Efficient Redactable Signature and Application to Anonymous Credentials

Olivier Sanders(✉)

Orange Labs, Applied Crypto Group, Cesson-Sévigné, France
olivier.sanders@orange.com

Abstract. Let us assume that Alice has received a constant-size signature on a set of messages $\{m_i\}_{i=1}^n$ from some organization. Depending on the situation, Alice might need to disclose, prove relations about or hide some of these messages. Ideally, the complexity of the corresponding protocols should not depend on the hidden messages. In particular, if Alice wants to disclose only k messages, then the authenticity of the latter should be verifiable in at most $O(k)$ operations.

Many solutions were proposed over the past decades, but they only provide a partial answer to this problem. In particular, we note that they suffer either from the need to prove knowledge of the hidden elements or from the inability to prove that the latter satisfy some relations.

In this paper, we propose a very efficient constant-size redactable signature scheme that addresses all the problems above. Signatures can indeed be redacted to remain valid only on a subset of k messages included in $\{m_i\}_{i=1}^n$. The resulting redacted signature consists of 4 elements and can be verified with essentially k exponentiations. Different shows of the same signature can moreover be made unlinkable leading to a very efficient anonymous credentials system.

1 Introduction

Digital Signature is a major cryptographic tool that is used to attest the authenticity of a digital data, ensuring that not even one bit has been modified. This rigidity is a strength in many scenarios but it also comes with its drawbacks. One of them is that verification of a standard signature requires knowledge of the full signed message.

For example, let us consider the case of a database containing n elements $\{m_i\}_{i=1}^n$ that should be certified by some authority. If the latter signs the whole set $\{m_i\}_{i=1}^n$, there is only one signature σ but checking the authenticity of even one element requires to download the full database. Obviously, this problem could be avoided by signing separately each element but this would replace one signature by potentially billions (n) of them. Between these two solutions one can find different trade-offs, such as splitting $\{m_i\}_{i=1}^n$ into different subsets that would be signed individually, but none of them is fully satisfying. Even solutions based on hash functions, such as Merkle tree, require to download at

© International Association for Cryptologic Research 2020
A. Kiayias et al. (Eds.): PKC 2020, LNCS 12111, pp. 628–656, 2020.
https://doi.org/10.1007/978-3-030-45388-6_22

least a logarithmic number of elements. Moreover, using hash functions prevents efficient proofs of knowledge, which will cause further problems.

The problem described above is not just related to efficiency. It indeed means that, to check the validity of a signature without using hash functions, one must have access to all the signed messages which is also a privacy issue. This problem is probably more obvious in a context where a user gets his attributes (*e.g.* his name, birthdate, address, etc) certified by some authority and then needs to prove the authenticity of only one of them. For example, to benefit from a preferential rate, he might need to prove that he is under 25 years of age. With a standard digital signature, he needs to send all his attributes, even if the latter are totally irrelevant. This means that the merchant will not only have information on his age, but he will also know his name, address and so on.

This problem is far from new in cryptography and a very classical solution for the user is to prove knowledge of the hidden attributes and that the latter are indeed certified by a credential issued by the organization. This requires a digital signature scheme with some nice features but this is not a real problem as several such schemes [4,10,21] have already been proposed. Actually, most anonymous credentials (or attribute based credentials) systems [1,9,10,21] work this way to solve our problem. Moreover, such a primitive can provide additional security guarantees, such as unlinkability of different showings, which are particularly interesting in many contexts.

Regarding privacy, this solution is thus fully satisfying. Regarding efficiency, things got worse as the unnecessary attributes must now be hidden in proofs of knowledge whose cost is at least greater than the one of sending all the attributes in clear. We believe that this problem is inherent to constructions based on digital signatures. As we explain above the latter indeed require the whole set of signed messages to be verified and do not support efficient partial verification. It seems therefore necessary to find another building block to avoid this problem.

Another strategy could be based on cryptographic accumulators, such as the ones from [2,19]. An accumulator C indeed allows to accumulate many elements $\{m_i\} \in \mathcal{I}$ while remaining of constant size. Moreover, for any accumulated message m_i, it is possible to derive a witness w_i proving that m_i has indeed been accumulated in C. If C is further signed, then one gets efficient partial verification on a message m_i: given C, the signature on C and the witness w_i, one can indeed check the authenticity of m_i without knowing any other messages. By using appropriate zero-knowledge proofs, one could even achieve some privacy properties. Actually, this is reminiscent of the approach of [15]. The authors indeed extend Nguyen's accumulator [19] to enable efficient proof that a subset $\{m_i\}_{i\in\mathcal{I}} \subset \{m_i\}_{i=1}^{n}$ has been accumulated. They then show how to combine their accumulator with signatures on equivalence classes to construct an anonymous credentials system with very nice features. Unfortunately, with their solution, once elements are accumulated, one only has the possibility to disclose them, not to prove that they satisfy some relations while hiding them. Concretely, in our example with user's attributes, this means that the user can now reveal his birthdate and any other necessary attribute, but not just prove (efficiently) that he is under 25.

Compared to the previous anonymous credentials cited above, [15] thus solves the efficiency issue but by removing an important feature of anonymous credentials, which implicitly harms privacy.

Finally, the problem of checking the authenticity of parts of the signed messages while hiding the other ones has already been considered by papers on redactable signature [6,18,20]. This primitive allows the user to quote parts of the message signed under σ and yet to prove that the latter is valid on the disclosed parts. Actually, this might seem exactly what we need here but, unfortunately, most redactable signatures aim at achieving some properties, such as transparency (original signatures should be indistinguishable from redacted ones), that do not seem relevant in our context and that negatively impact efficiency. Nevertheless, in [7], Camenisch *et al.* introduce a new variant of redactable signature, called *unlinkable redactable signature* (URS), that does not consider such outlying properties and that is thus perfectly tailored for applications to privacy-preserving protocols. As an example, the authors construct from an URS an anonymous credentials system with remarkable asymptotic complexity. Unfortunately, in this case, asymptotic complexity is not indicative of concrete performances. Current instantiations are indeed still very costly and can hardly compete with the most efficient solutions in practice (see Sect. 7). Moreover, their construction makes use of a vector commitment scheme that shares commonalities with the accumulator used in [15], which leads to the same issue: attributes can be either disclosed or hidden, but proving that some of them satisfy non trivial relations cannot be done efficiently. Nevertheless, to be fair, we must note that [7] provides security in the UC framework [11], which explains in part the efficiency gap with alternative solutions

1.1 Our Contribution

In this work we follow the approach based on URS from [7], but with the aim of achieving extremely efficient protocols. To this end, we construct a very flexible redactable signature scheme, that can be made unlinkable at almost no cost. We then explain how to use it to construct an anonymous credentials system with remarkable efficiency and that still supports proof of relations about attributes, contrarily to [15].

Our starting point is the Pointcheval-Sanders (PS) signature scheme [21] that generates constant size signatures on blocks of messages $(m_1, \ldots, m_n)$. As shown in [21], it comes with a series of features that are extremely useful in a privacy preserving context, such as the ability to efficiently prove knowledge of a signature or to generate a signature on a committed message. Unfortunately, when used to construct anonymous credentials systems, this scheme suffers from the problems described above, namely the fact that non disclosed messages heavily impact complexity, because their knowledge must be proven. Concretely, if one discloses k attributes (and thus hide/redact the $n-k$ other ones), one must still send $O(n-k)$ elements to the verifier (besides the k disclosed attributes) and the latter must still perform $O(n)$ operations.

The first contribution of our paper is the construction of an efficient redactable signature scheme RS from PS signatures. At first sight, this problem might seem easy to solve due to the simple algebraic structure of PS signatures. Indeed, in a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, a PS signature on $(m_1, \ldots, m_n)$, issued with secret key $(x, y_1, \ldots, y_n)$, is a pair $(\widetilde{\sigma}_1, \widetilde{\sigma}_2) \in \mathbb{G}_2$ where $\widetilde{\sigma}_1$ is random and $\widetilde{\sigma}_2 = \widetilde{\sigma}_1^{x+\sum_{i=1}^{n} y_i \cdot m_i}$. By providing $(X, Y_1, \ldots, Y_n) = (g^x, g^{y_1}, \ldots, g^{y_n})$ for some generator $g \in \mathbb{G}_1$ in the public key, one can test the validity of $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$ using the pairing e:

$$e(X \cdot \prod_{i=1}^{n} Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2)$$

When one asks to verify the authenticity of only a subset $\{m_i\}_{i \in \mathcal{I}}$ of messages, it might be tempting to only send $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$ along with an element $\sigma_1 = \prod_{i \in [1,n] \setminus \mathcal{I}} Y_i^{m_i}$ that would accumulate all the redacted elements. The previous equation would then simply become

$$e(X \cdot \sigma_1 \cdot \prod_{i \in \mathcal{I}} Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2). \tag{1}$$

Such a scheme would be extremely efficient: only a constant number of elements[1] needs to be sent to the verifier and the later only needs to perform k exponentiations in $\mathbb{G}_1$. Moreover, the structure of the resulting scheme makes combination with Schnorr's proof of knowledge [23] trivial. One can then hide and prove relations about any m_i with $i \in \mathcal{I}$.

Unfortunately, such a scheme is not secure. We provide details on the problem in Sect. 4.1 but intuitively it stems from the fact that the adversary can hide anything in σ_1, including elements of the form $Y_i^{r_i}$ with $i \in \mathcal{I}$ that it could use to cheat the verifier. A solution could then be to prove that σ_1 only aggregates the elements $Y_i^{m_i}$ with $i \in [1, n] \setminus \mathcal{I}$, by running the classical Schnorr's protocol to prove knowledge of the corresponding m_i. Unfortunately, such a solution takes us back to square one: we need to send $O(n-k)$ elements and verification requires $O(n)$ operations.

Fortunately, we can do far better by observing that we do not really care about the elements accumulated in σ_1. If the adversary manages to add some elements to σ_1 such that the verification equation above is still verified, this is not a problem as long as the added elements are not of the form $Y_i^{r_i}$ for $i \in \mathcal{I}$ and known r_i. Actually, the ability to add random elements to σ_1 should be kept since it will be the key to achieve unlinkability, as we will explain.

To retain security, we must then force the user to prove that σ_1 does not aggregate an element of the above form. Surprisingly, this can be done very efficiently by noticing that the polynomial f defined by $e(g, \widetilde{g})^{f(y_1, \ldots, y_n)} = e(\sigma_1, \prod_{i \in \mathcal{I}} \widetilde{Y}_i)$ will necessarily contain a monomial of the form y_j^2 for some $j \in \mathcal{I}$ if the user has cheated. Conversely, with a valid σ_1, f will not contain such kind

[1] We here follow the convention of previous works that do not include the disclosed elements $\{m_i\}$ in the complexity evaluation.

of monomials, the only degree 2 monomials being of the form $y_i \cdot y_j$ for $i \neq j$. By providing appropriate elements in the public key we can enable the user to prove that f is of the right form by simply providing an element σ_2 in $\mathbb{G}_1$ such that $e(\sigma_1, \prod_{i\in\mathcal{I}} \widetilde{Y}_i) = e(\sigma_2, \widetilde{g})$. That is, we get a secure redactable signature with remarkable efficiency: redacted signatures contain 4 elements and can be verified with 4 pairings and k exponentiations, whatever the values of k and n.

We believe that such a redactable signature is of independent interest. However, although it is redactable and unforgeable, it is not unlinkable and so cannot be directly used to build an anonymous credentials system. Our next contribution is then to enhance it to construct an URS in the sense of [7].

Here, the transformation is based on our previous observation. Our "proof of validity" of σ_1 does not prove that σ_1 is of the expected form $\prod_{i\in[1,n]\setminus\mathcal{I}} Y_i^{m_i}$ but simply that it does not contain illicit elements $Y_i^{r_i}$, for $i \in \mathcal{I}$. In particular, we can aggregate anything in σ_1 as long as it is not of the latter form and Eq. (1) is verified. To satisfy both conditions, we will use the fact that PS signatures can be sequentially aggregated to add to $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$ a signature on a random message t under a dummy public key and then modify σ_1 and σ_2 appropriately. That is, a new derived signature on $\{m_i\}_{i\in\mathcal{I}}$ is the resulting aggregate signature whose messages $\{m_i\}_{i\in[1,n]\setminus\mathcal{I}}$ and t have been redacted. As we prove in our paper, the random elements added in the process perfectly blind the original signature and thus ensure unlinkability at almost no cost: few additional exponentiations to redact the signature, but the signature size and the verification process remain unchanged.

Once we have our unlinkable redactable signature scheme, the transformation into an anonymous credentials system is rather straightforward because we inherit most of the nice features of PS signatures. We just have to adapt the protocol to get a credential on a committed value from [21] and then to add a proof of knowledge of the user's secret key during the showing process. Regarding efficiency, there is almost no change and the resulting protocol compares favourably with the state-of-the-art (see Sect. 7). In particular, in our system, the user only has to send a constant number of elements to prove possession of k attributes and the verifier only has to perform $O(k)$ operations, even if the credential was initially issued on a much larger number n of attributes. The main difference with our URS construction is the anonymity proof that is more intricate and that now makes use of the DDH assumption.

In the end, we get a remarkably versatile system which can provide both security and privacy with very good performance.

1.2 Organisation

We recall in Sect. 2 the definition of bilinear groups and two computational assumptions that we use to prove the security of our schemes. The syntax and the security model of redactable signatures (resp. anonymous credentials) are provided in Sect. 3 (resp. Sect. 6). Our redactable signature scheme is presented in Sect. 4 along with a variant achieving additional properties. The security proofs

of our main construction can be found in the same section, those for the variant are provided in the full version [22] of this paper due to lack of space. Our anonymous credentials system is described, and proved secure, in Sect. 6. Finally, we compare the efficiency of our constructions with the one of the most relevant schemes from the state-of-the-art in Sect. 7.

2 Preliminaries

Bilinear Groups. Our construction requires bilinear groups whose definition is recalled below.

Definition 1. *Bilinear groups are a set of three groups* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$ *of order* p *along with a map, called pairing,* $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ *that is*

1. *bilinear: for any* $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$*, and* $a, b \in \mathbb{Z}_p$*,* $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$*;*
2. *non-degenerate: for any* $g \in \mathbb{G}_1^*$ *and* $\widetilde{g} \in \mathbb{G}_2^*$*,* $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$*;*
3. *efficient: for any* $g \in \mathbb{G}_1$ *and* $\widetilde{g} \in \mathbb{G}_2$*,* $e(g, \widetilde{g})$ *can be efficiently computed.*

In this work, we need bilinear groups of prime order with *type 3* pairings [16], meaning that no efficiently computable homomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$. We stress that this is not a significant restriction since this yields the most efficient parameters [12,17].

Computational Assumptions. The security analysis of our protocols will make use of the following two assumptions.

- DL assumption: Given $(g, g^a) \in \mathbb{G}^2$, the DL assumption in the group $\mathbb{G}$ states that it is hard to recover a.
- DDH assumption: Given $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, the DDH assumption in the group $\mathbb{G}$ states that it is hard to decide whether $c = a \cdot b$ or c is random.

3 Redactable Signatures

A signature σ on some set of messages $\{m_i\}_{i=1}^n$ is redactable if it is possible to derive from it a signature $\sigma_{\mathcal{I}}$ on a subset of messages $\{m_i\}_{i \in \mathcal{I}}$, with $\mathcal{I} \subset [1, n]$. The point is that the verification of $\sigma_{\mathcal{I}}$ no longer requires the knowledge of the messages m_i for $i \in \overline{\mathcal{I}}$, where $\overline{\mathcal{I}} = [1, n] \setminus \mathcal{I}$. This feature is particularly useful when one only needs to check the authenticity of a subset of the messages. However, redacting messages does not necessarily mean hiding them and so it is necessary to consider additional properties when privacy is required. This leads us to the following definition of redactable signatures, adapted from [7].

3.1 Syntax

A redactable signature consists of the 4 following algorithms.

- $\mathtt{Keygen}(1^k, n)$: On input a security parameter 1^k and an integer n, this algorithm returns a key pair $(\mathsf{sk}, \mathsf{pk})$ supporting signatures on sets of n messages $\{m_i\}_{i=1}^n$.
- $\mathtt{Sign}(\mathsf{sk}, \{m_i\}_{i=1}^n)$: On input n messages $\{m_i\}_{i=1}^n$ and the signing key sk, this algorithm outputs a signature σ.
- $\mathtt{Derive}(\mathsf{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$: On input a signature σ on $\{m_i\}_{i=1}^n$, the public key pk and a subset $\mathcal{I} \subset [1, n]$, this algorithm returns a redacted (or derived) signature σ_I on the subset of messages $\{m_i\}_{i\in\mathcal{I}}$. In this paper, we will omit the subscript $\mathcal{I}$ of $\sigma_\mathcal{I}$ if this set is clear from the context.
- $\mathtt{Verify}(\mathsf{pk}, \sigma, \{m_i\}_{i\in\mathcal{I}})$: On input the public key pk, a set of messages $\{m_i\}_{i\in\mathcal{I}}$ and a signature σ (redacted or not), this algorithm outputs 1 (valid) or 0 (invalid).

Notation. In this paper, we will consider *sets* of messages $\{m_i\}_{i=1}^n$ instead of *vectors* $(m_1, \ldots, m_n)$ to highlight the benefits of redactability. Indeed, with this notation, a redacted signature σ_I can be verified only with the knowledge of the $|\mathcal{I}|$ elements in $\{m_i\}_{i\in\mathcal{I}}$. Conversely, with a vector notation, verification of σ_I would still require to send a vector of n elements $(m'_1, \ldots, m'_n)$, with $m'_i = \perp$ for $i \in \overline{\mathcal{I}}$.

We nevertheless stress that it is only a notation issue. In particular, even with our notation, the position of the messages (indicated by their index) remains crucial. For example, if $m_1 = m'_2$ and $m_2 = m'_1$, then we stress that a valid signature on $\{m_1, m_2\}$ is *not* valid on $\{m'_1, m'_2\}$ (this would be considered as a valid forgery in our security game). More generally, we will consider in this paper that $\{m_i\}_{i\in\mathcal{I}} \subset \{m'_i\}_{i=1}^n$ when $m_i = m'_i \ \forall i \in \mathcal{I}$.

3.2 Security Model

Correctness. Correctness requires that, for honestly generated keys, honestly generated and honestly derived signatures always verify.

Unforgeability. In [7], the authors consider a very strong notion of unforgeability. Indeed, besides the natural requirements for a signature scheme, their definition considers a signature $\sigma_\mathcal{I}$ on $\{m_i\}_{i\in\mathcal{I}}$, derived from a signature σ valid on $\{m_i\}_{i=1}^n$, as a forgery if the adversary only had access to other redacted versions $\sigma_{\mathcal{J}_k}$ of σ with $\mathcal{J}_k \neq \mathcal{I}$. Concretely, this means that the adversary succeeds if it can produce a new redacted version of a signature, even if the signer has actually signed the messages $\{m_i\}_{i\in\mathcal{I}}$. In this paper, we will call this security notion "strong unforgeability" because it is reminiscent of the eponymous notion for standard digital signature schemes.

Although we will show that our unlinkable redactable signature scheme of Sect. 4.2 satisfies this strong property, we believe that it is too strong for many

scenarios. For example our anonymous credentials construction only needs a weaker version, that we simply call "unforgeability", where new derivations of a signature are no longer considered as a forgery. Moreover, the strong unforgeability notion forbids some nice features, such as the ability to further redact a redacted signature. Finally, as we will show in Sect. 4.1, we can construct more efficient schemes if we only aim at achieving our unforgeability property.

We therefore think that it is relevant to consider this new notion that we define below. However, for completeness, we also recall the original one from [7].

Our security experiments in Fig. 1 make use of a counter c and three tables, Q_1, Q_2 and Q_3, along with the following oracles:

- $\mathcal{O}\mathtt{Sign}^*(\{m_i\}_{i=1}^n)$: on input a set of n messages, this oracle returns $\mathtt{Sign}(\mathsf{sk}, \{m_i\}_{i=1}^n)$, stores $Q_1[c] = (\sigma, \{m_i\}_{i=1}^n)$ and increments $c \leftarrow c+1$.
- $\mathcal{O}\mathtt{Sign}(\{m_i\}_{i=1}^n)$: on input a set of n messages, this oracle computes $\sigma \leftarrow \mathtt{Sign}(\mathsf{sk}, \{m_i\}_{i=1}^n)$, stores $Q_1[c] = (\sigma, \{m_i\}_{i=1}^n)$ and increments $c \leftarrow c+1$.
- $\mathcal{O}\mathtt{Derive}(k, \mathcal{I})$: on input an index k and a set $\mathcal{I}$, this algorithm returns $\perp$ if $Q_1[k] = \emptyset$ or if $\mathcal{I} \nsubseteq [1, n]$. Else, it uses σ and $\{m_i\}_{i=1}^n$ stored in $Q_1[k]$ to return $\mathtt{Derive}(\mathsf{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$. The set $\{m_i\}_{i\in\mathcal{I}}$ is then added to Q_2.
- $\mathcal{O}\mathtt{Reveal}(k)$: on input an index k, this algorithm returns $\perp$ if $Q_1[k] = \emptyset$ and $Q_1[k] = (\sigma, \{m_i\}_{i=1}^n)$ otherwise. The set $\{m_i\}_{i=1}^n$ is then added to Q_3.

We note that the only difference between $\mathcal{O}\mathtt{Sign}^*$ and $\mathcal{O}\mathtt{Sign}$ is that the former returns the signature, contrarily to the latter that does not return anything. Our unforgeability experiment only uses $\mathcal{O}\mathtt{Sign}^*$, which makes the $\mathcal{O}\mathtt{Reveal}$ and $\mathcal{O}\mathtt{Derive}$ oracles useless. For convenience, the set of messages $\{m_i\}_{i=1}^n$ stored in $Q_1[j]$ will be denoted $\{m_i^{(j)}\}_{i=1}^n$.

A redactable signature scheme is *unforgeable* if $\mathtt{Adv}^{uf}(\mathcal{A}) = |\Pr[\mathtt{Exp}_{\mathcal{A}}^{uf}(1^k, n) = 1]|$ is negligible for any polynomial time adversary $\mathcal{A}$. A redactable signature scheme is *strongly unforgeable* if $\mathtt{Adv}^{suf}(\mathcal{A}) = |\Pr[\mathtt{Exp}_{\mathcal{A}}^{suf}(1^k, n) = 1]|$ is negligible for any polynomial time adversary $\mathcal{A}$.

Unlinkability. Unlinkability states that it should be hard to link back a derived signature $\sigma_{\mathcal{I}}$ to its origin σ, unless the disclosed (non redacted) messages $\{m_i\}_{i\in\mathcal{I}}$ trivially allow to do so. In particular, this implies that $\sigma_{\mathcal{I}}$ does not leak any information on the redacted messages $\{m_i\}_{i\in\overline{\mathcal{I}}}$, even for an adversary that has generated the public key pk. This property is formally defined by the experiment $\mathtt{Exp}_{\mathcal{A}}^{unl-b}(1^k, n)$ of Fig. 1. A redactable signature scheme is unlinkable if $\mathtt{Adv}^{unl} = |\Pr[\mathtt{Exp}_{\mathcal{A}}^{unl-1}(1^k, n) = 1] - \Pr[\mathtt{Exp}_{\mathcal{A}}^{unl-0}(1^k, n) = 1]|$ is negligible for any polynomial time adversary $\mathcal{A}$.

4 Short Redactable Signatures

4.1 Our Construction

Our main building block to construct an unlinkable redactable signature or an anonymous credentials system will be the following redactable signature scheme

Unforgeability
$\mathtt{Exp}^{uf}_{\mathcal{A}}(1^k, n)$

1. $c \leftarrow 0$; $Q_1 \leftarrow \emptyset$;
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Keygen}(1^k, n)$
3. $(\sigma^*, \{m_i\}_{i\in\mathcal{I}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign}^*}(\mathsf{pk})$
4. Return 1 if $\mathcal{I} \neq \emptyset$
   and $\mathtt{Verify}(\mathsf{pk}, \sigma^*, \{m_i\}_{i\in\mathcal{I}}) = 1$
   and $\forall j < c, \exists k_j \in \mathcal{I} : m_{k_j} \neq m^{(j)}_{k_j}$
5. Else, return 0

Strong Unforgeability
$\mathtt{Exp}^{suf}_{\mathcal{A}}(1^k, n)$

1. $Q_1, Q_2, Q_3 \leftarrow \emptyset$;
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Keygen}(1^k, n)$
3. $(\sigma^*, \{m_i\}_{i\in\mathcal{I}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign}, \mathcal{O}\mathtt{Derive}, \mathcal{O}\mathtt{Reveal}}(\mathsf{pk})$
4. Return 1 if $\mathcal{I} \neq \emptyset$
   and $\mathtt{Verify}(\mathsf{pk}, \sigma^*, \{m_i\}_{i\in\mathcal{I}}) = 1$
   and $\{m_i\}_{i\in\mathcal{I}} \notin Q_2$
   and $\forall \{m'_i\}_{i=1}^n \in Q_3$:
   $\exists k_j \in \mathcal{I} : m_{k_j} \neq m'_{k_j}$
5. Else, return 0

Unlinkability
$\mathtt{Exp}^{unl-b}_{\mathcal{A}}(1^k, n)$

1. $(\mathsf{pk}, \mathcal{I}, \{m^{(0)}_i\}_{i=1}^n, \{m^{(1)}_i\}_{i=1}^n, \sigma^{(0)}, \sigma^{(1)}) \leftarrow \mathcal{A}()$
2. If $\mathtt{Verify}(\mathsf{pk}, \sigma^{(0)}, \{m^{(0)}_i\}_{i=1}^n) = 0$, return 0
3. If $\mathtt{Verify}(\mathsf{pk}, \sigma^{(1)}, \{m^{(1)}_i\}_{i=1}^n) = 0$, return 0
4. If $\exists j \in \mathcal{I} : m^{(0)}_j \neq m^{(1)}_j$, return 0
5. $\sigma^{(b)}_{\mathcal{I}} \leftarrow \mathtt{Derive}(\mathsf{pk}, \sigma^{(b)}, \{m^{(b)}_i\}_{i=1}^n, \mathcal{I})$
6. $b' \leftarrow \mathcal{A}(\sigma^{(b)}_{\mathcal{I}})$
7. Return b'.

Fig. 1. Security notions for redactable signatures

RS. The latter is unforgeable but it is clearly not unlinkable. We will explain in the next section how to enhance it to achieve this property. Nevertheless, we believe that this construction might be of independent interest due to its efficiency, for scenarios where privacy is not necessary.

Intuition. The signatures output by our Sign algorithm are PS signatures [21] on the messages $(m_1, ..., m_n)$. However, such signatures do not support partial verification, on a subset of $\{m_i\}_{i=1}^n$: all the signed messages must be disclosed, or one must prove knowledge of them, which in all cases imply to send at least n elements and to perform n exponentiations to verify the signature.

When considering the verification equation of PS signatures $e(X \prod_{i=1}^n Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2)$, it might be tempting to circumvent this problem by simply regrouping all the elements $\{Y_i^{m_i}\}_{i\in\overline{\mathcal{I}}}$ in $\sigma_1 = \prod_{i\in\overline{\mathcal{I}}} Y_i^{m_i}$. The verification equation would then become:

$$e(X \cdot \sigma_1 \prod_{i\in\mathcal{I}} Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2).$$

Unfortunately, the resulting scheme would clearly be insecure. Indeed, nothing prevents a dishonest user from hiding some parts of the disclosed messages in σ_1 to deceive the verifier. For example, if one receives a signature on $\{m_i\}_{i=1}^n$,

one can set $\sigma_1 = Y_1^r \cdot \prod_{i=2}^{n} Y_i^{m_i}$ and then claims a signature on $m_1 - r$, for any $r \in \mathbb{Z}_p$. Indeed, in such a case

$$e(X \cdot \sigma_1 \cdot Y_1^{m_1 - r}, \widetilde{\sigma}_1) = e(X \cdot Y_1^{m_1 - r} \cdot Y_1^r \prod_{i=2}^{r} Y_i^{m_i}, \widetilde{\sigma}_1)$$
$$= e(g, \widetilde{\sigma}_2)$$

so the equation would be verified. The element σ_1 cannot therefore be any element of $\mathbb{G}_1$, it is necessary to prove that it only accumulates messages whose index is not in $\mathcal{I}$. The conceptually simplest solution, that is actually used in most anonymous credentials constructions, is to prove knowledge of the undisclosed messages. However, as we have explained, this leads to a cost at least linear in the size of $\overline{\mathcal{I}}$. We therefore use here a much more efficient solution, based on the following idea. If σ is a signature that has been honestly derived for $\mathcal{I}$, then the pairing $e(\sigma_1, \prod_{i \in \mathcal{I}} \widetilde{Y}_i)$ evaluates to $e(g, \widetilde{g})^{f(y_1, \ldots, y_n)}$ for some polynomial f whose monomials are of the form $y_i \cdot y_j$, for $i \neq j$. Conversely, if one tries to hide some parts of the disclosed messages in σ_1, as in the attack we sketched above, then f now contains monomials of the form y_i^2, for some $i \in [1, n]$.

These two cases can easily be distinguished by adding the elements $Z_{i,j} = g^{y_i \cdot y_j}$ to the public key, for $i \neq j$. Indeed, these elements can trivially be used to reconstruct f in the former case, whereas they will not be sufficient in the latter case. Concretely, an honest user can compute $\sigma_2 \leftarrow \prod_{i \in \mathcal{I}, j \in \overline{\mathcal{I}}} Z_{i,j}^{m_j}$ and then prove that σ_1 is well formed with our second verification equation:

$$e(\sigma_1, \prod_{i \in \mathcal{I}} \widetilde{Y}_i) = e(\sigma_2, \widetilde{g})$$

Providing a similar element for an invalid σ_1 is equivalent to computing $g^{\sum_{j \in \mathcal{J}} y_j^2}$, for some $\mathcal{J} \subset [1, n]$, which is thought to be impossible in bilinear groups, given only the elements of the public key. A formal security analysis is provided in Sect. 4.3. We nevertheless recall that our redactable signature scheme `RS` is *not* strongly unforgeable. Such a property is achieved by our construction `URS` as a (positive) side effect of unlinkability.

The Scheme.

- `Keygen`$(1^k, n)$: on input a security parameter 1^k and an integer n, this algorithm generates $(g, \widetilde{g}) \xleftarrow{\$} \mathbb{G}_1^* \times \mathbb{G}_2^*$ along with $(n + 1)$ random scalars $x, y_1, \ldots, y_n \xleftarrow{\$} \mathbb{Z}_p$ and computes the following elements:
  - $X \leftarrow g^x$
  - $Y_i \leftarrow g^{y_i}$, $\forall 1 \leq i \leq n$
  - $\widetilde{Y}_i \leftarrow \widetilde{g}^{y_i}$, $\forall 1 \leq i \leq n$
  - $Z_{i,j} \leftarrow g^{y_i \cdot y_j}$, $\forall 1 \leq i \neq j \leq n$

  The secret key sk is then $(x, y_1, \ldots, y_n)$ whereas the public key pk is $(X, \{(Y_i, \widetilde{Y}_i)\}_{1 \leq i \leq n}, \{Z_{i,j}\}_{1 \leq i \neq j \leq n})$.

- $\texttt{Sign}(\mathsf{sk}, \{m_i\}_{i=1}^n)$: To sign n messages $m_1, \ldots, m_n$, the signer selects a random element $\widetilde{\sigma}_1 \xleftarrow{\$} \mathbb{G}_2$, computes $\widetilde{\sigma}_2 \leftarrow \widetilde{\sigma}_1^{x+\sum_{i=1}^n y_i \cdot m_i}$ and then outputs the signature $\sigma = (1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$.
- $\texttt{Derive}(\mathsf{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$: on input a signature $\sigma = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ on $\{m_i\}_{i=1}^n$, the public key pk and a subset $\mathcal{I} \subset [1, n]$, this algorithm generates:
  - $\sigma_1' = \prod_{j \in \overline{\mathcal{I}}} Y_j^{m_j}$
  - $\sigma_2' = \prod_{i \in \mathcal{I}, j \in \overline{\mathcal{I}}} Z_{i,j}^{m_j} = \prod_{j \in \overline{\mathcal{I}}} (\prod_{i \in \mathcal{I}} Z_{i,j})^{m_j}$

  where $\overline{\mathcal{I}} = [1, n] \setminus \mathcal{I}$. If $\mathcal{I} = [1, n]$, then $\overline{\mathcal{I}} = \emptyset$ and $\sigma_1' = \sigma_2' = 1_{\mathbb{G}_1}$. In all cases, the signer returns the derived signature $\sigma_I = (\sigma_1', \sigma_2', \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ on $\{m_i\}_{i \in \mathcal{I}}$.
- $\texttt{Verify}(\mathsf{pk}, \sigma, \{m_i\}_{i \in \mathcal{I}})$: A signature $\sigma = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2) \in \mathbb{G}_1^2 \times (\mathbb{G}_2^*)^2$ is valid on a subset of messages $\{m_i\}_{i \in \mathcal{I}}$ if the following equalities hold, in which case the algorithm returns 1:
  1. $e(X \cdot \sigma_1 \prod_{i \in \mathcal{I}} Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2)$
  2. $e(\sigma_1, \prod_{i \in \mathcal{I}} \widetilde{Y}_i) = e(\sigma_2, \widetilde{g})$

  If (at least) one of these equations is not satisfied, then the algorithm returns 0.

Remark 2. We add $(1_{\mathbb{G}_1}, 1_{\mathbb{G}_1})$ to the signatures returned by `Sign` so that they have the same structure as derived signatures, produced by `Derive`. We note that, for such signatures, the second verification equation is trivially satisfied and does not require pairing computations: both pairings evaluate to $1_{\mathbb{G}_T}$.

We stress that any signature derived for a subset $\mathcal{I}$ can be verified without knowledge of the redacted messages (those whose indices are in $\overline{\mathcal{I}}$). Moreover, the computational cost for the verifier does not depend on the number of redacted messages, namely $|\overline{\mathcal{I}}|$.

Remark 3. One can note that $Z_{i,j} = Z_{j,i}$ for all $1 \leq i \neq j \leq n$. Therefore the public pk contains $1 + \frac{n(n+3)}{2}$ elements. Nevertheless, we note that verification does not require the knowledge of the elements $Z_{i,j}$ that are only useful to derive signatures. In practice, one could then define a verification key $\mathsf{vk} = (X, \{(Y_i, \widetilde{Y}_i)\}_{1 \leq i \leq n})$, containing only $1 + 2n$ elements, that is sufficient to verify any signature (derived or not).

4.2 Achieving Unlinkability

The redactable scheme `RS` described in Sect. 4.1 is unforgeable but it is not unlinkable. As in [21], we could try to rerandomize each element by raising it to some random power, but this would only work for the $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$ part of the signature. Rerandomizing similarly the other half of the signature seems to be much more complex and is likely to require more elements and more pairing equations to prove validity of the resulting signature.

We therefore use two tricks to achieve unlinkability. The first one is that we can add in σ_1 any element that is not of the form Y_i^r, for $i \in \mathcal{I}$ and some known scalar r. The second one is the ability of PS signatures to be sequentially

aggregated. Concretely, we will aggregate a signature on a random message t under a dummy public key to the original signature and we will then include t in the set of redacted messages. Intuitively, the randomness of t will hide any information on the messages $m_i \in \overline{\mathcal{I}}$, thus ensuring unlinkability. Moreover, it remains easy to prove well-formedness of the resulting σ_1 due to the use of a dummy public key for which we know the corresponding secret key (in practice we will define the latter value as 1, but any other value would work). We thus get unlinkable signatures of the same size as previously and whose generation only requires few additional computations.

An Unlinkable Redactable Signature. The only differences between our unlinkable scheme `URS` and the one described in the previous section can be found in the `Derive` algorithm. Therefore, we here only describe this algorithm and refer to Sect. 4.1 for the description of the other algorithms that remain unchanged.

– $\texttt{Derive}(\mathsf{pk}, \sigma, \{m_i\}_{i=1}^n, \mathcal{I})$: on input a signature $\sigma = (\widetilde{\sigma}_1, \widetilde{\sigma}_2)$ on $\{m_i\}_{i=1}^n$, the public key pk and a subset $\mathcal{I} \subset [1, n]$, this algorithm generates 2 random scalars $r, t \xleftarrow{\$} \mathbb{Z}_p$ and computes the following elements:
  - $\widetilde{\sigma}'_1 \leftarrow \widetilde{\sigma}_1^r$
  - $\widetilde{\sigma}'_2 \leftarrow \widetilde{\sigma}_2^r \cdot (\widetilde{\sigma}'_1)^t$
  - $\sigma'_1 \leftarrow g^t \prod_{j \in \overline{\mathcal{I}}} Y_j^{m_j}$
  - $\sigma'_2 \leftarrow (\prod_{i \in \mathcal{I}} Y_i)^t \prod_{i \in \mathcal{I}, j \in \overline{\mathcal{I}}} Z_{i,j}^{m_j}$

  where $\overline{\mathcal{I}} = [1, n] \setminus \mathcal{I}$. If $\mathcal{I} = [1, n]$ then $\overline{\mathcal{I}} = \emptyset$ and $(\sigma'_1, \sigma'_2) = (g^t, \prod_{i=1}^n Y_i^t)$.

  In all cases, the signer returns the derived signature $\sigma_I = (\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2)$ on $\{m_i\}_{i \in \mathcal{I}}$.

The resulting derived signature has exactly the same size and the same structure as in the previous scheme `RS`. In particular, it is worthy to note that the verification algorithm remains unchanged and so that an unlinkable signature is also a valid signature for `RS`. Alternatively, we can see the `Derive` algorithm of the previous section as a particular case of this one, where $r = 1$ and $t = 0$.

Regarding the computational cost, we note that generating an unlinkable signature essentially requires 5 additional exponentiations (2 in $\mathbb{G}_1$ and 3 in $\mathbb{G}_2$) compared to the scheme `RS`.

Correctness. Let $\sigma_I = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ be a derived signature on $\{m_i\}_{i \in \mathcal{I}}$ outputted by this new `Derive` algorithm. We then have:

$$\begin{aligned} e(X \cdot \sigma_1 \prod_{i \in \mathcal{I}} Y_i^{m_i}, \widetilde{\sigma}_1) &= e(g^{t+x+\sum_{i=1}^n y_i \cdot m_i}, \widetilde{\sigma}_1) \\ &= e(g, \widetilde{\sigma}_2) \end{aligned}$$

and

$$\begin{aligned} e(\sigma_1, \prod_{i\in\mathcal{I}} \widetilde{Y}_i) &= e(g^t \prod_{j\in\overline{\mathcal{I}}} Y_j^{m_j}, \prod_{i\in\mathcal{I}} \widetilde{Y}_i) \\ &= e((\prod_{i\in\mathcal{I}} Y_i)^t (\prod_{j\in\overline{\mathcal{I}}} Y_j^{m_j})^{\sum_{i\in\mathcal{I}} y_i}, \widetilde{g}) \\ &= e((\prod_{i\in\mathcal{I}} Y_i)^t \prod_{i\in\mathcal{I}, j\in\overline{\mathcal{I}}} Z_{i,j}^{m_j}, \widetilde{g}) \\ &= e(\sigma_2, \widetilde{g}) \end{aligned}$$

which proves correctness of our scheme.

4.3 Security Analysis

The unforgeability of the scheme URS directly relies on the one of RS, proven in the generic group model. Proving strong unforgeability of URS requires to adapt the previous proof, which is done in the full version [22] of this paper. In all cases, we recall that we only consider type 3 pairings in this paper. Regarding unlinkability, we prove that the randomness added to our derived signatures perfectly hide the undisclosed messages and the original signatures. This is formally stated by the following theorem.

Theorem 4. – RS *is an unforgeable redactable signature scheme in the generic group model.*
– URS *is an unforgeable redactable signature scheme if* RS *is unforgeable.*
– URS *is a strongly unforgeable redactable signature scheme in the generic group model.*
– URS *is an unconditionally unlinkable redactable signature scheme.*

Proofs of Unforgeability. We proceed in two steps and first show the unforgeability of the scheme RS described in Sect. 4.1. We next extend this result to the unlinkable construction URS of Sect. 4.2.

Lemma 5. *In the generic group model, no adversary can break the unforgeability of the scheme* RS *with probability greater than* $3(4q_O + q_G + \frac{1+n(n+3)}{2})^2/2p$*, where* q_G *is a bound on the number of group oracle queries and* q_O *is a bound on the number of* $\mathcal{O}\mathtt{Sign}^*$ *queries.*

Proof. The adversary has access to the group elements provided in the public key $\mathsf{pk} = (X, \{(Y_i, \widetilde{Y}_i)\}_{1\leq i\leq n}, \{Z_{i,j}\}_{1\leq i\neq j\leq n})$ and those contained by the signatures $\sigma^{(i)}$ returned by the $\mathcal{O}\mathtt{Sign}^*$ oracle on $(m_{i,1}, \ldots, m_{i,n})$. In the following, each group element is associated with a polynomial whose formal variables are the scalars unknown to the adversary, namely x, $y_1, \ldots, y_n$ and r_i such that $\widetilde{\sigma}_{i,1} = \widetilde{g}^{r_i}$. We must first prove that the adversary is unable to symbolically produce a valid forgery $(\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ for some subset of messages $\{m_i\}_{i\in\mathcal{I}}$.

In the generic group model, the only way for the adversary to generate new group elements is to use the group oracle queries. This means that there are known scalars $(a, b, \{c_i\}_{i=1}^n, \{d_{i,j}\}_{1\leq i\neq j\leq n})$, $(a', b', \{c'_i\}_{i=1}^n, \{d'_{i,j}\}_{1\leq i\neq j\leq n})$, $(\alpha, \{\beta_i\}_{i=1}^n, \{\gamma_i\}_{i=1}^{q_O}, \{\delta_i\}_{i=1}^{q_O})$ and $(\alpha', \{\beta'_i\}_{i=1}^n, \{\gamma'_i\}_{i=1}^{q_O}, \{\delta'_i\}_{i=1}^{q_O})$ such that:

$$\sigma_1 = g^a \cdot X^b \cdot \prod_{i=1}^{n} Y_i^{c_i} \cdot \prod_{1\leq i\neq j\leq n} Z_{i,j}^{d_{i,j}}$$

$$\sigma_2 = g^{a'} \cdot X^{b'} \cdot \prod_{i=1}^{n} Y_i^{c'_i} \cdot \prod_{1\leq i\neq j\leq n} Z_{i,j}^{d'_{i,j}}$$

$$\widetilde{\sigma}_1 = \widetilde{g}^{\alpha} \cdot \prod_{i=1}^{n} \widetilde{Y}_i^{\beta_i} \cdot \prod_{i=1}^{q_O} \widetilde{\sigma}_{i,1}^{\gamma_i} \cdot \prod_{i=1}^{q_O} \widetilde{\sigma}_{i,2}^{\delta_i}$$

$$\widetilde{\sigma}_2 = \widetilde{g}^{\alpha'} \cdot \prod_{i=1}^{n} \widetilde{Y}_i^{\beta'_i} \cdot \prod_{i=1}^{q_O} \widetilde{\sigma}_{i,1}^{\gamma'_i} \cdot \prod_{i=1}^{q_O} \widetilde{\sigma}_{i,2}^{\delta'_i}$$

We do not consider separately the elements $\sigma_{i,1}$ and $\sigma_{i,2}$ because they are public combinations of $\{Y_i\}_{i=1}^n$ and $\{Z_{i,j}\}_{1\leq i\neq j\leq n}$.

Since $(\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ is a valid signature on $\{m_i\}_{i\in\mathcal{I}}$, we know that:

1. $e(X \cdot \sigma_1 \prod_{i\in\mathcal{I}} Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2)$
2. $e(\sigma_1, \prod_{i\in\mathcal{I}} \widetilde{Y}_i) = e(\sigma_2, \widetilde{g})$

Moreover, $(\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ is a valid forgery only if it cannot be trivially derived from the output of the $\mathcal{O}\mathtt{Sign}^*$ oracle. Concretely, this means that, for any $\ell \in [1, q_0]$, there is at least one index $k_\ell \in \mathcal{I}$ such that $m_{\ell,k_\ell} \neq m_{k_\ell}$.

Now, if we consider the second equation we get the following polynomial relation:

$$(a+b\cdot x+\sum_{i=1}^{n} c_i\cdot y_i+\sum_{1\leq i\neq j\leq n} d_{i,j}\cdot y_i\cdot y_j)\sum_{i\in\mathcal{I}} y_i = a'+b'\cdot x+\sum_{i=1}^{n} c'_i\cdot y_i+\sum_{1\leq i\neq j\leq n} d'_{i,j}\cdot y_i\cdot y_j$$

Since $\mathcal{I} \neq \emptyset$, for each monomial of the left member, there is at least an index $i \in [1, n]$ such that the monomial is a multiple of y_i. Therefore me must have $a' = b' = 0$. Moreover, if one of the coefficients $d_{i,j}$ were not zero, then the left member would be of degree 3 whereas the right one would be of degree 2. We can then conclude that $d_{i,j} = 0\ \forall 1 \leq i \neq j \leq n$ and thus get:

$$(a + b \cdot x + \sum_{i=1}^{n} c_i \cdot y_i) \sum_{i\in\mathcal{I}} y_i = \sum_{i=1}^{n} c'_i \cdot y_i + \sum_{1\leq i\neq j\leq n} d'_{i,j} \cdot y_i \cdot y_j$$

We can then note that there is no longer any term in x in the right member, which implies that $b = 0$. Moreover, there is no term in y_i^2 in the right member which means that $c_i = 0$, $\forall i \in \mathcal{I}$. We can therefore conclude that:

$$\sigma_1 = g^a \cdot \prod_{i \in \overline{\mathcal{I}}} Y_i^{c_i}$$

$$\sigma_2 = \prod_{i=1}^{n} Y_i^{c'_i} \cdot \prod_{1 \leq i \neq j \leq n} Z_{i,j}^{d'_{i,j}}$$

Now, let us consider the first equation, which gives the following polynomial relation:

$$(x + a + \sum_{i \in \overline{\mathcal{I}}} c_i \cdot y_i + \sum_{i \in \mathcal{I}} y_i \cdot m_i)(\sum_{i=1}^{q_O} \gamma_i \cdot r_i + \sum_{i=1}^{q_O} \delta_i \cdot r_i(x + \sum_{j=1}^{n} y_j \cdot m_{i,j}) +$$
$$\alpha + \sum_{i=1}^{n} \beta_i \cdot y_i) = \alpha' + \sum_{i=1}^{n} \beta'_i \cdot y_i + \sum_{i=1}^{q_O} \gamma'_i \cdot r_i + \sum_{i=1}^{q_O} \delta'_i \cdot r_i(x + \sum_{j=1}^{n} y_j \cdot m_{i,j})$$

On the left side, there is a unique monomial of the form $\delta_i \cdot r_i \cdot x^2$, $\forall i \in [1, n]$, whereas there is no term in x^2 on the right side. We can then conclude that $\delta_i = 0$, $\forall i \in [1, n]$:

$$(x + a + \sum_{i \in \overline{\mathcal{I}}} c_i \cdot y_i + \sum_{i \in \mathcal{I}} y_i \cdot m_i)(\sum_{i=1}^{q_O} \gamma_i \cdot r_i + \alpha + \sum_{i=1}^{n} \beta_i \cdot y_i)$$
$$= \alpha' + \sum_{i=1}^{n} \beta'_i \cdot y_i + \sum_{i=1}^{q_O} \gamma'_i \cdot r_i + \sum_{i=1}^{q_O} \delta'_i \cdot r_i(x + \sum_{j=1}^{n} y_j \cdot m_{i,j})$$

One can then note that, in the right member, all the monomials of degree 1 in x are also a multiple of some r_i. Therefore, we can conclude that $\alpha = 0$ and that $\beta_i = 0$, $\forall i \in [1, n]$. It then no longer remains any constant term in the left member, which implies that $\alpha' = 0$:

$$(x + a + \sum_{i \in \overline{\mathcal{I}}} c_i \cdot y_i + \sum_{i \in \mathcal{I}} y_i \cdot m_i)(\sum_{i=1}^{q_O} \gamma_i \cdot r_i)$$
$$= \sum_{i=1}^{n} \beta'_i \cdot y_i + \sum_{i=1}^{q_O} \gamma'_i \cdot r_i + \sum_{i=1}^{q_O} \delta'_i \cdot r_i(x + \sum_{j=1}^{n} y_j \cdot m_{i,j})$$

The factor $\sum_{i=1}^{q_O} \gamma_i \cdot r_i$ on the left side implies that all monomials are a multiple of some r_i. This means that $\beta'_i = 0$, $\forall i \in [1, n]$:

$$(x + a + \sum_{i \in \overline{\mathcal{I}}} c_i \cdot y_i + \sum_{i \in \mathcal{I}} y_i \cdot m_i)(\sum_{i=1}^{q_O} \gamma_i \cdot r_i) = \sum_{i=1}^{q_O} \gamma'_i \cdot r_i + \sum_{i=1}^{q_O} \delta'_i \cdot r_i(x + \sum_{j=1}^{n} y_j \cdot m_{i,j})$$

Now, if we consider this relation as an equality between polynomials in the variables r_i, we get, for each $\ell \in [1, q_O]$:

$$(x + a + \sum_{i \in \overline{\mathcal{I}}} c_i \cdot y_i + \sum_{i \in \mathcal{I}} y_i \cdot m_i)\gamma_\ell = \gamma'_\ell + \delta'_\ell(x + \sum_{j=1}^{n} y_j \cdot m_{i,j})$$

However, we know that, for any $\ell \in [1, q_0]$, there is at least one index $k_\ell \in \mathcal{I}$ such that $m_{\ell,k_\ell} \neq m_{k_\ell}$. This implies that $\delta'_\ell = \gamma_\ell = \gamma'_\ell = 0 \ \forall \ell \in [1, q_0]$, which is impossible. The adversary cannot therefore symbolically produce a valid forgery.

It remains to assess the probability of an accidental validity, when two different polynomials evaluate to the same value. All the polynomials considered in this proof are of degree at most 3. Since there are at most $(4q_O + q_G + \frac{1+n(n+3)}{2})$ polynomials, the probability of an accidental validity is bounded by $3(4q_O + q_G + \frac{1+n(n+3)}{2})^2/2p$ according to the Schwartz-Zippel lemma, which is negligible. □

Our next lemma shows that the unforgeability of RS implies the one of our unlinkable scheme URS from Sect. 4.2.

Lemma 6. *Any adversary $\mathcal{A}$ against the unforgeability of our unlinkable scheme* URS *can be converted into an adversary against the unforgeability of* RS*, succeeding with the same probability.*

Proof. Our reduction $\mathcal{R}$ uses $\mathcal{A}$, an adversary against the unforgeability of URS, to break the unforgeability of RS. There will be then two unforgeability games. To avoid any confusion, we will refer to the unforgeability game of our basic scheme as the "RS game" and to the one of our unlinkable scheme as the "URS game".

$\mathcal{R}$ starts the RS game and then obtains a public key pk that it forwards to $\mathcal{A}$. When it receives a $\mathcal{O}\mathtt{Sign}^*$ query, it simply forwards it to the corresponding oracle of the RS game and then receives a valid signature $\sigma = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ for RS. It then selects two random scalars r and t and computes:

- $\sigma'_1 = \sigma_1 \cdot g^t$;
- $\sigma'_2 = \sigma_2 \cdot (\prod_{i \in \mathcal{I}} Y_i)^t$;
- $\widetilde{\sigma}'_1 = \widetilde{\sigma}_1^r$;
- $\widetilde{\sigma}'_2 = \widetilde{\sigma}_2^r \cdot (\widetilde{\sigma}'_1)^t$.

Finally, it returns $\sigma = (\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2)$ to the adversary $\mathcal{A}$.

The fact that $\mathcal{R}$ forwards each query to the oracles of the RS game implies that the sets of messages stored in Q_1 are exactly the same for both games. Since the oracle of the URS game is perfectly simulated, the adversary eventually outputs a forgery which is a valid derived signature σ^* for URS. Since RS and URS have the same verification algorithm, σ^* is also a valid signature for RS. Moreover, our previous remark on Q_1 means that σ^* is also a valid forgery for the RS game. $\mathcal{R}$ then never fails when $\mathcal{A}$ succeeds, which concludes the proof.

Proof of Unlinkability. We prove here that a signature $\sigma_{\mathcal{I}}$ on $\{m_i\}_{i \in \mathcal{I}}$ derived from an original signature σ on $\{m_i\}_{i=1}^n$ is distributed independently of σ and $\{m_i\}_{i \in \overline{\mathcal{I}}}$. Since the messages output by the adversary in the unlinkability game satisfy $m_i^{(0)} = m_i^{(1)}$, $\forall i \in \mathcal{I}$, this means that the advantage of the adversary can only be negligible in this game.

Concretely, let $\widetilde{\tau}$ be a random element of $\mathbb{G}_2$ and u be a random scalar. For a signature $\sigma = (\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ on $\{m_i\}_{i=1}^n$ and any subset $\mathcal{I} \subset [1, n]$, we

define $t = u - \sum_{i\in\overline{\mathcal{I}}} y_i \cdot m_i$ and $r = \frac{v}{s}$, where v and s are such that $\widetilde{\tau} = \widetilde{g}^v$ and $\widetilde{\sigma}_1 = \widetilde{g}^s$. Since u and $\widetilde{\tau}$ are random, r and t are also random and so are distributed as specified in the `Derive` algorithm. Running the latter algorithm on $(\sigma, \{m_i\}_{i=1}^n, \mathcal{I})$ with these values would then lead to the derived signature $\sigma_I = (\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2)$ with:

- $\widetilde{\sigma}'_1 = \widetilde{\sigma}_1^r = \widetilde{\tau}$
- $\widetilde{\sigma}'_2 = \widetilde{\sigma}_2^r \cdot (\widetilde{\sigma}'_1)^t = \widetilde{\tau}^{x+\sum_{i\in\mathcal{I}} y_i \cdot m_i} \cdot \widetilde{\tau}^u$
- $\sigma'_1 = g^t \prod_{j\in\overline{\mathcal{I}}} Y_j^{m_j} = g^u$
- $\sigma'_2 = (\prod_{i\in\mathcal{I}} Y_i)^t \prod_{i\in\mathcal{I}, j\in\overline{\mathcal{I}}} Z_{i,j}^{m_j} = g^{u\cdot\sum_{i\in\mathcal{I}} y_i}$

Since u and $\widetilde{\tau}$ are random, the derived signature σ_I is clearly independent of the original signature and of the messages $\{m_i\}_{i\in\overline{\mathcal{I}}}$, which concludes the proof.

5 Anonymous Credentials

Anonymous credential (also called attribute-based credential) is a broad notion that usually encompasses any system that allows some organization to issue a credential on users' attributes such that (1) the users can later prove that their attributes are certified and (2) the elements revealed by the users when they *show* their credential cannot be linked to a specific issuance (unless the revealed attributes trivially allow to do so).

However, there is no unique, commonly accepted definition of anonymous credentials, but rather several variants of the same intuitive notion. For example, some definitions [5,14] assume that the credential are only shown once, whereas others support multiple (and unlinkable) showing of a credential [10,15,21]. We follow in this section the definition from [15] that consider multiple, interactive showings.

5.1 Syntax

An anonymous credentials system is defined by the following algorithms.

- $\mathtt{OrgKeygen}(1^k, n)$: This algorithm takes as input a security parameter 1^k and an integer n defining a bound on the number of attributes to certify and returns the organization key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathtt{UserKeygen}(\mathsf{pk})$: This algorithm returns a user's key pair $(\mathsf{usk}, \mathsf{upk})$ from the organization public key pk.
- $(\mathtt{Obtain}(\mathsf{usk}, \mathsf{pk}, \{m_i\}_{i=1}^n), \mathtt{Issue}(\mathsf{upk}, \mathsf{sk}, \{m_i\}_{i=1}^n)$: To obtain an anonymous credential on a set of attributes $\{m_i\}_{i=1}^n$, the user, running `Obtain`, interacts with the organization, running `Issue`. The former algorithm additionally requires the user's secret key usk and the organization public key pk whereas the latter requires upk and sk. At the end of the protocol, `Obtain` returns either a credential σ or $\perp$.

– $(\mathtt{Show}(\mathsf{pk}, \mathsf{usk}, \{m_i\}_{i=1}^n, \mathcal{I}, \sigma), \mathtt{Verify}(\mathsf{pk}, \{m_i\}_{i\in\mathcal{I}}))$: These algorithms are run by a user and a verifier, respectively, who interact during execution. `Show` enables the user to prove that a subset $\{m_i\}_{i\in\mathcal{I}}$ of his attributes, with $\mathcal{I} \subset [1, n]$, has been certified. It takes as input the credential σ, the organization public key pk, the whole set of attributes $\{m_i\}_{i=1}^n$ along with the intended subset $\mathcal{I}$. The `Verify` algorithm only takes as input pk and the subset $\{m_i\}_{i\in\mathcal{I}}$ and returns either 1 (accept) or 0 (reject).

5.2 Security Model

The security model considered here is the one from [15], that we slightly modify to harmonize this section with the one on redactable signature (Sect. 3).

Besides correctness, an anonymous credentials system must achieve *unforgeability* and *anonymity* that essentially mirror the unforgeability and unlinkability notions for redactable signatures. As in Sect. 3, we define these properties by the experiments described in Fig. 2 that use the following oracles along with two sets: `HU`, the set containing the identities of *honest* users and `CU`, that contains the ones of *corrupt* users. We additionally define the set `Att` that stores $\{i, \{m_j\}_{j=1}^n\}$ each time a credential is generated for user i on $\{m_j\}_{j=1}^n$ by the oracles $\mathcal{O}\mathtt{ObtIss}$ and $\mathcal{O}\mathtt{Issue}$ below. We say that $\{i, \{m_j\}_{j\in\mathcal{I}}\} \subset \mathtt{Att}$ if $\exists\{i, \{m'_j\}_{j=1}^n\} \in \mathtt{Att}$ with $m'_j = m_j$ for all $j \in \mathcal{I}$.

– $\mathcal{O}\mathtt{HU}(i)$: on input an identity i, this oracle returns $\perp$ if $i \in \mathtt{HU} \cup \mathtt{CU}$. Else it generates a key pair $(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \mathtt{UserKeygen}(\mathsf{pk})$ and returns upk_i. The identity i is then added to `HU`.
– $\mathcal{O}\mathtt{CU}(i, \mathsf{upk})$: on input an identity i and optionally a public key upk, this oracle registers a new corrupt user with public key upk if $i \notin \mathtt{HU}$ and returns usk_i and all the associated credentials otherwise. In the latter case, i is removed from `HU`. In all cases, i is added to `CU`.
– $\mathcal{O}\mathtt{ObtIss}(i, \{m_j\}_{j=1}^n)$: on input an identity $i \in \mathtt{HU}$ and a set of attributes $\{m_j\}_{j=1}^n$, this oracle runs $(\mathtt{Obtain}(\mathsf{usk}_i, \mathsf{pk}, \{m_j\}_{j=1}^n), \mathtt{Issue}(\mathsf{upk}_i, \mathsf{sk}, \{m_j\}_{j=1}^n)$ and stores the resulting output. The elements $\{i, \{m_j\}_{j=1}^n\}$ are then added to `Att`. If $i \notin \mathtt{HU}$, the oracle returns $\perp$.
– $\mathcal{O}\mathtt{Obtain}(i, \{m_j\}_{j=1}^n)$: on input an identity $i \in \mathtt{HU}$ and a set of attributes $\{m_j\}_{j=1}^n$, this oracle runs $\mathtt{Obtain}(\mathsf{usk}_i, \mathsf{pk}, \{m_j\}_{j=1}^n)$ and stores the resulting output. If $i \notin \mathtt{HU}$, the oracle returns $\perp$. This oracle is used by an adversary impersonating the organization to issue a credential to an honest user.
– $\mathcal{O}\mathtt{Issue}(i, \{m_j\}_{j=1}^n)$: on input an identity $i \in \mathtt{CU}$ and a set of attributes $\{m_j\}_{j=1}^n$, this oracle runs $\mathtt{Issue}(\mathsf{upk}_i, \mathsf{sk}, \{m_j\}_{j=1}^n)$. The elements $\{i, \{m_j\}_{j=1}^n\}$ are then added to `Att`. If $i \notin \mathtt{CU}$, the oracle returns $\perp$. This oracle is used by an adversary playing a malicious user to get a certificate from an honest organization.
– $\mathcal{O}\mathtt{Show}(k, \mathcal{I})$: Let $\sigma^{(k)}$ be the credential issued on $\{m_j^{(k)}\}_{j=1}^n$ for a user i_k during the k-th query to $\mathcal{O}\mathtt{ObtIss}$ or $\mathcal{O}\mathtt{Obtain}$. If $i_k \notin \mathtt{HU}$, this oracle returns $\perp$. Else, this oracle runs $\mathtt{Show}(\mathsf{pk}, \mathsf{usk}_{i_k}, \{m_j^{(k)}\}_{j=1}^n, \mathcal{I}, \sigma^{(k)})$ with the adversary playing a malicious verifier.

Unforgeability
$\mathtt{Exp}^{uf}_{\mathcal{A}}(1^k, n)$

1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Keygen}(1^k, n)$
2. $\{m_j\}_{j\in\mathcal{I}} \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{HU},\mathcal{O}\mathtt{CU},\mathcal{O}\mathtt{ObtIss},\mathcal{O}\mathtt{Issue},\mathcal{O}\mathtt{Show}}(\mathsf{pk})$
3. $b \leftarrow (\mathcal{A}(), \mathtt{Verify}(\mathsf{pk}, \{m_j\}_{j\in\mathcal{I}}))$
4. If $\{i, \{m_j\}_{j\in\mathcal{I}}\} \subset \mathtt{Att}$ with $i \in \mathtt{CU}$ or if $b = 0$, return 0
5. Return 1.

Anonymity
$\mathtt{Exp}^{ano-b}_{\mathcal{A}}(1^k, n)$

1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{Keygen}(1^k, n)$
2. $(j_0, j_1, \{m_i\}_{i\in\mathcal{I}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{HU},\mathcal{O}\mathtt{CU},\mathcal{O}\mathtt{Obtain},\mathcal{O}\mathtt{Show}}(\mathsf{sk})$
3. If $\{j_{b'}, \{m_i\}_{i\in\mathcal{I}}\} \not\subset \mathtt{Att}$ for $b' \in \{0, 1\}$, return 0
4. $(\mathtt{Show}(\mathsf{pk}, \mathsf{usk}_{j_b}, \{m_i^{(j_b)}\}_{j=1}^n, \mathcal{I}, \sigma^{(k)}), \mathcal{A}())$
5. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{HU},\mathcal{O}\mathtt{CU},\mathcal{O}\mathtt{Obtain},\mathcal{O}\mathtt{Show}}(\mathsf{sk})$
6. If $\mathcal{O}\mathtt{CU}$ has been queried on $j_{b'}$ for $b' \in \{0, 1\}$, return 0
7. Return b^*.

Fig. 2. Security notions for anonymous credentials

Correctness. A showing of a credential σ with respect to a set $\{m_i\}_{i\in\mathcal{I}}$ always verify if σ was honestly issued on $\{m_i\}_{i=1}^n$, with $\mathcal{I} \in [1, n]$.

Unforgeability. A credential system is *unforgeable* if $\mathtt{Adv}^{uf}(\mathcal{A}) = |\Pr[\mathtt{Exp}^{uf}_{\mathcal{A}}(1^k, n) = 1]|$ is negligible for any polynomial time adversary $\mathcal{A}$.

Anonymity. The anonymity property is defined by the $\mathtt{Exp}^{ano-b}_{\mathcal{A}}$ experiment in Fig. 2, for $b \in \{0, 1\}$. A credential system is *anonymous* if $\mathtt{Adv}^{ano} = |\Pr[\mathtt{Exp}^{ano-1}_{\mathcal{A}}(1^k, n) = 1] - \Pr[\mathtt{Exp}^{ano-0}_{\mathcal{A}}(1^k, n) = 1]|$ is negligible for any polynomial time adversary $\mathcal{A}$.

Our definition assumes that the organization key pair $(\mathsf{sk}, \mathsf{pk})$ is honestly generated and then sent to the adversary, contrarily to [15] that lets the adversary generates its own key pair. This modification indeed allows us to reduce the size of the public key pk in our next construction. Nevertheless, we stress that the latter can satisfy the original definition from [15] if we add a non-interactive zero-knowledge proof of knowledge of sk in pk.

6 Our Anonymous Credentials System

As noticed in [7,15], an unlinkable redactable signature scheme is very similar to an anonymous credentials system [8], also called attribute-based credentials system. Indeed, it can be used to prove that some data have been certified without being traced, while hiding (redacting) all the other signed data. To achieve all the properties expected from an anonymous credentials system, it

thus essentially lacks the ability to issue credentials on the user's secret key and then to present the credentials with respect to this key.

In this paper we use the definition of anonymous credentials provided in [15] and thus consider an interactive presentation protocol. However, the latter can easily be made non interactive by using the Fiat-Shamir heuristic [13] on the proof of knowledge that it contains.

6.1 Our Construction

In our system, the user's secret key usk is simply a random scalar that defines the public key upk as $\widetilde{g}^{\mathsf{usk}}$. Using the protocols described in [21], that we slightly modify, the user is able to get a redactable signature σ on usk and a set of attributes $\{m_i\}_{i=1}^n$ without revealing usk. Such a signature σ then acts as a credential for this user. To show a credential on some attributes $\{m_i\}_{i\in\mathcal{I}}$, the user essentially runs the $\mathtt{Derive}$ algorithm on σ and $\{\mathsf{usk}\} \cup \{m_i\}_{i\in\mathcal{I}}$ and then prove knowledge of usk.

Our construction can thus be seen as an interactive version of our URS scheme supporting proofs of knowledge of secret attributes. However, such modifications make the security proofs more intricate. In particular, anonymity no longer holds unconditionally, but under the $\mathtt{DDH}$ assumption in $\mathbb{G}_2$. Intuitively, this is due to the fact that usk must be kept secret but cannot either be aggregated to the set of undisclosed messages. Therefore, the distribution of derived signatures can no longer be made independent of usk and thus we cannot rely on the same arguments as those used in the security proof of Sect. 4.3.

- $\mathtt{OrgKeygen}(1^k, n)$: On input a security parameter 1^k and an integer n, this algorithm generates $(n+2)$ random scalars $x, y_0, y_1, \ldots, y_n \xleftarrow{\$} \mathbb{Z}_p$ and computes the following elements:
  - $X \leftarrow g^x$
  - $Y_i \leftarrow g^{y_i}$, $\forall 0 \leq i \leq n$
  - $\widetilde{Y}_i \leftarrow \widetilde{g}^{y_i}$, $\forall 0 \leq i \leq n$
  - $Z_{i,j} \leftarrow g^{y_i \cdot y_j}$, $\forall 0 \leq i \neq j \leq n$

  The secret key sk is then $(x, y_0, y_1, \ldots, y_n)$ whereas the public key pk is $(X, \{(Y_i, \widetilde{Y}_i)\}_{0\leq i\leq n}, \{Z_{i,j}\}_{0\leq i\neq j\leq n})$
- $\mathtt{UserKeygen}(\mathsf{pk})$: To generate a key pair $(\mathsf{usk}, \mathsf{upk})$ for a user, this algorithm selects a random $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p$ and computes $\mathsf{upk} \leftarrow \widetilde{g}^{\mathsf{usk}}$.
- $(\mathtt{Obtain}(\mathsf{usk}, \mathsf{pk}, \{m_i\}_{i=1}^n), \mathtt{Issue}(\mathsf{upk}, \mathsf{sk}, \{m_i\}_{i=1}^n))$: To obtain an anonymous credential on a set of attributes $\{m_i\}_{i=1}^n$, the user first sends her public key upk along with a proof of knowledge of usk, using for example the Schnorr's protocol [23]. If the proof is correct, then the organization selects a random $r \xleftarrow{\$} \mathbb{Z}_p$ and returns $\sigma = (\widetilde{\sigma}_1, \widetilde{\sigma}_2) \leftarrow (\widetilde{g}^r, \mathsf{upk}^{r\cdot y_0} \cdot \widetilde{g}^{r(x+\sum_{i=1}^n y_i \cdot m_i)})$ to the user.
- $(\mathtt{Show}(\mathsf{pk}, \mathsf{usk}, \{m_i\}_{i=1}^n, \mathcal{I}, \sigma), \mathtt{Verify}(\mathsf{pk}, \{m_i\}_{i\in\mathcal{I}}))$: For $\mathcal{I} \subset [1, n]$, we define $\mathcal{I}_0 = \{0\} \cup \mathcal{I}$. The protocol to show a credential on a subset $\{m_i\}_{i\in\mathcal{I}}$ is described in Fig. 3.

$\text{User}(\mathsf{pk}, \mathsf{usk}, \{m_i\}_{i=1}^n, \mathcal{I}, \sigma)$		$\text{Verifier}(\mathsf{pk}, \{m_i\}_{i\in\mathcal{I}})$
$k, r, t \xleftarrow{\$} \mathbb{Z}_p$		
$\sigma'_1 \leftarrow g^t \cdot \prod_{j\in[1,n]\setminus\mathcal{I}} Y_j^{m_j}$		
$\sigma'_2 \leftarrow (\prod_{i\in\mathcal{I}_0} Y_i)^t \cdot \prod_{i\in\mathcal{I}_0, j\in[1,n]\setminus\mathcal{I}} Z_{i,j}^{m_j}$		
$\widetilde{\sigma}'_1 \leftarrow \widetilde{\sigma}_1^r$		
$\widetilde{\sigma}'_2 \leftarrow \widetilde{\sigma}_2^r \cdot (\widetilde{\sigma}'_1)^t$		
$C \leftarrow e(Y_0^k, \widetilde{\sigma}'_1)$	$\xrightarrow{(\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2), C}$	If $(\widetilde{\sigma}'_1, \widetilde{\sigma}'_2) \notin (\mathbb{G}_2^*)^2$, return 0.
		Else, $c \xleftarrow{\$} \mathbb{Z}_p$
	$\xleftarrow{c}$	$B = e(X \cdot \sigma'_1 \prod_{i\in\mathcal{I}} Y_i^{m_i}, (\widetilde{\sigma}'_1)^{-1})$
$s = k + c \cdot \mathsf{usk}$	$\xrightarrow{s}$	If $e(Y_0^s, \widetilde{\sigma}'_1) \cdot C^{-1} = [B \cdot e(g, \widetilde{\sigma}'_2)]^c$
		and $e(\sigma'_1, \prod_{i\in\mathcal{I}_0} \widetilde{Y}_i) = e(\sigma'_2, \widetilde{g})$
		return 1
		Else, return 0

Fig. 3. A protocol to show a credential σ on a subset $\{m_i\}_{i\in\mathcal{I}}$

Correctness. For a valid credential σ issued on $(m_1, \ldots, m_n)$ and usk, we have:

$$e(X \cdot Y_0^{\mathsf{usk}} \cdot \prod_{i=1}^n Y_i^{m_i}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2)$$

which is equivalent to:

$$e(Y_0^{\mathsf{usk}}, \widetilde{\sigma}_1) = e(g, \widetilde{\sigma}_2) \cdot e(X \cdot \prod_{i=1}^n Y_i^{m_i}, \widetilde{\sigma}_1)^{-1}$$

Therefore:

$$\begin{aligned}
e(Y_0^s, \widetilde{\sigma}'_1) \cdot C^{-1} &= e(Y_0^{\mathsf{usk}}, \widetilde{\sigma}_1)^{r\cdot c} \\
&= [e(g, \widetilde{\sigma}_2) \cdot e(X \cdot \prod_{i=1}^n Y_i^{m_i}, \widetilde{\sigma}_1)^{-1}]^{r\cdot c} \\
&= [e(g, \widetilde{\sigma}'_2) \cdot e(g, \widetilde{\sigma}'_1)^{-t} \cdot e(X \cdot \prod_{i=1}^n Y_i^{m_i}, (\widetilde{\sigma}'_1)^{-1})]^c \\
&= [e(g, \widetilde{\sigma}'_2) \cdot e(X \cdot g^t \cdot \prod_{i\in\mathcal{I}} Y_i^{m_i} \cdot \prod_{i\in[1,n]\setminus\mathcal{I}} Y_i^{m_i}, (\widetilde{\sigma}'_1)^{-1})]^c \\
&= [e(g, \widetilde{\sigma}'_2) \cdot e(X \cdot \sigma'_1 \cdot \prod_{i\in\mathcal{I}} Y_i^{m_i}, (\widetilde{\sigma}'_1)^{-1})]^c \\
&= [e(g, \widetilde{\sigma}'_2) \cdot B]^c
\end{aligned}$$

and

$$\begin{aligned} e(\sigma'_1, \prod_{i\in\mathcal{I}_0} \widetilde{Y}_i) &= e(\prod_{j\in[1,n]\setminus\mathcal{I}} Y_j^{m_j}, \prod_{i\in\mathcal{I}_0} \widetilde{Y}_i) \cdot e(g^t, \prod_{i\in\mathcal{I}_0} \widetilde{Y}_i) \\ &= e((\prod_{j\in[1,n]\setminus\mathcal{I}} Y_j^{m_j})^{\sum_{i\in\mathcal{I}_0} y_i}, \widetilde{g}) \cdot e((\prod_{i\in\mathcal{I}_0} Y_i)^t, \widetilde{g}) \\ &= e(\sigma'_2, \widetilde{g}) \end{aligned}$$

which implies correctness of our protocol.

Proving Knowledge of Attributes. As we have explained, our `Show` protocol essentially consists in deriving a signature on $\mathsf{usk} \cup \{m_i\}_{i\in\mathcal{I}}$ and then proving knowledge of usk. The latter proof is very easy to produce using Schnorr's protocol because usk is an exponent in the verification equation. We note that this is also true for every attribute m_i such that $i \in \mathcal{I}$. Therefore, the protocol of Fig. 3 can easily be extended to hide and prove knowledge of the attributes $\{m_j\}_{j\in\mathcal{J}}$, for any subset $\mathcal{J} \subset \mathcal{I}$.

6.2 Security Analysis

The structure of our `Show` protocol makes the unforgeability proof rather straightforward: if an adversary is able to prove possession of a credential on a set of attributes that it does not own, then it is able to produce a valid forgery for our `URS` system or to impersonate an honest user. Since our protocol requires that the users prove knowledge of their secret key, the latter case implies an attack against the discrete logarithm. Proving anonymity of our credential system is more subtle as we cannot simply rely on the unlinkability of `URS`.

Theorem 7. – *Our credential system is unforgeable if* `URS` *is unforgeable and if the* DL *assumption holds in* $\mathbb{G}_2$.
– *Our credential system is anonymous under the* DDH *assumption in* $\mathbb{G}_2$.

Proof of Unforgeability. Let $\mathcal{A}$ be an adversary against the unforgeability of our anonymous credentials system. During the game, $\mathcal{A}$ returns a set of attributes $\{m_i\}_{i\in\mathcal{I}}$ and then proves possession of a credential on this set. Obviously, the credentials issued by oracles to corrupt users cannot be valid on $\{m_i\}_{i\in\mathcal{I}}$. However, honest users could possess a credential on such attributes, which leads to consider two different cases in our proof. Let usk be the secret key whose knowledge is proved by the adversary when it shows the credential on $\{m_i\}_{i\in\mathcal{I}}$, we distinguish two types of adversary:

– Type 1: $\exists i \in \mathtt{HU}$ such that $\mathsf{usk}_i = \mathsf{usk}$
– Type 2: $\forall i \in \mathtt{HU}$, $\mathsf{usk}_i \neq \mathsf{usk}$.

Lemma 8. *Any type 1 adversary $\mathcal{A}$ succeeding with probability ϵ can be converted into an adversary against the discrete logarithm assumption in $\mathbb{G}_2$ succeeding with probability $\frac{\epsilon}{q}$, where q is a bound on the number of honest users.*

Proof. Let $(\widetilde{g}, \widetilde{g}^a)$ be a DL challenge. Our reduction $\mathcal{R}$ generates the organisation key pair using $\widetilde{g}$ as the generator for $\mathbb{G}_2$ and returns pk to $\mathcal{A}$. Since we consider a type 1 adversary, we know that there is an index i such that $\mathcal{A}$ will try to impersonate the i-th honest user. Our reduction $\mathcal{R}$ then makes a guess on $i \in [1, q]$ and proceeds as follows.

- $\mathcal{O}\mathtt{HU}$: Let j be the index query to this oracle. If $j \neq i$, then $\mathcal{R}$ proceeds as usual. Else, it returns $\mathsf{upk}_i = \widetilde{g}^a$.
- $\mathcal{O}\mathtt{CU}$: If $\mathcal{R}$ receives a corruption query on an honest user j, it returns usk_j if $j \neq i$ and aborts otherwise.
- $\mathcal{O}\mathtt{ObtIss}$: $\mathcal{R}$ knows the organization secret sk and so perfectly simulates the organization's side of this protocol. It can also play the role of any honest user j if $j \neq i$. Else, it simulates the proof of knowledge of usk_i.
- $\mathcal{O}\mathtt{Issue}$: $\mathcal{R}$ knows sk and so is perfectly able to answer any query.
- $\mathcal{O}\mathtt{Show}$: If the queried credential belongs to $j \neq i$, then $\mathcal{R}$ is able to run the Show protocol defined in Fig. 3. Else, it runs the first steps of the protocols but simulates the knowledge of usk_i.

One can note that the game is perfectly simulated if the guess on i is correct, which occurs with probability $\frac{1}{q}$. In such a case, a successful adversary $\mathcal{A}$ proves knowledge of $\mathsf{usk}_i = a$ when it shows its credential. $\mathcal{R}$ can then run the extractor of the proof of knowledge to recover a, that it returns as a valid solution to the DL problem. The probability of success of $\mathcal{R}$ is then $\frac{\epsilon}{q}$. □

Lemma 9. *Any type 2 adversary $\mathcal{A}$ can be converted into an adversary against the unforgeability of the* URS *scheme succeeding with the same probability.*

Proof. Our reduction $\mathcal{R}$ runs the unforgeability game of the URS scheme for the parameter $n + 1$ and so receives a public key $(X, \{(Y_i, \widetilde{Y}_i)\}_{1 \leq i \leq n+1}, \{Z_{i,j}\}_{1 \leq i \neq j \leq n+1})$. $\mathcal{R}$ changes the indices of the elements of the public key, starting from 0 instead of 1, and then returns $\mathsf{pk} = (X, \{(Y_i, \widetilde{Y}_i)\}_{0 \leq i \leq n}, \{Z_{i,j}\}_{0 \leq i \neq j \leq n})$ to $\mathcal{A}$. It can then answers oracle queries as follows.

- $\mathcal{O}\mathtt{HU}$: $\mathcal{R}$ proceeds as usual, and stores the corresponding secret key.
- $\mathcal{O}\mathtt{CU}$: Here again, $\mathcal{R}$ proceeds as usual.
- $\mathcal{O}\mathtt{ObtIss}$: Let $i \in \mathtt{HU}$ and $\{m_i\}_{i=1}^n$ be the input of this oracle. The reduction recovers the secret key usk_i that it has generated for user i and then submits $(\mathsf{usk}_i, m_1, \ldots, m_n)$ to the signing oracle $\mathcal{O}\mathtt{Sign}^*$. It then receives a URS signature $(\sigma_1, \sigma_2, \widetilde{\sigma}_1, \widetilde{\sigma}_2)$ whose first two elements are $1_{\mathbb{G}_1}$. $\mathcal{R}$ then discards σ_1 and σ_2 and stores the resulting credential $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$.
- $\mathcal{O}\mathtt{Issue}$: Let $i \in \mathtt{CU}$ and $\{m_i\}_{i=1}^n$ be the input of this oracle. $\mathcal{R}$ extracts usk_i from the proof of knowledge produced by $\mathcal{A}$ and then proceeds as previously to get a URS signature on $(\mathsf{usk}_i, m_1, \ldots, m_n)$. Here again, the new credential is defined as $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$.

– $\mathcal{O}\texttt{Show}$: Let i and $\{m_i\}_{i\in\mathcal{I}}$ be the inputs of this oracle. A show query can only be made for a credential that has been issued through the $\mathcal{O}\texttt{ObtIss}$ oracle. Since the latter oracle uses the $\mathcal{O}\texttt{Sign}^*$ oracle of the unforgeability game of the URS scheme, there is a corresponding signature σ on $(\mathsf{usk}_i, m_1, \ldots, m_n)$ in the table Q_1. $\mathcal{R}$ can then run the Derive algorithm on σ and $\{\mathsf{usk}_i\}\cup\{m_i\}_{i\in\mathcal{I}}$ and gets $(\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2)$ such that:
  - $\widetilde{\sigma}'_2 = (\widetilde{\sigma}'_1)^{t+x+y_0\cdot\mathsf{usk}_i+\sum_{i=1}^{n} y_i\cdot m_i}$
  - $\sigma'_1 \leftarrow g^t \cdot \prod_{j\in[1,n]\setminus\mathcal{I}} Y_j^{m_j}$
  - $\sigma'_2 \leftarrow (\prod_{i\in\mathcal{I}_0} Y_i)^t \cdot \prod_{i\in\mathcal{I}_0, j\in[1,n]\setminus\mathcal{I}} Z_{i,j}^{m_j}$

  The elements σ'_1, σ'_2, $\widetilde{\sigma}'_1$ and $\widetilde{\sigma}'_2$ are therefore distributed as in the Show protocol of Fig. 3. It then only remains to compute $C = e(Y_0^k, \widetilde{\sigma}'_1)$ for some random k and to return a valid s using usk_i.

$\mathcal{R}$ can handle any oracle query and never aborts. Therefore, at the end of the game, $\mathcal{A}$ is able, with some probability ϵ, to prove possession of a credential on $\{m_i\}_{i\in\mathcal{I}}$. Our reduction extracts from the proof of knowledge contained in the Show protocol the value usk and stores the elements σ'_1, σ'_2, $\widetilde{\sigma}'_1$ and $\widetilde{\sigma}'_2$. The latter constitute a valid derived signature on $\{\mathsf{usk}\} \cup \{m_i\}_{i\in\mathcal{I}}$.

Since we here consider a type 2 adversary, usk must be different from usk_i, for any honest user i. Moreover, to be considered as an attack against unforgeability, no credential owned by corrupt users can be valid on this set of messages. This means that, for any credential on $(\mathsf{usk}_i, m'_1, \ldots, m'_n)$ with $i \in \texttt{CU}$, we have either $\mathsf{usk} \neq \mathsf{usk}_i$ or $\exists j \in \mathcal{I}$ such that $m_j \neq m'_j$. In all cases, this means that $\sigma = (\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2)$ and $\{\mathsf{usk}\} \cup \{m_i\}_{i\in\mathcal{I}}$ is a valid forgery against our URS scheme, which concludes our proof. □

Proof of Anonymity. Let $(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c)$ be a DDH challenge in $\mathbb{G}_2$. We construct a reduction $\mathcal{R}$ that uses $\mathcal{A}$, an adversary succeeding against the anonymity of our credential system with advantage ϵ, to decide whether $c = a \cdot b$.

At the beginning of the game $\texttt{Exp}_{\mathcal{A}}^{ano-b}$, $\mathcal{R}$ generates the organization key pair $(\mathsf{sk}, \mathsf{pk})$ and forwards it to $\mathcal{A}$ that eventually returns $(j_0, j_1, \{m_i\}_{i\in\mathcal{I}})$. $\mathcal{R}$ then makes a guess on the identity of the user i_b that will possess the credential $\sigma^{(j_b)}$ targeted by $\mathcal{A}$ and answers the oracle queries as follows.

– $\mathcal{O}\texttt{HU}$: Let j be the identity submitted to this oracle. If $j \neq i_b$, then $\mathcal{R}$ proceeds as usual. Else, it returns $\widetilde{g}^a$ as the public key upk_{i_b} of user i_b.
– $\mathcal{O}\texttt{CU}$: $\mathcal{R}$ proceeds as usual, unless this oracle is queried on i_b, in which case $\mathcal{R}$ aborts.
– $\mathcal{O}\texttt{Obtain}$: For any $j \neq i_b$, $\mathcal{R}$ knows usk_j and so is able to run the Obtain protocol as usual. If $j = i_b$, then $\mathcal{R}$ sends the public key upk_{i_b} and simulates the proof of knowledge of a.
– $\mathcal{O}\texttt{Show}$: Here again, $\mathcal{R}$ proceeds as usual or by simulating the proof of knowledge of the secret key if the credential belongs to i_b.

At some point in the game, the adversary outputs the identifiers j_0 and j_1 of two users, along with a set of attributes $\{m_i\}_{i\in\mathcal{I}}$. If $j_b \neq i_b$, then $\mathcal{R}$ aborts. Else, it proceeds as follows.

$\mathcal{R}$ first selects two random scalars k and α and sets $\widetilde{\sigma}'_1 = g^b$. It then computes:

- $\widetilde{\sigma}'_2 \leftarrow (\widetilde{\sigma}'_1)^{\alpha+x+\sum_{i\in\mathcal{I}} y_i\cdot m_i} \cdot (g^c)^{y_0}$
- $\sigma'_1 = g^\alpha$
- $\sigma'_2 = (\sigma'_1)^{\sum_{i\in\mathcal{I}_0} y_i}$

and simulates knowledge of a.

If $c = a \cdot b$, then, by setting $t = \alpha - \sum_{i\in[1,n]\setminus\mathcal{I}} y_i \cdot m_i$, one can see that $(\sigma'_1, \sigma'_2, \widetilde{\sigma}'_1, \widetilde{\sigma}'_2)$ is distributed as in the protocol of Fig. 3. Else, c is random, which means that $\widetilde{\sigma}'_2$ is a random element of $\mathbb{G}_2$. Since σ'_1, σ'_2 and $\widetilde{\sigma}'_1$ are independent of a and $\{m_i\}_{i\in[1,n]\setminus\mathcal{I}}$, $\mathcal{A}$ cannot succeed in this game with non negligible advantage. Therefore any change in the behaviour of $\mathcal{A}$ can be used to solve the DDH problem in $\mathbb{G}_2$, unless $\mathcal{R}$ aborts. The advantage of $\mathcal{R}$ is then at least $\frac{\epsilon}{q}$, where q is a bound on the number of honest users.

7 Efficiency

We describe in this section the complexity of the redactable signature schemes RS and URS before comparing the one of our anonymous credentials system with the most relevant systems of the state-of-the-art.

Redactable Signatures. Table 1 provides the most important figures regarding the size and computational complexity of the schemes RS and URS. For sake of clarity, we only consider the most expensive operations, such as exponentiations and pairings, and do not take into account the other ones. As in Remark 3, we define the subset vk of the elements of the public key pk that are necessary to verify signatures. Our efficiency analysis is based on the descriptions of the schemes from Sects. 4.1 and 4.2 that aim at minimizing the complexity in $\mathbb{G}_2$, where operations are usually less efficient and elements are larger than in $\mathbb{G}_1$. Nevertheless, we note that we can safely switch $\mathbb{G}_1$ and $\mathbb{G}_2$ if needed.

Table 1. Complexity of our Redactable Signature Schemes. The costs of Derive and Verify are provided for a set $\{m_i\}_{i\in\mathcal{I}}$ of k elements. Here, r_2 denotes the generation of a random element in $\mathbb{G}_2$, e_i denotes an exponentiation in $\mathbb{G}_i$, for $i \in \{1, 2\}$, and p_i denotes an equation involving i pairings.

	vk	pk	σ	Sign	Derive	Verify
RS	$(n+1)\mathbb{G}_1$ $+n\mathbb{G}_2$	$\frac{n^2+n+2}{2}\mathbb{G}_1$ $+n\mathbb{G}_2$	$2\mathbb{G}_1 + 2\mathbb{G}_2$	$1\mathsf{r}_2 + 1\mathsf{e}_2$	$2(n-k)\mathsf{e}_1$	$k\mathsf{e}_1 + 2\mathsf{p}_2$
URS	$(n+1)\mathbb{G}_1$ $+n\mathbb{G}_2$	$\frac{n^2+n+2}{2}\mathbb{G}_1$ $+n\mathbb{G}_2$	$2\mathbb{G}_1 + 2\mathbb{G}_2$	$1\mathsf{r}_2 + 1\mathsf{e}_2$	$2(n-k+1)\mathsf{e}_1$ $+3\mathsf{e}_2$	$k\mathsf{e}_1 + 2\mathsf{p}_2$

Anonymous Credentials. We compare in Table 2 the efficiency of our anonymous credentials system from Sect. 6 with the one of different approaches supporting multiple unlinkable showings of credentials. Most of the references and figures are extracted from the comparison in [15]. The latter shows that the existing solutions mostly differ in the size of the public key and of the credential and in the complexity of the showing process. For sake of clarity, we therefore only consider these features in our table and, for example, do not take into account the complexity of the Issuing process. We nevertheless note that our issuing process is among the most efficient ones. Similarly, we do not indicate in our table the computational assumptions that underlie the security of the constructions and refer to [15] for this information. We indeed note that, except for [9], all of them rely on the generic group model (GGM) or on non-standard assumptions (that are themselves proven in the GGM), which seems to be the price for efficiency and functionalities.

Table 2. Comparison of different anonymous credentials systems. The pk, vk and σ colomns refer to the size of the public key, of the verification key and of the credential, respectively. |Show| indicates the number of elements exchanged by the user and the verifier when the former shows k attributes. The Show and Verify colomns indicate the computational complexity for the user and the verifier, respectively. The last colomn indicates whether the scheme only supports *selective* (s) disclosure, or if it also allows to prove *relations* (r) about the attributes.

Scheme	pk/vk	σ	|Show|	Show	Verify	Proof
[9]	$O(n)/O(n)$	$O(1)$	$O(n-k)$	$O(n-k)$	$O(n)$	r
[10]	$O(n)/O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	r
[1]	$O(n)/O(n)$	$O(1)$	$O(n-k)$	$O(n-k)$	$O(n)$	r
[21]	$O(n)/O(n)$	$O(1)$	$O(n-k)$	$O(n-k)$	$O(n)$	r
[7]	$O(n)/O(n)$	$O(1)$	$O(1)$	$O(n-k)$	$O(k)$	s
[15]	$O(n)/O(n)$	$O(1)$	$O(1)$	$O(n-k)$	$O(k)$	s
Sect. 6	$O(n^2)/O(n)$	$O(1)$	$O(1)$	$O(n-k)$	$O(k)$	r

This table shows that, for a long time, a credential issued on n attributes needed $O(n)$ operations to be verified, even if the user only showed k attributes. Moreover, it was necessary to prove knowledge of the $(n-k)$ hidden attributes, which implied to send $O(n-k)$ elements during the protocol.

Our protocol circumvents this problem and proposes a constant size credential with a constant number of elements to send during Show. Moreover, a verifier who only needs to check k attributes only has to perform k operations, which seems optimal. However one can note that our scheme is not the first one to achieve such remarkable features. We therefore need to go beyond asymptotic comparison when it comes to [7] and [15].

Regarding [7], the situation is quite simple. Although it has nice asymptotic complexity, the $O(1)$ notation for |Show| hides about 100 groups elements to show a credential (see [15]). It is therefore far less practical than our scheme and the

one from [15]. Nevertheless, we must mention that it is the only one to achieve strong security in the UC framework [11], which may justify the efficiency gap.

Regarding [15], we note that our public key is larger, although it can be restricted to $O(n)$ elements if we only consider elements necessary for the verification, as explained in Remark 3. Our credential only consists of 2 elements of $\mathbb{G}_2$ and so is roughly twice shorter than the one from [15] that consists of 3 elements of $\mathbb{G}_1$, 1 of $\mathbb{G}_2$ and 2 scalars.

In our case, to show a credential, a user must send 2 elements of $\mathbb{G}_1$, 2 of $\mathbb{G}_2$, 1 of $\mathbb{G}_T$ and one scalar, contrarily to 8 elements of $\mathbb{G}_1$, 1 of $\mathbb{G}_2$ and two scalars in [15]. If we use Barreto-Naehrig curves [3] to instantiate the bilinear group, we get roughly the same complexity because of the element in $\mathbb{G}_T$ in our protocol. However, we note that the latter is the commitment of a Schnorr's proof and so could be replaced by a scalar if we choose to make our protocol non-interactive using the Fiat-Shamir heuristic [13]. In such a case, our `Show` protocol would be twice more efficient than the one from [15].

Finally, we believe that the main difference between these two schemes can be found in the ability to prove relations about the attributes. Indeed, in our protocol, each disclosed element is involved as an exponent of some public element in the verification equation so it is easy to hide it using Schnorr's proof of knowledge [23] and then to prove that it satisfies another relation (hence the "r" in the last column). Conversely, in [15], the disclosed attributes are roots of some polynomial $f_T(a)$ that is involved in the verification equation, with a a secret parameter of their scheme. Proving knowledge of these attributes is thus much more complex than in our case, so [15] cannot be used if one needs to efficiently prove some relations about them.

8 Conclusion

In this paper, we have provided a remarkably versatile and efficient signature scheme. Given a signature σ on a set of messages $\{m_i\}_{i=1}^n$, one can indeed disclose, prove relations about or redact any subset of $\{m_i\}_{i=1}^n$. Moreover, the number $(n-k)$ of undisclosed messages does not impact communication or verification complexity, leading to very efficient partial verification of a signature when k is small.

This ability to redact or prove relations about parts of the message is particularly useful when privacy is critical and we show that our scheme can be used to construct an anonymous credentials system with the same features. The resulting protocol then combines almost all the best properties of previous solutions, with constant-size credentials and $O(k)$ verification complexity, along with the ability to prove relations about attributes.

We believe that anonymous credentials are just an example of application of our scheme and that the latter could be useful as a building block for other primitives, in particular privacy-preserving ones.

Acknowledgements. The authors are grateful for the support of the ANR through project ANR-16-CE39-0014 PERSOCLOUD and project ANR-18-CE-39-0019-02 MobiS5.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006). https://doi.org/10.1007/11832072_8
2. Baldimtsi, F., et al.: Accumulators with applications to anonymity-preserving revocation. In: EuroS&P 2017, pp. 301–315 (2017)
3. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_22
4. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008)
5. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy, January 2000
6. Brzuska, C., et al.: Redactable signatures for tree-structured data: definitions and constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13708-2_6
7. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: definitions and practical constructions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 262–288. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_11
8. Camenisch, J., Lysyanskaya, A.: An identity escrow scheme with appointed verifiers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 388–407. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_23
9. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36413-7_20
10. Camenisch, J., 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). https://doi.org/10.1007/978-3-540-28628-8_4
11. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
12. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings - the role of Ψ revisited. Discrete Appl. Math. **159**(13), 1311–1322 (2011)
13. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
14. Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 233–253. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_12
15. Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. J. Cryptol. **32**(2), 498–546 (2019)

16. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008)
17. Guillevic, A.: Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In: Jacobson Jr., M., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 357–372. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_22
18. Haber, S., et al.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: Abe, M., Gligor, V. (eds.) ASIACCS 2008, pp. 353–362. ACM Press, March 2008
19. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30574-3_19
20. Nojima, R., Tamura, J., Kadobayashi, Y., Kikuchi, H.: A storage efficient redactable signature in the standard model. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 326–337. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04474-8_26
21. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29485-8_7
22. Sanders, O.: Efficient redactable signature and application to anonymous credentials. IACR Cryptology ePrint Archive, vol. 1201 (2019)
23. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22