Multi-show Anonymous Credentials with Encrypted Attributes in the Standard Model

Sébastien Canard, Roch Lescuyer, and Jacques Traoré

Orange Labs, Applied Crypto Group, Caen, France
{sebastien.canard,roch.lescuyer,jacques.traore}@orange.com

Abstract. Anonymous credential systems allow users to obtain a certified credential (a driving license, a student card, etc.) from one organization and then later prove possession of this certified credential to another party, while minimizing the information given to the latter. At CANS 2010, Guajardo, Mennink and Schoenmakers have introduced the concept of anonymous credential schemes with encrypted attributes, where the attributes to be certified are encrypted and unknown to the user and/or issuing organization. Their construction is secure in the random oracle model and based on blind signatures, which, unfortunately, restrict the credentials to be used only once (one-show) to remain unlinkable. In their paper, Guajardo *et al.* left as an open problem to construct *multi-show* credential schemes with encrypted attributes, or to show the impossibility of such a construction. We here provide a positive answer to this problem: our multi-show anonymous credential scheme with encrypted attributes relies on the non-interactive Groth-Sahai proof system and the recent work on commuting signatures from Fuchsbauer (Eurocrypt 2011) and is proven secure in the standard model.

Keywords: Privacy, Anonymous credentials, Encrypted attributes.

1 Introduction

Anonymous credential systems, introduced by Chaum in [12], permit users to obtain the certification of their attributes by some authorized organizations. In this context, such a certification is called a "credential". For example, a university, as an organization, can deliver credentials on particular attributes (name, birth date, studies, etc.) to its students in order to certify their status. Such credential can next be used anonymously by users to prove to a third party the possession of the certified attributes, while minimizing the information given to this third party. For example, a legitimate student can prove that she is a student, namely that she owns a credential certified by a university, without revealing any other information. She can also prove that her credential attributes satisfy some properties, for example that she is under 25, without revealing her true age, nor her name or studies.

A lot of work has been done on anonymous credentials and, currently, there are mainly two kinds of constructions. The first one is based on the work from

Brands [6] and makes use of blind signatures [11]. Such constructions are secure in the random oracle model and very practical. Unfortunately, the resulting credentials are "one-show" as they become linkable if they are used several times. They are implemented by Microsoft in their U-Prove technology [24]. The second one is due to Camenisch and Lysyanskaya [8,10] and is based on the use of group signatures [13]. The resulting anonymous credential systems are less efficient than the Brands' based ones, but they are "multi-show" since they use the inherent unlinkability property of group signatures. This technology is implemented by IBM for their Idemix product [22]. Although the original scheme was secure in the random oracle model [8,10], recent variants such as [5] are secure in the standard model. More recent papers have also proposed several variants of anonymous credentials with additional features, such as the delegation of credentials [4,16], or revocation capabilities [9,7].

Anonymous Credential with Encrypted Attributes. At CANS 2010, Guajardo, Mennink and Schoenmakers [21] have introduced the concept of *anonymous credential schemes with encrypted attributes.* They argue that, in some practical scenarios, the user should not (or does not want to) learn the certified attributes. Anonymous credentials with encrypted attributes can also be used in the context of secure multi-party computation and in particular for the millionaires protocol (see [21] for details).

In [21], Guajardo *et al.* first give the security model for anonymous credential schemes with encrypted attributes. Such a scheme involves three kinds of participants: issuers (or organizations), users and verifiers. It is composed of three protocols: key generation, credential issuance and verification. The key generation protocol permits each party to compute their secret and public keys whereas the issuance protocol allows a user to obtain, from an issuer, credentials on some encrypted attributes. The idea here is that the user only has access to the attributes in encrypted form. Finally, a verification protocol is played between a user and a verifier, in which the user proves the possession of a credential on encrypted attributes (without obtaining them in clear) while the verifier may possess the decryption key to obtain the plain attributes. The authors also give some variations where the verifier (and sometimes the issuer) does not learn the attributes in the clear.

They next propose a practical construction of this new concept. Their scheme is based on the Brands anonymous credential scheme [6,24] and makes use of blind signatures. As a result, multiple uses (or multi-shows) of the same credential makes them, as with Brands' system, linkable: thus the resulting system is only "one-show" (*a.k.a.* "one-use"), as argued by the authors in [21].

The Multi-show Problem. In [21], the authors left as an open problem to construct *multi-show* credential schemes with encrypted attributes, or to show the impossibility of such a construction. In this paper, we provide a positive answer to this problem by giving a concrete construction.

The main difficulty is to obtain a system where, after one credential issuance with an issuer, the user can use the resulting credential several times in an unlinkable manner: in other words, nobody should be able to know whether two different verification protocols were played by the same user (using the same credential) or not. It is well-known [6,8,10] that blind signature based anonymous credential cannot reach such an unlinkability property and it seems, as argued in [21], that one may start from [8,10], which makes use of group signatures, that are by essence, unlinkable.

In a nutshell, a group signature based anonymous credential system works as follows. During the issuance protocol, the user obtains from the issuer a signature on her attributes. The verification protocol next consists in proving the possession of an issuer's signature on some attributes without revealing the signature (and thus reaching the unlinkability property) nor the private attributes. When trying to apply this technique to anonymous credentials with encrypted attributes, several solutions are conceivable.

- The issuer encrypts the attributes and next signs the resulting ciphertexts. The user therefore needs to prove the possession of an issuer's signature on the ciphertexts, without revealing the signature (which can be written, using classical notation for proofs of knowledge, $\textsc{Pok}(\sigma : \sigma = \textsc{Sign}_{\mathcal{I}}(c))$). However since the ciphertexts will remain unchanged, the unlinkability property collapses.
- One solution to the above problem would be to randomize the ciphertexts (provided that the underlying encryption scheme supports such randomization techniques). Unfortunately, the issuer's signature would not be valid on the resulting ciphertexts.
- One possibility to the above non validity of the given signature is for the issuer not to give the signature directly, but to prove the possession of such signature. It follows that the user next has to produce a proof of knowledge of such a proof of knowledge, which is known to be a meta proof [25]. However, the result would clearly be impractical.
- Another solution would be to keep the signature on the encrypted attributes. During a verification protocol, the user would first randomize the original ciphertext c to obtain $\tilde{c}$ and next prove that she knows a signature on a randomized version of $\tilde{c}$, without revealing the signature nor the ciphertext c (such a proof can be written $\textsc{Pok}(\sigma, c : \sigma = \textsc{Sign}_I(c) \wedge \tilde{c} = \textsc{ReRand}(c))$). The main problem is that, to the best of our knowledge, it does not exist a practical instantiation of such a proof, except by using some variants of commuting signatures [16], as we will do in our construction[1].

Our Solution. In this paper, we take a different approach which can be seen as a mix of the two last above solutions. We make use of the concept of commuting signature which has recently been introduced by Fuchsbauer [16]. Such signature schemes allow to use a $\textsc{ComSig}$ procedure which on input one or several

[1] Even if the above proof is not exactly the one we will use.

(extractable) commitments on some messages plus a signing secret key, outputs a(n extractable) commitment on a signature on the committed messages along with a (Groth-Sahai) proof [20] that the signature on the committed messages can be recovered from the given commitment, without revealing the signature nor the committed messages. Fuchsbauer also gives a concrete and efficient construction of commuting signatures based on automorphic signatures [2,15]. We next associate a commuting signature to the randomization techniques on extractable commitments and Groth-Sahai proofs to obtain our multi-show anonymous credential scheme with encrypted attributes, which is secure in the standard model.

Organization of the Paper. The paper is organized as follows. In Section 2, we recall the concept and give the model for anonymous credential scheme with encrypted attributes in the multi-show setting. In Section 3, we give some useful tools, such as extractable commitments, Groth-Sahai proof systems and automorphic signatures. In particular, we describe an SXDH based Groth-Sahai proof of equality under different commitment keys (the DLIN version being given in [18]). Section 4 is devoted to commuting signatures. In this section, we introduce the way to produce a commuting signature on a vector of messages that are committed using different commitment keys. To the best of our knowledge, this tool is new and may be of independent interest. Finally, Section 5 describes our new anonymous credential scheme with encrypted attributes. The above extension on commuting signatures allows us to extend the work of Guajardo *et al.* to the case where the issuer certifies encrypted attributes to possibly different verifiers.

2 A Model for Anonymous Credential Systems with Encrypted Attributes

In this section, we recall the definition of encrypted credential schemes as given in [21]. We slightly differ from the Guajardo *et al.* model since we need to take into account the multi-show case.

2.1 Protocols

In an anonymous credential scheme with encrypted attributes, there is an issuer $\mathcal{I}$ who issues credentials on encrypted attributes, a user $\mathcal{U}$ who obtains credentials on some of her attributes she does not know, before anonymously proving the possession of such credentials, and a verifier $\mathcal{V}$ who is able to verify the validity of credentials and may obtain the plain attributes. The list of certified attributes are denoted $\mathbf{M}$ when they are in plain, and $\mathbf{C}$ when they are encrypted. An anonymous credential with encrypted attributes scheme Π is next composed of the following procedure, where λ is a security parameter.

– The key generation process is divided into three parts. The first one, denoted ParGen is played by any designated entity (possibly the issuer $\mathcal{I}$). It takes as input the security parameter 1^λ and outputs some parameters param for

the whole system. Next, the issuer executes IssGen which on inputs 1^λ and param, outputs $\mathsf{sk}_\mathcal{I}$. Finally, the verifier $\mathcal{V}$ uses VerGen to generate $\mathsf{sk}_\mathcal{V}$. This step finally publishes gpk as well as λ, param and the public keys related to $\mathsf{sk}_\mathcal{I}$ and $\mathsf{sk}_\mathcal{V}$.
- An issuance protocol Issue is played by $\mathcal{I}$ and $\mathcal{U}$. It takes as input gpk. The issuer additionally takes as input $\mathsf{sk}_\mathcal{I}$ and either the list $\mathbf{M}$ of plain attributes or the corresponding list $\mathbf{C}$ of encrypted attributes. The user always takes as input $\mathbf{C}$. This protocol outputs for the user a credential cred on the list $\mathbf{C}$ of encrypted attributes related to $\mathbf{M}$. The issuer outputs its view $\mathsf{view}^{\mathrm{Iss}}$ of the protocol.
- A verification protocol Verify is played by $\mathcal{U}$ and $\mathcal{V}$. It takes as input gpk. The verifier (resp. the user) additionally takes as input $\mathsf{sk}_\mathcal{V}$ (resp. cred and $\mathbf{C}$). The verifier outputs a bit representing either 1 and optionally a list $\mathbf{M}$ of plain attributes (for acceptance) or 0 (for rejection).

Completeness. Such a scheme should verify the *completeness* property which states that for any (gpk, $\mathsf{sk}_\mathcal{I}$, $\mathsf{sk}_\mathcal{V}$) output by the key generation procedures and related to honest issuer and verifier, the credential cred obtained by $\mathcal{U}$ during Issue will be accepted in the Verify protocol with overwhelming probability.

2.2 Security Properties

In [21], Guajardo *et al.* have given the security properties for the case of a one-show anonymous credential scheme with encrypted attributes. We thus need to modify their security model to reach the multi-show case. We moreover give more formal definitions for some properties. Let us consider an anonymous credential with encrypted attributes scheme denoted Π.

Used Oracles. Before giving the security experiments, we first describe the different oracles that will be used by the adversary. The security of our scheme is conducted in an adaptive corruption model, where the challenger $\mathcal{C}$ generates public keys for all entities and allows the adversary to get secret keys for some of them (the *corrupted* ones). We thus introduce a general key generation procedure, denoted KeyGen which corresponds to the above execution of ParGen, IssGen and VerGen. This procedure, executed by the challenger, takes as input 1^λ and outputs $(\mathsf{gpk}, \mathsf{sk}_\mathcal{I}, \mathsf{sk}_V)$.

In the following experiments, the adversary can play either the issuer or the user during the Issue procedure. In the first case, the adversary requests the ObtainC oracle[2] on chosen attributes while in the second case, the requested oracle is denoted IssueC on attributes chosen by either the adversary or the challenger. When the role of the issuer is played by the challenger, the set of all issuer's views for the Issue protocol is denoted V. Each entry V_i of V includes the set $\mathbf{M}_i$ of certified plain attributes. We use similar notation for the Verify protocol, with a request to the ShowC when the adversary plays the role of the verifier and a request to VerifyC otherwise.

[2] By convention, the name of the oracle denotes the action executed by the challenger.

```
$\mathbf{Exp}^{\mathsf{unf}}_{\Pi,\mathcal{A}}(\lambda)$;
– $(\mathsf{gpk}, \mathsf{sk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{V}}) \leftarrow \textsc{KeyGen}(1^\lambda)$;
– $(st) \leftarrow \mathcal{A}_g^{\textsc{IssueC},\textsc{VerifyC},\textsc{ShowC}}(\mathsf{gpk}, \mathsf{sk}_{\mathcal{V}})$;
– $\textsc{Verify} : (\perp \leftarrow \mathcal{A}_c^{\mathcal{U}}(st)), \mathsf{out} \leftarrow \mathcal{C}^{\mathcal{V}}(\mathsf{sk}_{\mathcal{V}}))$;
– if $\mathsf{out} = 0$, then return 0;
– if $(\mathsf{out} = (1, \widetilde{\mathbf{M}}) \wedge \exists i : \mathbf{M}_i = \widetilde{\mathbf{M}}$, return 0;
– return 1.
```

```
$\mathbf{Exp}^{\mathsf{attmask}}_{\Pi,\mathcal{A}}(\lambda)$;
– $b \leftarrow \{0,1\}$;
– $(\mathsf{gpk}, \mathsf{sk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{V}}) \longleftarrow \textsc{KeyGen}(1^\lambda)$;
– $(\mathbf{M}_0, \mathbf{M}_1, st) \leftarrow \mathcal{A}_g^{\textsc{IssueC},\textsc{VerifyC}}(\mathsf{gpk})$;
– $\textsc{Issue} : (\perp \leftarrow \mathcal{C}^{\mathcal{I}}(\mathbf{M}_b)), (\tilde{st} \leftarrow \mathcal{A}_{ch}^{\mathcal{U}}(st))$;
– $b' \leftarrow \mathcal{A}_{gu}^{\textsc{IssueC},\textsc{VerifyC}}(\mathsf{gpk}, \tilde{st})$;
– return $(b = b')$.
```

```
$\mathbf{Exp}^{\mathsf{up}}_{\Pi,\mathcal{A}}(\lambda)$
– $b \leftarrow \{0,1\}$;
– $(\mathsf{gpk}, \mathsf{sk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{V}}) \longleftarrow \textsc{KeyGen}(1^\lambda)$;
– $(st, \mathbf{C}) \leftarrow \mathcal{A}_g^{\textsc{ObtainC},\textsc{ShowC}}(\mathsf{gpk}, \mathsf{sk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{V}})$;
– $\textsc{Issue} : (st_0 \leftarrow \mathcal{A}_{ch_1}^{\mathcal{I}}(st)), ((\mathsf{cred}_0) \leftarrow \mathcal{C}^{\mathcal{U}}(\mathbf{C}))$;
– $\textsc{Issue} : (st_1 \leftarrow \mathcal{A}_{ch_2}^{\mathcal{I}}(st_0)), ((\mathsf{cred}_1) \leftarrow \mathcal{C}^{\mathcal{U}}(\mathbf{C}))$;
– $\textsc{Verify} : (\perp \leftarrow \mathcal{C}^{\mathcal{U}}(\mathsf{cred}_b, \mathbf{C})), \tilde{st} \leftarrow \mathcal{A}_{ch_f}^{\mathcal{V}}(st_1))$;
– $b' \leftarrow \mathcal{A}_{gu}^{\textsc{ObtainC},\textsc{ShowC}}(\tilde{st})$;
– return $(b = b')$.
```

```
$\mathbf{Exp}^{\mathsf{hv\text{-}up}}_{\Pi,\mathcal{A}}(\lambda)$
– $b \leftarrow \{0,1\}$;
– $(\mathsf{gpk}, \mathsf{sk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{V}}) \longleftarrow \textsc{KeyGen}(1^\lambda)$;
– $(st, \mathbf{C}_0, \mathbf{C}_1) \leftarrow \mathcal{A}_g^{\textsc{ObtainC},\textsc{VerifyC}}(\mathsf{gpk}, \mathsf{sk}_{\mathcal{I}})$;
– if $|\mathbf{C}_0| \neq |\mathbf{C}_1|$, then return 0;
– $\textsc{Issue} : (st_0 \leftarrow \mathcal{A}_{ch_1}^{\mathcal{I}}(st)), ((\mathsf{cred}_0) \leftarrow \mathcal{C}^{\mathcal{U}}(\mathbf{C}_0))$;
– $\textsc{Issue} : (st_1 \leftarrow \mathcal{A}_{ch_2}^{\mathcal{I}}(st_0)), ((\mathsf{cred}_1) \leftarrow \mathcal{C}^{\mathcal{U}}(\mathbf{C}_1))$;
– $\textsc{Verify} : (\perp \leftarrow \mathcal{C}^{\mathcal{U}}(\mathsf{cred}_b, \mathbf{C}_b)), \tilde{st} \leftarrow \mathcal{A}_{ch_f}^{\mathcal{V}}(st_1))$;
– $b' \leftarrow \mathcal{A}_{gu}^{\textsc{ObtainC},\textsc{VerifyC}}(\tilde{st})$;
– return $(b = b')$.
```

Fig. 1. Security experiments

In the following, a protocol Prot between an entity $\mathcal{E}_0$, playing the role of $\mathcal{R}_0$ (of a user, a verifier or an issuer), taking on input i_0 and outputting o_0 and an entity $\mathcal{E}_1$, playing a role $\mathcal{R}_1$, taking on input i_1 and outputting o_1 is denoted $\textsc{Prot} : (o_0 \leftarrow \mathcal{E}_0^{\mathcal{R}_0}(i_0)), (o_1 \leftarrow \mathcal{E}_1^{\mathcal{R}_1}(i_1))$.

The different security experiments are next given in Figure 1 while the related security definitions are given as follows.

Unforgeability. As we are in the multi-show case and do not rely on blind signatures, we cannot use the same definition as Guajardo *et al.* [21] who ask the adversary to output more credentials than generated by the issuer. In fact, we give a single definition which embed both the *one-more unforgeability* and the *blinding invariance unforgeability* properties introduced in the original model [21]. In particular, the blinding invariance unforgeability property states in [21] that for any attribute list output by the adversary, the number of credentials on this list does not exceed the number of times a credential has been issued on this list and the one-more unforgeability property prevents an adversary from outputting $K+1$ distinct credentials after having requested only K credentials. In our setting, this can be simplified by preventing an adversary from being accepted during a Verify protocol with a set of attributes which has never been certified by the issuer.

More precisely, our experiment asks the adversary to successfully play a Verify protocol such that the embedded attributes have never been certified by the issuer. The *unforgeability* experiment $\mathsf{Exp}^{\mathsf{unf}}_{\Pi,\mathcal{A}}(1^\lambda)$ for the adversary $\mathcal{A}=(\mathcal{A}_g,\mathcal{A}_c)$, with security parameter λ, is given in Figure 1 and an anonymous credential with encrypted attributes scheme satisfies the *unforgeability* property iff there exists a negligible function $\nu(\lambda)$ such that for any adversary $\mathcal{A}$, $\Pr(\mathsf{Exp}^{\mathsf{unf}}_{\Pi,\mathcal{A}}(1^\lambda) \longrightarrow 1) < \nu(\lambda)$.

Attribute Masking. This property says that no unauthorized party should learn the encrypted attributes. We here consider, as in [21], the case where only the user does not learn the plain attributes. Other cases (*e.g.* attributes not known by the issuer) can easily be adapted. Contrary to [21], we here provide a formal security definition, for which the experiment $\mathbf{Exp}^{\mathsf{attmask}}_{\Pi,\mathcal{A}}(\lambda)$ is given in Figure 1. It follows that an anonymous credential with encrypted attributes scheme satisfies the *attribute masking* property iff there exists a negligible function $\nu(\lambda)$ such that for any adversary $\mathcal{A}=(\mathcal{A}_g,\mathcal{A}_{ch},\mathcal{A}_{gu})$, $\Pr\big(\mathsf{Exp}^{\mathsf{attmask}}_{\Pi,\mathcal{A}}(1^\lambda) \longrightarrow 1\big) < \frac{1}{2}+\nu(\lambda)$.

User Privacy. Contrary to the definition given in [21], we consider the case of a multi-show credential. Then, the user privacy should include the possibility for one single user to use several times the same credential, without being traced. In fact, there are two cases, depending on the possibility for the adversary to corrupt (*user privacy*) or not (*honest-verifier user privacy*) the verifier. In both cases, the adversary plays the role of the issuer $\mathcal{I}$ and executes two different Issue protocols with the challenger. Next, one of the two output credential is used by the challenger during a Verify protocol. If the verifier is corrupted, this experiment can easily be won by the adversary since the corrupted issuer can certify two different sets of attributes and the corrupted verifier can easily check which one is used during the Verify protocol by decrypting the encrypted attributes. Thus, when the verifier is corrupted, this experiment is only relevant when the plain attributes are similar in both Issue protocols. For this purpose (see $\mathbf{Exp}^{\mathsf{up}}$ in Figure 1), the adversary output one single set of encrypted attributes $\mathbf{C}$, which is used twice in both Issue protocols. We do not need this restriction for the case where the verifier is honest and the adversary thus output two different encrypted attributes $\mathbf{C}_0$ and $\mathbf{C}_1$ (see $\mathbf{Exp}^{\mathsf{hv\text{-}up}}$ in Figure 1).

Both experiments are given in Figure 1. Next, the scheme satisfies the *user privacy* (resp. *honest-verifier user privacy*) property iff there exists a negligible function $\nu(\lambda)$ such that for any adversary $\mathcal{A}=(\mathcal{A}_g,\mathcal{A}_{ch_1},\mathcal{A}_{ch_2},\mathcal{A}_{ch_f},\mathcal{A}_{gu})$, $\Pr\big(\mathsf{Exp}^{\mathsf{up}}_{\Pi,\mathcal{A}}(1^\lambda) \longrightarrow 1\big) < \frac{1}{2}+\nu(\lambda)$ (resp. $\Pr\big(\mathsf{Exp}^{\mathsf{hv\text{-}up}}_{\Pi,\mathcal{A}}(1^\lambda) \longrightarrow 1\big) < \frac{1}{2}+\nu(\lambda)$).

3 Cryptographic Tools

We here introduce the cryptographic tools we need in the following. This includes extractable commitment schemes, Groth-Sahai (GS) proofs [20] and automorphic signatures [2,15].

In the following, a bilinear environment is given by the tuple (p, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, e, g_1, g_2) where p is a prime number, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p, g_1 (resp. g_2) is a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$), and $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a pairing with the non-degeneracy ($e(g_1, g_2) \neq 1$) and bilinearity (for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$) properties. For vectors of group elements, "$\odot$" denotes the component-wise group operation.

As we use the Abe *et al.* proposal [3] for signing a vector of messages, we also need an injective mapping $\langle . \rangle : \{1, ..., n_{\max}\} \rightarrow \mathbb{G}_1 \times \mathbb{G}_2$ such that for all $n, n' \in \{1, ..., n_{\max}\}$, $\langle n \rangle \odot \langle n' \rangle \neq (1, 1)$, where $n_{max} \in \mathbb{N}$ is a fixed parameter. In the following, we consider $\langle . \rangle : n \mapsto ({g_1}^n, {g_2}^n)$ since for all reasonably small $n, n' \in \{1, ..., n_{\max}\}$ (we consider n_{max} as small in the following constructions), we have $({g_1}^{n+n'}, {g_2}^{n+n'}) \neq (1, 1)$.

3.1 Randomizable and Extractable Commitment Schemes

A *commitment scheme* permits one user to commit to a message, using some randomness, such that it is possible to further give the message and the randomness to prove that this was truly the committed message. The commitment becomes *extractable* when the commit process makes use of a public key which is related to a secret key allowing the owner of the latter to open any commitment and retrieve the initially committed message. Finally, the commitment scheme is said *randomizable* if it exists a public procedure which permits to randomize a given commitment, without obtaining any information about the initially committed message, and such that it is infeasible to know whether two given commitments are related to the same message or not. A formal definition of such a commitment can be found in [20].

SXDH Commitments. In the following, we will use SXDH randomizable and extractable commitment schemes [20], which can be described as follows.

Key generation. Given a bilinear environment, the extractable keys are $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ and the public key ck of the commitment schemes is composed of $\mathbf{u} := (\mathbf{u}_1, \mathbf{u}_2), \mathbf{v} := (\mathbf{v}_1, \mathbf{v}_2)$ where (for $t_1, t_2 \in \mathbb{Z}_p$)

$$\mathbf{u}_1 := (g_1, {g_1}^{\alpha_1}), \mathbf{u}_2 := ({g_1}^{t_1}, {g_1}^{\alpha_1 t_1}), \mathbf{v}_1 := (g_2, {g_2}^{\alpha_2}), \mathbf{v}_2 := ({g_2}^{t_2}, {g_2}^{\alpha_2 t_2}).$$

Commitment. The commitment to a group element X, with randomness $\rho = (\rho_1, \rho_2) \in \mathbb{Z}_p^2$ is

$$\mathbf{c} := (c_1, c_2) = ({u_{11}}^{\rho_1} \cdot {u_{21}}^{\rho_2}, X \cdot {u_{12}}^{\rho_1} \cdot {u_{22}}^{\rho_2}) \text{ if } X \in \mathbb{G}_1 \text{ and}$$
$$\mathbf{c} := (c_1, c_2) = ({v_{11}}^{\rho_1} \cdot {v_{21}}^{\rho_2}, X \cdot {v_{12}}^{\rho_1} \cdot {v_{22}}^{\rho_2}) \text{ if } X \in \mathbb{G}_2.$$

Such a commitment is in the following denoted $\text{SXDHC}\textsc{om}(\mathsf{ck}, X, \rho)$.

Extraction. The extraction retrieves $X \in \mathbb{G}_1$ (resp. $X \in \mathbb{G}_2$) by computing $c_2 \cdot c_1^{-\alpha_1}$ (resp. $c_2 \cdot c_1^{-\alpha_2}$).

Randomization. Given a commitment $\mathbf{c}$ and some fresh randomness $\rho' = (\rho'_1, \rho'_2) \in \mathbb{Z}_p^2$, the randomization of $\mathbf{c}$ is done by computing

$$\mathbf{c}' := (c_1 \cdot {u_{11}}^{\rho'_1} \cdot {u_{21}}^{\rho'_2}, c_2 \cdot {u_{12}}^{\rho'_1} \cdot {u_{22}}^{\rho'_2}) \text{ if } \mathbf{c} \in \mathbb{G}_1^2 \text{ and}$$
$$\mathbf{c}' := (c_1 \cdot {v_{11}}^{\rho'_1} \cdot {v_{21}}^{\rho'_2}, c_2 \cdot {v_{12}}^{\rho'_1} \cdot {v_{22}}^{\rho'_2}) \text{ if } \mathbf{c} \in \mathbb{G}_2^2.$$

3.2 (SXDH) Groth-Sahai Proofs

Groth and Sahai have described in [20] a witness indistinguishable proof system, for a class of *pairing-product equations* (PPE for short) over variables $X_1, \ldots, X_m \in \mathbb{G}_1$ and $Y_1, \ldots, Y_n \in \mathbb{G}_2$ as

$$E(X_1, \ldots, X_m; Y_1, \ldots, Y_n) \; : \; \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{j=1}^{n} e(A_j, \underline{Y_j}) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} = \mathbf{t}_{\mathcal{T}}$$

defined by elements $A_j \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $\gamma_{i,j} \in \mathbb{Z}_p$ for $i \in [1,m]$, $j \in [1,n]$ and $\mathbf{t}_{\mathcal{T}} \in \mathbb{G}_T$ and where the notation $\underline{X}$ means that the variable X is a secret value. In this paper, we use the SXDH version of GS proofs. Such a proof is denoted $\textsc{Prove}(\mathsf{ck}, E, (X_1, \ldots, X_m; Y_1, \ldots, Y_n), (r_1, \ldots, r_m; s_1, \ldots, s_m))$ where r_i, $s_j \in \mathbb{Z}_p$. We refer the reader to *e.g.* [20,16] for details.

In [4], it has been shown that such proofs can be publicly randomized, in such a way that it is infeasible to link the original proof to the randomized one. This procedure is in the following denoted $\textsc{RdProof}(\mathsf{ck}, E, (\mathbf{c}_i, r'_i)_{i=1}^{m}, (\mathbf{d}_j, s'_j)_{j=1}^{n}, \pi)$ where $\mathbf{c}_i$, for $i \in [1,m]$, denotes a commitment to X_i and $\mathbf{d}_j$, for $j \in [1,n]$, a commitment to Y_j, π is the proof to be randomized and the r'_i, s'_j correspond to the new randomness.

Diffie-Hellman Pairing-Product Equation. In the sequel, we will need several times to provide a GS proof with a *DH pairing-product equation.* This equation is denoted $E_{\mathcal{DH}}$ and, on the values $(M,N) \in \mathbb{G}_1 \times \mathbb{G}_2$ and where $g_1, g_2 \in \mathbb{G}_1 \times \mathbb{G}_2$, is given by $E_{\mathcal{DH}}^{(g_1,g_2)}(M,N) \; : \; e(\underline{M}, g_2) \cdot e({g_1}^{-1}, \underline{N}) = 1$.

3.3 GS Proof of Equality under Different Commitment Keys

In the following, we need to prove that two values X_1 and X_2, committed with two different keys, are equal. Such GS proof has already been given in [18] for the DLIN case but we here need it in the SXDH one. Such a proof is in the following denoted

$$\pi_{\mathrm{eq}} \leftarrow d\textsc{Prove}^{\mathrm{eq}}((\mathsf{ck}, \mathsf{ck}'), E_{\mathrm{eq}}, (X_1, X_2), (r, s, r', s')).$$

When the values X_1 and X_2 are committed using the same key, the GS proof is related to the PPE

$$E_{\mathrm{eq}}(X_1, X_2) \quad : \quad e(\underline{X_1}, g_2) \cdot e(\underline{X_2}, {g_2}^{-1}) = 1. \tag{1}$$

Consider now two commitments $\mathbf{c}_1, \mathbf{c}_2$ of X_1, X_2 under *different* commitment keys $\mathsf{ck} := (\mathbf{u}, \mathbf{v})$ and $\mathsf{ck}' := (\mathbf{u}', \mathbf{v}')$ respectively. We want to construct a witness-indistinguishable proof system that X_1 and X_2 satisfy E_{eq} from $\mathbf{c}_1$ and $\mathbf{c}_2$. We have $\mathbf{c}_1 := ({u_{11}}^{r_1} \cdot {u_{21}}^{r_2}, X_1 \cdot {u_{12}}^{r_1} \cdot {u_{22}}^{r_2})$ for uniformly chosen $r_1, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$. If we fix X_2, the proof that the committed value X_1 satisfies equation $e(\underline{X_1}, g_2) = e(X_2, g_2)$ can be reduced to $(\phi_{12} := {g_2}^{r_1}, \phi_{22} := {g_2}^{r_2})$ which can be checked[3] by

$$e(\mathbf{c}_{11}, g_2) = e(u_{11}, \phi_{12}) \cdot e(u_{21}, \phi_{22}) \text{ and} \tag{2a}$$

$$e(\mathbf{c}_{12}, g_2) = e(X_2, g_2) \cdot e(u_{12}, \phi_{12}) \cdot e(u_{22}, \phi_{22}), \tag{2b}$$

which gives us the first part of our GS proof.

Regarding now (2a) and (2b) as a set of equations over variables X_2, ϕ_{21} and ϕ_{22}, we use the GS proof system a second time by committing to these new variables under key $\mathbf{u}'$. Note that as we have already treated the case of $\mathbf{c}_{11}$ and $\mathbf{c}_{12}$, we consider them as fixed in the second part of the GS proof. This leads us to apply the proof algorithm on the following equations.

$$E_{\text{eq}'1}: \qquad e(u_{11}, \underline{{g_2}^{r_1}}) \cdot e(u_{21}, \underline{{g_2}^{r_2}}) = e(\mathbf{c}_{11}, g_2) \tag{3a}$$

$$E_{\text{eq}'2}: \qquad e(\underline{X_2}, g_2) \cdot e(u_{12}, \underline{{g_2}^{r_1}}) \cdot e(u_{22}, \underline{{g_2}^{r_2}}) = e(\mathbf{c}_{12}, g_2) \tag{3b}$$

The resulting complete GS proof will be given in the full version of the paper. Regarding the randomization of such a proof, one need to update commitments and proofs to the new randomness for the commitment on X_1.

3.4 Automorphic Signatures

Automorphic signatures have been introduced in [2,15] as a new signature scheme where (i) the verification keys lie in the message space, (ii) messages and signatures consist of elements of a bilinear group, and (iii) verification is done by evaluating a set of pairing-product equations. Automorphic signatures are used in [16] to construct commuting signatures, where a commuting signature is concretely a verifiably encrypted automorphic signature. We now described the instantiation given in [15].

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be a bilinear environment as defined above. We also need $h, k, u \in \mathbb{G}_1$. The message space is $\mathcal{DH} := \{({g_1}^m, {g_2}^m) \mid m \in \mathbb{Z}_p\}$. The secret key is $x \in \mathbb{Z}_p^*$ and the related public verification key is $\mathsf{vk} := (X, Y) = ({g_1}^x, {g_2}^x)$. The signature of a message $(M, N) \in \mathcal{DH}$ is done by picking $c, r \in \mathbb{Z}_p$ at random and computing

$$\sigma := \big(A := \big(h \cdot M \cdot k^r\big)^{\frac{1}{x+c}}, B := u^c, D := {g_2}^c, R := {g_1}^r, S := {g_2}^r\big).$$

A signature $\sigma = (A, B, D, R, S)$ on a message $(M, N) \in \mathcal{DH}$ is valid iff $e(A, Y \cdot D) = e(h \cdot M, g_2) \cdot e(k, S)$, $e(B, g_2) = e(u, D)$ and $e(R, g_2) = e(g_1, S)$. Details can be found in [15,16].

[3] As explained in Section 6.1 of the full version of [20], two group elements are enough for such a proof.

4 Commuting Signatures and Some New Extensions

We here introduce commuting signatures and some extensions which are of independent interest. In our main scheme, we need to sign a vector of messages while individual messages can be committed using different commitment extraction keys. To the best of our knowledge, this description is new and we here give a general way to treat such a case.

4.1 Additional Commitments

Fuchsbauer [16], when constructing commuting signatures, makes use of commitment on Diffie-Hellman (DH) tuples which are messages signed by the automorphic signature scheme introduced above.

Commitment on Diffie-Hellman Tuples. Let $k \in \mathbb{G}_1$. A commitment on a DH tuple (M, N) takes as input some randomness $(t, \mu, \nu, \rho, \sigma)$. It first computes $(P, Q) = ({g_1}^t, {g_2}^t)$ and $U = M \cdot k^t$. It next computes SXDH commitments $\mathbf{c}_M = \text{SXDH}\textsc{Com}(\mathsf{ck}, M, \mu)$, $\mathbf{c}_N = \text{SXDH}\textsc{Com}(\mathsf{ck}, N, \nu)$, $\mathbf{c}_P = \text{SXDH}\textsc{Com}(\mathsf{ck}, P, \rho)$ and $\mathbf{c}_Q = \text{SXDH}\textsc{Com}(\mathsf{ck}, Q, \sigma)$. Finally, it executes the SXDH GS proofs $\pi_M = \textsc{Prove}(ck, E_{\mathcal{DH}}^{(g_1,g_2)}, (M, N), (\mu, \nu))$, $\pi_P = \textsc{Prove}(ck, E_{\mathcal{DH}}^{(g_1,g_2)}, (P, Q), (\rho, \sigma))$ and $\pi_U = \textsc{Prove}(ck, E_U, (M, Q), (\mu, \sigma))$ where

$$E_U(M, Q) : e(k^{-1}, \underline{Q}) \cdot e(\underline{M}, {g_2}^{-1}) = e(U, g_2)^{-1}$$

The commitment on (M, N) is $C = \textsc{Commit}\text{DH}(\mathsf{ck}, (M, N), (t, \mu, \nu, \rho, \sigma)) = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$. Such a commitment is randomizable [16] using the randomization techniques for SXDH commitments and GS proofs.

More Simple Commitment on DH Tuples. Before producing a commuting signature on a message (M, N), it is necessary to produce such a commitment on the message. In some other cases, we need to produce a commitment on a DH tuple but without any relation with the execution of a commuting signature on the underlying message. In this case, we do not need to make so a complicated commitment on a DH tuple. In fact, the values P and Q are in this case not necessary (see [16] for details). The above commitment, denoted $\textsc{Commit}\text{DH}^{-}(\mathsf{ck}, (M, N), (\mu, \nu))$ is thus reduced to the values $(\mathbf{c}_M, \mathbf{c}_N, \pi_M)$.

Commitment to an Automorphic Signature When the Message Is Known But Committed. An automorphic signature (see [15] and Section 3.4) can be committed as follows, when the signed message (M, N) is known[4] but committed as a DH tuple: $\textsc{Commit}\text{DH}(\mathsf{ck}, (M, N), (\mu, \nu))$.

Let $\sigma = (A, B, D, R, S)$ be an automorphic signature on the message (M, N). A commitment on a signature corresponds to $(\mathbf{c}_\sigma, \pi_\sigma) = ((\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S), (\pi_A, \pi_B, \pi_R))$ where c_X corresponds to $\text{SXDH}\textsc{Com}(\mathsf{ck}, X, \alpha_X)$ (for $X \in \{A, B, D, R, S\}$) and where $\pi_A = \textsc{Prove}(\mathsf{ck}, E_A, (A, \alpha_A), (M, \mu), (D, \alpha_D), (S, \alpha_S))$,

[4] The case where the message is not known is different, see [16].

$\pi_B = \textsc{Prove}(\mathsf{ck}, E_{\mathcal{DH}}^{(u,g_2)}, (B, \alpha_B), (D, \alpha_D))$ and finally $\pi_R \leftarrow \textsc{Prove}(\mathsf{ck}, E_{\mathcal{DH}}^{(g_1,g_2)}, (R, \alpha_R), (S, \alpha_S))$ with

$$E_A(A, M, D, S) : e(k^{-1}, \underline{S}) \cdot e(\underline{A}, Y) \cdot e(\underline{M}, {g_2}^{-1}) \cdot e(\underline{A}, \underline{D}) = e(h, g_2).$$

In the following, such a procedure is denoted $\textsc{CommitSign}(\mathsf{ck}, \mathsf{vk}, (M, N), \sigma, (\mu, \nu, \alpha_A, \alpha_B, \alpha_D, \alpha_R, \alpha_S))$.

Commitment to an Automorphic Signature When the Message Is Known But not Committed. The case where the message (M, N) is publicly known (and thus not committed) can be simplified since the equation E_A becomes

$$E_A(A, M, D, S) : e(k^{-1}, \underline{S}) \cdot e(\underline{A}, Y) \cdot e(\underline{A}, \underline{D}) = e(h \cdot M, g_2).$$

The rest of the procedure is done similarly and the result is $\textsc{CommitSign}^{-}(\mathsf{ck}, \mathsf{vk}, (M, N), \sigma, (\mu, \nu, \alpha_A, \alpha_B, \alpha_D, \alpha_R, \alpha_S))$ in the following.

4.2 Simple Commuting Signature: One Committed Message and One Commitment Key

We now recall the SigCom algorithms to sign a DH tuple committed using the above commitment scheme for DH tuples as described in [16]. The secret signing key is $\mathsf{sk} = x \in \mathbb{Z}_p^*$ and the corresponding public key is the DH tuple $\mathsf{vk} = (X = {g_1}^x, Y = {g_2}^x)$, as for an automorphic signature. The signature of a DH commitment $C = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ on (M, N) is done as follows (see [16] for details).

- The signer first picks fresh random values $c, r \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha, \beta, \delta, \rho', \sigma' \xleftarrow{\$} \mathbb{Z}_p^2$.
- She next computes an automorphic signature (see [2,15] and Section 3.4) $A = (h \cdot U \cdot k^r)^{\frac{1}{x+c}}$, $B = u^c$, $D = {g_2}^c$, $R = {g_1}^r$ and $S = {g_2}^r$.
- She also computes the SXDH commitments $\mathbf{c}_A = \textsc{SXDHCom}(ck, A, \alpha)$, $\mathbf{c}_B = \textsc{SXDHCom}(\mathsf{ck}, B, \beta)$, $\mathbf{c}_D = \textsc{SXDHCom}(\mathsf{ck}, D, \delta)$, $\mathbf{c}_R = \mathbf{c}_P \odot \textsc{SXDHCom}(\mathsf{ck}, R, \rho')$ and $\mathbf{c}_S = \mathbf{c}_Q \odot \textsc{SXDHCom}(\mathsf{ck}, S, \sigma')$.
- She makes the GS proofs: $\pi'_A := \pi_U \odot \textsc{Prove}(\mathsf{ck}, E_{A^\dagger}, (A, D), (\alpha, \delta))$, $\pi_A := \textsc{RdProof}(\mathsf{ck}, E_A, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_D, 0), (\mathbf{c}_S, \sigma'), \pi'_A)$, $\pi_B = \textsc{Prove}(ck, E_{\mathcal{DH}}, (B, D), (\beta, \delta))$ and $\pi_R := \textsc{RdProof}(\mathsf{ck}, E_R, (\mathbf{c}_R, \rho'), (\mathbf{c}_S, \sigma'), \pi_P)$ where

$$\begin{aligned} E_A(A, M, D, S) &: e(k^{-1}, \underline{S}) \cdot e(\underline{A}, Y) \cdot e(\underline{M}, {g_2}^{-1}) \cdot e(\underline{A}, \underline{D}) = e(h, g_2) \\ E_{A^\dagger}(A, D) &: e(\underline{A}, Y) \cdot e(\underline{A}, \underline{D}) = 1 \\ E_B(B, D) &: e(u^{-1}, \underline{D}) \cdot e(\underline{B}, g_2) = 1 \\ E_R(R, S) &: e({g_1}^{-1}, \underline{S}) \cdot e(\underline{R}, g_2) = 1 \end{aligned}$$

The PPE $E_{A^\dagger}$ is not truly verified but is necessary to produce the GS proof on E_A (see [16]). The signature is $\Sigma := \big(\mathbf{c}_\Sigma = (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S), \pi_\Sigma = (\pi_A, \pi_B, \pi_R)\big)$.

4.3 Vector of Committed Messages and One Commitment Key

In our anonymous credential scheme with encrypted attributes, we need to sign several messages at the same time. For this purpose, we need to adapt the above commuting signature, which can easily be done by using the recent technique of [3] and [17]. On input a signing key x and a vector $(C_1, \ldots, C_n)$ of commitments to the DH tuples $(M_1, N_1), \ldots, (M_n, N_n)$, the whole procedure works as follows. Let $n_{\max} \in \mathbb{N}$ be the maximum number of messages we can sign together and let $\mathsf{sk} = x \in \mathbb{Z}_p^*$ be the used secret signing key, related to the public key $\mathsf{vk} = (X = {g_1}^x, Y = {g_2}^x)$.

1. Run the signing key generation $n+1$ times to get $(\mathsf{vk}_i = (X_i, Y_i), \mathsf{sk}_i = x_i)_{i=0}^n$, where $\mathsf{sk}_i \in \mathbb{Z}_p^*$.
2. Compute the following automorphic signatures (see Section 3.4): Γ_0 on vk_0 under sk and Δ_0 on $\langle n \rangle$ (as defined at the beginning of Section 3) under sk_0. The message vk_0 is next committed using CommitDH$^-$ to obtain $\mathbf{c}_{\mathsf{vk}_0}$. Finally, the signatures Γ_0 and Δ_0 are also committed using the procedures CommitSign (resp. CommitSign$^-$ since $\langle n \rangle$ is not committed), and obtain $(\mathbf{c}_{\Gamma_0}, \pi_{\Gamma_0})$ (resp. $(\mathbf{c}_{\Delta_0}, \pi_{\Delta_0})$).
3. Executes n times the following: produce an automorphic signature Γ_i on vk_i under sk_0 and a signature Δ_i on $\mathsf{vk}_i \odot \langle i \rangle$ under sk_0. In addition, commit to each message and each signature. Again, for $i \in [1, n]$, the message vk_i is committed using CommitDH$^-$ to obtain $\mathbf{c}_{\mathsf{vk}_i}$. The signatures Γ_i and Δ_i are also committed using CommitSign (this time, the message $\mathsf{vk}_i \odot \langle i \rangle$ related to Δ_i is not totally known), and obtain $(\mathbf{c}_{\Gamma_i}, \pi_{\Gamma_i})$ and $(\mathbf{c}_{\Delta_i}, \pi_{\Delta_i})$ respectively.
4. Executes n times the SigCom procedure for a single message (see Section 4.2 above). The i-th execution takes as inputs the commitment C_i and the related signing key sk_i and outputs a commuting signature $\Sigma_i = (\mathbf{c}_{\Sigma_i}, \pi_{\Sigma_i})$.
5. Commit to all public keys vk_i as $\mathbf{c}_{\mathsf{vk}_i} = (\mathbf{c}_{X_i} = \text{SXDHCom}(\mathsf{ck}, X_i, \zeta_i), \mathbf{c}_{Y_i} = \text{SXDHCom}(\mathsf{ck}, Y_i, \psi_i))$ and $\pi_{X_i} = \text{Prove}(\mathsf{ck}, E_{\mathcal{DH}}^{(g_1, g_2)}, (X_i, \zeta_i), (Y_i, \psi_i))$.
6. As the above signatures Σ_i are valid under the plain public keys vk_i, we need to use next the AdPrC$_K$ procedure[5] [16] to adapt Σ_i so that it is valid under the public keys which are committed in the $\mathbf{c}_{\mathsf{vk}_i}$'s. Given $\Sigma_i = (\mathbf{c}_{A_i}, \mathbf{c}_{B_i}, \mathbf{c}_{D_i}, \mathbf{c}_{R_i}, \mathbf{c}_{S_i}, \pi_{A_i}, \pi_{B_i}, \pi_{R_i})$, the resulting commuting signature is next $\Sigma_i' = (\mathbf{c}_{A_i}, \mathbf{c}_{B_i}, \mathbf{c}_{D_i}, \mathbf{c}_{R_i}, \mathbf{c}_{S_i}, \pi'_{A_i}, \pi_{B_i}, \pi_{R_i})$ where the proof π'_{A_i} = RdProof(ck, $E_{\hat{A}}$, $(\mathbf{c}_{A_i}, 0)$, $(\mathbf{c}_{M_i}, 0)$, $(\mathbf{c}_{S_i}, 0)$, (SXDHCom(ck, Y_i, 0), ψ_i), $(\mathbf{c}_{A_i}, 0)$, π_{A_i}), with

$$E_{\hat{A}}(A, M, S, Y, D) : e(k^{-1}, \underline{S}) \cdot e(\underline{A}, \underline{Y}) \cdot e(\underline{M}, {g_2}^{-1}) \cdot e(\underline{A}, \underline{D}) = e(h, g_2).$$

The whole commuting signature on the vector $\big((M_1, N_1), \ldots, (M_n, N_n)\big)$ is finally $\Sigma = \big((\mathbf{c}_{\mathsf{vk}_i}, \mathbf{c}_{\Gamma_i}, \mathbf{c}_{\Delta_i}, \pi_{\mathsf{vk}_i}, \pi_{\Gamma_i}, \pi_{\Delta_i})_{i=0}^n, (\Sigma_i')_{i=1}^n\big)$.

[5] The AdPrC$_K$ procedure (Adapt Proof when Committing to the Key) allows to adapt proofs when committing or decommitting to the verification key.

4.4 Vector of Committed Messages and Several Commitment Keys

We now introduce the way we will use such signatures in our anonymous credential scheme with encrypted attributes. To the best of our knowledge, this procedure is new and may be of independent interest.

We assume having the DH tuple commitments $(C_1, \dots, C_n)$ on the messages (M_1, N_1), ..., (M_n, N_n) under possibly several commitment keys $(\mathsf{ck}_1, \dots, \mathsf{ck}_n)$, using the randomness (μ_1, ν_1), ..., (μ_n, ν_n). We use a commuting signature with the commitment key denoted ck and the secret signing key $\mathsf{sk} = x$.

The idea is to use the above procedure. We commit to each message using the same commitment key and prove that the committed values are equals, using our new procedure given in Section 3.3. More precisely, we have the following.

1. For all $i \in [1, n]$, produce a DH tuple commitment on (M_i, N_i) using ck, which outputs $C_i^{(\mathsf{ck})} = \textsc{CommitDH}(\mathsf{ck}, (M_i, N_i), (\mu_i, \nu_i))$.
2. For all $i \in [1, n]$, produce a GS proof of equality under different commitment keys (see Section 3.3): $\pi_{\mathrm{eq}_i} = d\textsc{Prove}^{\mathrm{eq}}((\mathsf{ck}, \mathsf{ck}_i), E_{\mathrm{eq}}, (C_i^{(\mathsf{ck})}, C_i), (\mu_i^{(\mathsf{ck})}, \nu_i^{(\mathsf{ck})}, \mu_i, \nu_i))$.
3. Compute the commuting signature, using the secret key x, for the vector of messages $(C_1^{(\mathsf{ck})}, \dots, C_n^{(\mathsf{ck})})$ on the single commitment key ck, as described in Section 4.3 just above. This procedure outputs Σ.

In this procedure, we remark that the two first steps are not necessarily executed by one single actor. If the messages are known by different parties, each one can execute these two first procedures and the owner of the signing key can next produce the commuting signature.

Note that not all the elements need to be hidden. In particular, thanks to the homomorphic properties of the SDXH commitment, there is no need to commit to $\mathsf{vk}_i \odot \langle i \rangle$ once vk_i is already committed. The proofs of validity are done *w.r.t.* the same commitment $\mathbf{c}_{\mathsf{vk}_i}$. Furthermore, $\langle n \rangle$ must stay in clear, since the length of the vector has to be checkable.

Security Considerations. Regarding the security of the above constructions, we first note that signatures on vector of committed messages have the same probability distribution as directly generated signatures on vector elements. Then the reduction from [3] is adapted to the unforgeability notion of commuting signatures. An adversary, given an access to an oracle which signs committed values, is not able to forge committed signatures on values that were not queried to the oracle in a committed form (thanks to the simulation algorithm, as in [16]). Thus, from a forgery on a vector of committed messages, we extract a forgery on the underlying commuting signature scheme with non-negligible probability.

4.5 Commuting Signatures in Privacy Enhancing Cryptography

In this paper, we mainly focus on anonymous credential systems as described and used in [12,8,21]. It exists in the literature several related cryptographic tools which also aims at preserving the privacy of consumers with close methods and problems. This is for example the case for group [13], blind [11] and traceable [23] signatures.

Commuting signatures have been introduced in [16] and the existing construction is based on automorphic signatures [2,15]. Automorphic signatures, as said in [2,15], permits to efficiently create blind signatures. It is also possible to give a more efficient variant of Groth's group signature scheme [19] by using automorphic signatures instead of certified signatures. For traceable signatures, recent papers [14,1] have also proposed variants of traceable signature, also using automorphic signatures. In particular, the signature confirmation/denial [1] and the efficient tracing [14] techniques can be incorporated into traditional anonymous credential (and related tools) systems.

In our case, this is slightly different since automorphic signatures are not enough. In fact, if commuting signatures can, in most cases, be used instead of automorphic signatures (even if the result is obviously less efficient), the contrary is false. As for delegatable anonymous credentials [4], we need at the same time (i) a signature process on a message which is not known by the signer (or the receiver in our case) and (ii) a process where a unique signature is used several times while being untraceable, even for the signer. This is exactly the aim of commuting signatures, as explained in [16], since signing and encrypting/commuting commute. The fact that a unique signature can be used several times, using the randomization techniques of commitment schemes and Groth-Sahai proofs, is what permits our scheme to be multi-show, while this was not the case for the Guajardo *et al.* construction [21] since they make use of blind signatures.

5 A Multi-show Anonymous Credential Scheme with Encrypted Attributes

We have now introduced all the elements we need to describe our multi-show anonymous credential scheme with encrypted attributes. We first sketch an overview before giving details and security arguments.

5.1 Overview of Our Solution

We want to design an anonymous credential scheme with encrypted attributes. We first take as a basis the work given on non-interactive anonymous credential schemes by Belenkiy *et al.* in [5], and later refined by Fuchsbauer [16] by using commuting signatures. To introduce the property of encrypted attributes, we make use of extractable commitments, arguing that a perfectly binding extractable committed value exactly corresponds to a ciphertext of a public key encryption scheme.

Our scheme next works as follows. In a nutshell, the issuance protocol consists for the issuer in producing a commuting signature on the attributes, using the commitment key of the verifier to encrypt/commit to the attributes. The resulting commuting signature is next given to the user who can play a verification protocol with the verifier by randomizing the commuting signature, which is composed of commitments and Groth-Sahai proofs. As two different randomizations of the same inputs are indistinguishable, we obtain the multi-show property, as expected.

Following [21], the user, during the issuance protocol, generates a random secret α for the issued credential cred. This secret (or the related public key) should be signed by the issuer in the above commuting signature. However, the verifier should not be able to obtain it and we thus need to use a commitment key which is different from the verifier's one for this particular message α. For this purpose, we use the mechanism proposed in Section 4.4, which permits to produce a commuting signature on messages committed with possibly different commitment keys. The other commitment extraction key is related to the verifier (who need to be known at the issuing process, as for the scheme in [21]) and the related commitment scheme is used by the issuer to commit to the attributes.

During the verification process, the user should prove the knowledge of the secret α, without revealing it, for obvious reasons. The GS proof system does not permit such a proof of knowledge and we need to do something more. In fact, we use the powerfulness of automorphic signatures [2,15] for which the verification keys lie in the message space. Thus, during the verification process, the user produces an automorphic signature on some plain message related to the context (or sent by the verifier), using the committed secret key α.

Finally, as we can use the general commuting signature scheme for a vector of committed messages and several commitment keys, we are also able to deal with several verifiers. Thus, the issuance protocol permits the issuer to create a credential with several encrypted attributes, for potentially several different verifiers, which is a new property not proposed in [21]. More formally, we give the following construction.

5.2 Algorithms and Protocols

Following [21], we outline the case where the issuer encrypts attributes for verifiers. Thus, the plain attributes remain hidden to the user. Our scheme is easily adaptable to other policies, giving the extraction key and making the extractable commitment accordingly.

Key Generation. Our scheme works on a bilinear environment (p, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, e, g_1, g_2) as defined in Section 3. We also need randomly picked generators $h, k, u, v \in \mathbb{G}_1$. As shown above (see Section 4.4), we need a commitment key for the commuting signature. For this purpose, we generate a commitment public key $\mathsf{ck} := (\mathbf{u}, \mathbf{v})$, while the corresponding secret key is known by nobody[6]. We finally set $\mathsf{grp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, h, k, u, v, g_2, \mathsf{ck})$

[6] As we use Groth-Sahai proofs, we are in the common reference string model.

Issuer key generation. Each issuer generates her own keys. For this purpose, she picks at random $x_{\mathcal{I}} \in \mathbb{Z}_p^*$ as her secret key $\mathsf{sk}_{\mathcal{I}}$ and computes the public key as a DH tuple: $\mathsf{pk}_{\mathcal{I}} = (X_{\mathcal{I}}, Y_{\mathcal{I}}) = ({g_1}^{x_{\mathcal{I}}}, {g_2}^{x_{\mathcal{I}}})$.

Verifier key generation. Each verifier also generates her keys. As said before, these corresponds to the ones for an SXDH commitment scheme: the secret key is $(\alpha_{\mathcal{V}_1}, \alpha_{\mathcal{V}_2}) \in (\mathbb{Z}_p^*)^2$ and the corresponding public key is $\mathsf{ck}_{\mathcal{V}} := (\mathbf{u}_{\mathcal{V}}, \mathbf{v}_{\mathcal{V}})$ as defined in Section 3.1.

Issuance Protocol. This protocol is played by an issuer with keys $(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}})$ and is related to the attributes denoted $(m_1, \ldots, m_N)$, such that each $m_i = (M_i, N_i)$. We here consider that each attribute m_i is "encrypted" for the verifier i with public key $\mathsf{ck}_{\mathcal{V}_i}$ (with possibly several times the same verifier), which is not proposed in [21].

We first assume that some entity (possibly the issuer itself) first encrypts the plain attributes $(m_1, \ldots, m_N)$. For this purpose, it produces, for all $i \in [1, N]$, a commitment C_i, using $\mathsf{ck}_{\mathcal{V}_i}$, on each (M_i, N_i) and using the DH tuple commitment scheme described in Section 4.1: $c_i := \textsc{CommitDH}(\mathsf{ck}_{\mathcal{V}_i}, (M_i, N_i), (\mu_i, \nu_i))$. These commitments are next given on input to both the issuer (if necessary) and the user. The issuance protocol is next divided into several steps.

User Generate an automorphic signing secret key $\alpha \in \mathbb{Z}_p^*$. The pair $(X = {g_1}^{\alpha}, Y = {g_2}^{\alpha})$ corresponds to the public verification key of an automorphic signature scheme related to α and is also used to commit to α. Thus, the user next commits to α, using the commitment key ck of the commuting signature scheme: $C_0 = \textsc{CommitDH}(\mathsf{ck}, (X, Y), (\xi, \xi'))$. The result is sent to the issuer.

Issuer The issuer next produces a commuting signature on all the commitments $C_0, C_1, \cdots, C_N$, using the algorithm given in Section 4.4, the secret key $\mathsf{sk}_{\mathcal{I}}$ as the signing secret key, and the key ck for the related commitment scheme. As C_0 is already committed using ck, it is not necessary to commit again to it and produce a GS proof of equality. This is however necessary for the other C_i's.
The resulting signature Σ is sent to the user, together with the C_i's.

User Verify the commuting signature Σ (see [16]) and save the C_i's and the credential $((\alpha, \xi, \xi'), \mathsf{cred} := \Sigma)$.

Verification Protocol. Let $\mathcal{U}$ be a user having beforehand carried out an issuance protocol with an issuer, and thus having a credential $((\alpha, \xi, \xi'), \mathsf{cred})$ as defined above, and on some encrypted/committed attributes $(C_1, \ldots, C_N)$. She now interacts with a verifier having access to a decryption/extraction key $(\alpha_{\mathcal{V}_1}, \alpha_{\mathcal{V}_2})$ related to one commitment key $\mathsf{ck}_{\mathcal{V}}$ used to create cred. For the sake of simplicity, we assume that $\mathsf{ck}_{\mathcal{V}} = \mathsf{ck}_{\mathcal{V}_1}$ in the above issuance protocol.

User Randomize her credential cred and commitments C_i's (to obtain the $\tilde{C}_i$'s) by using the randomization technique of commuting signatures [16].

The new commuting signature is $\tilde{\Sigma}$. Let $(\tilde{\xi}, \tilde{\xi}')$ be the new randomness associated to $\tilde{C}_0$ and α. User U produces[7] a signature σ on some fresh message related to the context[8] (or sent by $\mathcal{V}$), using the secret key α. Let us recall that α is related to $(X, Y) := ({g_1}^{\alpha}, {g_2}^{\alpha})$ and that (X, Y) is committed in $\tilde{C}_0$. The fresh message is hashed to $m \in \mathbb{Z}_p$ and mapped to $(M, N) := ({g_1}^{m}, {g_2}^{m})$. User produces an automorphic signature (see Section 3.4) $\sigma := (A, B, D, R, S) \leftarrow \text{Sign}(\alpha, (M, N))$ and proves that the signature σ is valid under the verification key committed in $\tilde{C}_0$ by computing $\pi \leftarrow \text{Prove}(\mathsf{ck}, E_Y, (X, Y), (\tilde{\xi}, \tilde{\xi}'))$ with

$$E_Y(Y) : e(A, \underline{Y} \cdot D) = e(h \cdot M, g_2) \cdot e(k, S).$$

She next sends to the verifier $\tilde{\Sigma}$, π and σ.

Verifier The verifier checks the commuting signature $\tilde{\Sigma}$, the proof π and the signature σ. She is next able to use her secret key $(\alpha_{\mathcal{V}_{1,1}}, \alpha_{\mathcal{V}_{1,2}})$ to extract the attribute (M_1, N_1) committed in C_1 (see Section 3.1).

Remark 1. The verifier retrieves a plain attribute as a DH tuple. The way for her to retrieve an *understandable* attribute can be treated by either considering bits (as done in [21]), or very small messages (to test all possibilities) or (for bigger messages) to publish a cross-reference table between understandable messages and corresponding DH tuples.

Regarding the security of our new construction, we give the following theorem, while the assumptions are given in Appendix A and the proof will be given in the full version of the paper.

Theorem 1. *Our anonymous credential scheme with encrypted attributes ensures the unforgeability, attribute masking and (honest-verifier) user privacy properties under the q-ADHSDH, the AWFCDH and the SXDH assumptions in* $(\mathbb{G}_1, \mathbb{G}_2)$.

Acknowledgments. This work has been supported by the French Agence Nationale de la Recherche under the PACE 07 TCOM Project, and by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II. We are also grateful to Georg Fuchsbauer for his suggestions of improvement on Section 4.3, to Sherman Chow for his help on this final version and to anonymous referees for their valuable comments.

References

1. Abe, M., Chow, S.S.M., Haralambiev, K., Ohkubo, M.: Double-Trapdoor Anonymous Tags For traceable Signatures. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 183–200. Springer, Heidelberg (2011)

[7] This additional signature is added in order to prevent credentials sharing. This aspect is not taken into account in the model. To adopt an *all-or-nothing* policy, each α contained in each credential may be the same value, and this value is a user secret necessary to prove possession of a credential.

[8] Like the concatenation of the current time and the verifier public key.

2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
3. Abe, M., Haralambiev, K., Ohkubo, M.: Efficient Message Space Extension for Automorphic Signatures. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 319–330. Springer, Heidelberg (2011)
4. 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), `http://eprint.iacr.org/2008/428`
5. 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), `http://eprint.iacr.org/2007/384`
6. Brands, S.: Rethinking PKI and digital certificates - building in privacy. PhD thesis, Eindhoven Institute of Technology (1999)
7. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
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.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
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. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–203 (1983)
12. Chaum, D., Evertse, J.-H.: A Secure and Privacy-Protecting Protocol for Transmitting Personal Information Between Organizations. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 118–167. Springer, Heidelberg (1987)
13. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
14. Chow, S.S.M.: Real Traceable Signatures. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 92–107. Springer, Heidelberg (2009)
15. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), `http://eprint.iacr.org/`
16. Fuchsbauer, G.: Commuting Signatures and Verifiable Encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 224–245. Springer, Heidelberg (2011)
17. Fuchsbauer, G.: Personal Communication (2011)
18. Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable Constant-Size Fair E-Cash. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 226–247. Springer, Heidelberg (2009)
19. Groth, J.: Fully Anonymous Group Signatures without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)

20. Groth, J., Sahai, A.: Efficient non-Interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
21. Guajardo, J., Mennink, B., Schoenmakers, B.: Anonymous Credential Schemes with Encrypted Attributes. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 314–333. Springer, Heidelberg (2010)
22. IBM. Identity mixer - Idemix, `http://www.zurich.ibm.com/security/idemix/`
23. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
24. Microsoft. Microsoft U-Prove, `https://connect.microsoft.com/site1188`
25. De Santis, A., Yung, M.: Cryptographic Applications of the Non-Interactive Metaproof and many-Prover Systems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 366–377. Springer, Heidelberg (1991)

A Used Assumptions

[q-ADHSDH] The q-Asymmetrical Double Hidden Strong Diffie-Hellman Problem. Given $(g_1, X = {g_1}^x,\ h,\ u, g_2, Y = {g_2}^x) \in {\mathbb{G}_1}^4 \times {\mathbb{G}_2}^2$, $q-1$ tuples $\left\{(A_i = (h \cdot {g_1}^{v_i})^{\frac{1}{x+c_i}}, B_i = u^{c_i}, D_i = {g_2}^{c_i}, V_i = {g_1}^{v_i}, W_i = {g_2}^{v_i})\right\}_{i=1}^{q-1}$ for $c_i, v_i \overset{\$}{\leftarrow} \mathbb{Z}_p$ find a new tuple (A, B, D, V, W) such that $e(A, Y \cdot D) = e(h \cdot V, g_2)$, $e(B, g_2) = e(u, D)$ and $e(V, g_2) = e(g_1, W)$.

[AWFCDH] The Asymmetric Weak Flexible Computational Diffie-Hellman Problem. Given $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, $A = {g_1}^a$ for $a \overset{\$}{\leftarrow} \mathbb{Z}_p$, find a tuple $(R, S, M, N) \in (\mathbb{G}_1^*)^2 \times (\mathbb{G}_2^*)^2$ such that $e(A, S) = e(M, g_2) \quad e(M, g_2) = e(g_1, N) \quad e(R, g_2) = e(g_1, S)$, i.e. there exists $r \in \mathbb{Z}_p$ such that $(R, S, M, N) = ({g_1}^r, {g_2}^r, {g_1}^{ra}, {g_2}^{ra})$

[SXDH] The Symmetric External Diffie-Hellman Problem. Given $({g_1}^r, {g_1}^s, {g_1}^t)$ (resp. $({g_2}^r, {g_2}^s, {g_2}^t)$) for random $r, s \in \mathbb{Z}_p$ (resp. $r', s' \in \mathbb{Z}_p$), decide whether $t = rs \mod p$ or t is uniform in $\mathbb{Z}_p$.

For each problem given above, the corresponding assumption states that the problem is hard in $(\mathbb{G}_1, \mathbb{G}_2)$.