$\mathcal{PCS}$, A Privacy-Preserving Certification Scheme

Nesrine Kaaniche(✉), Maryline Laurent, Pierre-Olivier Rocher, Christophe Kiennert, and Joaquin Garcia-Alfaro

SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, Evry, France
{nesrine.kaaniche,joaquin.garcia_alfaro}@telecom-sudparis.eu

Abstract. We present $\mathcal{PCS}$, a privacy-preserving certification mechanism that allows users to conduct anonymous and unlinkable actions. The mechanism is built over an attribute-based signature construction. The proposal is proved secure against forgery and anonymity attacks. A use case on the integration of $\mathcal{PCS}$ to enhance the privacy of learners of an e-assessment environment, and some details of the ongoing implementation, are briefly presented.

Keywords: Attribute-based signatures · Attribute-based credentials · Anonymity · Bilinear pairings · Anonymous certification

1 Introduction

We present $\mathcal{PCS}$, a privacy-preserving certification scheme that provides the possibility of conducting anonymous authentication. This allows organizations to issue certificates to end-users in a way that they can demonstrate their possession in a series of transactions without being linked. $\mathcal{PCS}$ builds over an existing attribute-based signature scheme previously presented by Kaaniche and Laurent in ESORICS 2016 [10], called $\mathcal{HABS}$ (for *Homomorphic Attribute Based Signatures*). The objective of $\mathcal{HABS}$ is to enable users to anonymously authenticate with verifiers. At the same time, users minimize the amount of information submitted to the service provider, with respect to a given presentation policy. In [20,21], Vergnaud reported some limitations of $\mathcal{HABS}$ and proved that some of its security assumptions may fail in the random oracle model. $\mathcal{PCS}$ takes over $\mathcal{HABS}$ and addresses the limitations reported by Vergnaud. An ongoing implementation of the $\mathcal{PCS}$ proposal for e-learning scenarios, under the scope of a EU-funded project (cf. http://tesla-project.eu/ for further information), is available online[1] to facilitate its understanding and validation.

Paper Organization — Sections 2 and 3 provide additional background on the use of Anonymous Credentials (AC) and Attribute-based Signatures (ABS). Sections 4 and 5 provide a generic presentation of the $\mathcal{PCS}$ construction, as well as the main differences with respect to the previous $\mathcal{HABS}$ scheme. Section 6 presents the security analysis of $\mathcal{PCS}$. Section 7 briefly discusses a use case of $\mathcal{PCS}$ for e-assessment environments. Section 8 concludes the paper.

[1] Source code snippets available at http://j.mp/PKIPCSgit.

J. Garcia-Alfaro et al. (Eds.): DPM/CBT 2017, LNCS 10436, pp. 239–256, 2017.
DOI: 10.1007/978-3-319-67816-0_14

2 Background on Anonymous Credentials (AC)

In [5], Chaum introduced the notion of Anonymous Credentials (AC). Camenisch and Lysyanskaya fully formalized the concept in [3,4]. AC, also referred to as privacy-preserving attribute credentials, involve several entities and procedures. It fulfills some well-identified security and functional requirements. In the sequel, we present some further details about the type of entities, procedures and requirements associated to traditional AC schemes.

2.1 Entities

An anonymous credential system involves several entities. This includes mandatory entities (e.g., *users*, *verifiers* and *issuing organizations*) and optional entities (e.g., *revocation authorities* and *inspectors*) [2]. The central entity in AC is the *user* entity. Its interest is to obtain a privacy-preserving access to a series of services. The providers of such services are denoted as *verifiers*. Each verifier enforces an access control policy with regard to its resources and services. This access control is based on the credentials owned by the users. The related information is included in what is called the *presentation tokens*.

With the purpose of accessing the resources, a user has to obtain its credentials from a series of *issuing organizations*. Then, the user selects the appropriate information with regard to the issued credentials and shows the selected information to the requesting verifier, under a presentation token. The access control policy associated to the verifier is referred to as the *presentation policy*. Both the user and the verifier have to obtain the most recent revocation information from the *revocation authority* to either generate or verify the presentation tokens. The *revocation authority* may eventually revoke some issued credentials and maintain the list of valid credentials in the system. When a credential is revoked, the associated user will no longer be able to derive the corresponding presentation tokens. An additional trusted entity, denoted as the *inspector*, holds the technical capabilities to remove the anonymity of a user, if needed.

2.2 Procedures

An anonymous credential system mainly relies on the execution of the following series of procedures and algorithms:

- Setup — It takes as input a security parameter ξ that represents the security level; and returns some public parameters, as well as the public (pk) and secret (sk) key pair of the issuing organization, denoted as (pk_o, sk_o).
- UserKeyGen — Returns the key pairs of users. For instance, let $j \in \mathbb{N}$ represent the request of user j, it returns a key pair denoted as (pk_{u_j}, sk_{u_j}).
- Obtain $\leftrightarrow$ Issue — It presents the issuance procedure. The Issue procedure is executed by the issuing organization. It takes as input some public parameters, the secret key of the issuing organization sk_o, the public key of the user pk_u and the set of attributes $\{a_i\}_{i=1}^{N}$. N is the number of attributes.

The Obtain procedure is executed by the user and takes as input the secret key of the user sk_u and the public key of the issuing organization pk_o. At the end of this phase, the user receives a credential C.

– Show ↔ Verify — It represents the procedures between the user and the verifier. With respect to the presentation policy, the Show procedure takes as input the secret key of the user sk_u, the public key of the issuing organization pk_o, the credential C and the set of required attributes $\{a_i\}_{i=1}^{N'}$. N' is the number of required attributes. The resulting output of this algorithm is the presentation token. The Verify procedure is publicly executed by the verifier. It takes as input the public key of the issuing organization pk_o, as well as the set of attributes $\{a_i\}_{i=1}^{N'}$ and the presentation token. The Verify procedure provides as output a bit value $b \in \{0, 1\}$, denoting either the success or the failure associated to the verification process.

2.3 Requirements of AC Systems

An AC system has to fulfill the following requirements:

– *Correctness* — Honest users shall always succeed in anonymously proving validity proofs to the verifiers.
– *Anonymity* — Honest users shall remain anonymous with regard to other system users while conducting the presentation procedure in front of a series of verifiers.
– *Unforgeability* — Users that fail at holding an appropriate set of legitimate credentials shall not be able to generate presentation tokens for the system.
– *Unlinkability* — Honest users shall not be related to two or more observed items of the system. This requirement is often divided in two subproperties:
  - *Issue-show unlinkability.* It ensures that data gathered during the procedure of issuing credentials cannot be used by system entities to link a presentation token to the original credential.
  - *Multi-show unlinkability.* Presentation tokens derived from the same credentials and transmitted over different system sessions cannot be linked together by the verifiers.

Privacy-preserving attribute credential systems have to ensure some additional functional requirements, such as revocation, inspection and *selective disclosure*. Selective disclosure refers to the ability of the system users to present only partial information to the verifiers. Such information may be derived from the user credentials, in order to prove, e.g., that the user is at least eighteen years old to be eligible for accessing a service, without revealing the exact age.

3 Attribute-Based Signatures for AC Support

Attribute-based Signatures (ABS for short) is a cryptographic primitive that enables users to sign data with fine-grained control over the required identifying information [14]. To use ABS, a user shall possess a set of attributes and a secret signing key per attribute. The signing key must be provided by a trusted authority. The user can sign, e.g., a document, with respect to a predicate satisfied by the set of attributes. Common settings for ABS must include a Signature Trustee (ST), an Attribute Authority (AA), and several signers and verifiers. The ST acts as a global entity that generates valid global system parameters. The AA issues the signing keys for the set of attributes of the users (e.g., the signers). The role of the ST and the AA can be provided by the same entity. The AA can hold knowledge about the signing keys and the attributes of the users. However, the AA should not be capable to identifying which attributes have been used in a given valid signature. This way, the AA will not be able to link the signature to the source user. The AA should not be able to link back the signatures to the signers. This is a fundamental requirement from ABS, in order to fulfill common privacy requirements.

3.1 Related Work

Several ABS schemes exist in the related literature, considering different design directions. This includes ABS solutions in which (i) the attribute value can be a binary-bit string [9,13–16] or general-purpose data structures [22]; (ii) ABS solutions satisfying access structures under threshold policies [9,13,16], monotonic policies [14,22] and non-monotonic policies [15]; and (iii) ABS solutions in which the secret keys associated to the attributes are either issued by a single authority [14,16,22] or by a group of authorities [14,15]. General-purpose threshold cryptosystems can also be adapted in order to achieve traceability protection [7,8].

A simple ABS system can rely on using only one single AA entity. The AA entity derives the secret keys $\{sk_1, \cdots, sk_N\}$, with respect to the attribute set that identifies a given signer, denoted by $\mathcal{S} = \{a_1, \cdots, a_N\}$. N is the number of attributes. The procedure to generate the secret keys is performed using the master key of the AA entity, as well as some additional public parameters. These elements shall be generated during the setup procedure. A message m is sent by the verifier to the user, along with a signing predicate Υ. In order to sign m, the signing user shall hold a secret key and a set of attributes satisfying the predicate Υ. The verifier shall be able to verify whether the signing user holds the set of attributes satisfying the predicate associated to the signed message.

In [10], Kaaniche and Laurent presented an anonymous certification primitive, called $\mathcal{HABS}$, and constructed over the use of ABS. In addition to common requirements such as *privacy* and *unforgeability*, $\mathcal{HABS}$ was designed with these additional properties in mind:

- *Signature traceability* — $\mathcal{HABS}$ includes a procedure denoted as Inspec, in order to grant some entities the ability of identifying the user originating an ABS signature. To prevent common issuing organizations from tracing the system users, the Inspec procedure is provided only to a tracing authority. This authority, typically an inspector, shall hold a secret key. The *Signature traceability* is important to guarantee accountability and prevent fraud.

- *Issuer unlinkability* — When a user requests multiple authorities to issue credentials with respect to a set of attributes, common ABS authorities can link the set of credentials to one user through the corresponding public key. $\mathcal{HABS}$ includes an issuance procedure to avoid this situation.
- *Replaying sessions* — To mitigate the possibility of replay attacks (common to ABS setups), $\mathcal{HABS}$ forces its verifiers to generate for each authentication session, a new message. Such a message shall depend on the session data, e.g., the identity of the verifier and a timestamp.

In [20,21], some of the requirements imposed by $\mathcal{HABS}$ were questioned by Vergnaud. The concrete realization of the $\mathcal{HABS}$ primitive was proved unsatisfactory with regard to the expected unforgeability and privacy properties under the random oracle model. The privacy-preserving certification scheme presented in this paper addresses such limitations. We present next the revisited primitives and procedures, and answer some of the claims reported by Vergnaud in [20,21].

4 The $\mathcal{PCS}$ Construction

4.1 System Model

The $\mathcal{PCS}$ construction relies on a series of modified algorithms with regard to the original $\mathcal{HABS}$ construction reported in [10], involving several users (i.e., signers). To ease the comparison to the initial approach, we denote by $\mathcal{PCS}$ the modifications, and by $\mathcal{HABS}$ the main algorithms originally defined in [10].

- $\mathcal{PCS}$.Setup – It runs the original $\mathcal{HABS}$.Setup algorithm. It takes as input the security parameter ξ and returns a set of global public parameters. All the algorithms include as default input such global public parameters.
- $\mathcal{PCS}$.KeyGen – This algorithm returns the key pairs of either users or issuing organization. The key pairs are denoted (pk_u, sk_u) for the users, e.g., (pk_{u_j}, sk_{u_j}) for a user j; and (pk_o, sk_o) for the issuing organization.
- $\mathcal{PCS}$.Obtain $\leftrightarrow$ $\mathcal{PCS}$.Issue – The $\mathcal{PCS}$.Issue algorithm executed by the issuing organization takes as input the secret key of the issuing organization sk_o, the public key of the user pk_u, and a set of attributes $\mathcal{S} \subset \mathbb{S}$. $\mathcal{S} = \{a_i\}_{i=1}^{N}$, where N is the number of attributes. $\mathbb{S}$ is the attribute universe. The algorithm returns a signed commitment C over the set of attributes $\mathcal{S}$.

  The $\mathcal{PCS}$.Obtain algorithm is executed by the user and corresponds to the collection of the certified credentials from the issuer. The user can verify the correctness of the received signed commitment over the provided attributes. In case the user wants to conduct the verification process, the $\mathcal{PCS}$.Obtain algorithm takes as input the signed commitment C, the secret key of the user sk_u and the public key of the issuing organization pk_o. It returns a bit $b \in \{0, 1\}$ with the result of the verification (either success or failure).

– $\mathcal{PCS}$.Show $\leftrightarrow$ $\mathcal{PCS}$.Verify – It enables the verifier to check whether a user has previously obtained credentials on some attributes from a certified issuing organization, to get granted access to a service with respect to a given access policy. The verifier has to send a blinded group element $\mathtt{M}$ based on a random message m sent to the user. Following the $\mathcal{HABS}$ construction, and in order to avoid replay attacks, each authentication session is personalized with a nonce — for instance, the identity of the verifier concatenated with a timestamp. By using the credentials, the user signs the nonce. To do so, the user selects some attributes satisfying the signing predicate Υ ($\Upsilon(\mathcal{S}') = 1$) and signs the value of $\mathtt{M}$. The resulting signature Σ is sent to the verifier.

  The $\mathcal{PCS}$.Show algorithm takes as input the randomized message $\mathtt{M}$, a signing predicate Υ, the secret key of the user sk_u, the credential C and a subset of the user attributes $\mathcal{S}'$, such as $\Upsilon(\mathcal{S}') = 1$. The algorithm returns a signature Σ (or an error message $\perp$).

  The $\mathcal{PCS}$.Verify algorithm takes as input the received signature Σ, the public key of the issuing organization(s) pk_o, the signing predicate Υ and the message m. It returns a bit $b \in \{0, 1\}$ with the result of the verification, where 1 denotes *acceptance* for a successful verification of the signature; and 0 denotes *rejection*.

4.2 Security Model

We present in this section the threat models assumed to validate the requirements of $\mathcal{PCS}$. We first assume a traditional *honest but curious* model for the verifier and the issuing organization entities. Under such a model, the verifiers and the issuing organizations are honest in the sense that they provide proper inputs and outputs, at each step of their respective algorithms, as well as properly performing the computations that are supposed to be conducted; but they are curious in the sense that they may attempt to gain some extra information they are not supposed to obtain. We assume the honest but curious threat model against the validation of the privacy requirements of $\mathcal{PCS}$, i.e., with respect to the anonymity and unlinkability properties. We consider as second threat model the case of malicious users trying to override their rights. That is, malicious users that misuse some of the steps of their associated algorithms, e.g., by providing invalid inputs or outputs. We assume this second threat model against the unforgeability requirement of $\mathcal{PCS}$ provided below.

4.2.1 Unforgeability

The unforgeability requirement expects that it is not possible to forge a valid credential — in case of the Issue algorithm (respectively, the presentation token of the user – in case of the Show algorithm). This requirement ensures that colluding users will not be able to frame a user who did not generate a valid presentation token. The unforgeability requirement is defined with respect to three security games, as presented in [10]. Each security game is defined between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, that simulates the system procedures to interact with the adversary.

Definition 1. ***Unforgeability*** — *$\mathcal{PCS}$ satisfies the unforgeability requirement if for every Probabilistic Polynomial Time (PPT) adversary $\mathcal{A}$, there exists a negligible function ϵ such that:*

$$Pr[\boldsymbol{Exp}_{\mathcal{A}}{}^{unforg}(1^{\xi}) = 1] \leq \epsilon(\xi)$$

where $\boldsymbol{Exp}_{\mathcal{A}}{}^{unforg}$ is the security experiment against the unforgeability requirement, with respect to the `MC-Game`, `MU-Game` and `Col-Game` games, as presented in the original $\mathcal{HABS}$ construction [10].

The aforementioned security games are defined as follows:

- `MC-Game` – $\mathcal{A}$ is allowed to conduct an unbounded number of queries to the $\mathcal{PCS}$.Issue algorithm for different sets of attributes with respect to a fixed user public key and issuing organization secret key (i.e., the secret key of the issuing organization is not known by $\mathcal{A}$). To successfully win the `MC-Game`, the adversary shall obtain a valid credential C^* for a challenge set of attributes $\mathcal{S}^*$, and this shall be accepted by the $\mathcal{PCS}$.Obtain algorithm.
- `MU-Game` – given a user public key pk_u, a set of attributes $\mathcal{S}$ and a credential C over $\mathcal{S}$ for pk_u, the adversary $\mathcal{A}$ can conduct an unbounded number of presentation queries — as a verifier — for any signing predicate Υ such that $\Upsilon(\mathcal{S})$ equals one. To successfully win the `MU-Game`, $\mathcal{A}$ shall obtain a valid presentation token for a credential C accepted by an honest verifier.
- `Col-Game` – given two pairs of public and secret keys (pk_{u_1}, sk_{u_1}) and (pk_{u_2}, sk_{u_2}), two disjoint and non-empty sets of attributes $\mathcal{S}_1$ and $\mathcal{S}_2$, and two credentials C_1 associated to $\mathcal{S}_1$ for pk_{u_1} and C_2 associated to $\mathcal{S}_2$ for pk_{u_2}, the adversary $\mathcal{A}$ shall be able to generate a valid presentation token for a key pair (pk_{u_j}, sk_{u_j}) for $j \in \{1, 2\}$ with respect to a signing predicate Υ such that $\Upsilon(\mathcal{S}_j) \neq 1$.

4.2.2 Privacy

The privacy requirement covers the anonymity, the issue-show and the multi-show requirements, as defined in Sect. 2. We introduce three security games based on an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, similarly to the $\mathcal{HABS}$ construction [10]. We assume that $\mathcal{A}$ does not directly run or control the $\mathcal{PCS}$.Obtain $\leftrightarrow$ Issue or $\mathcal{PCS}$.Show $\leftrightarrow$ Verify algorithms, but may request the results of these algorithms to the challenger $\mathcal{C}$ in charge of such algorithms.

Definition 2. ***Privacy*** – *$\mathcal{PCS}$ satisfies the privacy requirement, if for every PPT adversary $\mathcal{A}$, there exists a negligible function ϵ such that:*

$$Pr[\boldsymbol{Exp}_{\mathcal{A}}{}^{priv}(1^{\xi}) = 1] = \frac{1}{2} \pm \epsilon(\xi)$$

where $\boldsymbol{Exp}_{\mathcal{A}}{}^{priv}$ *is the security experiment against the privacy requirement, with respect to the* `PP-Game`*,* `MS-Game` *and* `IS-Game` *games, as presented in the original* $\mathcal{HABS}$ *construction [10].*

In the aforementioned indistinguishability security games, $\mathcal{A}$ is given two pairs of public and secret keys $((pk_{u_1}, sk_{u_1})$ and $(pk_{u_2}, sk_{u_2}))$ and a set of attributes $\mathcal{S}$. The adversary can conduct an unbounded number of presentation queries — as a verifier — for any signing predicate Υ satisfied by $\mathcal{S}$; or a subset of $\mathcal{S}$ for two fixed credentials C_1 associated to $\mathcal{S}$ for pk_{u_1} and C_2 associated to $\mathcal{S}$ for pk_{u_2}. To successfully win one of the following security games, $\mathcal{A}$ should be able to guess, with a probability greater than a half:

– `PP-Game` – which key pair (pk_{u_j}, sk_{u_j}) for $j \in \{1, 2\}$, was used in the presentation procedure, with respect to a fixed signing predicate Υ and a chosen set of attributes $\mathcal{S}$.
– `MS-Game` – whether the same key pair (pk_{u_j}, sk_{u_j}) for $j \in \{1, 2\}$ was used in two different presentation procedures with respect to a chosen signing predicate Υ and a set of attributes $\mathcal{S}$.
– `IS-Game` – which key pair (pk_{u_j}, sk_{u_j}) and related credential C_j for $j \in \{1, 2\}$, was used in the presentation procedure, with respect to a fixed signing predicate Υ and a set of attributes $\mathcal{S}$.

Notice that the `PP-Game` and `IS-Game` formalize the notions of anonymity. The `MS-Game` formalizes the unlinkability requirement.

5 Concrete Construction

In this section, we complement the elements provided in previous sections to conclude the concrete construction of $\mathcal{PCS}$.

5.1 Access Structures

Definition 3 ***(Monotone Access Structure [1]).*** *Let* $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ *be a set of parties. Let* $\mathbb{A}$ *be an access structure, i.e., a collection of non-empty subsets of* $\{P_1, P_2, \cdots, P_n\}$*. Then, a collection* $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ *is called monotone if for all* $B, C \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$*, it holds that* $B \in \mathbb{A}$*,* $B \subseteq C$ *and* $C \in \mathbb{A}$*. The sets in* $\mathbb{A}$ *are known as the authorized sets. The remainder sets, not in* $\mathbb{A}$*, are known as the unauthorized sets.*

Definition 4 ***(Linear Secret Sharing Schemes (LSSS) [1]).*** *A secret sharing scheme* Π *over a set* $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ *is called linear (over* $\mathbb{Z}_p$*) if:*

1. *The share assigned to each party forms a vector over* $\mathbb{Z}_p$*;*
2. *There exists a matrix* M *with* l *rows, called the sharing generating matrix for* Π*, such that for each* $i \in [1, l]$*, we can define a function* ρ*, where* $\rho(i)$ *corresponds to the party associated to the* i^{th} *row of* M*. If we consider the column vector* $\boldsymbol{v} = (v_1, \cdots, v_k)^T$*, where* $v_1 = s \in \mathbb{Z}_p$ *is the secret to be shared,*

such that $v_t \in \mathbb{Z}_p$ and $t \in [2, k]$ are chosen at random, then $M \cdot \boldsymbol{v}$ is the vector of l shares of s according to Π. The share $\lambda_i = (M \cdot v)_i$ shall belong to the party designed by $\rho(i)$.

Assume Π is an LSSS for the access structure $\mathbb{A}$. Let S be an authorized set, such that $S \in \mathbb{A}$ and $I \subseteq \{1, 2, \cdot, l\}$ is defined as $I = \{i : \rho(i) \in S\}$. If $\{\lambda_i\}_{i \in I}$ are valid shares of a secret s according to Π, then there shall exist some constant $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ that can be computed in polynomial time, such that $\sum_{i \in I} \lambda_i w_i = s$ [1].

It is known that any monotonic boolean formula can be converted into a valid LSSS representation. Generally, boolean formulae are used to describe the access policy, and their equivalent LSSS matrices are used to sign and verify the signatures. The labeled matrix in Definition 4 is also known in the related literature as monotone span program [11,14].

Definition 5 ***(Monotone Span Programs (MSP)** [11,14]**).*** *A Monotone Span Program (MSP) is a tuple $(\mathbb{K}, M, \rho, \boldsymbol{t})$, such that $\mathbb{K}$ is a field, M is a $l \times c$ matrix (where l is the number of rows and c the numbers of columns), $\rho : [l] \to [n]$ is the labeling function and $\boldsymbol{t}$ is the target vector. The size of the MSP is the number l of rows. Since ρ is the function labeling each row i of M to a party $P_{\rho(i)}$, each party can be considered as* associated *to one or more rows. For any set of parties $S \subseteq \mathcal{P}$, the sub-matrix consisting of rows associated to the parties in S is denoted as M_S. The span of a matrix M, denoted as span(M), corresponds to the subspace generated by the rows of M, i.e., all vectors of the form $\boldsymbol{v} \cdot M$. An MSP is said to compute an access structure $\mathbb{A}$ if for each $S \in \mathbb{A}$ then the target vector t is in span(M_S). This can be formally described as follows:*

$$\mathbb{A}(S) = 1 \iff \exists \boldsymbol{v} \in \mathbb{K}^{1 \times l} : \boldsymbol{v}M = \boldsymbol{t}$$

5.2 Bilinear Maps

Consider three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p, such that g_1 and g_2 are the generators of, respectively, $\mathbb{G}_1$ and $\mathbb{G}_2$. A bilinear map $\hat{e}$ is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that the following properties are satisfied:

– (i) for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ (i.e., bilinearity property);
– (ii) $\hat{e}(g_1, g_2) \neq 1$ (i.e., non-degeneracy property);
– (iii) there exists an efficient algorithm that can compute $\hat{e}(g_1, g_2)$ for any $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ (i.e., computability property).

5.3 Complexity Assumptions

For our construction, we shall consider the following complexity assumptions:

– ***q-Diffie Hellman Exponent Problem (q-DHE)*** – Let $\mathbb{G}$ be a multiplicative cyclic group of a prime order p. Let g be a generator of $\mathbb{G}$. Then, the q-DHE problem can be stated as follows: given a tuple of elements

$(g, g_1, \cdots, g_q, g_{q+2}, \cdots, g_{2q})$, such that $g_i = g^{\alpha^i}$, where $i \in \{1, \cdots, q, q+2, \cdots, 2q\}$ and $\alpha \xleftarrow{R} \mathbb{Z}_p$, there is no efficient probabilistic algorithm $\mathcal{A}_{qDHE}$ that can compute the missing group element $g_{q+1} = g^{\alpha^{q+1}}$.

- ***Discrete Logarithm Problem (DLP)*** – Let $\mathbb{G}$ be a multiplicative cyclic group of a prime order p. Let g be a generator of $\mathbb{G}$. Then, DLP problem can be stated as follows [18]. Given the public element $y = g^x \in \mathbb{G}$, there is no efficient probabilistic algorithm $\mathcal{A}_{DLP}$ that can compute the integer x.
- ***Computational Diffie Hellman Assumption (CDH)*** – Let $\mathbb{G}$ be a group of a prime order p. Let g be a generator of $\mathbb{G}$. The CDH problem, whose complexity is assumed stronger than DLP, is stated as follows: given the tuple of elements (g, g^a, g^b), where $\{a, b\} \xleftarrow{R} \mathbb{Z}_p$, there is no efficient probabilistic algorithm $\mathcal{A}_{CDH}$ that computes g^{ab}.

5.4 Resulting Construction

Find below the revisited set of algorithms that conclude the $\mathcal{PCS}$ construction:

- Setup — It takes as input the security parameter ξ and returns the public parameters *params*. The public parameters are considered an auxiliary input to all the algorithms of $\mathcal{PCS}$.
  Global Public Parameters params – the Setup algorithm first generates an asymmetric bilinear group environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ where $\hat{e}$ is an asymmetric pairing function such as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.
  The random generators $g_1, h_1 = {g_1}^{\alpha}, \{\gamma_i\}_{i \in [1,\mathcal{N}]} \in \mathbb{G}_1$ and $g_2, h_2 = {g_2}^{\alpha} \in \mathbb{G}_2$ are also generated, as well as $\alpha \in \mathbb{Z}_p$ where $\mathcal{N}$ denotes the maximum number of attributes supported by the span program. We note that each value γ_i is used to create the secret key corresponding to an attribute a_i. Let $\mathcal{H}$ be a cryptographic hash function. The global parameters of the system are denoted as follows:

$$params = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, p, g_1, \{\gamma_i\}_{i \in [1,\mathcal{N}]}, g_2, h_1, h_2, \mathcal{H}\}$$

- KeyGen — It returns a pair of secret and public keys for each participating entity (i.e., issuing organization and user). In other words, the user gets a key pair (pk_u, sk_u) where sk_u is chosen at random from $\mathbb{Z}_p$; and $pk_u = {h_1}^{sk_u}$ is the corresponding public key. The issuing organization also gets a key pair (pk_o, sk_o). The issuing organization secret key sk_o relies on the couple defined as $sk_o = (s_o, x_o)$, where s_o is chosen at random from $\mathbb{Z}_p$ and $x_o = {g_1}^{s_o}$. The public key of the issuing organization pk_o corresponds to the couple $(X_o, Y_o) = (\hat{e}(g_1, g_2)^{s_o}, {h_2}^{s_o})$.
- Issue — It is executed by the issuing organization. The goal is to issue the credential to the user with respect to a pre-shared set of attributes $\mathcal{S} \subset \mathbb{S}$, such that $\mathbb{S}$ represents the attribute universe, defined as: $\mathcal{S} = \{a_1, a_2, \cdots, a_N\}$, where N is the number of attributes such that $N < \mathcal{N}$.
  The Issue algorithm takes as input the public key of the user pk_u, the set of

attributes $\mathcal{S}$ and the secret key of the issuing organization sk_o. It also selects an integer r at random and returns the credential C defined as:

$$C = (C_1, C_2, \{C_{3,i}\}_{i \in [1,N]}) = (x_o \cdot [{pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}}] \cdot {h_1}^r, {g_2}^r, \{{\gamma_i}^r\}_{i \in [1,N]})$$

where $\mathcal{H}(\mathcal{S}) = \mathcal{H}(a_1)\mathcal{H}(a_2)\cdots\mathcal{H}(a_N)$ and ${\gamma_i}^r$ represent the secret key associated to the attribute a_i, where $i \in [1, N]$.

- Obtain — It is executed by the user. It takes as input the credential C, the secret key of the user sk_u, the public key of the issuing organization pk_o and the set of attributes $\mathcal{S}$. It returns 1 if Eq. 1 is true (0 otherwise).

$$\hat{e}(C_1, g_2) \stackrel{?}{=} X_o \cdot \hat{e}(g_1^{sk_u \mathcal{H}(\mathcal{S})^{-1}}, Y_o) \cdot \hat{e}(h_1, C_2) \quad (1)$$

- Show — It is also executed by the user. The goal is to authenticate itself. The rationale is as follows. The user sends a request to the verifier to get granted access to a service. The verifier sends a presentation policy to the user. The presentation policy is given by a randomized message $\mathtt{M}$, a predicate Υ and the set of attributes that have to be revealed by the user. The user signs the message $\mathtt{M} = {g_1}^m$ with respect to the predicate Υ, satisfying a subset of attributes in $\mathcal{S}$. As introduced in Sect. 4, m is different for each authentication session.

  In the following, we denote by $\mathcal{S}_R$, the set of attributes revealed to the verifier, and $\mathcal{S}_H$ the set of non-revealed attributes, such as $\mathcal{S} = \mathcal{S}_R \cup \mathcal{S}_H$. The signing predicate Υ is represented by an LSSS access structure (M, ρ), i.e., M is a $l \times k$ matrix, and ρ is an injective function that maps each row of the matrix M to an attribute. The Show algorithm takes as input the user secret key sk_u, the credential C, the attribute set $\mathcal{S}$, the message m and the predicate Υ such that $\Upsilon(\mathcal{S}) = 1$. The process works as follows:

  1. The credentials of the user are randomized by choosing an integer $r' \in \mathbb{Z}_p$ at random, and conducting the following operations:

$$\begin{cases} C'_1 = C_1 \cdot {h_1}^{r'} = x_o \cdot [{pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}}] \cdot {h_1}^{r+r'} \\ C'_2 = C_2 \cdot {g_2}^{r'} = {g_2}^{r+r'} \\ C'_{3,i} = C'_{3,i} \cdot {\gamma_i}^{r'} = {\gamma_i}^{r+r'} \end{cases}$$

     The resulting credential C' is set as follows:

$$C' = (C'_1, C'_2, \{C'_{3,i}\}_{i \in [1,N]}) = (x_o \cdot [{pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}}] \cdot {h_1}^{r+r'}, {g_2}^{r+r'}, \{{\gamma_i}^{r+r'}\}_{i \in [1,N]})$$

  2. As the attributes of the user in $\mathcal{S}$ satisfy Υ, the user can compute a vector $\boldsymbol{v} = (v_1, \cdots, v_l)$ that also satisfies $\boldsymbol{v}M = (1, 0, \cdots, 0)$ according to Definition 5.

3. For each attribute a_i, where $i \in [1, l]$, the user computes $\omega_i = {C'_2}^{v_i}$ and calculates a quantity B that depends on $\{C'_{3,i}\}_{i \in [1,N]}$ such that $B = \prod_{i=1}^{l} (\gamma'_{\rho(i)})^{v_i}$.
4. Afterwards, the user selects a random r_m and computes the couple $(\sigma_1, \sigma_2) = (C'_1 \cdot B \cdot \mathtt{M}^{r_m}, {g_1}^{r_m})$. Notice that the user may not have knowledge about the secret value of each attribute in Υ. If this happens, v_i is set to 0, so to exclude the necessity of this value.
5. Using now the secret key of the user, it is possible to compute an accumulator on non-revealed attributes as follows:

$$A = {Y_o}^{\frac{sk_u \mathcal{H}(\mathcal{S}_H)^{-1}}{r_m}}$$

The user returns the presentation token $\Sigma = (\Omega, \sigma_1, \sigma_2, C'_2, A, \mathcal{S}_R)$, that includes the signature of the message $\mathtt{M}$ with respect to the predicate Υ, and where $\Omega = \{\omega_1, \cdots, \omega_l\}$ is the set of committed element values of the vector $\boldsymbol{v}$, based on the credential's item C'_2.

– Verify — Given the presentation token Σ, the public key of the issuing organization pk_o, the set of revealed attributes $\mathcal{S}_R$, the message m and the signing predicate Υ corresponding to $(M_{l \times k}, \rho)$, the verifier checks the received set of revealed attributes $\mathcal{S}_R$, and computes an accumulator A_R such that $A_R = {\sigma_2}^{\mathcal{H}(\mathcal{S}_R)^{-1}}$. Then, the verifier picks uniformly at random $k-1$ integers $\mu_2, \cdots, \mu_k$ and calculates l integers $\tau_i \in \mathbb{Z}_p$ for $i \in \{1, \cdots, l\}$, such that $\tau_i = \sum_{j=1}^{k} \mu_j M_{i,j}$, and where $M_{i,j}$ is an element of the matrix M. The verifier accepts the presentation token as valid (i.e., it returns 1) if Eq. 2 holds true:

$$\hat{e}(\sigma_1, g_2) \stackrel{?}{=} X_o \hat{e}(A_R, A) \hat{e}(, h_1, C'_2) \prod_{i=1}^{l} \hat{e}(\gamma_{\rho(i)} {h_1}^{\tau_i}, \omega_i) \hat{e}(\sigma_2, {g_2}^{m}) \quad (2)$$

6 Security Analysis

Theorem 1. ***Correctness*** – *$\mathcal{PCS}$ is correct if for all $(params) \leftarrow$ Setup(ξ), all pairs of public and secret keys $\{(pk_o, sk_o), (pk_u, sk_u)\} \leftarrow$ KeyGen$(params)$, all attribute sets $\mathcal{S}$, all credentials $C \leftarrow$ Issue $(\mathcal{S}, sk_o, pk_u)$, all claiming predicates Υ such as $\Upsilon(\mathcal{S}) = 1$ and all presentation tokens $\Sigma \leftarrow$ Show $(C, sk_u, \mathtt{M}, \Upsilon)$, we have* Obtain $(C, sk_u, pk_o, \mathcal{S}) = 1$ *and* Verify $(\Sigma, m, \Upsilon, pk_o) = 1$.

Proof. The correctness of Theorem 1 relies on Eqs. 1 and 2 (cf. Sect. 5.4). The correctness of Eq. 1 is straightforward by following the bilinearity requirement of pairing functions (cf. Sect. 5.2), summarized as follows:

$$\begin{aligned}
\hat{e}(C_1, g_2) &= \hat{e}(x_o \cdot [{pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}}] \cdot {h_1}^{r}, g_2) \\
&= \hat{e}({g_1}^{s_o}, g_2) \cdot \hat{e}({h_1}^{sk_u s_o \mathcal{H}(\mathcal{S})^{-1}}, g_2) \cdot \hat{e}({h_1}^{r}, g_2) \\
&= \hat{e}(g_1, g_2)^{s_o} \cdot \hat{e}({g_1}^{sk_u \mathcal{H}(\mathcal{S})^{-1}}, {h_2}^{s_o}) \cdot \hat{e}(h_1, {g_2}^{r}) \\
&= X_o \cdot \hat{e}({g_1}^{sk_u \mathcal{H}(\mathcal{S})^{-1}}, Y_o) \cdot \hat{e}(h_1, C_2)
\end{aligned}$$

Recall that the correctness of the presentation token is validated by the verifier. It verifies if the received token $\Sigma = (\Omega, \sigma_1, \sigma_2, C'_2, A, \mathcal{S}_R)$ holds a valid signature of message $\mathtt{M}$, based on the predicate Υ. For this purpose, the verifier checks the set of revealed attributes $\mathcal{S}_R$ and computes an accumulator A_R of the revealed attributes' values, using σ_2, such as $A_R = {\sigma_2}^{\mathcal{H}(\mathcal{S}_R)^{-1}}$, where $\mathcal{H}(\mathcal{S}_R) = \prod_{a_i \in \mathcal{S}_R} \mathcal{H}(a_i)^{-1}$. The value of σ_1 can be expressed as follows:

$$\begin{aligned}\sigma_1 &= C'_1 \cdot B \cdot \mathtt{M}^{r_m} \\ &= C'_1 \cdot \prod_{i=1}^{l} (\gamma'_{\rho(i)})^{v_i} \cdot {g_1}^{r_m m} \\ &= x_o \cdot {pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}} \cdot {h_1}^{r+r'} \cdot \prod_{i=1}^{l} (\gamma_{\rho(i)})^{(r+r')v_i} \cdot {g_1}^{r_m m}\end{aligned}$$

To prove the correctness of the presentation token verification, let us denote $(r + r')$ by R, and the first side of Eq. 2 by Ⓢ, such that:

$$\begin{aligned}Ⓢ &= \hat{e}(x_o \cdot {pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}} \cdot {h_1}^{r+r'} \cdot \prod_{i=1}^{l} (\gamma_{\rho(i)})^{R v_i} \cdot \mathtt{M}^{r_m}, g_2) \\ &= \hat{e}(x_o, g_2) \cdot \hat{e}({pk_u}^{s_o \mathcal{H}(\mathcal{S})^{-1}}, g_2) \cdot \hat{e}({h_1}^R, g_2) \cdot \hat{e}({g_1}^{r_m m}, g_2) \cdot \hat{e}(\prod_{i=1}^{l} {\gamma_{\rho(i)}}^{R v_i}, g_2) \\ &= \hat{e}(g_1, g_2)^{s_o} \cdot \hat{e}({g_1}^{sk_u \mathcal{H}(\mathcal{S}_R \cup \mathcal{S}_H)^{-1}}, {g_2}^{\alpha s_o}) \cdot \hat{e}({h_1}^R, g_2) \cdot \hat{e}(\sigma_2, {g_2}^m) \cdot \prod_{i=1}^{l} \hat{e}({\gamma_{\rho(i)}}^{R v_i}, g_2) \\ &= X_o \cdot \hat{e}([{g_1}^{sk_u}]^{\mathcal{H}(\mathcal{S}_R)^{-1} \mathcal{H}(\mathcal{S}_H)^{-1}}, {h_2}^{s_o}) \cdot \hat{e}(h_1, {g_2}^R) \cdot \hat{e}(\sigma_2, {g_2}^m) \cdot \prod_{i=1}^{l} \hat{e}(\gamma_{\rho(i)}, {g_2}^{R v_i}) \\ &= X_o \cdot \hat{e}({g_1}^{\mathcal{H}(\mathcal{S}_R)^{-1}}, [{Y_o}^{sk_u}]^{\mathcal{H}(\mathcal{S}_H)^{-1}}) \cdot \hat{e}(h_1, C'_2) \cdot \hat{e}(\sigma_2, {g_2}^m) \cdot \prod_{i=1}^{l} \hat{e}(\gamma_{\rho(i)}, \omega_i) \\ &= X_o \cdot \hat{e}(A_R, A) \cdot \hat{e}(h_1, C'_2) \cdot \prod_{i=1}^{l} \cdot \hat{e}(\gamma_{\rho(i)} {h_1}^{\tau_i}, \omega_i) \cdot \hat{e}(\sigma_2, {g_2}^m)\end{aligned}$$

Given that $\tau_i = \sum_{i=1}^{k} \mu_j M_{i,j}$, then the last equality is simplified to:

$$\sum_{i=1}^{l} \tau_i (v_i R) = R \sum_{i=1}^{l} \tau_i v_i = R \cdot 1 = R$$

and the term $\hat{e}({h_1}^R, g_2)$ leads to $\hat{e}({h_1}^R, g_2) = \prod_{i=1}^{l} \hat{e}({h_1}^{R\tau_i}, {g_2}^{R v_i})$ □

Theorem 2. ***Unforgeability*** – *The $\mathcal{PCS}$ scheme ensures the unforgeability requirement, under the CDH, q-DHE and DLP cryptographic assumptions.*

Sketch of proof. To prove that $\mathcal{PCS}$ satisfies the unforgeability requirement, we show that an adversary $\mathcal{A}$ who does not own an appropriate legitimate credential, is not able to generate a valid presentation token. Thus, $\mathcal{A}$ cannot violate the statements of Theorem 2 by reaching the advantage $Pr[\mathbf{Exp}_{\mathcal{A}}{}^{unforg}(1^{\xi}) = 1] \geq \epsilon(\xi)$.

Theorem 2 is based on the security games presented in Sect. 4.2 for the unforgeability requirement, namely `MC-Game`, `MU-Game` and `Col-Game`. We recall that the $\mathcal{PCS}$ scheme mainly relies on the $\mathcal{HABS}$ mechanism [10] for the $\mathcal{PCS}$.Obtain $\leftrightarrow$ $\mathcal{PCS}$.Issue and $\mathcal{PCS}$.Show $\leftrightarrow$ $\mathcal{PCS}$.Verify algorithms. It is, therefore, similarly resistant to forgery attacks under the CDH, q-DHE and DLP assumptions.

For the first game, namely `MC-game`, $\mathcal{A}$ may try a forgery attack against the CDH assumption, considering that the credential element C_1 is a product of an accumulator over the set of user attributes, the secret key of the issuing organization x_o and a randomization of the public group element h_1. Knowing that this randomization is required for deriving the remaining credential elements, $\mathcal{A}$ is led to violate the CDH assumption. In [20,21], Vergnaud details a forgery attack against the $\mathcal{HABS}$ construction. The assumption is to imagine a situation in which $\mathcal{A}$ overrides the granted rights by multiplying the first credential element C_1 such that $C_1 = C_1 \cdot X_u{}^{-\mathcal{H}(\mathcal{S})^{-1}} \cdot X_u{}^{\mathcal{H}(\mathcal{S}')^{-1}}$, where X_u is the public key of the user, $\mathcal{S} = \{a_1, \cdots, a_N\}$, $\mathcal{S}' = \{a_1, \cdots, a_M\}$ and $N < M$. This attack does not affect the $\mathcal{PCS}$ construction, since the secret key of the issuing organization is used during the generation of the credential element C_1. This protects the $\mathcal{PCS}$ construction from the attack reported by Vergnaud against $\mathcal{HABS}$ in [20,21].

By building over the previous attack, Vergnaud also states in [20,21] that an adversary $\mathcal{A}$ can override the granted rights by conducting a collusion attack (i.e., `Col-Game`) based on two different credentials C_{u_1} for pk_{u_1} and C_{u_2} for pk_{u_2}. The use of the secret key of the issuing organization for the derivation of the credential element C_1 also makes unfeasible this forgery attack against $\mathcal{PCS}$.

Similarly, and under the `MU-Game`, Vergnaud states in [20,21] that an adversary can try a forgery attack against the $\mathcal{HABS}$ construction, by eavesdropping the communication of a presentation protocol for a signing predicate Υ and a public key (pk_u); then, by impersonating the same user during the following sessions under the same predicate Υ. In fact, $\mathcal{A}$ can compute $\sigma_1{}' = \sigma_1 - \sigma_2(m' - m) = C_1' \cdot B \cdot g_1{}^{mr_m}$, for some known r_m. This attack does not affect the $\mathcal{PCS}$ construction, since the signing message m is properly randomized, and only the corresponding group element $\mathtt{M} = g_1{}^m$ is provided to the signer.

Finally, $\mathcal{PCS}$ is also resistant to replay attacks. The randomness elements appended by the challenger, for each request addresses the issue. Therefore, the $\mathcal{PCS}$ scheme ensures the unforgeability requirement, under the q-DHE, CDH and DLP assumptions, with respect to `MC-Game`, `MU-Game` and `Col-Game`.

Theorem 3. ***Privacy*** – *$\mathcal{PCS}$ satisfies the privacy requirement, with respect to the anonymity and unlinkability properties.*

Sketch of proof. Theorem 3 relies on the security games introduced in Sect. 4.2, namely `PP-Game`, `MS-Game` and `IS-Game`. They assume an adversary $\mathcal{A}$ trying to distinguish between two honestly derived presentation tokens for different settings with respect to every security game. As in the original $\mathcal{HABS}$ proposal [10], each specific setting of the $\mathcal{PCS}$ construction randomizes the secret keys of the users, as well as the presentation tokens.

During the `PP-Game`, since a new presentation token for the same message `M` and the same access predicate Υ is computed from random nonces, generated by $\mathcal{C}$, both presentation tokens are identically distributed in both cases. Then, an adversary $\mathcal{A}$, against the issue-show requirement — with respect to `IS-Game` — has an access to the *Issue* oracle for generating users' credentials. However, an honest user produces a different presentation token for each presentation session $\mathcal{PCS}$.Show, by using the randomness introduced by the user while generating the presentation token. As such, the probability of predicting j is bounded by $\frac{1}{2}$. In [20,21], Vergnaud identifies an anonymity attack against $\mathcal{HABS}$ with respect to the `PP-Game` and the `IS-Game`. Vergnaud states in [20,21] that an adversary $\mathcal{A}$ can compute $A^{\mathcal{H}(S_H)r_m} = {g_2}^{sk_{u_j}\mathcal{H}(S_H)^{-1}{r_m}^{-1}} = {g_2}^{sk_{u_j}}$ for some known r_m and $j \in \{1, 2\}$, in order to identify the signing user. This attack does not affect the $\mathcal{PCS}$ construction, since the secret key of the issuer is used during the generation of the credential element C_1.

Similarly, the `MS-Game` relies on a *left-or-right* oracle, where an adversary $\mathcal{A}$ cannot distinguish the oracle's outputs better than just flipping a coin. In fact, both presentation tokens for the same message `M` and the same access predicate Υ sent to different users, such as $\Upsilon(\mathcal{S}_{u_1}) = \Upsilon(\mathcal{S}_{u_2}) = 1$, are statistically indistinguishable. Using the previous attack against the $\mathcal{HABS}$ construction, Vergnaud states in [20,21] that the adversary can check whether two presentation tokens $\Sigma^{(1)}$ and $\Sigma^{(2)}$ were generated using the same pair of public and secret keys (sk_{u_j}, pk_{u_j}), by computing two group elements $T_1 = {C'_1}^{(2)}/{C'_1}^{(2)}$ and $T_2 = {C'_2}^{(2)}/{C'_2}^{(2)}$, hence evaluating the equality between two bilinear maps values $\hat{e}(T_1, g_2)$ and $\hat{e}({g_1}^{-1}, T_2)$. This same attack does not affect the $\mathcal{PCS}$ construction, since C'_1 and C'_2 are no longer provided with the presentation token. Indeed, the adversary $\mathcal{A}$ cannot distinguish two different presentations tokens with probability $\mathbf{Adv}(\mathcal{A}, t) \neq \frac{1}{2} + \epsilon$. As such, $\mathcal{PCS}$ is unlinkable, ensuring as well the privacy requirement.

7 E-assessment Use Case for $\mathcal{PCS}$

E-assessment is an innovative form for the evaluation of learners' knowledge and skills in online education, where part of the assessment activities is carried out online. As e-assessment involves online communication channel between learners and educators, as well as data transfer and storage, security measures are required to protect the environment against system and network attacks. Issues concerning the security and privacy of learners is a challenging topic. Such issues are discussed under the scope of the TeSLA project (cf. http://tesla-project.eu/ for further information), a EU-funded project that aims at providing learners with an innovative environment that allows them to take assessments remotely, thus avoiding mandatory attendance constraints.

In [12], the security of the TeSLA e-assessment system was analyzed and discussed w.r.t. the General Data Protection Regulation (GDPR) [6] recommendations. To meet such recommendations, it is necessary to ensure a reasonable level of privacy in the system. TeSLA implements several privacy technological filters. For instance, a randomized system identifier is associated to each learner. This identifier is used each time the learner accesses the TeSLA system, hence ensuring pseudo-anonymity to every learner — full anonymity not being an option in TeSLA for legal reasons. Yet, a randomized identifier alone cannot protect the learners against more complex threats such as unwanted traceability. The system can still be able to link two different sessions of the same learner. To handle such issues, the $\mathcal{PCS}$ construction is being integrated along with the security framework of the TeSLA architecture.

Available as a multi-platform `C++` source code at http://j.mp/PKIPCSgit, and mainly based on existing cryptographic libraries such as PBC [19] and MCL [17], the construction is available online to facilitate understanding, comparison and validation of the solution. For the time being, the integration of $\mathcal{PCS}$ in TeSLA is expected to allow learner-related tools to prove they are authorized to access a resource without revealing more than needed about the identity of the learners. For example, learners can be issued with certified attributes that may be required by the system verifier, such as *enrolled on engineering courses* or *conducting graduate studies*. When the learners want to prove that they own the right set of attributes, they perform a digital signature based on the required attributes, allowing the system verifier to check if a precise user is authorized, sometimes without even knowing precisely which attributes were used.

Such an approach can be easily integrated to access electronic resources on e-learning environments such as Moodle (cf. https://moodle.org/). It should be enough to prove that the learner comes from an allowed university or that the learner is registered for a given e-learning course. That way, it becomes impossible for the learning environment to follow some unnecessary information of each learner, while still letting them access specific resources of the system (e.g., anonymous quizzes and polls, to quickly validate the percentage of understanding of the learners, prior the final e-assessment). Similarly, when a learner takes the final e-assessment, the learner's work can be anonymously sent to anti-cheating tools (such as anti-plagiarism). With anonymous certification, each tool might receive a request for the same work without being able to know which learner wrote it, but also without being able to correlate the requests and decide whether they were issued by the same learner. Some further information about the integration of $\mathcal{PCS}$ into the TeSLA platform is under evaluation for testing purposes. It will be reported soon, in a forthcoming publication.

8 Conclusion

We have proposed an anonymous certification scheme called $\mathcal{PCS}$, as a building block of new privacy-friendly electronic identity systems. By using $\mathcal{PCS}$, a user can anonymously agree with a verifier about the possession of a set of attributes, such as age, citizenship and similar authorization attributes. While staying anonymous and having the control over all the released data, users can preserve their privacy during the verification procedure.

$\mathcal{PCS}$ builds over $\mathcal{HABS}$ (short for Homomorphic Attribute Based Signatures), presented by Kaaniche and Laurent in ESORICS 2016 [10]. $\mathcal{PCS}$ revisits the previous construction and addresses some security and privacy concerns reported by Vergnaud in [20,21]. Based on several security games, $\mathcal{PCS}$ handles the limitations in $\mathcal{HABS}$ with respect to forgery and anonymity. $\mathcal{PCS}$ supports a flexible selective disclosure mechanism with no-extra processing cost, which is directly inherited from the expressiveness of attribute-based signatures for defining access policies. A use case dealing with the integration of $\mathcal{PCS}$ to allow the learners of an e-assessment platform to reveal only required information to certificate authority providers has also been briefly presented. Multi-platform `C++` snippets of code, available at http://j.mp/PKIPCSgit, and based on two different cryptographic libraries [17,19], are released to facilitate the understanding, comparison and validation of $\mathcal{PCS}$, with regard to $\mathcal{HABS}$.

Acknowledgements. This work is supported by the H2020-ICT-2015/H2020-ICT-2015 TeSLA project *An Adaptive Trust-based e-assessment System for Learning*, Number 688520.

References

1. Beimel, A.: Secret sharing and key distribution. Research thesis (1996)
2. Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G.L., Neven, G., Pederson, M.O.: Scientific comparison of ABC protocols: part i - formal treatment of privacy-enhancing credential systems (2014)
3. Camenisch, J., 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). doi:10.1007/3-540-44987-6_7
4. Camenisch, J., Mödersheim, S., Sommer, D.: A formal model of identity mixer. In: Kowalewski, S., Roveri, M. (eds.) FMICS 2010. LNCS, vol. 6371, pp. 198–214. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15898-8_13
5. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
6. Europe, C.: Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In: General Data Protection Regulation, January 2016 (2016)
7. Garcia-Alfaro, J., Barbeau, M., Kranakis, E.: A proactive threshold secret sharing scheme handling Gen2 privacy threats. Technical report, Carleton University, March 2009
8. Garcia-Alfaro, J., Barbeau, M., Kranakis, E.: Proactive threshold cryptosystem for EPC tags. Ad hoc Sensor Wirel. Netw. **12**(3–4), 187–208 (2011)
9. Herranz, J., Laguillaumie, F., Libert, B., Ràfols, C.: Short attribute-based signatures for threshold predicates. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 51–67. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27954-6_4

10. Kaaniche, N., Laurent, M.: Attribute-based signatures for supporting anonymous certification. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 279–300. Springer, Cham (2016). doi:10.1007/978-3-319-45744-4_14
11. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of the 8th IEEE Structure in Complexity Theory (1993)
12. Kiennert, C., Rocher, P.O., Ivanova, M., Rozeva, A., Durcheva, M., Garcia-Alfaro, J.: Security challenges in e-assessment and technical solutions. In 8th International Workshop on Interactive Environments and Emerging Technologies for eLearning, 21st International Conference on Information Visualization, London, UK (2017)
13. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: ASIACCS 2010 (2010)
14. 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). doi:10.1007/978-3-642-19074-2_24
15. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19379-8_3
16. 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). doi:10.1007/978-3-642-02384-2_13
17. Shigeo, M.: MCL - Generic and fast pairing-based cryptography library. https://github.com/herumi/mcl. Version: release20170402
18. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
19. Stanford University: PBC - The Pairing-Based Cryptography Library. https://crypto.stanford.edu/pbc/. Version: 0.5.14
20. Vergnaud, D.: Comment on "attribute-based signatures for supporting anonymous certification" by N. Kaaniche and M. Laurent (ESORICS 2016). IACR Cryptology ePrint Archive (2016)
21. Vergnaud, D.: Comment on attribute-based signatures for supporting anonymous certification by N. Kaaniche and M. Laurent (ESORICS 2016). Comput. J. 1–8 (2017)
22. Zhang, Y., Feng, D.: Efficient attribute proofs in anonymous credential using attribute-based cryptography. In: Chim, T.W., Yuen, T.H. (eds.) ICICS 2012. LNCS, vol. 7618, pp. 408–415. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34129-8_39