Anonymous Credentials from (Indexed) Aggregate Signatures

Sébastien Canard
Orange Labs, Caen, France
Applied Crypto Group
sebastien.canard@orange-ftgroup.com

Roch Lescuyer
Orange Labs, Caen, France
Applied Crypto Group
roch.lescuyer@orange-ftgroup.com

ABSTRACT

Anonymous credential systems allow users to obtain certified credentials (a driving license, a student card, etc.) from organizations and then later to prove the possession of one (or more) credential(s) to another party, while minimizing the information given to the latter. While current constructions use zero-knowledge proofs of knowledge of a signature or blinding mechanisms, we keep in this paper a new approach, based on aggregate signature schemes.

In particular, we use the notion of *indexed aggregate signature* by which we aggregate several signatures into a single one, but only if they are initially related to the same index. The resulting anonymous credential system is the first one which efficiently enables a user to prove the possession, in an untraceable way, of several credentials issued by possibly *several* organizations.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms

Algorithms, Design, Performance, Security

Keywords

Cryptographic protocols, Anonymous credentials, Privacy-enhancing systems, Aggregate signatures

1. INTRODUCTION

Privacy is today one of the most interesting problem our modern society needs to face to. Among privacy principles, the minimization of users' data when they are interacting with service providers is of great importance. For example, suppose that a service provider wants to offer some privileges to its customers when they are *e.g.* students and/or under 25. This service provider does not necessarily need to obtain all the information about the customer, such as her name, exact address, exact age, etc. However, it needs to be sure that these informations, called attributes, are certified by an authorized entity (in our example, the university or the country's authority). This use case is typically addressed by the concept of anonymous credential, which has been introduced by Chaum in [13].

Related work. Several anonymous credential systems have been proposed, using semi-trusted third parties [14], or not. Among the previously proposed systems, some of them are theoretical only [18, 25] while the others are practical, such as those proposed by Camenisch and Lysyanskaya [10, 11] or by Brands [7]. More recent papers also address variants of anonymous credentials, such as the delegation of credentials [2, 21], or the problem of the revocation [9].

The Camenisch-Lysyanskaya framework [10, 11], proposed in [3] in the standard model and implemented by IBM as the Idemix system [23], is related to the use of group signature schemes. The Brands's view [7], based on the use of blind signatures and implemented in the Microsoft U-Prove technology [26], is much more efficient but induces some kind of traceability of the (still anonymous) user. This drawback can be avoided by asking for several certifications of the same credentials to the same organization, at the cost of a less efficient and easy-to-use solution.

Multi-organizations. In our above practical example, if the user has to prove that she is both student and under 25, she may need to use both her student card (delivered by the university) and her identity card (delivered by the country's authority). Since this case can occur in real life applications of anonymous credentials, cryptographic schemes should take into account the case where a user manipulates at the same time several credentials certified by possibly different organizations. However, none of existing schemes directly considers this case. The sole exception is the Verheul proposal [29] but the construction is not proven secure. Other constructions, based on group or blind signatures, formally consider multi-organizations (see *e.g.* [11]) but not the above case where a user has to prove the possession of several credentials not issued by the same organization.

Regarding constructions in [10, 11] based on group signatures, each credential is represented by the belonging to a group managed by the related organization. When one user has to prove the possession of several credentials, she has to prove that she belongs to two different groups, which necessitates a (time and space) complexity which depends on the

DIM'11, October 21, 2011, Chicago, Illinois, USA.

number of manipulated credentials. This is similar for blind signature based solutions [7] since each organization is the signer of the used blind signature scheme. In this paper, we consequently enrich the current model and next propose a practical scheme with this additional feature.

Aggregate Signatures. For this purpose, we adopt a new approach for anonymous credential systems, which is based on the use of aggregate signatures [6, 24]. Such signature schemes permit to aggregate several signatures into a single one, while initial signatures have been produced under potentially different messages and public keys. As the aggregation should not be done to create false relations by using certified credentials given to different users, we need to introduce a new variant (in the flavour of the synchronized aggregate signatures [22, 1]) we call *indexed* aggregate signature scheme. In our case, aggregation is only possible if signatures have been produced using the same initial index. This type of signature scheme is not necessarily related to anonymous credentials and may be of independent interest.

Our contribution. As a conclusion, our contribution is threefold. First, we propose a model for anonymous credentials where the multi-organizations case is made explicit. Secondly, we propose to achieve this with the help of indexed aggregate signatures. Finally we propose a concrete scheme.

Organization of the paper. The paper is organized as follows. In Section 2, we formally describe our new model. Section 3 introduces the new notion of indexed aggregate signature scheme, with the formal model and a concrete scheme. In Section 4, we give some additional cryptographic building blocks we need and our new anonymous credential system built upon indexed aggregate signatures. We finally conclude and discuss open problems in Section 5.

2. MODEL FOR ANONYMOUS CREDENTIAL SYSTEMS

In this section, we introduce a model for anonymous credentials, based on the work on [10, 29] and making some modifications that are, from our point of view, more relevant to real-life needs. In particular, we add the possibility for a user to prove the possession of several certified credentials coming from potentially different organizations.

2.1 Procedures

An *anonymous credential system* is given by the following algorithms, involving organizations, users and verifiers.

SETUP. This algorithm takes as input a security parameter λ and outputs the scheme parameters param, which are given to all algorithms as auxiliary input.

USERKG, ORGKG. These algorithms produce keys pairs $(\mathsf{pk}_U, \mathsf{sk}_U)$ for users and $(\mathsf{pk}_O, \mathsf{sk}_O)$ for organizations respectively.

OBTAIN $\leftrightarrow$ ISSUE. The OBTAIN algorithm takes a user secret key sk_U and an organization public key pk_O. The ISSUE algorithm takes an organization secret key sk_O, a user public key pk_U and a list of attributes $\{m_i\}_{i=1}^{l}$. At the end of the protocol, user U has obtained a credential C on $\{m_i\}_{i=1}^{l}$, or an error message $\perp$.

SHOW $\leftrightarrow$ VERIFY. The SHOW algorithm takes a user secret key sk_U, a list of organization public keys together with a list of attributes[1] $\{(\mathsf{pk}_{O_n}, m_n)\}_{n=1}^{N}$ and a list of credentials $\{C_j\}_{j=1}^{J}$. The VERIFY algorithm takes the same list of public keys and messages $\{(\mathsf{pk}_{O_n}, m_n)\}_{n=1}^{N}$. At the end of the protocol, the VERIFY algorithm outputs a bit $b \in \{0, 1\}$.

First of all we require a system to be *correct*: any association of correctly generated credentials has to be accepted by a verifier if protocols are carried out honestly. More formally, we require that for all keys of a user $(\mathsf{pk}_U, \mathsf{sk}_U)$, and of some organizations $\{(\mathsf{pk}_{O_j}, \mathsf{sk}_{O_j})\}_{j=1}^{J}$, and for the credentials $\{C_j\}_{j=1}^{J}$ on attributes $\{\{m_{j,i}\}_{i=1}^{l_j}\}_{j=1}^{J}$ output by the OBTAIN/ISSUE protocol with this user and the different organizations, the output of the verification protocol is 1 with SHOW$(\mathsf{sk}_U, \{(\mathsf{pk}_{O_n}, m_n)\}_{n=1}^{N}, \{C_j\}_{j=1}^{J})$ and VERIFY$(\{(\mathsf{pk}_{O_n}, m_n)\}_{n=1}^{N})$, for each combination where, for all n, there exists j, i s.t. $m_n = m_{j,i}$ and $\mathsf{pk}_n = \mathsf{pk}_{O_j}$.

Then, as usual for anonymous credential systems, we require that a system satisfies the *unforgeability* and the *anonymity* properties. Before stating these properties formally as experiments, we introduce some oracles and variables.

2.2 Oracles and Variables

During the below experiments, the adversary may interact with some oracles and manipulate some global variables (all initialized with empty). The set of honest users (resp. organizations) is denoted HU (resp. HO), while the set of corrupted users (resp. organizations) is denoted CU (resp. CO). The table **upk** (resp. **opk**) contains the users' (resp organizations') public keys, while **usk** (resp. **osk**) contains the related users' (resp organizations') secret keys. Finally, the set **cred** records the credentials obtained by all the users. For example, the entry $\mathbf{cred}[i][j]$ contains the set of all the credentials belonging to user i and issued by organization j. Similarly, the entry $\mathbf{att}[i][j]$ contains the attributes of user i that are certified by the organization j. The oracles are next given in Figure 1.

We are now ready to define the unforgeability and anonymity properties. We denote by $\mathcal{O}$ the set of oracles that are available to an adversary.

2.3 Unforgeability

Regarding the unforgeability, the aim of the adversary is to prove that it is in possession of some credentials delivered by some honest organizations while this is not the case. In particular, the adversary can (i) interact with organizations in order to obtain credentials (using the SNDTOI oracle), and (ii) ask honest users to obtain credentials (using the GETCRED oracle). The adversary is accepted if it outputs an attribute which has never been certified or if it is able to combine several honestly obtained credentials while this is not allowed. More precisely, the unforgeability experiment is defined below. Informally speaking, the condition (5b) says that no user (honest or not) has received all the claimed credentials. We say that an anonymous credential system $\mathcal{AC}$ is *unforgeable* if for any adversary $(\mathcal{A}_1, \mathcal{A}_2)$, the probability that the UNFORGEABILITY$_{\mathcal{A}}^{\mathcal{AC}}$ outputs 1 is negligible (as a function of λ).

[1]Using this notation, we remark that an organization can be given several times. As a consequence, N can be greater than the number of existing organizations.

AddU(i). If it does not already exists, it adds i to the set HU of honest users, executes UserKG and updates accordingly the i-th entry of **upk** and **usk**. It finally returns **upk**$[i]$.

AddO(j). It works similarly to the AddU oracle, but for organizations, using OrgKG, updating **opk** and **osk** and outputting **opk**$[j]$.

USK(i) (resp. OSK(j)). The oracle simply returns (**usk**$[i]$, **cred**$[i]$) (resp. **osk**$[j]$) (participants are not corrupted, but keys leak).

CrptU(i, pk_U) (resp. CrptO(j, pk_O)). It adds i (resp. j) to the set CU (resp. CO) of corrupted users (resp. organizations), sets **upk**$[i] \leftarrow pk_U$ (resp. **opk**$[j] \leftarrow pk_O$) and outputs 1 if no error occurs.

GetCred($i, j, (m_1, \ldots, m_l)$). This oracle permits the honest user i to obtain certified credentials on the attributes $(m_1, \ldots, m_l)$ from the honest organization j. For this purpose, the oracle plays the role of both the user and the organization, using the appropriate keys. The attributes (and credentials) are next added to **att**$[i][j]$ and **cred**$[i][j]$ and the external communications are outputted.

SndToU(i, j, m) (resp. SndToI(i, j, m)). This oracle plays the role of the user i (resp. organization j) which receives the message m from the corrupted organization j (resp. user i). This may modify the entries **att**$[i][j]$ and **cred**$[i][j]$.

Ch($b, i_0, i_1, \{(n, m_n)\}_{n=1}^N$). This oracle is related to the anonymity experiment and depends on the bit b set by this experiment. It takes as input a pair i_0 and i_1 of honest users and a set $\{(n, m_n)\}_{n=1}^N$ of N attributes (chosen by the adversary) certified by different (corrupted) organizations to both i_0 and i_1. It plays the role of the user i_b with the set of credentials $\{(n, m_n)\}_{n=1}^N$ with the adversary, playing the role of the verifier, during a verification protocol. It outputs the external view of the whole protocol.

Figure 1: Oracles

Unforgeability$_{\mathcal{A}}^{\mathcal{AC}}(\lambda)$

1. $\mathsf{param} \leftarrow \text{Setup}(1^\lambda)$;
2. $\mathcal{O} \leftarrow \{\text{AddU}(\cdot), \text{AddO}(\cdot), \text{USK}(\cdot), \text{OSK}(\cdot), \text{CrptU}(\cdot), \text{CrptO}(\cdot), \text{GetCred}(\cdot,\cdot,\cdot)\}$, SndToI$(\cdot,\cdot,\cdot)$;
3. $(N, \{(n, m_n)\}_{n=1}^N, st) \leftarrow \mathcal{A}_1{}^{\mathcal{O}}(\mathsf{param})$;
4. A Show/Verify protocol is executed between the adversary $\mathcal{A}_2$, taking on input st and playing the role of the user, and the challenger, playing the role of the verifier and taking on input $\{(\mathsf{pk}_{O_n}, m_n)\}_{n=1}^N$;
5. Return 1 if
   (a) the Verify procedure outputs 1 and
   (b) for all $i^* \in CU \cup HU$, there exists $\mathsf{n}^* \in [1, N] \cap HO$ such that $m_{\mathsf{n}^*} \notin \mathbf{att}[i^*][\mathsf{n}^*]$.

2.4 Anonymity

Regarding the anonymity, the aim of the adversary is to distinguish between two chosen honest users which one is showing her credentials. For this purpose, the adversary has access to the Ch oracle given in Figure 1. Moreover it can interact with honest users to issue them some credentials with the help of the SndToU oracle. The formal experiment and the definition are as follows.

Anonymity$_{\mathcal{A}}^{\mathcal{AC}}(\lambda)$

1. $\mathsf{param} \leftarrow \text{Setup}(1^\lambda)$; $b \xleftarrow{\$} \{0, 1\}$;
2. $\mathcal{O} \leftarrow \{\text{AddU}(\cdot), \text{AddO}(\cdot), \text{USK}(\cdot), \text{OSK}(\cdot), \text{CrptU}(\cdot), \text{CrptO}(\cdot), \text{SndToU}(\cdot,\cdot), \text{Ch}(b,\cdot,\cdot,\cdot)\}$;
3. $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{param})$;
4. Return 1 if $b = b'$.

We say that an anonymous credential system $\mathcal{AC}$ is *anonymous* if for any adversary $\mathcal{A}$, the probability that the experiment Anonymity$_{\mathcal{A}}^{\mathcal{AC}}$ outputs 1 is negligibly close to 1/2 (as a function of λ).

3. INDEXED AGGREGATE SIGNATURES

An aggregate signature scheme [6, 24] is a classical signature scheme augmented with a specific algorithm which allows anyone to aggregate, into a single signature, a list of signatures valid under possibly different public keys and messages. In this section, we introduce the notion of *indexed aggregate signatures*, which will be useful in our anonymous credential system and which may be of independent interest.

3.1 Traditional Aggregate Signatures and Introduction of the Index

We first recall the traditional notion of aggregate signature scheme such as described in *e.g.* [6, 24]. In a nutshell, such a scheme consists in five procedures: Setup, KeyGen, Sign, Aggregate and Verify.

The Setup and KeyGen procedures permit to generate the different parameters and keys for the signer, while the Sign algorithm corresponds to the signature procedure which on input a message and a secret key, output a signature on the message.

Next, on input a set of N triplets of the form $(\mathsf{pk}_{S_n}, \mathsf{m}_n, \sigma_n)$, the Aggregate algorithm outputs a signature σ on the list of messages $(\mathsf{m}_1, \cdots, \mathsf{m}_N)$, under the public keys $(\mathsf{pk}_{S_1}, \cdots, \mathsf{pk}_{S_N})$ (with possibly several time the same key). Finally, the Verify algorithm takes as input an integer N (possibly equal to 1), a set of messages $(\mathsf{m}_1, \cdots, \mathsf{m}_N)$, a set of public keys $(\mathsf{pk}_{S_1}, \cdots, \mathsf{pk}_{S_N})$, and one signature σ and test whether the signature is valid or not.

In our anonymous credential system, we should keep the control on the message signature pairs which could be aggregated together. For this purpose, we add one specific value, called the *message index*, and we argue that the Aggregate procedure can yield valid signatures only if all the input message signature pairs have the same message index. This gives us the notion of *indexed aggregate signature* which we formally define in the following.

3.2 Related Concepts

Regarding the literature, there are only few aggregate signature schemes and its seems to be a concept which is hard to build. One consequence is that several alternate notions can however be found. In [24], the authors address the notion of *sequential* aggregation. An aggregate signature under different signing keys has to pass from all the signers to be produced. We argue that the use of such schemes here will lose the benefits intended from aggregate signatures, since in our need, the aggregation has to be done on the fly by the user in each verification, and without the help of the signers.

More recently, *synchronized* aggregate signatures have been introduced in [1], based on the work of [22]. In this case, the aggregation is possible only if the signers are synchronized on a specific value when they produce their individual signature. Such a policy of synchronization avoids the need for interactions between signers. We are not far from our index notion. In fact, the main difference is that a constant number of aggregations on the same synchronized value is allowed in the synchronized model, and the security is broken if the number of aggregations exceeds this bound. In our case, we don't impose such a bound since it is undesirable for designing an anonymous credential system.

3.3 Formal Definitions

An *indexed aggregate signature scheme* is given by the following five algorithms.

SETUP. On input a security parameter λ, this algorithm outputs the parameters of the system, which are given to all algorithms as auxiliary input. In particular, a message space $\mathcal{M}$ and an indices space $\mathcal{I}$ are specified.

KEYGEN. This algorithm, taking on input param and k, outputs the key pair $(\mathsf{sk}_S, \mathsf{pk}_S)$ for a signer.

SIGN. This algorithm takes as input a message index $\mathsf{i} \in \mathcal{I}$, a message $m \in \mathcal{M}$ and a secret key sk_S. It outputs a signature σ on m and i, under the public key pk_S.

AGGREGATE. This algorithm takes as input a message index $\mathsf{i} \in \mathcal{I}$ and a set of N triplets $(\mathsf{pk}_{S_n}, m_n, \sigma_n)$. It outputs either an error message $\perp$, or a signature σ on i and $(m_1, \cdots, m_N)$ under $(\mathsf{pk}_{S_1}, \cdots, \mathsf{pk}_{S_N})$ (with possibly several time the same key).

VERIFY. This algorithm takes as input a message index i, a set of public keys/messages $\{(\mathsf{pk}_{S_n}, m_n)\}_{n=1}^N$ and a signature σ. It outputs $b = 1$ if the signature σ is valid and $b = 0$ otherwise.

3.3.1 Correctness

We require such a scheme to be correct, i.e. a signature correctly computed by the signature procedure or the aggregation one is necessarily accepted by the verification algorithm. More formally, for a given λ, $\mathsf{param} \leftarrow \text{SETUP}(1^\lambda)$, $N \geq 1$ and $\mathsf{i} \in \mathcal{I}$, let $\{(\mathsf{pk}_{S_n}, \mathsf{sk}_{S_n})\}_{n=1}^N$ be a set of key pairs, $\{m_n\}_{n=1}^N \in \mathcal{M}^N$ be a set of messages, and $\sigma_n := \text{SIGN}(\mathsf{i}, m_n, \mathsf{sk}_{S_n})$ be individual signatures for $n \in [1, N]$. An indexed aggregate signature is said *correct* if the verification $\text{VERIFY}(\mathsf{i}, \{(\mathsf{pk}_{S_n}, m_n)\}_{n=1}^N, \sigma) = 1$ holds for each σ such that $\sigma := \text{AGGREGATE}(\mathsf{i}, \{(\mathsf{pk}_{S_n}, m_n, \sigma_n)\}_{n=1}^N)$.

Next, a *secure* indexed aggregate signature scheme should verify the *unforgeability property* which can be described as follows, based on the initial work in [6] for standard aggregate signature schemes.

3.3.2 Unforgeability

The aim of an adversary is to forge an aggregate signature. It thus outputs a valid signature on an index and a list of public keys and messages. This is considered as a forge if one message has never been signed with the specified index. A challenge public key is given to the adversary, and the latter may have access to a signing oracle denoted SIGN. Each time this oracle is asked on an index i and a message m, we update the global variable $\mathbf{M}$ by adding m to $\mathbf{M}[\mathsf{i}]$. More precisely, the unforgeability experiment is given as follows.

$\text{UNFORGEABILITY}_{\mathcal{A}}^{\mathcal{AG}}(\lambda)$

1. $\mathsf{param} \leftarrow \text{SETUP}(1^\lambda)$; $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \text{KEYGEN}()$; $\mathbf{M} \leftarrow \varepsilon$
2. $(\mathsf{i}, \{(\mathsf{pk}_{S_n}, m_n)\}_{n=1}^N, \sigma) \leftarrow \mathcal{A}^{\text{SIGN}(\mathsf{sk}_S, \cdot, \cdot)}(\mathsf{param}, \mathsf{pk}_S)$
3. Return 1 if
   (a) $\text{VERIFY}(\mathsf{i}, \{(\mathsf{pk}_{S_n}, m_n)\}_{n=1}^N, \sigma) = 1$, and
   (b) $\exists \mathsf{n} \in [1, N]$ such that $\mathsf{pk}_{S_\mathsf{n}} = \mathsf{pk}_S$ and $m_\mathsf{n} \notin \mathbf{M}[\mathsf{i}]$

We say that an indexed aggregate signature scheme $\mathcal{AG}$ is $(t, N_{\max}, q_S, \varepsilon)$-*unforgeable* if the probability for an adversary $\mathcal{A}$ to win the $\text{UNFORGEABILITY}_{\mathcal{A}}^{\mathcal{AG}}$ game within time t, with no more than $N_{\max}$ individual signatures aggregated in the forge, after q_S queries to the sign oracle is less than ε.

Remark 1 *In the following, we consider that we are in the* chosen-key model. *It means that an adversary has not to register the different keys that are involved in a forgery. In the* registered-key model*, such a registration is necessary.*

3.4 Intuition for Our Construction

Before giving our formal construction, let us take a look at some difficulties to construct such a scheme and explain how we get out of it.

Let $\mathcal{A}$ be an adversary against the above unforgeability game which outputs a valid and non trivial aggregate signature $(\mathsf{i}, \{(\mathsf{pk}_{S_n}, m_n)\}_{n=1}^N, \sigma)$. We distinguish several cases.

The first case happens when the index i has already been seen, but when a new message appears in the list of the forgery associated with the challenge public key. In this case, we are able to exhibit an individual forgery against the aggregate signature scheme upon which our construction is built, namely the BGLS scheme [6] (augmented with the trick from [4]). As a consequence, a signature on a message m in our scheme is close to the BGLS form: $\mathcal{H}(g_1{}^\gamma \| g_2{}^\gamma \| m)^\gamma$, where γ is the signing secret key. It follows that the aggregation is simply the multiplication of the signatures.

The second case happens when the index and all messages in the forgery have already been seen. It may happen, for example, when corrupted users collude to aggregate some differently obtained signatures. To handle this case, the signature process corresponds to a signature on both the message m and the index i, using the BGLS signature scheme as said above. The security of our scheme regarding such an attack is next ensured by the fact that it is not possible to disaggregate aggregated elements. Our signature is then of the form $\sigma := \left(\mathsf{i} \cdot \mathcal{H}(g_1{}^\gamma \| g_2{}^\gamma \| m)\right)^\gamma$.

The third and last case happens when $\mathcal{A}$ outputs a new index i, i.e. an index never involved in adversary's queries. Regarding the above form of our signature, it seems hard to prevent such attack since, for example, the multiplication of two differently obtained signature with indices i and i' introduces the index $\mathsf{i} \times \mathsf{i}'$ which is new and correct regarding the signature verification (admitting that $\mathsf{i} \times \mathsf{i}'$ is an acceptable index). To handle this case, we introduce a control over the indices. We see them as a signature from some general authority. It follows that the exhibition of a new index is equivalent to a forgery against the underlying signature scheme. In the following, we choose a Boneh-Boyen-like signature [5] which one corresponds to a couple

$\mathsf{i} = (A, s)$ such that $A = (u \cdot {g_1}^x)^{\frac{1}{\delta+s}}$, where x is some user secret and δ the signing secret key. Our signature is finally $\sigma := \big(A \cdot \mathcal{H}({g_1}^\gamma \| {g_2}^\gamma \| m)\big)^\gamma$.

3.5 Formal Description

We now give a concrete indexed aggregate signature scheme, denoted by **Agg**. As said above, our proposal is based on the initial work from [6] for standard aggregate signature schemes, with Boneh-Boyen-like certificates [5] as indices.

Agg.SETUP(1^λ). It first generates a bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ such that p is a prime number of length λ, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p and e is a bilinear pairing. It also chooses a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$, and generators $g_1, u, f \xleftarrow{\$} \mathbb{G}_1$ and $g_2 \xleftarrow{\$} \mathbb{G}_2$. A scalar $\delta \xleftarrow{\$} \mathbb{Z}_p$ is picked and $\Delta := ({g_1}^\delta, {g_2}^\delta)$ is computed. The message space is $\mathcal{M} \in \{0,1\}^*$. The indices space is defined as: $\mathcal{I} := \big\{((X,Y),(A,B,D)) \in (\mathbb{G}_1 \times \mathbb{G}_2) \times ({\mathbb{G}_1}^2 \times \mathbb{G}_2) \mid A = (u \cdot {g_1}^x)^{\frac{1}{\delta+s}} \wedge B = f^s \wedge D = {g_2}^s \wedge X = {g_1}^x \wedge Y = {g_2}^x\big\}$. In the following, such an index is said valid if $e(A, D \cdot \Delta_2) = e(u \cdot X, g_2)$, $e(B, g_2) = e(f, D)$ and $e(X, g_2) = e(g_1, Y)$.

Such an index can not be directly computed (see above in Section 3.4 why we need such kind of index) and we need to keep δ secret. Consequently, the SETUP additionally needs to publish a set of valid indices. Let $\mathcal{X} := \big\{s \in \mathbb{Z}_p \setminus \{-\delta\}\big\}$ be a polynomial[2] set of values in $\mathbb{Z}_p \setminus \{-\delta\}$ and let $\mathcal{BB} := \big\{((X,Y),(A,B,D)) \in \mathcal{I} \mid s \in \mathcal{X}\big\}$ be a set of possible indices. Each signer can in the following take one index in this set. Finally the procedure outputs $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \Delta, \mathcal{H}, \mathcal{BB})$.

Agg.KEYGEN(). This algorithm first chooses the secret $\gamma \xleftarrow{\$} \mathbb{Z}_p$ and computes $\Gamma_1 \leftarrow {g_1}^\gamma$ and $\Gamma_2 \leftarrow {g_2}^\gamma$. It returns $\mathsf{pk}_S := \Gamma = (\Gamma_1, \Gamma_2)$, $\mathsf{sk}_S := \gamma$.

Agg.SIGN(γ, i, m). After checking that i is valid, this procedure produces a signature on m and i by computing $\sigma := \big(A \cdot \mathcal{H}({g_1}^\gamma \| {g_2}^\gamma \| m)\big)^\gamma \in \mathbb{G}_1$

Agg.AGGREGATE($\{\sigma_n\}_{n=1}^N$). The aggregation procedure returns $\sigma_a := \prod_{n=1}^N \sigma_n$

Agg.VERIFY($\mathsf{i}, \{(\Gamma_n, m_n)\}_{n=1}^N, \sigma$). This procedure returns 1 if $\mathsf{i} := ((X,Y),(A,B,D))$ is valid and if

$$e(\sigma, g_2) = e(A, \prod_{n=1}^N \Gamma_{n,2}) \cdot \prod_{n=1}^N e(\mathcal{H}(\Gamma_{n,1} \| \Gamma_{n,2} \| m_n), \Gamma_{n,2})$$

This construction is correct since one can easily check that $e(\sigma, g_2) = e(\prod_{n=1}^N [A \cdot \mathcal{H}(\cdots \| m_n)]^{\gamma_n}, g_2) = e(A, \prod_{n=1}^N \Gamma_{n,2}) \cdot \prod_{n=1}^N e(\mathcal{H}(\cdots \| m_n), \Gamma_{n,2})$. We also have the following theorem regarding unforgeability, which is proven in Appendix B under assumptions that are stated in Appendix A.

Theorem 1 *The **Agg** scheme is an unforgeable indexed aggregate signature scheme under the q-ADHSDH assumption in the chosen-key model for aggregate signature if $\mathcal{H}$ is seen as a random oracle.*

[2]i.e. there exists a polynomial q such that $|\mathcal{X}| = q(\lambda)$.

4. ANONYMOUS CREDENTIALS FROM AGGREGATE SIGNATURES

We give now our main construction of anonymous credential schemes based on indexed aggregate signatures.

4.1 Zero-Knowledge Proofs

We need in our construction several zero-knowledge proofs of knowledge (ZKPK) [19]. Roughly speaking, such a tool describes the way an entity proves to another one that she knows a set of secret values $\alpha_1, \ldots, \alpha_q$ verifying a given relation $\mathcal{R}$ without revealing any information about her secrets. Such proofs are considered secure if they are *complete* (one knowing the secret is always accepted), (honest-verifier) *zero-knowledge* (the transcripts of the proof can be simulated) and *sound* (there exists an extractor of the secrets, which implies that a prover will be accepted only if she knows the claimed secrets). In the following, such a ZKPK is denoted by $\text{POK}\{\alpha_1, \ldots, \alpha_q : \mathcal{R}(\alpha_1, \ldots, \alpha_q) = 1\}$.

More precisely, we need in the following discrete logarithm based ZKPK [12, 8] which can be seen as any combination of discrete logarithm proofs [28], representation proofs [27] and equality of known discrete logarithm proofs [15].

4.2 Overview of Our Solution

We now informally show how to use an indexed aggregate signature scheme **Agg** to design a secure anonymous credential system. A more formal view is given Figure 2

.

ORGKEYGEN.

1. $(\mathsf{pk}_O, \mathsf{sk}_O) \leftarrow$ **Agg**.KEYGEN()

USERKEYGEN.

1. Any sk_U such that there exists a function UTOI that maps user secrets to indices.

OBTAIN/ISSUE.

1. User computes $\mathsf{i}_U :=$ UTOI(sk_U) and $\pi :=$ POK$\{\langle\alpha\rangle : \mathsf{i}_U =$ UTOI$(\alpha)\}$ and sends i_U, π to the organization.
2. Organization checks π and computes and individual signature $\sigma_n :=$ **Agg**.SIGN($\mathsf{sk}_O, \mathsf{i}_U, m_n$) for each $n \in [1, L]$ where the m_n's correspond to some attributes related to the user. The σ_n's are sent to the user.
3. The certificate is finally $\Sigma := \{\sigma_n\}_{n=1}^L$.

SHOW/VERIFY.

1. User first aggregates the different certificates she needs, as $\sigma \leftarrow$ **Agg**.AGGREGATE($\{\sigma_n\}_{n=1}^N$).
2. User next produces a zero-knowledge proof of knowledge

   POK$\{\langle\sigma, x\rangle :$ **Agg**.VERIFY$\big(\mathsf{i}_U, \{(\mathsf{pk}_{O_n}, m_n)\}_{n=1}^N, \sigma\big) = 1\}$

Figure 2: Generic construction

Organizations are signers and a credential on attributes are signatures. Let us consider a user having obtained several credentials on some attributes, may be from different organizations. Let us assume that this user interacts with a verifier asking for several attributes. Using the aggregation procedure on her credentials and related to the requested attributes, this user is able to output a single signature which can be used to answer the verifier. Three issues arise:

1. users should not be able to share their credentials: so we introduce *indexed* aggregate signature schemes ;

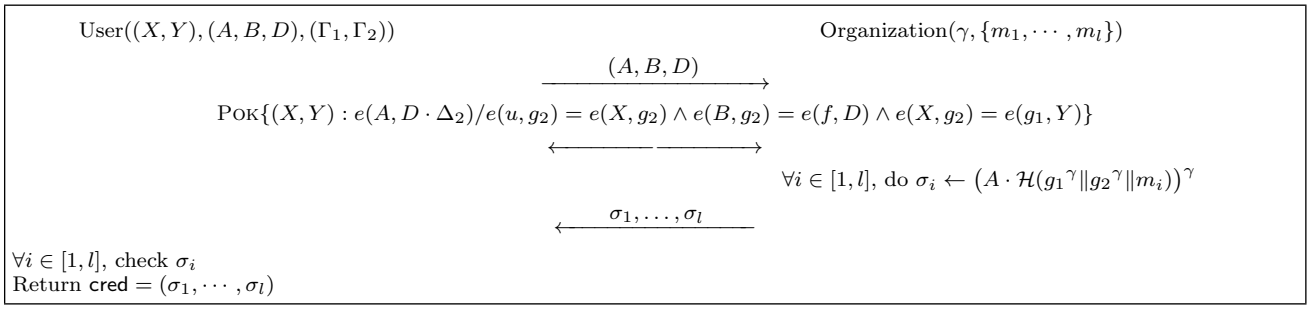

Figure 3: The issuance protocol

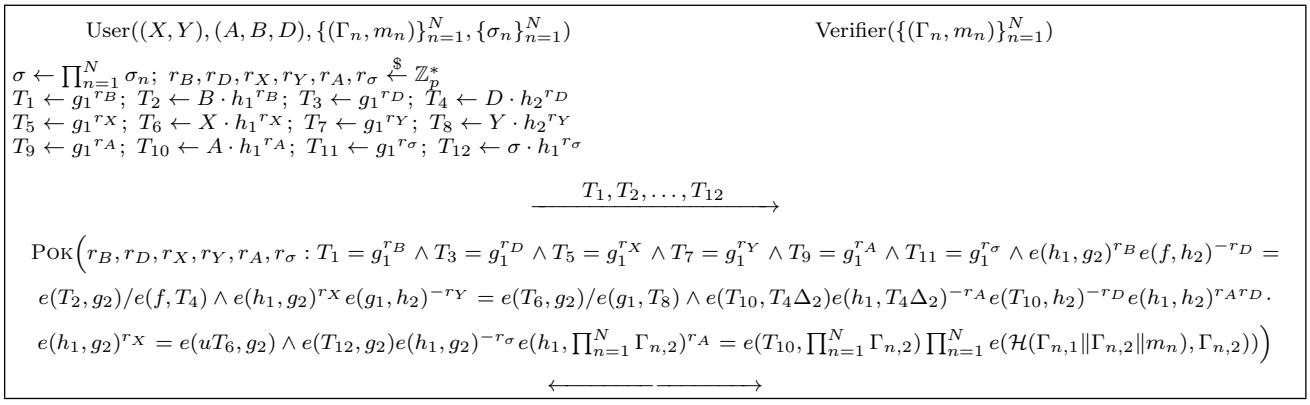

Figure 4: The showing protocol

2. the aggregate signature should not be disclose to the verifier since it compromises the anonymity of the user. Our solution is to produce a ZKPK (see above) of this signature (see Remark 2 below) ;

3. as the organization should not be able to use the certified credentials obtained by an honest user, the index message should correspond to a secret only known by the user. This way, different obtained credentials can only be aggregated if they are related to this secret, and thus to the same user.

Remark 2 *The reader should note that the proof of knowledge is constant-size since a single aggregate signature is proved to be known, independently of the number of involved credentials before the aggregation. This brings non-negligible advantages in bandwidth, specially if credentials are embedded in a smart-card. We argue that this solution is well-suited for embedded processors since a transmitted bit costs more than a computed bit into such environments.*

4.3 Our Anonymous Credential System

We now describe a concrete instantiation $\mathcal{AC}$ of our new anonymous credential system, based on the indexed aggregate signature scheme we introduced in Section 3.5.

Setup(1^λ). As in Section 3.5, generate a bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ and next uniformly at random $g_1, h_1, u, f \xleftarrow{\$} \mathbb{G}_1$ and $g_2, h_2 \xleftarrow{\$} \mathbb{G}_2$. Finally, let $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ be a cryptographically secure hash function. Let $\Delta := ({g_1}^\delta, {g_2}^\delta)$ for $\delta \xleftarrow{\$} \mathbb{Z}_p$ be a general registration key. The parameters of the system are $\mathsf{param} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, h_2, u, \Delta, \mathcal{H})$.

OrgKeyGen(). During this step, an organization chooses uniformly at random $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $(\Gamma_1, \Gamma_2) := ({g_1}^\gamma, {g_2}^\gamma)$. The organization public key is next $\mathsf{pk}_O := (\Gamma_1, \Gamma_2)$ and the corresponding secret key is $\mathsf{sk}_O := \gamma$.

UserKeyGen(). User secret keys are $\mathsf{sk}_U := (X, Y)$ where $X \leftarrow {g_1}^x$ and $Y \leftarrow {g_2}^x$ for an uniformly picked $x \xleftarrow{\$} \mathbb{Z}_p^*$. The mapping from user keys to indices is done by UtoI : $x \mapsto (u \cdot X)^{\frac{1}{\delta+s}}$. This mapping is done once during some kind of registration step, where the user interacts with some certification authority knowing δ. Such a "certificate" is valid if $e(B, g_2) = e(f, D)$, $e(X, g_2) = e(g_1, Y)$ and $e(A, D \cdot \Delta_2) = e(u \cdot X, g_2)$. The set of all "certificates" of all users corresponds to the set $\mathcal{BB}$ of the indexed aggregate signature scheme.

Obtain ↔ Issue. This protocol is given in Figure 3 and allows a user with index (X, Y), (A, B, D) to obtain a credential $\mathsf{cred} = \{\sigma_i\}_{i=1}^l$ on attributes $\{\mathsf{m}_i\}_{i=1}^l$, issued by the organization with the public key (Γ_1, Γ_2).

Show ↔ Verify. This protocol is described in Figure 4 and allows a user, on input several signatures $\{\sigma_n\}_{n=1}^N$ on attributes $\{\mathsf{m}_n\}_{n=1}^N$ under public keys $(\Gamma_{n,1}, \Gamma_{n,2})$, to prove the possession of these certificates to a verifier. This is done by first aggregating all the signatures and

	Keys for L attributes per credential		**Issuing Protocol**		
	Parameters	Credential + User Secret	User	Issuer	Bandwidth
UProve	$O(L) : \{(g_i, e_i, z_i)\}_{i=1}^{L}$	$O(\mathbf{1})$	$O(L) : G_q$	$O(L) : G_q$	O(1)
Idemix	$O(L) : \{R_i\}_{i=1}^{L}$	$O(\mathbf{1})$	$O(L) : \mathbb{Z}_n$	$O(L) : \mathbb{Z}_n$	$O(L-k)$
Ours	$O(\mathbf{1})$	$O(L)$	$O(L) : \mathbb{G}_2$	$O(L) : \mathbb{G}_1$	$O(L)$
	Showing Protocol				
	User | Verifier	Bandwidth	User	Verifier	Bandwidth
	One-Show 1 *Organization*		*One-Show* *N Organizations*		
UProve	$O(1)$	$O(1)$	$O(N) : G_q$		$O(N)$
Idemix	$O(L)$ mult. $\mathbb{Z}_n$	$O(1) : \mathbb{Z}$	$O(NL)$ mult. $\mathbb{Z}_n$		$O(N) : \mathbb{Z}$
Ours	$O(1)$ pair.; $O(L) : \lvert\mathbb{G}_2\rvert$	$O(\mathbf{1})$	$O(NL)$ pairings		$O(\mathbf{1})$
	One-Show 1 *Organization* *n-out-of-L hidden att.*		*K-Multi-Show* 1 *Organization*		
UProve	$O(n) : G_q$	$O(n) : \mathbb{Z}_q$	$O(K) : G_q$		$O(K)$
Idemix	$O(L)$ mult. $\mathbb{Z}_n$	$O(n) : \mathbb{Z}$	$O(L)$ mult. $\mathbb{Z}_n$		$O(L) : \mathbb{Z}$
Ours	$O(1)$ pairings; $O(L) : \lvert\mathbb{G}_2\rvert$	$O(\mathbf{1})$	$O(1)$ pairings; $O(L) : \lvert\mathbb{G}_2\rvert$		$O(\mathbf{1})$

Table 1: Comparison with the state of the art

next to prove the knowledge of the resulting signature, with index $((X, Y), (A, B, D))$.

The showing protocol is a zero-knowledge proof of knowledge of values $(X, Y), (A, B, D)$ and $\{\sigma_n\}_n$ such that the aggregation of the latter is a valid credential for the former. Group elements are committed in the T_is and the protocol is a standard Schnorr-like proof.

4.4 Security Arguments

The security of our scheme mainly relies on the security of the chosen indexed aggregate signature scheme (see Section 3) and on the security of the ZKPK (see Section 4.1). More formally, we have the following theorem. Assumptions are given in Appendix A and proofs in Appendix C.

Theorem 2 *The $\mathcal{AC}$ anonymous credential system is unforgeable under the q-ADHSDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ and anonymous under the DDH assumption in $\mathbb{G}_1$ if $\mathcal{H}$ is seen as a random oracle.*

4.5 Efficiency Comparison

We compare our new proposal with the related work on anonymous credentials, namely the Idemix [23] and the UProve [26] systems. The whole comparison is given in Table 1. We fix L as the number of attributes issued by an authority into a credential and analyse the computational, memory and bandwidth costs for the keys and protocols.

In our solution, an individual signature is issued for each attribute by the corresponding organization, whereas an Idemix certificate is constant-length per organization but for possibly several attributes. However, our signatures are only composed of one group element whereas an Idemix signature is much more larger. The best solution, in terms of user storage, is thus still the UProve one.

Next, this flexibility on prover side allows us to reach a constant bandwidth, which is not the case in the Idemix and UProve systems, specially if some attributes are hidden. We argued that bandwidth is a more precious resource than storage in application we have in mind. More precisely, when the user has to prove that she owns several attributes coming from different organization, the space complexity of our solution is constant. Regarding the time complexity, the prover only has one multiplication of L different signatures (for the aggregation process). The other multiplications depending on L are the ones related to the multiplication of the organization public keys, which can be easily pre-computed and/or published. On the other side, in such case, the Idemix and UProve systems necessitates to produce L different group signatures or show L different blind signatures respectively.

The next step is certainly to make a true implementation of our system to test it in concrete situations and make a complete comparison.

5. CONCLUSION

We proposed a new way to treat anonymous credentials by using aggregate signatures and we gave a concrete instantiation of this idea. More particularly, we address the problem of dealing with several credentials certified by several different organizations, which case as not been taken into account by previously known constructions.

We leave as a tricky open problem the construction of a scheme in the standard model, which appears to be difficult. Another open problem is to enable statements on hidden attributes. With our new system, we can hide an attribute by using some kind of disaggregation, but we can not prove any statement about a hidden value since we further apply a hash function on it.

Finally, we note that our scheme is valid only if the size of the actual indices space is sub-exponential in the size of the whole space of the possible values, as in the construction from [1]. We leave as an interesting open problem to show that there is a reason beneath.

Acknowledgements

This work has been partially supported by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II. We also thanks the anonymous reviewers for their useful comments.

6. REFERENCES

[1] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In *ACM CCS*, pages 473–484, 2010.

[2] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO*, pages 108–125, 2009.

[3] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC*, pages 356–374, 2008.
[4] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In *ICALP*, pages 411–422, 2007.
[5] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.
[6] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.
[7] Stefan Brands. *Rethinking PKI and digital certificates - building in privacy*. PhD thesis, Eindhoven Institute of Technology, 1999.
[8] Jan Camenisch, Aggelos Kiayias, and Moti Yung. On the portability of generalized schnorr proofs. In *EUROCRYPT*, pages 425–442, 2009.
[9] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC*, pages 481–500, 2009.
[10] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.
[11] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, pages 56–72, 2004.
[12] Sébastien Canard, Iwen Coisel, and Jacques Traoré. Complex zero-knowledge proofs of knowledge are easy to use. In *ProvSec*, pages 122–137, 2007.
[13] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
[14] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO*, pages 118–167, 1986.
[15] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.
[16] Jean-Sébastien Coron. Optimal security proofs for pss and other signature schemes. In *EUROCRYPT*, pages 272–287, 2002.
[17] Jean-Sébastien Coron and David Naccache. Boneh et al.'s k-element aggregate extraction assumption is equivalent to the diffie-hellman assumption. In *ASIACRYPT*, pages 392–397, 2003.
[18] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *CRYPTO*, pages 328–335, 1988.
[19] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.
[20] Georg Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009.
[21] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In *EUROCRYPT*, pages 224–245, 2011.
[22] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *PKC*, pages 257–273, 2006.
[23] IBM. Identity mixer - Idemix, 2010.
[24] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, pages 465–485, 2006.
[25] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *SAC*, pages 184–199, 1999.
[26] Microsoft. U-Prove community technology, 2010.
[27] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.
[28] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.
[29] Eric R. Verheul. Self-blindable credential certificates from the weil pairing. In *ASIACRYPT*, pages 533–551, 2001.

APPENDIX

A. ASSUMPTIONS IN BILINEAR GROUPS

[q-ADHSDH] Asymmetric Double Hidden Strong Diffie-Hellman

Given $(g_1, \Delta_1 = {g_1}^{\delta}, h, f, g_2, \Delta_2 = {g_2}^{\delta}) \in {\mathbb{G}_1}^4 \times {\mathbb{G}_2}^2$, tuples $\left\{(A_i = (h \cdot {g_1}^{x_i})^{\frac{1}{\delta + s_i}}, B_i = f^{s_i}, D_i = {g_2}^{s_i}, X_i = {g_1}^{x_i}, Y_i = {g_2}^{x_i})\right\}_{i=1}^{q-1}$ for $s_i, x_i \xleftarrow{\$} \mathbb{Z}_p$ find a new tuple (A, B, D, X, Y) such that $e(A, D \cdot \Delta_2) = e(h \cdot X, g_2)$, $e(B, g_2) = e(f, D)$ and $e(X, g_2) = e(g_1, Y)$.

This problem is an asymmetric variant of the DHSDH problem and has been proven generically secure in [20]. We refer the reader to [20] for more analysis.

[co-CDH] Bilinear Computational Diffie-Hellman

Given $(g_1, g_2, {g_1}^a, {g_2}^a, h)$, find h^a, where $g_1, h \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$.

[EAE] Element Aggregate Extraction

Given $(g_1, {g_1}^{\gamma}, {g_1}^a, {g_1}^b, g_2, {g_2}^{\gamma}, {g_2}^a, {g_2}^b, {g_1}^{\gamma(a+b)})$, find ${g_1}^{\gamma a}$ where $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $\gamma, a, b \in \mathbb{Z}_p$.

In [17], the authors show how a solver for the EAE problem can be use for computing a co-CDH challenge. The converse is straightforward. Therefore the EAE problem is equivalent to the co-CDH problem.

[DDH] Decisional Diffie-Hellman

Given (g, g^a, g^b, g^c), decide whether $g^c = g^{ab}$ or $g^c \xleftarrow{\$} G$.

B. UNFORGEABILITY OF OUR INDEXED AGGREGATE SIGNATURE SCHEME

We prove here Theorem 1 about the unforgeability property of our indexed aggregate signature scheme.

PROOF. Consider an adversary $\mathcal{A}$ that, after receiving parameters (g_1, g_2, u, Δ), a valid set $\mathcal{BB} := \{\mathsf{i} := ((X_1, X_2), (A, s))\}$ such that $|\mathcal{BB}| := q(\lambda)$ for some polynomial λ, and

a challenge public key Γ, is allowed to ask for q signatures σ_i on index/message pairs (i_i, m_i) of her choice and outputs a valid aggregate signature $(\mathsf{i}_*, \{(\Gamma_n, m_n)\}_{n=1}^{N_*}, \sigma_*)$ for an index $\mathsf{i}_* := ((X_{*,1}, X_{*,2}), (A_*, B_*, D_*))$. Let $M[\mathsf{i}]$ denote the messages issued on index i during the whole attack.

We distinguish three types of forgers. An adversary $\mathcal{A}$ is called of type I if $\mathsf{i}_* \notin \mathcal{BB}$ and of type II if $\mathsf{i}_* \in \mathcal{BB}$. $\mathcal{A}$ is called of type IIa if there exists $\mathsf{n} \in [1, N]$ such that $\Gamma_\mathsf{n} = \Gamma$ and, for all $\mathsf{i} \in \mathcal{BB}$, $m_\mathsf{n} \notin M[\mathsf{i}]$. $\mathcal{A}$ is called of type IIb if there exists $\mathsf{n} \in [1, N]$ and $\mathsf{i} \neq \mathsf{i}_*$ such that $\Gamma_\mathsf{n} = \Gamma$ and $m_\mathsf{n} \in M[\mathsf{i}]$. We will use the first type to break a q-ADHSDH challenge, the second to break a co-CDH challenge and the last to break an EAE challenge. Since, the q-ADHSDH assumption implies the others, theorem follows. Regarding the random oracle, a table $H[m]$ is maintained for each reduction to answer the same value for identical queries. Let ζ, $0 < \zeta < 1$, be a probability we will later optimize.

Type I. Let $(g_1, \Delta_1, h, f, g_2, \Delta_2, \{(A_i, B_i, D_i, X_i, Y_i)\}_{i=1}^{q-1})$ be a uniformly generated q-ADHSDH challenge. We set $\mathcal{BB} := \{(X_i, Y_i), (A_i, B_i, D_i)\}$, $u := h$ and $\Gamma := ({g_1}^\gamma, {g_2}^\gamma)$ for $\gamma \xleftarrow{\$} \mathbb{Z}_p$. We supply $(g_1, g_2, u, \Gamma, \Delta)$ and $\mathcal{BB}$ to adversary $\mathcal{A}$. A hash query for a message m is simulated as ${g_1}^{c_m}$ for $c_m \xleftarrow{\$} \mathbb{Z}_p$. The signing query on (i_i, m) is answered as $({A_{\mathsf{i}_i}}{g_1}^{c_m})^\gamma$. $\mathcal{A}$ eventually outputs $(\mathsf{i}_*, \{(\Gamma_n, m_n)\}_{n=1}^{N_*}, \sigma_*)$ where $\mathsf{i}_* \notin \mathcal{BB}$. i_* is returned as solution the initial problem.

Type IIa. Let $(g_1, g_2, {g_1}^a, {g_2}^a, h)$ be a uniformly generated bilinear CDH challenge. We pick $\delta, \alpha, \beta, s_1, x_1, \ldots, s_q$, $x_q \xleftarrow{\$} \mathbb{Z}_p$, set $u := {g_1}^\alpha, f := {g_1}^\beta, \Delta := ({g_1}^\delta, {g_2}^\delta)$. For $i \in [1, q]$, we set $\mathsf{i}_i := (({g_1}^{x_i}, {g_2}^{x_i}), ({g_1}^{\frac{\alpha + x_i}{\delta + s_i}}, s_i))$. Parameters $(g_1, g_2, u, f, \Gamma, \Delta, \mathcal{BB})$ are supplied to adversary $\mathcal{A}$ together with $\Gamma := ({g_1}^a, {g_2}^a)$ as challenge public key. For a hash query on m, if $m := \Gamma || m'$ for some m', we answer by ${h}{g_1}^{c_m}$ with probability ζ. Otherwise, we answer by ${g_1}^{c_m}$. For a sign query on (i, m), if $\mathcal{H}_{\text{SIM}}(m) = {g_1}^{c_m}$, we answer by $({g_1}^\gamma)^{\frac{\alpha + x_\mathsf{i}}{\delta + s_\mathsf{i}} + c_m}$. Otherwise we fail.

$\mathcal{A}$ eventually outputs $(\mathsf{i}_*, \{(\Gamma_n, m_n)\}_{n=1}^{N_*}, \sigma_*)$. Since the forgery is of type IIa, $\mathsf{i}_* \in \mathcal{BB}$ and there exists $\mathsf{n} \in [1, N]$ such that $\Gamma_\mathsf{n} = \Gamma$ and $m_\mathsf{n} \notin M[\mathsf{i}]$ for all $\mathsf{i} \in \mathcal{BB}$. We continue only if $\mathcal{H}_{\text{SIM}}(m_\mathsf{n}) = {h}{g_1}^{c_{m_\mathsf{n}}}$ and $\mathcal{H}_{\text{SIM}}(m_i) = {g_1}^{c_{m_i}}$ for all $i \in [1, N_*]$ such that $i \neq \mathsf{n}$ and $\Gamma_i = \Gamma$. In this case, we are able to compute

$$\left[\sigma_* \cdot \prod_{i=1}^{N_*} \left[{\Gamma_{i,1}}^{-\frac{\alpha + x_{\mathsf{i}_*}}{\delta + s_{\mathsf{i}_*}} - c_{m_i}}\right]\right]^{t^{-1} \mod p} = h^a$$

where $t := |\{i \mid (\Gamma_i, m_i) = (\Gamma, m_\mathsf{n})\}|$. Intuitively, since m_n has never been queried and hash responses are uniformly distributed, there is a non-negligible probability for the reduction not to fail. But we can't do this if m_n has already been queried with probability 1. So we must handle forgers of type IIb differently.

Type IIb. Let $(g_1, {g_1}^\gamma, {g_1}^{\gamma^2}, g_2, {g_2}^\gamma, {g_2}^a, {g_2}^b, {g_1}^{\gamma(a+b)})$ be a uniformly generated EAE challenge. We pick δ, α, β, $s_1, \ldots, s_q \xleftarrow{\$} \mathbb{Z}_p$, set $u := {g_1}^\alpha$, $f := {g_1}^\beta$, $\Delta := ({g_1}^\delta, {g_2}^\delta)$. We then uniformly choose $k \xleftarrow{\$} \{1, \ldots, q\}$ and $x_1, \ldots, x_{k-1}$, x'_k, $x_{k+1}, \ldots, x_q \xleftarrow{\$} \mathbb{Z}_p$. For $i \in [1, q]$, $i \neq k$, we set $\mathsf{i}_i := (({g_1}^{x_i}, {g_2}^{x_i}), ({g_1}^{\frac{\alpha + x_i}{\delta + s_i}}, s_i))$. For $i = k$ we set $\mathsf{i}_k := (({g_1}^b {g_1}^{x'_k}, {g_2}^b {g_2}^{x'_k}), (({g_1}^b {g_1}^{\alpha + x'_k})^{\frac{1}{\delta + s_i}}, s_i))$. One can check that i_k is a valid index : $e(X_{k,1}, g_2) = e(g_1, g_2)^{b + x'_k} = e(g_1, X_{k,2})$ but we don't know $x_k := b + x'_k$. Parameters $(g_1, g_2, u, f, \Gamma, \Delta, \mathcal{BB})$ are supplied to adversary $\mathcal{A}$. For a hash query on m, if $m := \Gamma || m'$ for some m', we answer by $({g_1}^a)^{\frac{1}{\delta + s_k}} {g_1}^{c_m}$ with probability ζ. Otherwise, we answer by ${g_1}^{c_m}$. For a sign query on (i, m), if $\mathcal{H}_{\text{SIM}}(m) = {g_1}^{c_m}$ and $\mathsf{i} \neq \mathsf{i}_k$, we answer by $({g_1}^\gamma)^{\frac{\alpha + x_\mathsf{i}}{\delta + s_\mathsf{i}} + c_m}$. If $\mathcal{H}_{\text{SIM}}(m) = ({g_1}^a)^{\frac{1}{\delta + s_k}} {g_1}^{c_m}$ and $\mathsf{i} = \mathsf{i}_k$, we answer by

$$({g_1}^\gamma)^{\frac{\alpha + x'_k}{\delta + s_k} + c_m} ({g_1}^{\gamma(a+b)})^{\frac{1}{\delta + s_k}} = \left(({u}{g_1}^{b + x'_k})^{\frac{1}{\delta + s_k}} \mathcal{H}_{\text{SIM}}(m)\right)^\gamma$$

We fail in the other cases.

$\mathcal{A}$ eventually outputs $(\mathsf{i}_*, \{(\Gamma_n, m_n)\}_{n=1}^{N_*}, \sigma_*)$. Since the forgery is of type IIb, there exists $\mathsf{n} \in [1, N]$ and $\mathsf{i} \neq \mathsf{i}_*$ such that $\Gamma_\mathsf{n} = \Gamma$ and $m_\mathsf{n} \in M[\mathsf{i}]$. We continue only if $\mathsf{i}_* = \mathsf{i}_k$ and $\mathcal{H}_{\text{SIM}}(m_i) = {g_1}^{c_{m_i}}$ for all $i \in [1, N_*]$. In this case, we are able to compute

$$\left[\sigma_* \cdot \prod_{i=1}^{N_*} \left[{\Gamma_{i,1}}^{-\frac{\alpha + x'_k}{\delta + s_k} - c_{m_i}}\right]\right]^{(\delta + s_k) t^{-1} \mod p} = {g_1}^{\gamma b}$$

It remains to show that both reductions of type II will not fail with non-negligible probability. Following Coron's analysis (cf. [16, 6]), an optimal bound is given by $\zeta := O(1/(q_S + N_{\max}))$. More explanations are given in the full version of our paper. □

C. PROOF FOR OUR ANONYMOUS CREDENTIAL SYSTEM

We first show that the showing protocol is a honest-verifier zero-knowledge proof of knowledge. We must show that this protocol is complete, can be simulated and has an extractor.

Lemma 3 *The showing protocol is complete.*

PROOF. Let assume that an honest prover in possession of $\{\sigma_n\}_{n=1}^N$ correctly computed $\sigma, T_1, \ldots, T_{12}$. We have $e(T_6, g_2)e(h_1, g_2)^{-r_X} e(g_1, h_2)^{r_Y} / e(g_1, T_8) = e(X {h_1}^{r_X} {h_1}^{-r_X}, g_2)$ $e(g_1, Y^{-1} {h_2}^{-r_Y} {h_2}^{r_Y}) = 1$ so this equations proves that (T_6, T_8) contains a valid Diffie-Hellman tuple. The same holds for the values contained in (T_2, T_4). We have $e(T_{10}, g_2)e(, T_{10}h_2)^{-r_D} e(h_1, T_4\Delta_2)^{-r_A} e(h_1, h_2)^{r_A r_D} = e(A {h_1}^{r_A} {h_1}^{-r_A}, D {h_2}^{r_D} {h_2}^{-r_D} \Delta_2)$ and $e(uT_6, g_2)e(h_1, g_2)^{-r_X} = e(u\ X {h_1}^{r_X}$ ${h_1}^{-r_X}, g_2)$ so the values in T_{10} and T_4 are a signature of the value in T_6. Finally we have $e(T_{12}, g_2)e(h_1, g_2)^{-r_\sigma} = e(\sigma$ ${h_1}^{r_\sigma} {h_1}^{-r_\sigma}, g_2) = e(A, \prod_n \Gamma_{n,2}) \prod_n e(\mathcal{H}(\Gamma_n \| m_n), \Gamma_{n,2}) = e($ $T_{10} {h_1}^{-r_A}, \prod_n \Gamma_{n,2}) \prod_n e(\mathcal{H}(\Gamma_n \| m_n), \Gamma_{n,2})$ so the value in T_{12} is a valid aggregate certificate. □

Lemma 4 (ZK) *For an honest-verifier, transcripts of the showing protocol can be simulated if DDH is hard in $\mathbb{G}_1$.*

PROOF. As in a Schnorr-like proof, we construct a simulator by randomly choosing $B, D, X, Y, A, \sigma \xleftarrow{\$} \mathbb{G}_1$ and $r_B, r_D, r_X, r_Y, r_A, r_\sigma \xleftarrow{\$} \mathbb{Z}_p$. We set $T_1 := {g_1}^{r_B}$, $T_2 := B {h_1}^{r_B}$, and for the rest as well. Under the DDH assumption in $\mathbb{G}_1$, the pair (T_1, T_2) and the others are indistinguishable from one outputs by a real prover. Next, we choose random values $c, s_B, s_D, s_X, s_Y, s_A, s_\sigma \xleftarrow{\$} \mathbb{Z}_p$ and compute the rest of the proof of knowledge accordingly. Values $(T_1, \ldots, R_1, \ldots, c, s_X, \ldots)$, where values $\{R_i\}, c, \{s_j\}$ come from the proof

of knowledge, satisfy the verification equations and are distributed as in a real transcript. □

Lemma 5 (PK) *The showing protocol has an extractor.*

PROOF. Using the rewinding technique, we extract values $\tilde{r}_B, \tilde{r}_D, \tilde{r}_X, \tilde{r}_Y, \tilde{r}_A, \tilde{r}_\sigma, s$ from the proof of knowledge. From these values we are able to compute $\tilde{\sigma} = T_{12} \cdot {h_1}^{-\tilde{r}_\sigma}$ which is a valid aggregate credential on index $(\tilde{X}, \tilde{Y}), (\tilde{A}, \tilde{B}, \tilde{D})$ where $\tilde{B} := T_2 \cdot {h_1}^{-\tilde{r}_B}$, $\tilde{D} := T_4 \cdot {h_1}^{-\tilde{r}_D}$, $\tilde{X} := T_6 \cdot {h_1}^{-\tilde{r}_X}$, $\tilde{Y} := T_8 \cdot {h_1}^{-\tilde{r}_Y}$ and $\tilde{A} := T_{10} \cdot {h_1}^{-\tilde{r}_A}$. □

Lemma 6 (Correctness) *The $\mathcal{AC}_1$ scheme is correct.*

PROOF. The scheme is correct if the verification is correct, so the correctness of the anonymous credential scheme follows from the protocol completeness (cf. Lemma 3). □

Lemma 7 (Unforgeability) *The $\mathcal{AC}$ scheme is unforgeable if the q-ADHSDH problem is hard and $\mathcal{H}$ is seen as a random oracle.*

PROOF. We reduce an adversary $\mathcal{A}$ against the unforgeability of our anonymous credential system $\mathcal{AC}$ to an adversary $\mathcal{B}$ against the unforgeability of our indexed aggregate signature $\mathcal{AG}$. $\mathcal{B}$ has access to her own random and signature oracles $\mathcal{H}$ and SIGN. Let $(g_1, u, f, g_2, \Delta, \mathcal{BB})$ be system parameters for the $\mathcal{AG}$ scheme and Γ be challenge public key. $\mathcal{B}$ picks $h_1 \in \mathbb{G}_1$, $h_2 \in \mathbb{G}_2$ and supplies parameters $(g_1, h_1, u, f, g_2, h_2)$ to $\mathcal{A}$. To simulate ADDO(j)[3], we generates for $s_j \xleftarrow{\$} \mathbb{Z}_p$ $\Gamma_j := (\Gamma_{,1} {g_1}^{s_j}, \Gamma_{,2} {g_2}^{s_j})$. To simulate hash queries $\mathcal{H}_{\text{SIM}}(m)$ we do: if m can be parsed as $G \| H \| m'$ where (G, H) is a valid Diffie-Hellman pair, then we make a call to $\mathcal{B}$'s own oracle $h \leftarrow \mathcal{H}(\Gamma_{,1} \| \Gamma_{,2} \| m')$, pick $c \xleftarrow{\$} \mathbb{Z}_p$ and set $w \leftarrow h {g_1}^c$ and record $H[m] \leftarrow (w, c)$. Else, we call $h \leftarrow \mathcal{H}(m)$ and record $H[m] \leftarrow (h, \perp)$.

We introduce a subroutine $\text{SIGN}_{\text{SIM}}(j, \mathsf{i}, m)$ in order to simulate individual signatures on index i from signer j. We retrieve the public key $(\Gamma_{j,1}, \Gamma_{j,2})$, simulate a hash query $\mathcal{H}_{\text{SIM}}(\Gamma_{j,1} \| \Gamma_{j,2} \| m)$ and retrieve the corresponding (w, c). We next carry out a signature query $s \leftarrow \text{SIGN}(\mathsf{i}, m)$ and return $\sigma := s\big(Aw\big)^{s_j} {\Gamma_{,1}}^c$.

To simulate the GETCRED oracle, we simply follow the issuing protocol Figure 3 with subroutine $\text{SIGN}_{\text{SIM}}(j, \mathbf{usk}[i], m)$ in order to issue credentials on each $m \in \{m_i\}_{i \in [l]}$. To simulate the SNDTOI oracle, we distinguish the two part of the protocol. Regarding the first part of the issuing protocol, adversary $\mathcal{A}$ supply a proof of knowledge of her keys we extract by rewinding techniques. We are then able to recover $\tilde{\mathsf{i}}$ and record it $\mathbf{usk}[i] \leftarrow \tilde{\mathsf{i}}$. Regarding the second part, $\mathcal{B}$ is able, thanks to $\tilde{\mathsf{i}}$, to issue credentials by querying the subroutine $\text{SIGN}_{\text{SIM}}(j, \tilde{\mathsf{i}}, m)$ for each $m \in \{m_i\}_{i \in [l]}$. There is no problem to simulate other oracles.

Let's assume that $\mathcal{A}$ successfully pass the showing protocol on tuples $\{(j_n, m_n)\}_{n=1}^N$ she specified. From Lemma 5 we are able to extract a tuple $(\tilde{\mathsf{i}}_*, \sigma_*)$ such that σ_* is a valid aggregate credential on $(\tilde{\mathsf{i}}_*, \{(j_n, m_n)\}_{n=1}^N)$. We retrieve random s_{j_n}'s for all $n \in [N]$ and compute

$$\sigma := \sigma_* \cdot \prod_{n=1}^{N} \Big[\big(\tilde{A}_* \mathcal{H}(\Gamma_{,1} \| \Gamma_{,2} \| m_n) {g_1}^{c_n} \big)^{-s_{j_n}} \cdot {\Gamma_{,1}}^{-c_n} \Big]$$

$\mathcal{B}$ returns $(\tilde{\mathsf{i}}_*, \{(\Gamma, m_n)\}_{n=1}^N, \sigma)$ as response to her challenge.

First we show that signatures outputs by the SIGN_{SIM} subroutine are correct. We have $e(\sigma, g_2) = e(s, g_2) e\big(\big(Aw\big)^{s_j} {\Gamma_{*,1}}^c, g_2\big) = e(A, \Gamma_{j,2}) e(\mathcal{H}_{\text{SIM}}(\Gamma_{j,1} \| \Gamma_{j,2} \| m), \Gamma_{j,2})$ so σ is a valid signature on (i, m) under Γ_j. Then we show that the computed $\mathcal{AG}$ forgery is valid if the $\mathcal{AC}$ forgery is valid. If σ_* is a valid aggregate credential, then

$$\begin{aligned} \sigma_* &= \prod_{n=1}^{N} \text{SIGN}_{\text{SIM}}(j_n, \tilde{\mathsf{i}}_*, m_n) = \prod_{n=1}^{N} \Big[\big(\tilde{A} \mathcal{H}_{\text{SIM}}(\Gamma_j \| m_n) \big)^{\gamma_{j_n}} \Big] \\ &= \prod_{n=1}^{N} \Big[\text{SIGN}(\tilde{\mathsf{i}}_*, m_n) \big(\tilde{A} \mathcal{H}(\Gamma \| m_n) {g_1}^{c_n} \big)^{s_{j_n}} {\Gamma_{,1}}^{c_n} \Big] \end{aligned}$$

We conclude that $\sigma = \prod_{n=1}^{N} \text{SIGN}(\tilde{\mathsf{i}}_*, m_n)$ therefore σ is a valid aggregate signature on $(\tilde{\mathsf{i}}_*, \{(\Gamma, m_n)\}_{n=1}^N)$. Finally we have to show that, if the $\mathcal{AC}$ forgery is non trivial, then the computed $\mathcal{AG}$ forgery is non trivial. This is done by distinguish three types of forger that match the one for the $\mathcal{AG}$ scheme (cf. the proof of Theorem 1 given in Appendix B).

Type I. The extracted index $\tilde{\mathsf{i}}$ does not match an existing user secret key $\mathbf{usk}[i]$ for $i \in HU \cup CU$. In this case, $\mathcal{B}$ successfully wins the game against $\mathcal{AG}$.

Type II. The extracted index $\tilde{\mathsf{i}}$ matches an existing user secret key $\mathbf{usk}[i]$ for $i \in HU \cup CU$. If $\mathcal{A}$ successfully wins the game, there is $\mathsf{n} \in [N]$ such that $j_n \in HO$ and $m_\mathsf{n} \notin \{m_i\}_{i \in [l]}$ for all messages $\{m_i\}_{i \in [l]}$ $\mathcal{A}$ asked. This means that honest organization j_n did not issue a credential on m_n. In particular, there was no call on $\text{SIGN}_{\text{SIM}}(\tilde{\mathsf{i}}_*, m_\mathsf{n})$ and, subsequently, no call on $\mathcal{B}$'s oracle $\text{SIGN}(\tilde{\mathsf{i}}_*, m_\mathsf{n})$. Hence $m_\mathsf{n} \notin M[\tilde{\mathsf{i}}_*]$ and that $\mathcal{B}$ successfully wins the game against $\mathcal{AG}$. □

Lemma 8 (Anonymity) *The $\mathcal{AC}$ scheme is anonymous if the DDH problem is hard in $\mathbb{G}_1$.*

PROOF. Let $(g, v := {g_1}^a, h, w := h^d)$ be a DDH challenge in $\mathbb{G}_1$. Among parameters, we set $g_1 \leftarrow g$. The other are honestly generated. We uniformly pick a bit $b \in \{0, 1\}$. Hash queries are simulated with a new random value in $\mathbb{G}_1$ for any new query. To simulate the SNDTOU oracle, we simply follows the OBTAIN algorithm Figure 3. Since checks are done during the protocol, we are sure that only valid credentials will be recorded by user i. To simulate the CH oracle, we pick $r, r' \xleftarrow{\$} \mathbb{Z}_p$ and computes $(v', w') \leftarrow (v^r g^{r'}, w^r g^{r'})$. Tuple (g, v', h, w') is a randomly distributed DDH instance derived from the challenge. We then compute $(T_{11}, T_{12}) := (v', \sigma w')$ where σ is the aggregation of credentials from user i_b. the same is done for values X, Y and A. From these values, we simulate a protocol transcript t as specified in Lemma 4. t is returned to $\mathcal{A}$. There is no problem to simulate other oracles. Finally, $\mathcal{A}$ outputs a bit b'. We output 1 if $b' = b$ and 0 otherwise as response to our own challenge. If the challenge is a DDH tuple, then algorithm $\mathcal{A}$ has advantage ε to distinguish b. If it is a random tuple, then the T_i's are independent of b and $\mathcal{A}$ has advantage 0 to distinguish the underlying user. We conclude $\mathcal{B}$ has advantage $\varepsilon/2$ to solve the DDH challenge. □

[3]To simplify, we disallow the ORGKG oracle. We can add it and lose a factor linear in the number of signers. Moreover, this oracle is relevant in case of indistinguishability property, not for the unforgeability one.