reason for the lack of a generic framework is that dedicated schemes for specific scenarios are often more efficient than generic solutions. However, defining, instantiating, and re-proving tailored variants of PABCs is hard and error-prone. Clearly, it is desirable to have a unified definitional approach providing security definitions for a full-fledged PABC system. It turns out that achieving such definitions is far from trivial as they quickly become very complex, in particular, if one allows for relations between hidden attributes when issuing and presenting credentials, as we shall discuss. Nevertheless, in this paper we take a major step towards such a unified framework for PABCs by formally defining the most relevant features, detached from specific instantiations or use cases. We further provide a generic construction of a PABC system based on a number of simpler building blocks such as blind signatures or revocation schemes, and a formal proof that this construction meets our security definitions. Finally, we give concrete instantiations of these components, with detailed security proofs provided in the full version.

Considered Features. Our definition of PABC systems comprises the richest set of features integrated into a holistic PABC scheme so far. It supports credentials with any fixed number of attributes, of which any subset can be revealed

during presentation. A single presentation can reveal attributes from *multiple credentials*. Users can *prove equality* of attributes, potentially across different credentials, without revealing their exact values. Users have *secret keys* from which arbitrarily many *scope-exclusive pseudonyms* can be derived. That is, for a given secret key and *scope*, only one unique pseudonym can be derived. Thus, by reusing the same scope in multiple presentations, users can intentionally create linkability between presentations; using different scopes yields mutually unlinkable pseudonyms.

Credentials can optionally be bound to the users' secret keys to prevent users from sharing their credentials. For a presentation that involves multiple credentials and/or a pseudonym, all credentials and the pseudonym must be bound to or derived from the same user secret key, respectively. Issuers can *revoke credentials*, so that they can no longer be used. During issuance, some attribute values may be hidden from the issuer or *"carried over"* from existing credentials. This latter *advanced issuance* means that the issuer does not learn their values but is guaranteed that they are equal to an attribute in an existing credential.

Our Contributions. We give formal security definitions for a full PABC system that incorporates all of the features mentioned above. We provide a generic construction from lower-level building blocks that satisfies our definitions and we present secure instantiations of the building blocks.

In terms of security, informally, we expect presentations to be *unforgeable*, i.e., users can only present attributes from legitimately obtained and unrevoked credentials, and to be *private*, i.e., they do not reveal anything more than intended. For privacy, we distinguish weak privacy, where presentations of a credential cannot be linked to a specific issuance session, and the strictly stronger notion of simulatable privacy, where in addition presentations of the same credential cannot be linked to each other. This allows us to cover (slight variants of) the most prevalent schemes used in practice, U-Prove and Identity Mixer.

Formally defining these properties is far from trivial because of the complexity of our envisaged system. For example, user can obtain credentials on (*i*) revealed, (*ii*) blindly carried-over, and (*iii*) fully blind attributes. Each type comes with different security expectations that must be covered by a single definition. Carried-over attributes, for example, present a challenge when defining unforgeability: While the issuer never learns the attributes, the adversary must not be able to present a value that was not previously issued as part of a pre-existing credential. For privacy, one challenge is to formalize the exact information that users intend to reveal, as they might reveal the same and possibly identifying attributes in different presentations. Revocation gives the issuer the power to exclude certain credentials from being used, which must be modeled without cementing trivial linking attacks into the model that would turn the definition moot.

As our definitions are rather complex, proving that a concrete scheme satisfies them from scratch can be a challenging and tedious task. Also, proofs for such monolithic definitions tend to be hard to verify. We thus further define

building blocks, strongly inspired by existing work, and show how to generically compose them to build a secure PABC system. Our construction is efficient in the sense that its complexity is roughly the sum of the complexities of the building blocks. Additionally, this construction allows for simple changes of individual components (e.g., the underlying revocation scheme) without affecting any other building blocks and without having to reprove the security of the system. Finally, we give concrete instantiations for all our building blocks based on existing protocols.

Related Work. Our definitions are inspired by the work of Chase et al. [18,25], who provide formal, property-based definitions of delegatable anonymous credential systems and give a generic construction from so-called P-signatures [26]. However, their work focus on pseudonymous access control with delegation, but lacks additional features such as attributes, revocation, and advanced issuance.

PABCs were first envisioned by Chaum [16,27], and they have been a vivid area of research over the last decade. The currently most mature solutions are IBM's Identity Mixer based on CL-signatures [7–10] and Microsoft's U-Prove based on Brands' signatures [11,12]. A first formal definition [8] in the ideal-real world paradigm covered the basic functionalities without attributes and revocation. Their definition is stand-alone and does not allow composability as honest parties never output any cryptographic values such as a credentials or pseudonyms. This restriction makes it infeasible to use their schemes as building block in a larger system. These drawbacks are shared by the definition of Garman et al. [19].

A recent MAC-based credential scheme [20] allows for multiple attributes per credential, but requires issuer and verifier to be the same entity. It does not cover pseudonyms, advanced issuance, or revocation and provides rather informal definitions only. Similarly, Hanser and Slamanig [28] do not consider any of these features nor blind issuance, i.e., the issuer always learns all the user's attributes.

Baldimtsi and Lysyanskaya [29] define a blind signature scheme with attributes and claim that it yields an efficient *linkable* anonymous credential scheme, again without giving formal definitions. The scheme can be seen as a weakened version of our signature building block without unlinkability or extractability of hidden attributes – properties that are crucial for our PABC system.

Camenisch et al. [24] provide a UC-definition for anonymous credentials and a construction based on bilinear maps. However, their definition does not cover a full-blown credential systems but just a primitive to issue and verify signatures on attributes, i.e., a primitive we call privacy-enhancing signatures in this paper.

Finally, existing definitions of attribute-based signatures, e.g., [30–32] differ substantially from those of our building block for privacy-enhancing attribute-based signature (cf. Sect. 4.3), as again they do not consider, e.g., blind issuance.

2 Notation

Algorithms and parties are denoted by sans-serif fonts, e.g., A, B. For deterministic (probabilistic) algorithms we write $a \leftarrow \mathsf{A}(in)$ ($a \xleftarrow{\$} \mathsf{A}(in)$), if a is the output of A on inputs in. For an interactive protocol (A, B) let $(out_{\mathsf{A}}, out_{\mathsf{B}}) \leftarrow \langle \mathsf{A}(in_{\mathsf{A}}), \mathsf{B}(in_{\mathsf{B}}) \rangle$ denote that, on private inputs in_{A} to A and in_{B} to B, A and B obtained outputs out_{A} and out_{B}, respectively. For a set $\mathcal{S}$, $s \xleftarrow{\$} \mathcal{S}$ denotes that s is drawn uniformly at random from $\mathcal{S}$. We write $\Pr[\mathcal{E} : \Omega]$ to denote the probability of event $\mathcal{E}$ over the probability space Ω. We write vectors as $\vec{x} = (x_i)_{i=1}^{k} = (x_1, \ldots, x_k)$.

A function $\nu : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every k and all sufficiently large n we have $\nu(n) < \frac{1}{n^k}$. Let $\pm\{0,1\}^k := [-2^k + 1, 2^k - 1] \cap \mathbb{Z}$, and $[n] := \{1, \ldots, n\}$.

Finally, κ is the main security parameter, and ε the empty string or list.

3 Privacy ABC Systems

A privacy-enhancing attribute-based credential system for an attribute space $\mathcal{AS}$ is a set of algorithms SPGen, UKGen, IKGen, $\mathsf{Present}$, Verify, ITGen, ITVf, and Revoke and a protocol $\langle \mathcal{U}.\mathsf{Issue}, \mathcal{I}.\mathsf{Issue} \rangle$. Wherever possible (i.e., except for (advanced) issuance which is inherently interactive), we opted for non-interactive protocols. This is because rounds of interaction are an expensive resource in practice, and should thus be kept as few as possible.

Parties are grouped into issuers, users, and verifiers. The publicly available system parameters are generated using SPGen by a trusted party (in practice, this might be implemented using multiparty techniques). Each issuer can issue and revoke credentials that certify a list of attribute values under his issuer public key. Users hold secret keys that can be used to derive *pseudonyms* that are unique for a given scope string, but are unlinkable across scopes. Using $\mathsf{Present}$, users can create non-interactive *presentation tokens* from their credentials that reveal any subset of attributes from any subset of their credentials, or prove that certain attributes are equal without revealing them. Presentation tokens can be publicly verified using Verify, on input the token and the issuers' public keys. To obtain a credential, a user generates an *issuance token* defining the attribute values of the new credential using ITGen. After the issuer verified the issuance token using ITVf, the user and the issuer run $\langle \mathcal{U}.\mathsf{Issue}, \mathcal{I}.\mathsf{Issue} \rangle$, at the end of which the user obtains a credential. Issuance can be combined with a presentation of existing credentials to hide some of the attribute values from the issuer, or to prove that they are equal to attributes in credentials that the user already owns. Hence, issuance tokens can be seen as an extension of presentation tokens. Credentials can optionally be bound to a user's secret key, meaning that knowledge of this key is required to prove possession of the credential. Now, if a pseudonym or multiple key-bound credentials are used in a presentation token, then all credentials and the pseudonym must be bound to the same key. Finally, an issuer can revoke a credential using the Revoke algorithm. This algorithm outputs some public revocation information RI that is published and should be used as input to the verification algorithm of presentation and issuance tokens.

Issuers and users agree on the parameters for issuance tokens including a revocation handle and the revealed attributes of the new credential (upon which the user has generated the issuance token) in a step preceding issuance. Issuers further verify the validity of these tokens before engaging in this protocol. There are no requirements on how revocation handles are chosen, but in practice they should be different for each credential an issuer issues.

3.1 Syntax

Before formalizing the security properties, we introduce the syntax of PABCs.

System parameter generation. The system parameters of a PABC-system are generated as $spar \xleftarrow{\$} \mathsf{SPGen}(1^\kappa)$. For simplicity we assume that the system parameters in particular contain an integer L specifying the maximum number of attributes that can be certified by one credential, as well as a description of the attribute space $\mathcal{AS}$. For the rest of this document, we assume that all honest parties only accept attributes from $\mathcal{AS}$ and abort otherwise.

The system parameters are input to all algorithms presented in the following. However, for notational convenience, we will sometimes not make this explicit.

User key generation. Each user generates a secret key as $usk \xleftarrow{\$} \mathsf{UKGen}(spar)$.

Issuer key generation. Each issuer generates a public/private issuer key pair and some initial revocation information as $(ipk, isk, RI) \xleftarrow{\$} \mathsf{IKGen}(spar)$. We assume that RI also defines the set of all supported revocation handles for this issuer, and that honest parties only accept such revocation handles and abort otherwise.

Presentation. A user generates a pseudonym nym and presentation token pt as $(nym, pt) \xleftarrow{\$} \mathsf{Present}\Big(usk, scope, \big(ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i\big)_{i=1}^{k}, E, M\Big)$, where

- usk is the user's secret key, which can be ε if $scope = \varepsilon$ and none of the credentials $(cred_i)_{i=1}^k$ is bound to a user secret key;
- $scope$ is the scope of the generated pseudonym nym, where $scope = \varepsilon$ if no pseudonym is to be generated (in which case $nym = \varepsilon$);
- $(cred_i)_{i=1}^k$ are k user-owned credentials that are involved in this presentation;
- ipk_i and RI_i are the public key and current revocation information of the issuer of $cred_i$;
- $(a_{i,j})_{j=1}^{n_i}$ is the list of attribute values certified in $cred_i$, where each $a_{i,j} \in \mathcal{AS}$;
- $R_i \subseteq [n_i]$ is the set of attribute indices for which the value is revealed;
- E induces an equivalence relation on $\{(i,j) : i \in [k] \ \wedge \ j \in [n_i]\}$, where $((i,j),(i',j')) \in E$ means that pt will prove that $a_{i,j} = a_{i',j'}$ without revealing the actual attribute values. That is, E enables one to prove equality predicates;
- $M \in \{0,1\}^*$ is a message to which the presentation token is to be bound. This might, e.g., be a nonce chosen by the verifier to prevent replay attacks in which a verifier uses a valid presentation token to impersonate a user.

If $k = 0$ and $scope \neq \varepsilon$, only a pseudonym nym is generated while $pt = \varepsilon$.

Presentation verification. A verifier can check the validity of a pseudonym *nym* and a presentation token *pt*:

$$\mathsf{accept/reject} \leftarrow \mathsf{Verify}\Big(nym, pt, scope, \big(ipk_i, RI_i, (a_{i,j})_{j\in R_i}\big)_{i=1}^{k}, E, M\Big),$$

where the inputs are as for presentation. For notational convenience from now on a term like $(a_{i,j})_{j\in R_i}$ implicitly also describes the set R_i.

Issuance token generation. Before issuing a credential, a user generates an *issuance token* defining the attributes of the credentials to be issued, where (some of) the attributes and the secret key can be hidden from the issuer and can be blindly "carried over" from credentials that the user already possesses (so that the issuer is guaranteed that hidden attributes were vouched for by another issuer). Similarly to a presentation token we have:

$$(nym, pit, sit) \xleftarrow{\$} \mathsf{ITGen}\Big(usk, scope, rh, \big(ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i\big)_{i=1}^{k+1}, E, M\Big),$$

where most of the inputs and outputs are as before, but

- *pit* and *sit* are the public and secret parts of the issuance token,
- $cred_{k+1} = \varepsilon$ is the new credential to be issued,
- *rh* is the revocation handle for $cred_{k+1}$ (maybe chosen by the issuer before),
- ipk_{k+1} and RI_{k+1} are the public key and current revocation information of the issuer of the new credential,
- $(a_{k+1,j})_{j\in R_{k+1}}$ are the attributes of $cred_{k+1}$ that are revealed to the issuer,
- $(a_{k+1,j})_{j\notin R_{k+1}}$ are the attributes of $cred_{k+1}$ that remain hidden, and
- $((k+1, j), (i', j')) \in E$ means that the j^{th} attribute of the new credential will have the same value as the j'^{th} attribute of the i'^{th} credential.

Issuance token verification. The issuer verifies an issuance token as follows:

$$\mathsf{accept/reject} \xleftarrow{\$} \mathsf{ITVf}\Big(nym, pit, scope, rh, \big(ipk_i, RI_i, (a_{i,j})_{j\in R_i}\big)_{i=1}^{k+1}, E, M\Big).$$

For $j \in [k]$ all inputs are as for Verify, but for $k+1$ they are for the new credential to be issued based on *pit*.

Issuance. Issuance of credentials is a protocol between a user and an issuer:

$$(cred, RI') \xleftarrow{\$} \langle(\mathcal{U}.\mathsf{Issue}(sit, pit); \mathcal{I}.\mathsf{Issue}(isk, pit, RI)\rangle.$$

The inputs are defined as before, and *pit* has been verified by the issuer before. The user obtains a credential as an output, while the issuer receives an updated revocation information RI'.

Revocation. To revoke a credential with revocation handle *rh*, the issuer runs: $RI' \xleftarrow{\$} \mathsf{Revoke}(isk, RI, rh)$ to generate the new revocation information RI' based on the issuer's secret key, the current revocation information, and the revocation handle to be revoked.

3.2 Oracles for Our Security Definitions

Our security definitions of PABC systems require a number of oracles, some of which are the same for different definitions. We therefore present them all in one place. The oracles are initialized with a set of honestly generated keys of n_{I} honest issuers $\{(ipk_i^*, isk_i^*, RI_i^*)\}_{i=1}^{n_{\mathrm{I}}}$ and n_{U} users with keys $\{usk_i^*\}_{i=1}^{n_{\mathrm{U}}}$, respectively. Let $IK^* = \{ipk_i^*\}_{i=1}^{n_{\mathrm{I}}}$. The oracles maintain initially empty sets $\mathcal{C}$, $\mathcal{HP}$, $\mathcal{IT}$, $\mathcal{IRH}$, $\mathcal{RRH}$, $\mathcal{RI}$. Here, $\mathcal{C}$ contains all credentials that honest users have obtained as instructed by the adversary, while $\mathcal{HP}$ contains all presentation tokens generated by honest users. All public issuance tokens that the adversary used in successful issuance protocols with honest issuers are stored in $\mathcal{IT}$. The set $\mathcal{IRH}$ contains all issued revocation handles, i.e., the revocation handles of credentials issued by honest issuers, while $\mathcal{RRH}$ contains the revoked handles per issuer. Finally, $\mathcal{RI}$ contains the history of the valid revocation information of honest issuers at any point in time. Time is kept per issuer I through a counter $epoch_I^*$ that is initially 0 and increased at each issuance and revocation by I.

Honest Issuer Oracle $\mathcal{O}^{\texttt{issuer}}$. This oracle allows the adversary to obtain and revoke credentials from honest issuers. It provides the following interfaces:

- On input $(\texttt{issue}, nym, pit, scope, rh, (ipk_i, RI_i, (a_{i,j})_{j\in R_i})_{i=1}^{k+1}, E, M)$ the oracle checks that ITVf accepts the issuance token pit and that the revocation information of all honest issuers is authentic, i.e., that a tuple $(ipk_i, RI_i, \cdot)$ exists in $\mathcal{RI}$ for all honest issuers ipk_i. Further, it verifies that ipk_{k+1} is the public key of an honest issuer ipk_I^* with corresponding secret key isk_I^* and current revocation information $RI_I^* = RI_{k+1}$. If one of the checks fails, the oracle outputs $\perp$ and aborts.
  The oracle then runs $\mathcal{I}.\mathsf{Issue}(isk_{k+1}, pit, RI_I^*)$ in interaction with A until the protocol outputs RI_{k+1}. It returns RI_{k+1} to A, sets $RI_I^* \leftarrow RI_{k+1}$, increases $epoch_I^*$, adds $(ipk_I^*, RI_I^*, epoch_I^*)$ to $\mathcal{RI}$. It adds $\big(nym, pit, scope, (ipk_i, RI_i, (a_{i,j})_{j\in R_i})_{i=1}^{k+1}, E, M\big)$ to $\mathcal{IT}$ and adds (ipk_I^*, rh) to the set $\mathcal{IRH}$.
- On input $(\texttt{revoke}, I, rh)$ the oracle checks that the revocation handle rh has been the input of a successful issuance protocol with an honest issuer with key ipk_I^*, or returns $\perp$ otherwise. The oracle runs $RI_I^* \xleftarrow{\$} \mathsf{Revoke}(isk_I^*, RI_I^*, rh)$, increases $epoch_I^*$, adds $(ipk_I^*, rh, epoch_I^*)$ to $\mathcal{RRH}$, adds $(ipk_I^*, RI_I^*, epoch_I^*)$ to $\mathcal{RI}$, and returns RI_I^* to the adversary.

Honest User Oracle $\mathcal{O}^{\texttt{user}}$. This oracle gives the adversary access to honest users, which he can trigger to obtain credentials and request presentation tokens on inputs of his choice. The adversary does not get the actual credentials, but only unique credential identifiers cid by which he can refer to the credentials. It provides the following interfaces:

- On input $(\texttt{present}, U, scope, (ipk_i, RI_i, cid_i, R_i)_{i=1}^{k}, E, M)$ the oracle checks if $U \in [n_{\mathrm{U}}]$ and if, for all $i \in [k]$, a tuple $(U, cid_i, cred_i, (a_{i,j})_{j=1}^{n_i}) \in \mathcal{C}$ exists. For all credentials from honest issuers, $\mathcal{O}^{\texttt{user}}$ verifies if RI_i is authentic, i.e., if for all honest issuer public keys $ipk_i = ipk_I^*$ there exists $(ipk_i, RI_i, \cdot) \in \mathcal{RI}$.

If any check fails, $\mathcal{O}^{\texttt{user}}$ returns $\perp$, otherwise it computes $(nym, pt) \xleftarrow{\$} \mathsf{Present}\big(usk_U^*, scope, (ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k}, E, M\big)$ and adds $\big(nym, pt, scope, (ipk_i, RI_i, (a_{i,j})_{j\in R_i})_{i=1}^{k}, E, M\big)$ to $\mathcal{HP}$. It returns (nym, pt) to A.

- On input $(\texttt{obtain}, U, scope, rh, (ipk_i, RI_i, cid_i, R_i)_{i=1}^{k+1}, E, M, (a_{k+1,j})_{j=1}^{n_{k+1}})$ the oracle checks if $U \in [n_{\mathrm{U}}]$ and, for all $i \in [k]$, a tuple $(U, cid_i, cred_i, (a_{i,j})_{j=1}^{n_i}) \in \mathcal{C}$ exist. It further checks that the revocation information of honest issuers is authentic, i.e., that a tuple $(ipk_i, RI_i, \cdot) \in \mathcal{RI}$ exists for all honest issuers $ipk_i \in IK^*$. The oracle computes the issuance token $(nym, pit, sit) \xleftarrow{\$} \mathsf{ITGen}\big(usk_U^*, scope, rh, (ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k+1}, E, M\big)$. If $ipk_{k+1} \notin IK^*$, the oracle sends (nym, pit) to the adversary and runs $\mathcal{U}.\mathsf{Issue}(sit, pit)$ in interaction with the adversary until it returns a credential $cred$ and stores $(U, cid_{k+1}, cred, (a_{k+1,j})_{j=1}^{n_{k+1}})$ in $\mathcal{C}$. If $ipk_{k+1} = ipk_I^* \in IK^*$, the oracle runs $\mathcal{U}.\mathsf{Issue}(sit, pit)$ internally against $\mathcal{I}.\mathsf{Issue}(isk_I^*, pit)$ until they output a credential $cred$ and revocation information RI', respectively. The oracle adds $(U, cid_{k+1}, cred, (a_{k+1,j})_{j=1}^{n_{k+1}})$ to $\mathcal{C}$ and adds (ipk_I^*, rh) to $\mathcal{IRH}$. It further increases $epoch_I^*$, sets $RI_I^* \leftarrow RI'$, and adds $(ipk_I^*, RI_I^*, epoch_I^*)$ to $\mathcal{RI}$. Finally, the oracle chooses a fresh and unique credential identifier cid_{k+1} for the new credential and outputs it to A (note that A's choice of cid_{k+1} is ignored).

3.3 Security Definitions for PABCs

We now formalize the security properties required from PABC-systems.

Correctness. The correctness requirements are what one would expect, i.e., whenever all parties are honest and run all algorithms correctly, none of the algorithms aborts or outputs reject. That is, (*i*) if all inputs are correct, Issue always outputs a valid credential, (*ii*) holding valid credentials satisfying the specified relations allows a user to generate valid presentation- and issuance tokens, and (*iii*) issuers are able to revoke any attribute of their choice.

Pseudonym Collision-Resistance. On input $spar \xleftarrow{\$} \mathsf{SPGen}(1^\kappa)$, no PPT adversary can come up with two user secret keys $usk_0 \neq usk_1$ and a scope $scope$ such that $\mathsf{NymGen}(spar, usk_0, scope) = \mathsf{NymGen}(spar, usk_1, scope)$ with non-negligible probability.

Furthermore, we require that a pseudonym is a deterministic function of the system parameters, user secret and the scope. That is, we require that for some deterministic function NymGen, and for all $usk, scope, rh, E, E', M, M'$ and all honest tuples C, C' of the form $(ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k}$ it holds that:

$$\begin{aligned}\Pr[nym = nym_{\mathrm{i}} = nym_{\mathrm{p}} : {} & spar \xleftarrow{\$} \mathsf{SPGen}(1^\kappa), nym \leftarrow \mathsf{NymGen}(spar, usk, scope),\\ & (nym_{\mathrm{i}}, pit, sit) \xleftarrow{\$} \mathsf{ITGen}(usk, scope, rh, C', E', M'),\\ & (nym_{\mathrm{p}}, pt) \xleftarrow{\$} \mathsf{Present}(usk, scope, C, E, M)] = 1.\end{aligned}$$

Unforgeability. In the unforgeability game, the adversary can request credentials from honest issuers, or trigger honest users to receive or present credentials, using the oracles from Sect. 3.2. After this interaction, the adversary outputs a number of presentation tokens and pseudonyms, i.e., a set $\mathcal{FT}$ of tuples $(nym, pt, scope, (ipk_i, RI_i, (a_{i,j})_{j\in R_i})_{i=1}^{k}), E, M)$ as defined in the syntax of the Verify algorithm. The adversary wins the game if at least one of the presentation tokens is a forgery or if at least one of the issuance tokens submitted to the honest issuer oracle was a forgery. Note that unforgeability of credentials is implied by this definition, as the adversary can always derive a forged presentation token from a forged credential.

Experiment $\mathsf{Forge}_{\mathsf{A}}(1^{\kappa}, n_{\mathrm{I}}, n_{\mathrm{U}})$**:**
$spar \xleftarrow{\$} \mathsf{SPGen}(1^{\kappa})$
$(ipk_I^*, isk_I^*, RI_I^*) \xleftarrow{\$} \mathsf{IKGen}(spar)$ for $I = 1, \ldots, n_{\mathrm{I}}$
$usk_U^* \xleftarrow{\$} \mathsf{UKGen}(spar)$ for $U = 1, \ldots, n_{\mathrm{U}}$
$\mathcal{FT} \xleftarrow{\$} \mathsf{A}^{\mathcal{O}^{\mathtt{issuer}}, \mathcal{O}^{\mathtt{user}}}(spar, (ipk_I^*, RI_I^*)_{I=1}^{n_{\mathrm{I}}}, n_{\mathrm{U}})$
Return 1 if and only if:
$\mathcal{FT}$, $\mathcal{IT}$, $\mathcal{HP}$, $\mathcal{IRH}$, $\mathcal{RRH}$, and $\mathcal{RI}$ are not consistent.

Fig. 1. $\mathsf{Forge}_{\mathsf{A}}(1^{\kappa}, n_{\mathrm{I}}, n_{\mathrm{U}})$

Informally, a forgery is an issuance or presentation token for which the corresponding credentials were not issued to the adversary or are not supposed to be valid w.r.t. the revocation information stated in the token. Now, as the issuer does not see all attributes nor the user secret key of issued credentials, it is often not clear whether or not a given issued credential is one of the credentials corresponding to a token. However, if we assume that we knew all hidden values for each credential issued (including the user secret key), then we can efficiently test whether or not a given issuance or presentation token is a forgery. Thus, if there is an assignment for all the hidden values of the issued credentials such that all the issuance and presentation tokens presented by the adversary correspond to valid credentials, then there is no forgery among the tokens. Or, in other words, if there is no such assignment, then the adversary has produced a forgery and wins the game. Regarding the validity of credentials, the adversary also wins if he outputs a valid token for a credential that was already revoked with respect to the revocation information specified by the adversary (which may not necessarily be the latest published revocation information).

Definition 1 (Unforgeability). *A* PABC*-scheme satisfies* unforgeability*, if for every PPT adversary* A *and all* $n_{\mathrm{U}}, n_{\mathrm{I}} \in \mathbb{N}$ *there exists a negligible function* ν *such that* $\Pr[\mathsf{Forge}_{\mathsf{A}} = 1] \leq \nu$*, where the experiment is described in Fig. 1, and the oracles* $\mathcal{O}^{\mathtt{issuer}}$ *and* $\mathcal{O}^{\mathtt{user}}$ *are as in Sect. 3.2.*

We now define what *consistency* of the sets $\mathcal{FT}$, $\mathcal{IT}$, $\mathcal{HP}$, $\mathcal{IRH}$, $\mathcal{RRH}$, and $\mathcal{RI}$ means. First, the set $\mathcal{FT}$ must only contain "fresh" and valid presentation tokens, meaning that for each tuple $(nym, pt, scope, (ipk_i, RI_i, (a_{i,j})_{j\in R_i})_{i=1}^{k})$,

E, M) in $\mathcal{FT}$ must pass Verify and that for all $i \in [k]$ where $ipk_i \in IK^*$ there exists a tuple $(ipk_i, RI_i, \cdot) \in \mathcal{RI}$. If any tuple in $\mathcal{FT}$ does not satisfy these conditions, then the adversary loses the game. Let $USK \subset \{0,1\}^*$ be a hypothetical set containing the user secret keys that the adversary may have used throughout the game, and let $CRED$ be a hypothetical set containing the credentials from honest issuers that the adversary may have collected during the game. In the latter set, we write $(usk, ipk_I^*, rh, (\alpha_1, \ldots, \alpha_n)) \in CRED$ if the adversary obtained a credential from issuer ipk_I^* with revocation handle rh and attribute values $(\alpha_1, \ldots, \alpha_n)$ bound to user secret usk. Now, we consider the sets $\mathcal{FT}$, $\mathcal{IT}$, $\mathcal{HP}$, $\mathcal{IRH}$, $\mathcal{RRH}$ and $\mathcal{RI}$ to be *consistent* if there exist sets USK and $CRED$ such that the following conditions hold:

1. *Each credential is the result of a successful issuance.* For all revocation handles rh and honest issuer public keys ipk_I^*, the number of tuples $(\cdot, ipk_I^*, rh, \cdot) \in CRED$ is at most the number of tuples $(ipk_I^*, rh) \in \mathcal{IRH}$.
2. *All presentation or issuance tokens correspond to honestly obtained unrevoked credentials.* For every tuple $\big(nym, pt, scope, (ipk_i, RI_i, epoch_i, (a_{i,j})_{j \in R_i})_{i=1}^{k}, E, M\big) \in \mathcal{FT} \cup \mathcal{IT}$ there exists a $usk \in USK$ and a set of credentials $\{cred_i = (usk_i, ipk_i, rh_i, (\alpha_{i,j})_{j=1}^{n_i}) : ipk_i \in IK^*\} \subseteq CRED$ such that:
   (a) $usk_i \in \{usk, \varepsilon\}$ (all key-bound credentials are bound to the same key),
   (b) $nym = scope = \varepsilon$ or $nym =$ NymGen$(spar, usk, scope)$ (if there is a pseudonym, it is for usk and $scope$),
   (c) $\alpha_{i,j} = a_{i,j}$ for all $j \in R_i$ (the revealed attribute values are correct),
   (d) $\alpha_{i,j} = \alpha_{i',j'}$ for $((i,j),(i',j')) \in E$ with $ipk_{i'} \in IK^*$ (the attributes satisfy the equality relations to other credentials from honest issuers), and
   (e) there exists $(ipk_i, RI_i, epoch_i) \in \mathcal{RI}$ such that there exists no tuple $(ipk_i, rh_i, epoch_i') \in \mathcal{RRH}$ with $epoch_i' \leq epoch_i$ (the credentials were not revoked in the epoch where they were presented).

Thus, the adversary wins the game, if there do not exist sets USK and $CRED$ that satisfy all the information captured in $\mathcal{FT}$, $\mathcal{IT}$, $\mathcal{HP}$, $\mathcal{IRH}$, $\mathcal{RRH}$, and $\mathcal{RI}$.

We had to make a number of design decisions for our definitions, which we want to explain in the following:

- First, we do not require strong unforgeability, i.e., we do not consider a token a forgery if it contains the identical elements as a pt or pit generated by an honest user. In practice, tokens will be bound to messages M that uniquely identify the current session, and thus an adversary will neither be able to reuse pt or pit, nor will it be able to benefit from a weak forgery thereof.
- Next, we do not require that adversarially generated issuance tokens are satisfied until the issuance protocol finishes. That is, we consider forgery of issuance tokens to be a problem only if they are later used in a successful issuance protocol. This is acceptable, as an invalid issuance token does not give an adversary any meaningful power if he cannot use it to obtain a credential. Requiring the stronger property that issuance tokens are unforgeable by themselves is possible, but would further increase the complexity of our definitions – without providing stronger guarantees in practice.

- For blindly issued attributes we require that they satisfy E, but do not forbid, e.g., that an adversary runs the issuance protocol twice with the same issuance token, but with the resulting credentials containing different values for the blinded attributes as long as they satisfy E. This is not a real-world problem, as the adversary could otherwise just run multiple sessions for different issuance tokens, as the issuer will, by definition of blind attributes, not obtain any guarantees on those attributes apart from what E specifies.
- Finally, we allow presentation tokens for earlier epochs than the one underlying a credential to be generated. This makes sense for off-line verifiers who cannot update their revocation information continuously. However, our definition and construction could easily be modified to forbid such tokens.

Simulatable Privacy. For privacy, all issuance protocols and presentation tokens performed by honest users must be simulatable using only the public information that is explicitly revealed during the issuance or presentation. That is, the simulator is not given the credentials, values of hidden attributes, or even the index of the user that is supposed to perform the presentation or issuance, but must provide a view to A that is indistinguishable from a real user.

Experiment $\mathsf{Privacy}_{\mathsf{A}}(1^\kappa, n_{\mathrm{U}})$**:**
$b \xleftarrow{\$} \{0,1\}$
If $b = 0$:
  $spar \xleftarrow{\$} \mathsf{SPGen}(1^\kappa)$
  $usk_U^* \xleftarrow{\$} \mathsf{UKGen}(spar)$ for $U = 1, \ldots, n_{\mathrm{U}}$
  $USK^* = \{usk_U^*\}_{U=1}^{n_{\mathrm{U}}}$
  $b' \xleftarrow{\$} \mathsf{A}^{\mathcal{O}^{\mathrm{user}}}(spar, n_{\mathrm{U}})$
If $b = 1$:
  $(spar, \tau) \xleftarrow{\$} \mathsf{S}_1(1^\kappa)$
  $b' \xleftarrow{\$} \mathsf{A}^{\mathcal{F}(n_{\mathrm{U}},\cdot)|\mathsf{S}_2(\tau)}(spar, n_{\mathrm{U}})$
Return 1 if and only if $b = b'$.

Fig. 2. $\mathsf{Privacy}_{\mathsf{A}}(1^\kappa, n_{\mathrm{U}})$

Formalizing this is not straightforward, however. It does not suffice to require two separate simulators that work for issuance and presentation, respectively, because pseudonyms and revocation introduce explicit dependencies across different issuance and presentation queries that must also be reflected in the simulation. Moreover, the simulator must not generate presentation tokens that could not have been generated in the real world, e.g., because the user does not have the required credentials. But as the simulator does not see any user indices or hidden attribute values, it cannot know which queries can be satisfied.

We therefore define a game where an adversary A either runs in a real world with access to an honest user oracle performing the actual protocols, or runs in a simulated world, where oracle queries are first filtered by a *filter* $\mathcal{F}$ and then responded to by a stateful simulator S. The filter's role is to sanitize the queries from non-public information such as user indices, credential identifiers, etc., and to intercept queries that could not be satisfied in the real world. Note that the filter thereby enforces that the adversary can only obtain presentation

tokens for valid inputs. This must be guaranteed by the credential system as well, otherwise the adversary could distinguish between both worlds.

Definition 2 (Privacy). *A* PABC *system is* private*, if there exist PPT algorithms* $\mathsf{S}_1, \mathsf{S}_2$ *such that for every PPT adversary* A *and every* $n_{\mathrm{U}} \in \mathbb{N}$ *there exists a negligible function* ν *such that:* $\Pr[\mathsf{Privacy}_{\mathsf{A}}(1^\kappa, n_{\mathrm{U}}) = 1] \leq 1/2 + \nu(\kappa)$*, cf. (Fig. 2).*

Here, $\mathcal{O}^{\mathtt{user}}$ *is as described in Sect. 3.2, while* $\mathcal{F}$ *maintains initially empty lists* C *and* P*, a counter* $ctr = 0$*, and internal state* $\mathsf{st}_{\mathsf{S}} = \tau$*, and behaves as follows:*

- *On input* $(\mathtt{present}, U, scope, (ipk_i, RI_i, cid_i, R_i)_{i=1}^{k}, E, M)$*, the filter checks if* $U \in [n_{\mathrm{U}}]$ *and if, for all* $i \in [k]$*, a tuple* $(U, cid_i, ipk_i, (a_{i,j})_{j=1}^{n_i}, \mathsf{rev}_i) \in C$ *exists. Here,* rev_i *is the code of an algorithm that on input* RI_i *outputs a bit indicating whether* $cred_i$ *is to be considered revoked.* $\mathcal{F}$ *checks if* $\mathsf{rev}_i(RI_i) = 0\ \forall i \in [k]$ *and that* $a_{i,j} = a_{i',j'}$ *for all* $((i,j),(i',j')) \in E$*. If a check fails, the filter returns* $\perp$*. If* $scope \neq \varepsilon$ *and* $(U, scope, p) \notin P$ *then* $\mathcal{F}$ *sets* $ctr \leftarrow ctr + 1$*,* $p \leftarrow ctr$*, and adds* $(U, scope, p)$ *to* P*. It then executes* $(\mathsf{st}_{\mathsf{S}}, nym, pt) \xleftarrow{\$} \mathsf{S}_2(\mathsf{st}_{\mathsf{S}}, \mathtt{present}, scope, p, (ipk_i, (a_{i,j})_{j \in R_i})_{i=1}^{k}, E, M)$*. Finally, it returns* (nym, pt) *to* A*.*
- *On input* $(\mathtt{obtain}, U, scope, rh, (ipk_i, RI_i, cid_i, R_i)_{i=1}^{k+1}, E, M, (a_{k+1,j})_{j=1}^{n_{k+1}})$*, the filter checks if* $U \in [n_{\mathrm{U}}]$ *and if, for all* $i \in [k]$*, a tuple* $(U, cid_i, ipk_i, (a_{i,j})_{j=1}^{n_i}, \mathsf{rev}_i) \in C$ *exists. For all credentials,* $\mathcal{F}$ *checks if* $\mathsf{rev}_i(RI_i) = 0\ \forall i \in [k]$ *and that* $a_{i,j} = a_{i',j'}$ *for all* $((i,j),(i',j')) \in E$*. If any of the checks fails, the filter returns* $\perp$*. The filter then looks up the same value* p *as in* $\mathcal{O}^{\mathtt{present}}$*. It sets* $(\mathsf{st}_{\mathsf{S}}, nym, pit) \xleftarrow{\$} \mathsf{S}_2(\mathsf{st}_{\mathsf{S}}, \mathtt{obtain}, scope, p, (ipk_i, (a_{i,j})_{j \in R_i})_{i=1}^{k+1}, E, M, rh)$ *and returns* (nym, pit) *to* A*. For the subsequent flows in the issuance protocol,* $\mathcal{F}$ *answers each incoming message* M_{in} *from* A *by running* $(\mathsf{st}_{\mathsf{S}}, M_{\mathrm{out}}) \xleftarrow{\$} \mathsf{S}_2(\mathsf{st}_{\mathsf{S}}, M_{\mathrm{in}})$*. At the last flow,* S_2 *returns a tuple* $(\mathsf{st}_{\mathsf{S}}, M_{\mathrm{out}}, cid, \mathsf{rev})$*. If* $cid \neq \perp$*,* $\mathcal{F}$ *adds* $(U, cid, ipk_{k+1}, (a_{k+1,j})_{j=1}^{n_{k+1}}, \mathsf{rev})$ *to* C *and returns* M_{out} *to* A*.*

Weak Privacy. The definition above ensures a very strong notion of privacy that is not satisfied by all existing PABC schemes as they do not provide unlinkability across multiple presentation tokens that were derived from the same credential. For instance, for U-Prove [11,12] an arbitrary number of presentations cannot be linked to a specific issuance session, but any two presentations of the same credential can be linked to each other. We thus also introduce a strictly weaker privacy notion called *weak privacy*. Informally, we there give the simulator some more information to be able to generate "linkable" presentation tokens if the adversary requests multiple presentations tokens for some credential. This is done by giving S_2 "anonymized" pointers to credential identifiers as input, and thus, the simulator is aware if the same credential is used in multiple presentation sessions and can prepare the simulated token accordingly. Due to the anonymization, the simulator still does not learn the connection between an issued credential and a presentation token, thus untraceability (meaning presentation sessions cannot be linked to the actual issuance of the credential) still holds.

4 Building Blocks

We next introduce the syntax for the building blocks needed in our construction. We omit detailed security definitions of the building blocks here and refer to the full version [1], where we also present concrete instantiations of all components. Compared to PABC-systems, most of the security requirements presented in the following are relatively easy to formalize and prove for a specific instantiation. However, in Sect. 5 we will show that these properties are actually sufficient to obtain PABC-systems, by giving a generic construction for PABC-systems from these building blocks. We stress that the following definitions are heavily inspired by existing work, but have been adapted to facilitate the generic construction.

4.1 Global Setup

Global system parameters are parameters that are shared by all building blocks.

Global System Parameter Generation. The global system parameters are generated as $spar_{\mathsf{g}} = (1^\kappa, \mathcal{AS}, \ell, L, spar_{\mathsf{c}}, ck, spar'_{\mathsf{g}}) \xleftarrow{\$} \mathsf{SPGen}_{\mathsf{g}}(1^\kappa)$, where $\mathcal{AS} \subseteq \pm\{0,1\}^\ell$ is the message space of the signature scheme, and the revocation and pseudonym systems support inputs from at least $\pm\{0,1\}^\ell$. The integer L specifies the maximum number of attributes that can be signed by one signature. Furthermore, $spar_{\mathsf{c}} \xleftarrow{\$} \mathsf{SPGen}_{\mathsf{c}}(1^\kappa, \ell)$ and $ck \xleftarrow{\$} \mathsf{ComKGen}(spar_{\mathsf{c}})$ is a public master commitment key. Finally, $spar'_{\mathsf{g}}$ potentially specifies further parameters.

4.2 Commitment Schemes

A commitment scheme is a tuple of six algorithms $(\mathsf{SPGen}_{\mathsf{c}}, \mathsf{ComKGen}, \mathsf{Com}, \mathsf{ComOpenVf}, \mathsf{ComPf}, \mathsf{ComProofVf})$, where $\mathsf{SPGen}_{\mathsf{c}}$ generates commitment parameters $spar_{\mathsf{c}}$. Taking these as inputs, ComKGen generates commitment keys ck, which can be used to compute a commitment/opening pair (c, o) to a message m using the commitment algorithm Com. This pair can be verified using ComOpenVf. Furthermore, we require that one can generate a non-interactive proof of knowledge π of the content m of a commitment c, and the corresponding opening c, using ComPf. This proof can then be publicly verified using ComProofVf.

4.3 Privacy-Enhancing Signatures

We next define the main building block: privacy-enhancing attribute-based signatures (PABS). Informally, parties are split into issuers signing attributes, users obtaining signatures, and verifiers checking whether users possess valid signatures on certain attributes. After setting up some PABS-specific system parameters, each issuer computes his signing/verification key pair, such that everybody can verify that keys are well-formed. At issuance time, users can reveal certain attributes to the issuer and get the remaining attributes signed blindly. Having received a signature, a user can verify its correctness. Presentation is then

done in a non-interactive manner: users compute signature presentation tokens, potentially revealing certain attributes, and verifiers can check these tokens.

System Parameter Generation. The signature system parameters are generated as $spar_{\mathsf{s}} = (spar_{\mathsf{g}}, spar'_{\mathsf{s}}) \xleftarrow{\$} \mathsf{SPGen}_{\mathsf{s}}(spar_{\mathsf{g}})$, where the input are global system parameters and $spar'_{\mathsf{s}}$ potentially specifies further parameters.

Similar to Sect. 3.1, we assume that the respective system parameters are input to all the other algorithms of the given scheme. However, for notational convenience, we will sometimes not make this explicit.

Key Generation. An issuer generates a key pair $(ipk, isk) \xleftarrow{\$} \mathsf{IKGen}(spar_{\mathsf{s}})$. We assume that the issuer public key implicitly also defines a maximum L of attributes a signature may contain.

Key Verification. An issuer public key ipk can be verified for correctness with respect to κ as $\mathsf{accept/reject} \leftarrow \mathsf{KeyVf}(ipk)$.

Signature Issuance. Issuance of a signature is a protocol between a user and a signature issuer:

$$(sig/\bot, \varepsilon) \xleftarrow{\$} \langle \mathcal{U}.\mathsf{Sign}(ipk, (c_j, o_j)_{j \notin R}, \vec{a}); \mathcal{I}.\mathsf{Sign}(isk, (a_i)_{i \in R}, (c_j, \pi_j)_{j \notin R}) \rangle, \text{where}$$

- $\vec{a} = (a_i)_{i=1}^{L} \in \mathcal{AS}^L$ are the attributes to be signed,
- R denotes indices of attributes that are revealed to the issuer
- c_j is a verified commitment to a_j, o_j is the associated opening information, and π_j is a non-interactive proof of knowledge of the opening of c_j. In particular, c_j might be re-used from a preceding signature presentation, allowing users to blindly carry over attributes into new credentials.

Signature Verification. The correctness of a signature can be verified using SigVf on input a signature sig, attributes $\vec{a}$ and a issuer public key ipk:

$$\mathsf{accept/reject} \leftarrow \mathsf{SigVf}(sig, \vec{a}, ipk).$$

Signature Presentation Token Generation. The user can compute a signature presentation token spt that proves that he possesses a signature for a set of revealed attributes R and committed attributes $(c_j, o_j)_{j \in C}$ where $C \cap R = \emptyset$. Furthermore, a signature presentation token can be bound to a specific message M specifying, e.g., some context information or random nonce to disable adversaries to re-use signature presentation tokens in subsequent sessions:

$$spt/\bot \xleftarrow{\$} \mathsf{SignTokenGen}(ipk, sig, \vec{a}, R, (c_j, o_j)_{j \in C}, M).$$

Signature Presentation Token Verification. A signature presentation token can be verified as $\mathsf{accept/reject} \xleftarrow{\$} \mathsf{SignTokenVf}(ipk, spt, (a_i)_{i \in R}, (c_j)_{j \in C}, M)$.

4.4 Revocation Schemes

Suppose a set of users, e.g., employees, who are granted access to some online resource. Then this set will often change over time. While adding users would be possible with the features presented so far, revoking access for specific users would not. In the following we thus define revocation for signature systems.

We chose a blacklisting approach rather than whitelisting, as whitelists require verifiers to update their local copy of the revocation information every time a new signature gets issued, which makes it harder to realize offline applications. For blacklists, different verifiers may obtain updates at different intervals, depending on their security policies. As a consequence, our generic construction also only supports blacklisting, while the interfaces and definitions from Sect. 3 would also support whitelisting or hybrid approaches.

After having set up system parameters, a revocation authority generates a secret revocation key, together with some public revocation key and revocation information. Using its secret key, the authority can revoke revocation handles by updating the revocation information accordingly. Proving that a revocation handle has not yet been revoked is again done non-interactively: a user can generate a token showing that some commitment contains an unrevoked handle. This token can later be publicly verified.

System Parameter Generation. On input global system parameters, revocation parameters are generated as $spar_{\mathtt{r}} = (spar_{\mathtt{g}}, \mathcal{RS}, spar'_{\mathtt{r}}) \xleftarrow{\$} \mathsf{SPGen}_{\mathtt{r}}(spar_{\mathtt{g}})$, where $\mathcal{RS}$ specifies the set of supported revocation handles, and $spar'_{\mathtt{r}}$ potentially specifies further parameters.

Revocation Setup. The revocation authority (in the definitions of Sect. 3 this role is taken by the issuer) runs the revocation setup to obtain a secret key rsk, an associated public key rpk and a public revocation information RI:

$$(rsk, rpk, RI) \xleftarrow{\$} \mathsf{RKGen}(spar_{\mathtt{r}}).$$

Attribute Revocation. A revocation handle rh can get revoked by a revocation authority by updating the public revocation information RI to incorporate rh:

$$RI' \xleftarrow{\$} \mathsf{Revoke}(rsk, RI, rh).$$

Revocation Token Generation. A user can generate a token proving that a certain revocation handle, committed to in c, has not been revoked before:

$$rt/\perp \xleftarrow{\$} \mathsf{RevTokenGen}(rh, c, o, RI, rpk).$$

In practice, a revocation presentation will always be tied to a signature presentation to prove that the signature presentation is valid. The value of c in the former will therefore be one of the commitments from the latter.

Revocation Token Verification. A revocation token is verified by:

$$\mathsf{accept/reject} \leftarrow \mathsf{RevTokenVf}(rt, c, RI, rpk).$$

4.5 Pseudonyms

Users can be known to different issuers and verifiers under unlinkable pseudonyms.

System Parameter Generation. The parameters of a pseudonym system are generated as $spar_{\mathrm{p}} = (spar_{\mathrm{g}}, spar'_{\mathrm{p}}) \xleftarrow{\$} \mathsf{SPGen_p}(spar_{\mathrm{g}})$, where the input are global system parameters, and $spar'_{\mathrm{p}}$ potentially specifies further pseudonym parameters.

User key generation. A user generates his secret key as $usk \xleftarrow{\$} \mathsf{UKGen}(spar_{\mathrm{p}})$.

Pseudonym generation. A pseudonym nym for given usk and $scope \in \{0,1\}^*$ is computed deterministically as $nym \leftarrow \mathsf{NymGen}(usk, scope)$.

Pseudonym presentation. On input a user's secret key usk, a commitment c to usk with opening information o, and a scope string $scope \in \{0,1\}^*$, the pseudonym presentation algorithm generates a pseudonym nym with a proof π:

$$(nym, \pi) \xleftarrow{\$} \mathsf{NymPres}(usk, c, o, scope).$$

Pseudonym verification. A pseudonym nym and proof π are verified for a commitment c and scope $scope$ as $\mathsf{accept/reject} \leftarrow \mathsf{NymVf}(spar_{\mathrm{p}}, c, scope, nym, \pi)$.

5 Generic Construction of PABCs

We next present a generic construction of PABC systems from the building blocks introduced above. Our construction uses a global setup procedure, a commitment scheme, a PABS scheme, a revocation scheme, as well as a pseudonym scheme, where the syntax is as introduced in Sect. 4.

The idea underlying our construction is to use the pseudonym and revocation schemes unchanged to obtain the according properties for the PABC-system. Issuance and presentation are realized via the given PABS-scheme. However, instead of just signing the attributes, the issuer additionally signs the user secret key and the revocation handle whenever applicable. Similarly, whenever a user computes a presentation token for a set of credentials, it proves knowledge of the corresponding signature on the contained attributes, the revocation handle, and the user secret key, where the latter is always treated as an unrevealed attribute.

These independent components are linked together using commitments. For instance, to show that indeed the revocation handle contained in a credential was shown to be unrevoked, the same commitment/opening pair is used to generate revocation and signature presentation tokens.

5.1 Formal Description of the Construction

Let eq be a function mapping an attribute index (i, j) to its equivalence class as induced by E, i.e., $\mathrm{eq}(i,j) = \{(i,j)\} \cup \{(i',j') : ((i,j),(i',j')) \in E\}$. Let $\overline{E}$ be the set of all these equivalence classes, and $(a_{\overline{e}})_{\overline{e} \in \overline{E}}$ be the corresponding attribute values, i.e., $\mathrm{eq}(i,j) = \overline{e} \Rightarrow a_{i,j} = a_{\overline{e}}$. Finally, let $E_i = \{j : ((i,j),(i',j')) \in E\}$.

As some algorithm names from PABC schemes also appear in the building blocks, we stress that all algorithms called in the construction are those from the building blocks and never from the PABC scheme.

System parameter generation. This algorithm outputs $spar = (spar_{\mathtt{g}}, spar_{\mathtt{s}}, spar_{\mathtt{r}}, spar_{\mathtt{p}})$, where the different parts are generated using the system parameter algorithms of the building blocks. We assume that the algorithms of all building blocks take their respective parameters as implicit inputs.

User key generation. Users generate their secret keys as $usk \xleftarrow{\$} \mathsf{UKGen}(1^\kappa)$.

Issuer key generation. Issuers generate signature keys $(ipk', isk') \xleftarrow{\$} \mathsf{IKGen}(spar_{\mathtt{s}})$ and revocation keys $(rsk, rpk, RI') \xleftarrow{\$} \mathsf{RKGen}(spar_{\mathtt{r}})$. The algorithm outputs $(ipk, isk, RI) = ((ipk', rpk), (isk', rsk), RI')$.

Presentation. On inputs $(usk, scope, (ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k}, E, M)$, this algorithm outputs whatever AuxPresent (cf. Fig. 3) outputs, where $ipk_i = (ipk'_i, rpk_i)$, $cred_i = (sig_i, rh_i)$, and $\hat{M} = \mathtt{pres} \| M$.

Presentation verification. On inputs $(nym, pt, scope, (ipk_i, RI_i, (a_{i,j})_{j \in R_i})_{i=1}^{k}, E, M)$, Verify outputs whatever AuxVerify described in Fig. 4 outputs.

Issuance token generation. An issuance token for inputs $(usk, scope, rh_{k+1}, (ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k+1}, E, M)$, is generated as specified in Fig. 5.

$(c_{usk}, o_{usk}) \xleftarrow{\$} \mathsf{Com}(usk)$
$(nym, \pi_{nym}) = (\varepsilon, \varepsilon)$
if $scope \neq \varepsilon$:
 $(nym, \pi_{nym}) \xleftarrow{\$} \mathsf{NymPres}(usk, c_{usk}, o_{usk}, scope)$
$(c_{\overline{e}}, o_{\overline{e}}) \xleftarrow{\$} \mathsf{Com}(a_{\overline{e}})\ \forall \overline{e} \in \overline{E}$
for $i = 1, \ldots, k$ **do:**
 $(c_{rh,i}, o_{rh,i}) \leftarrow (\varepsilon, \varepsilon)$
 if $cred_i$ **is revocable:**
  $(c_{rh,i}, o_{rh,i}) \xleftarrow{\$} \mathsf{Com}(rh_i)$
  $rt_i \xleftarrow{\$} \mathsf{RevTokenGen}(rh_i, c_{rh,i}, o_{rh,i}, RI_i, rpk_i)$
$\hat{M}' = \hat{M} \| scope \| (ipk_i, (a_{i,j})_{j \in R_i})_{i=1}^{k} \| E$
for $i = 1, \ldots, k$ **do:**
 $spt_i \xleftarrow{\$} \mathsf{SignTokenGen}(ipk'_i, sig_i, ((a_{i,j})_{j=1}^{n_i}, usk, rh_i), R_i,$
 $((c_{\mathrm{eq}(i,j)}, o_{\mathrm{eq}(i,j)})_{j \in E_i}, c_{usk}, o_{usk}, c_{rh,i}, o_{rh,i}), \hat{M}')$
$pt = (c_{usk}, \pi_{nym}, (c_{\overline{e}})_{\overline{e} \in \overline{E}}, (c_{rh,i}, rt_i, spt_i)_{i=1}^{k})$
if $rt_i \neq \perp$ **and** $spt_i \neq \perp$ **for** $i = 1, \ldots, k$:
 Output (nym, pt)
Output $(\perp, \perp)$

Fig. 3. $\mathsf{AuxPresent}(usk, scope, (ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k}, E, \hat{M})$

$\hat{M}' = \hat{M} \| \mathit{scope} \| (\mathit{ipk}_i, (a_{i,j})_{j \in R_i})_{i=1}^{k} \| E$
if $\mathit{scope} \neq \varepsilon \wedge \mathsf{NymVf}(c_{usk}, \mathit{scope}, \mathit{nym}, \pi_{nym}) = \text{reject}$:
 Output reject
for $i = 1, \ldots, k$ do:
 if $\mathsf{RevTokenVf}(\mathit{rt}_i, c_{rh,i}, RI_i, \mathit{rpk}_i) = \text{reject}$ or
  $\mathsf{SignTokenVf}(\mathit{ipk}_i, \mathit{spt}_i, (a_{i,j})_{j \in R_i}, ((c_{\mathrm{eq}(i,j)})_{j \in E_i}, c_{usk}, c_{rh,i})_{i=1}^{k}, \hat{M}') = \text{reject}$:
  Output reject
Output accept

Fig. 4. $\mathsf{AuxVerify}(nym, pt, scope, (ipk_i, RI_i, (a_{i,j})_{j \in R_i})_{i=1}^{k}, E, \hat{M})$

$(\mathit{nym}, \mathit{pt}) \xleftarrow{\$} \mathsf{AuxPresent}(\mathit{usk}, \mathit{scope}, (\mathit{ipk}_i, RI_i, \mathit{cred}_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k}, E, \mathtt{iss} \| M)$
 thereby saving the used $(c_{\overline{e}}, o_{\overline{e}})_{\overline{e} \in \overline{E}}$
$(c_j, o_j) \xleftarrow{\$} \mathsf{Com}(a_{k+1,j}) \ \forall j \notin R_{k+1} \cup E_{k+1}$
$\pi_j \xleftarrow{\$} \mathsf{ComPf}(c_{\mathrm{eq}(k+1,j)}, o_{\mathrm{eq}(k+1,j)}, a_{k+1,j}) \ \forall j \in E_{k+1}$
$\pi_j \xleftarrow{\$} \mathsf{ComPf}(c_j, o_j, a_{k+1,j}) \ \forall j \notin R_{k+1} \cup E_{k+1}$
$\pi_{usk} \xleftarrow{\$} \mathsf{ComPf}(c_{usk}, o_{usk}, \mathit{usk})$
$\mathit{pit} = (\mathit{pt}, \mathit{rh}_{k+1}, (c_j)_{j \notin R_{k+1} \cup E_{k+1}}, \pi_{usk}, (\pi_j)_{j \notin R_{k+1}})$
$\mathit{sit} = ((c_{\mathrm{eq}(k+1,j)}, o_{\mathrm{eq}(k+1,j)})_{j \in E_{k+1}}, (c_j, o_j)_{j \notin R_{k+1} \cup E_{k+1}}, c_{usk}, o_{usk}, \mathit{ipk}'_{k+1}, (a_{k+1,j})_{j=1}^{n_{k+1}}, \mathit{usk}, \mathit{rh}_{k+1})$
Output $(\mathit{pit}, \mathit{sit}, \mathit{nym})$

Fig. 5. $\mathsf{ITGen}(usk, scope, rh_{k+1}, (ipk_i, RI_i, cred_i, (a_{i,j})_{j=1}^{n_i}, R_i)_{i=1}^{k+1}, E, M)$

Issuance token verification. To verify $pit = (pt, rh_{k+1}, (c_{k+1,j}, \pi_{k+1,j})_{j \notin R_{k+1}}, \pi_{usk})$, the verifier returns the output of:

$$\mathsf{AuxVerify}\big(nym, pt, scope, (ipk_i, RI_i, (a_{i,j})_{j \in R_i})_{i=1}^{k}, E, \mathtt{iss} \| M\big).$$

Issuance. For issuance, the user and the issuer run:

$$\begin{aligned}\langle \mathcal{U}.\mathsf{Sign}(&ipk'_{k+1}, ((c_{\mathrm{eq}(k+1,j)}, o_{\mathrm{eq}(k+1,j)})_{j \in E_{k+1}}, (c_j, o_j)_{j \notin R_{k+1} \cup E_{k+1}}, c_{usk}, o_{usk}),\\ &((a_{k+1,j})_{j=1}^{n_{k+1}}, usk, rh_{k+1})); \\ \mathcal{I}.\mathsf{Sign}(&isk', ((a_i)_{i \in R_{k+1}}, rh_{k+1}), ((c_{\mathrm{eq}(k+1,j)}, \pi_j)_{j \in E_{k+1}}, (c_j, \pi_j)_{j \notin R_{k+1} \cup E_{k+1}},\\ &(c_{usk}, \pi_{usk})))\rangle,\end{aligned}$$

where they extract their inputs from *sit*, and *isk* and *pit*, respectively. When the user's protocol returns *sig*, the user outputs $cred = (sig, rh_{k+1})$.

Revocation. On input an issuer secret key $isk = (isk', rsk)$, a revocation information RI and a revocation handle rh, the revocation algorithm returns $RI' \xleftarrow{\$} \mathsf{Revoke}(rsk, RI, rh)$.

The formal statements and proofs of the following theorem are given in [1].

Theorem 1 (informal). *Let the used building blocks satisfy all security properties introduced in Sect. 4. Then the* PABC *scheme resulting from the above construction is secure and simulatably private according Sect. 3.3. Furthermore, if the* PABS *scheme is weakly user private, the resulting scheme is secure and weakly private.*

6 Conclusion

We provided security definitions, a modular construction, and secure instantiations of a PABC system. Our framework encompasses a rich feature set including multi-attribute credentials, multi-credential presentations, key binding, pseudonyms, attribute equality proofs, revocation, and advanced credential issuance with carried-over attributes. Prior to our work, most of these features found provably secure instantiations in isolation, but their combination into a bigger PABC system was never proved secure, nor even defined formally.

Proving formal implications among existing definitions and ours might require substantial further research as for each related work, all definitions would first have to be reduced to the set of commonly considered features.

Finally, even though we think that our feature set is rich enough to cover a wide range of use cases (cf. Sect. 1), there are more features that can be added to our framework. Among these features are *inspection*, where presentation tokens can be de-anonymized by trusted inspectors, or *attribute predicates*, allowing to prove for example greater-than relations between attributes.

References

1. Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G.L., Neven, G., Pedersen, M.O.: Formal Treatment of Privacy-Enhancing Credential Systems. ePrint, 2014/708 (2014)
2. ABC4Trust - Attribute-based Credentials for Trust: EU FP7 Project (2015). http://www.abc4trust.eu
3. 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)
4. European Parliament and Council of the European Union: Regulation (EC) No 45/2001. Official Journal of the European Union (2001)
5. European Parliament and Council of the European Union: Directive 2009/136/EC. Official Journal of the European Union (2009)
6. Schmidt, H.A.: National strategy for trusted identities in cyberspace. Cyberwar-Resources Guide, Item 163 (2010)
7. Camenisch, J., Herreweghen, E.V.: Design and Implementation of the idemix Anonymous Credential System. In: Atluri, V. (ed.) ACM CCS 02, pp. 21–30. ACM (2002)
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. Brands, S.: Rethinking Public Key Infrastructure and Digital Certificates - Building in Privacy. Ph.D. thesis, Eindhoven Institute of Technology (1999)
12. Paquin, C., Zaverucha, G.: U-prove Cryptographic Specification v1.1 (Revision 2). Technical report, Microsoft Corporation (2013)
13. IRMA - I Reveal My Attributes: Research Project (2015). https://www.irmacard.org
14. IBM Research Security Team: Specification of the Identity Mixer Cryptographic Library. IBM Technical report RZ 3730 (99740) (2010)
15. Corporation, M.: Proof of Concept on integrating German Identity Scheme with U-Prove technology (2011). http://www.microsoft.com/mscorp/twc/endtoendtrust/vision/eid.aspx
16. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981)
17. Verheul, E.R.: Self-blindable credential certificates from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 533. Springer, Heidelberg (2001)
18. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
19. Garman, C., Green, M., Miers, I.: Decentralized anonymous credentials. In: NDSS 2014. The Internet Society (2014)
20. Chase, M., Meiklejohn, S., Zaverucha, G.M.: Algebraic MACs and Keyed-Verification Anonymous Credentials. eprint, 2013/516 (2013)
21. Nguyen, L., Paquin, C.: U-Prove Designated-Verifier Accumulator Revocation Extension. Technical report MSR-TR-2013-87 (2013)
22. Zaverucha, G.: U-Prove ID escrow extension. Technical report MSR-TR-2013-86 (2013)
23. Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 82–99. Springer, Heidelberg (2013)
24. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable & modular anonymous credentials: definitions and practical constructions. In: Iwata, T., Jung, H.C. (eds.) ASIACRYPT 2015, PartII. LNCS, vol. 9453, pp. 262–288. Springer, Heidelberg (2015)
25. Chase, M.: Efficient Non-Interactive Zero-Knowledge Proofs for Privacy Applications. Ph.D. thesis, Brown University (2008)
26. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
27. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
28. Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 491–511. Springer, Heidelberg (2014)
29. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: ACM CCS 13, pp. 1087–1098. ACM (2013)
30. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: Feng, D., Basin, D.A., Liu, P. (eds.) ASIACCS 10, pp. 60–69. ACM (2010)

31. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011)
32. Shahandashti, S.F., Safavi-Naini, R.: Threshold attribute-based signatures and their application to anonymous credential systems. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 198–216. Springer, Heidelberg (2009)