Short Randomizable Signatures

David Pointcheval[1](✉) and Olivier Sanders[2]

[1] École Normale Supérieure, CNRS, INRIA, PSL Research University, Paris, France
David.Pointcheval@ens.fr
[2] DGA-MI, Bruz, France

Abstract. Digital signature is a fundamental primitive with numerous applications. Following the development of pairing-based cryptography, several taking advantage of this setting have been proposed. Among them, the Camenisch-Lysyanskaya (CL) signature scheme is one of the most flexible and has been used as a building block for many other protocols. Unfortunately, this scheme suffers from a linear size in the number of messages to be signed which limits its use in many situations.

In this paper, we propose a new signature scheme with the same features as CL-signatures but without the linear-size drawback: our signature consists of only two elements, whatever the message length, and our algorithms are more efficient. This construction takes advantage of using type 3 pairings, that are already widely used for security and efficiency reasons.

We prove the security of our scheme without random oracles but in the generic group model. Finally, we show that protocols using CL-signatures can easily be instantiated with ours, leading to much more efficient constructions.

1 Introduction

Digital signature is one of the main cryptographic primitives which can be used in its own right, to provide the electronic version of handwritten signatures, but also as a building block for more complex primitives. Whereas efficiency is the main concern of the first case, the latter case usually requires a signature scheme with additional features. Indeed, when used as a building block, signatures must not just be efficient, they also have to be compatible with the goals and the other building blocks of the protocol. For example, privacy-preserving primitives usually require a signature scheme which allows signatures on committed secret values and compatible with zero-knowledge proofs.

1.1 Related Works

Constructing a versatile signature scheme that is both efficient and secure is not easy. One of the first construction specifically designed as a building block for other applications was proposed by Camenisch and Lysyanskaya [18].

O. Sanders—Work done while being at Orange Labs.

K. Sako (Ed.): CT-RSA 2016, LNCS 9610, pp. 111–126, 2016.
DOI: 10.1007/978-3-319-29485-8_7

Their construction, relying on the Strong RSA assumption [6], allows indeed signatures on committed values and proofs of knowledge of a signature.

The emergence of pairing-based cryptography [13,34] has created a need for such signature schemes compatible with this new setting. Indeed, many cryptographic protocols now use bilinear groups, *i.e.* a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. In 2004, Camenisch and Lysyanskaya proposed a new pairing-based signature scheme [19] whose flexibility has allowed it to be used in several applications, such as group signatures [10], direct anonymous attestations [9,25], aggregate signatures [35] or E-cash systems [21]. One of its most interesting features is probably the ability of its signatures to be randomized: given a valid CL-signature $\sigma = (a, b, c)$ on a message m, anyone can generate another valid signature on the same message by selecting a random scalar t and computing (a^t, b^t, c^t). The latter is indistinguishable from a fresh signature on m. Let us consider a typical situation for anonymous credentials [17], direct anonymous attestations [15], or group signatures [24]: a user first gets a signature σ on some secret value s and then has to prove, several times, that s is certified still keeping the proofs unlinkable. If σ were issued using a conventional signature scheme, it would have to be committed and the user would have to prove that the commitment opens to a valid signature on a secret value which is a rather complex statement to prove, even in the Random Oracle Model (ROM) [7]. Now, if σ is a CL-signature, then the user can simply compute a randomized version σ' of σ, sends it and proves that it is valid on the secret value. This idea underlies the efficiency of the constructions described in [9,10,25]. For these constructions, unlinkability relies on the DDH assumption in $\mathbb{G}_1$, and so requires the use of asymmetric pairings. But this is not a strong assumption, since they offer the best efficiency (see [29]).

One might have thought that the seminal work of Groth and Sahai [32], providing the first practical non-interactive zero-knowledge proofs (NIZKs) in the standard model, in conjunction with the recent structure-preserving signatures [1–3,23], has decreased interest for CL-signatures. However, that has not happened due to the huge performance gap between constructions in the standard model and constructions in the ROM: for example, the most efficient group signature in the standard model [31] consists of 50 group elements whereas [10], in the ROM, consists of only 3 group elements and two scalars. And for real-life applications, where time constraints are particularly challenging, constructions with NIZK proofs in the ROM seem unavoidable.

As a consequence, signatures schemes, such as the CL-signatures, compatible with NIZKs in the ROM still remain of huge practical interest.

Another primitive for which efficiency considerations are central is anonymous credentials. Unfortunately, even if they are one of the applications proposed for CL-signatures, most of these schemes [4,5,16,20] use other constructions, such as the one proposed by Boneh, Boyen and Shacham (BBS) [12]. This is due to a large extent to the size of CL-signatures, which is linear in the number of messages to be signed. Since a user of an anonymous credential system may have several attributes to be certified, this cost quickly becomes prohibitive. This is unfortunate because, here again, the randomizability of CL-signatures could lead to more efficient protocols.

1.2 Our Contribution

In this paper, we propose a new signature scheme, with the same features as CL-signatures, but with a remarkable efficiency. Indeed, whereas the original CL-signatures [19] on blocks of r messages consist of $1 + 2r$ elements of $\mathbb{G}_1$, ours only require 2 elements of $\mathbb{G}_1$, whatever r is. Moreover, as illustrated in Fig. 1 (see Sect. 7), our signature and verification algorithms are much more efficient.

Our work proceeds from the observation that most of the recent protocols [9,10,25] using CL-signatures require type 3 pairings for efficiency and security reasons (see [29]). However, CL-signatures, as most of the constructions from the beginnings of pairing-based cryptography, were designed for type 1 pairings. Unfortunately, this setting usually leads to more complex protocols since they cannot rely on assumptions which would have held with pairings of other types. This has been illustrated by the recent results [2,23] on structure-preserving signatures, which show that designing schemes specifically for type 3 pairings results in more efficient constructions.

Following the same rationale, we propose a signature scheme suited to such pairings: it can be seen as CL-signatures, but taking advantage of the full potential of type 3 pairings. The separation between the space of the signatures ($\mathbb{G}_1$) and the one of the public key ($\mathbb{G}_2$) allows indeed more efficient constructions since the elements of the latter can no longer be used to build forgeries in the former. Unfortunately, the security of our scheme does not rely on any standard assumption and so is proved in the generic group model, which does not provide the same guarantees. However, as illustrated by [2,11,19], relying on proofs in the generic group model or on non-standard assumptions (themselves proved in this model), allows more efficient constructions. For some applications with challenging time constraints, such as public transport where authentication must be performed in less than 300 ms [27,33], we argue that this trade-off, between efficiency and the security assumption, is reasonable. By providing short signatures with efficient algorithms, our solution may then contribute to make all features of modern cryptography more accessible.

Improving the efficiency of primitives with practical applications was also the concern of the authors of [22]. They proved, in the generic group model, the security of the MAC scheme introduced in [28] and used it to construct keyed-verification anonymous credentials (the secret-key analogue of standard anonymous credentials). Although our signature shares similarities with this scheme, it offers much more flexibility. Indeed, the construction described in [22,28] does not achieve public verifiability and so only fits the case where the verifier is also the issuer. Moreover, the protocols for obtaining or proving knowledge of a MAC on committed messages are more complex than the ones, for a signature, we describe in this paper.

Besides efficiency, one of the main advantages of our scheme is that it acts as a plug-in replacement for CL-signatures. Indeed, since they achieve the same properties than the latter, our signatures can be used to instantiate most of the protocols initially designed for CL ones. To illustrate this point, we convert our signature scheme into a sequential aggregate signature scheme [37] using an idea similar to the one of Lee, Lee and Yung [35]. The resulting aggregate signature only consists of 2 elements in $\mathbb{G}_1$ and so is shorter than theirs. Similar gains can be achieved for many other applications such as group signatures or anonymous credentials.

1.3 Organization

We review some definitions and notations in Sect. 2 and present new computational assumptions in Sect. 3. Section 4 describes our signature scheme whose conversion into a sequential aggregate signature scheme is described in Sect. 5. Section 6 describes a variant of our scheme allowing to sign committed values along with a protocol for proving knowledge of a signature. Section 7 provides a comparison with related works. Finally, we describe some applications and provide the security proofs in the appendices.

2 Preliminaries

2.1 Bilinear Groups

Bilinear groups are a set of three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1. for all $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{a \cdot b}$;
2. for $g \neq 1_{\mathbb{G}_1}$ and $\widetilde{g} \neq 1_{\mathbb{G}_2}$, $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$;
3. the map e is efficiently computable.

Galbraith, Paterson, and Smart [29] defined three types of pairings: in type 1, $\mathbb{G}_1 = \mathbb{G}_2$; in type 2, $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists an efficient homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, while no efficient one exists in the other direction; in type 3, $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction.

Although type 1 pairings were mostly used in the early-age of pairing-based cryptography, they have been gradually discarded in favour of type 3 pairings. Indeed, the latter offer a better efficiency and are compatible with several computational assumptions, such as the Decision Diffie-Hellman assumption in $\mathbb{G}_1$ or $\mathbb{G}_2$, also known as the XDH assumption, which does not hold in type 1 pairings.

In this work, we only consider type 3 pairings. We stress that using type 1 or type 2 pairings would make our signature scheme totally insecure.

2.2 Digital Signature Scheme

Syntax. A digital signature scheme Σ is defined by four algorithms:

- the $\texttt{Setup}$ algorithm which, on input a security parameter k, outputs pp, a description of the public parameters;
- the key generation algorithm $\texttt{Keygen}$ which, on input pp, outputs a pair of signing and verification keys $(\mathsf{sk}, \mathsf{pk})$ – we assume that sk contains pk, and that pk contains pp;
- the signing algorithm $\texttt{Sign}$ which, on input the signing key sk and a message m, outputs a signature σ;
- the verification algorithm $\texttt{Verify}$ which, on input m, σ and pk, outputs 1 if σ is a valid signature on m under pk, and 0 otherwise.

Security Notion. The standard security notion for a signature scheme is *existential unforgeability under chosen message attacks* (EUF-CMA) [30] which means that it is hard, even given access to a signing oracle, to output a valid pair (m, σ) for a message m never asked to the signing oracle. It is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- **Setup:** $\mathcal{C}$ runs the $\texttt{Setup}$ and the $\texttt{Keygen}$ algorithms to obtain sk and pk. The adversary is given the public key pk;
- **Queries:** $\mathcal{A}$ adaptively requests signatures on at most q messages $m_1, \ldots, m_q$. $\mathcal{C}$ answers each query by returning $\sigma_i \leftarrow \texttt{Sign}(\mathsf{sk}, m_i)$;
- **Output:** $\mathcal{A}$ eventually outputs a message-signature pair (m^*, σ^*) and wins the game if $\texttt{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$ and if $m^* \neq m_i \ \forall i \in [1, q]$.

A signature scheme is EUF-CMA secure if no probabilistic polynomial-time adversary $\mathcal{A}$ can win this game with non-negligible probability.

2.3 Sequential Aggregate Signature

Syntax. Sequential aggregate signature [37] is a special type of aggregate signature (introduced by Boneh *et al.* [14]) where the final signature on the list of messages is computed sequentially by each signer, who adds his signature on his message. It is defined by the four algorithms described below:

- the $\texttt{AS.Setup}$ algorithm which, on input a security parameter k, outputs pp, a description of the public parameters;
- the key generation algorithm $\texttt{AS.Keygen}$ which, on input pp, outputs a pair of signing and verification keys $(\mathsf{sk}, \mathsf{pk})$ – we assume that sk contains pk, and that pk contains pp;
- the signing algorithm $\texttt{AS.Sign}$ which, on input an aggregate signature σ on messages $(m_1, \ldots, m_r)$ under public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$, a message m and a signing key sk such that $\mathsf{pk} \notin \{\mathsf{pk}_i\}_{i=1}^r$, outputs a new aggregate signature σ' on $(m_1, \ldots, m_r, m)$;
- the verification algorithm $\texttt{AS.Verify}$ which, on input $(m_1, \ldots, m_r)$, σ and distinct public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$, outputs 1 if σ is a valid aggregate signature on $(m_1, \ldots, m_r)$ under $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$, and 0 otherwise.

Security Model. The security property for a sequential aggregate signature scheme is *existential unforgeability under chosen message attacks* which requires that no adversary is able to forge an aggregate signature, on a set of messages of its choice, by a set of users whose secret keys are not all known to it. It is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

– **Setup:** $\mathcal{C}$ first initializes a key list $\texttt{KeyList}$ as empty. Next it runs the $\texttt{AS.Setup}$ algorithm to get *pp* and the $\texttt{AS.Keygen}$ algorithm to get the signing and verification keys $(\mathsf{sk}^*, \mathsf{pk}^*)$. The verification key pk^* is given to $\mathcal{A}$;
– **Join Queries:** $\mathcal{A}$ adaptively asks to add the public keys pk_i to $\texttt{KeyList}$;
– **Signature Query:** $\mathcal{A}$ adaptively requests aggregate signatures on at most q messages $m_1, \ldots, m_q$ under the challenge public key pk^*. For each query, it provides an aggregate signature σ_i on the messages $(m_{i,1}, \ldots, m_{i,r_i})$ under the public keys $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i})$, all in $\texttt{KeyList}$. Then $\mathcal{C}$ returns the aggregation $\texttt{AS.Sign}(\mathsf{sk}^*, \sigma_i, (m_{i,1}, \ldots, m_{i,r_i}), (\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i}), m_i)$;
– **Output:** $\mathcal{A}$ eventually outputs an aggregate signature σ on the messages $(m_1^*, \ldots, m_r^*)$ under the public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$ and wins the game if the following conditions are all satisfied:
  - $\texttt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), (m_1^*, \ldots, m_r^*), \sigma) = 1$;
  - For all $\mathsf{pk}_j \neq \mathsf{pk}^*$, $\mathsf{pk}_j \in \texttt{KeyList}$;
  - For some $j^* \in [1, r]$, $\mathsf{pk}^* = \mathsf{pk}_{j^*}$ and $m_{j^*}^*$ has not been queried to the signing oracle, *i.e.* $m_{j^*}^* \neq m_i$, for $i = 1, \ldots, q$.

A sequential aggregate signature scheme is EUF-CMA secure if no probabilistic polynomial-time adversary $\mathcal{A}$ can win this game with non-negligible probability.

Certified Keys. As in [35], we consider the setting proposed by Lu *et al.* [36] where users must prove knowledge of their signing key sk when they want to add a public key pk in $\texttt{KeyList}$. In the security proof, this enables the simulator to answer every signature query made by the adversary $\mathcal{A}$. As a consequence, in the **Join Query**, when $\mathcal{A}$ asks to add pk to $\texttt{KeyList}$, it additionally proves its knowledge of the corresponding secret key sk.

3 Assumption

A by-now classical assumption is the so-called LRSW [38], applied to many privacy-preserving protocols, such as the CL-signatures [19], that admit two protocols: an issuing protocol that allows a user to get a signature σ on a message x, just by sending a commitment of x to the signer, and a proving protocol that allows the user to prove, in a zero-knowledge way, his knowledge of a signature on a commitment of x. They lead to efficient anonymous credentials.

Definition 1 (LRSW Assumption). *Let $\mathbb{G}$ be a cyclic group of prime order p, with a generator g. For $X = g^x$ and $Y = g^y$, where x and y are random scalars in $\mathbb{Z}_p$, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $h \in \mathbb{G}$ and outputs the triple $T = (h, h^y, h^{x+mxy})$. Given (X, Y) and unlimited access to this oracle, no adversary can efficiently generate such a triple for a new scalar m^*, not asked to $\mathcal{O}$.*

This assumption has been introduced in [38] and proven in the generic group model, as modeled by Shoup [42].

We now propose two similar assumptions in bilinear groups of type 3 that will provide even more efficient protocols. We then prove them to hold in the bilinear generic group model.

Definition 2 (Assumption 1). *Let* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *a bilinear group setting of type 3, with* g *(resp.* $\widetilde{g}$*) a generator of* $\mathbb{G}_1$ *(resp.* $\mathbb{G}_2$*). For* $(X = g^x, Y = g^y)$ *and* $(\widetilde{X} = \widetilde{g}^x, \widetilde{Y} = \widetilde{g}^y)$*, where* x *and* y *are random scalars in* $\mathbb{Z}_p$*, we define the oracle* $\mathcal{O}(m)$ *on input* $m \in \mathbb{Z}_p$ *that chooses a random* $h \in \mathbb{G}_1$ *and outputs the pair* $P = (h, h^{x+my})$*. Given* $(g, Y, \widetilde{g}, \widetilde{X}, \widetilde{Y})$ *and unlimited access to this oracle, no adversary can efficiently generate such a pair, with* $h \neq 1_{\mathbb{G}_1}$*, for a new scalar* m^**, not asked to* $\mathcal{O}$*.*

One can note that using pairings, an output of the adversary can be checked since the pair $P = (P_1, P_2)$ should satisfy $e(P_1, \widetilde{X} \cdot \widetilde{Y}^m) = e(P_2, \widetilde{g})$. In addition, (X, Y) are enough to answer oracle queries: on a scalar $m \in \mathbb{Z}_p$, one computes $(g^r, (X \cdot Y^m)^r)$. This requires 3 exponentiations per query, while knowing (x, y) just requires a random sampling in $\mathbb{G}_1$ and one exponentiation.

In some situations, a weaker assumption will be enough, where Y is not given to the adversary:

Definition 3 (Assumption 2). *Let* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ *a bilinear group setting of type 3, with* g *(resp.* $\widetilde{g}$*) a generator of* $\mathbb{G}_1$ *(resp.* $\mathbb{G}_2$*). For* $(\widetilde{X} = \widetilde{g}^x, \widetilde{Y} = \widetilde{g}^y)$ *where* x *and* y *are random scalars in* $\mathbb{Z}_p$*, we define the oracle* $\mathcal{O}(m)$ *on input* $m \in \mathbb{Z}_p$ *that chooses a random* $h \in \mathbb{G}$ *and outputs the pair* $P = (h, h^{x+my})$*. Given* $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$ *and unlimited access to this oracle, no adversary can efficiently generate such a pair, with* $h \neq 1_{\mathbb{G}_1}$*, for a new scalar* m^**, not asked to* $\mathcal{O}$*.*

Theorem 4. *The above Assumption 1 (and thus the Assumption 2) holds in the generic bilinear group model: after* q *oracle queries and* q_G *group-oracle queries, no adversary can generate a valid pair for a new scalar with probability greater than* $6(q + q_G)^2/p$*.*

The proof can be found in the full version [40].

4 Our Randomizable Digital Signature Scheme

For the sake of clarity, for our signature scheme, we first describe the specific case where only one message is signed. We then present an extension allowing to sign several messages and show that the security of the latter scheme holds under the security of the former (which holds under the weak Assumption 2).

4.1 A Single-Message Signature Scheme

Description. Our signature scheme to sign a message $m \in \mathbb{Z}_p$ consists of the following algorithms:

- $\mathtt{Setup}(1^k)$: Given a security parameter k, this algorithm outputs $pp \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. These bilinear groups must be of type 3. In the following, we denote $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$;
- $\mathtt{Keygen}(pp)$: This algorithm selects $\widetilde{g} \overset{\$}{\leftarrow} \mathbb{G}_2$ and $(x, y) \overset{\$}{\leftarrow} \mathbb{Z}_p^2$, computes $(\widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^x, \widetilde{g}^y)$ and sets sk as (x, y) and pk as $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$;
- $\mathtt{Sign}(\mathsf{sk}, m)$: This algorithm selects a random $h \overset{\$}{\leftarrow} \mathbb{G}_1^*$ and outputs $\sigma \leftarrow (h, h^{(x+y\cdot m)})$;
- $\mathtt{Verify}(\mathsf{pk}, m, \sigma)$: This algorithm parses σ as (σ_1, σ_2) and checks whether $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^m) = e(\sigma_2, \widetilde{g})$ are both satisfied. In the positive case, it outputs 1, and 0 otherwise.

Correctness: If $\sigma = (\sigma_1 = h, \sigma_2 = h^{(x+y\cdot m)})$, then

$$e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^m) = e(h, \widetilde{X} \cdot \widetilde{Y}^m) = e(h, \widetilde{g})^{(x+y\cdot m)} = e(h^{(x+y\cdot m)}, \widetilde{g}) = e(\sigma_2, \widetilde{g}).$$

Remark 5. As already remarked above, the signature could be generated with the secret key being either (x, y) or $(X = g^x, Y = g^y)$. But the former leads a more efficient signature scheme.

Randomizability. As the CL-signatures, a signature $\sigma = (\sigma_1, \sigma_2)$ on a message m can be randomized by selecting a random $t \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and computing $\sigma' \leftarrow (\sigma_1^t, \sigma_2^t)$ which is still a valid signature on m: it corresponds to replace $h \in \mathbb{G}_1^*$ by $h' = h^t \in \mathbb{G}_1^*$.

Security Analysis. EUF-CMA is exactly the above Assumption 2, since a signing oracle is perfectly equivalent to the oracle $\mathcal{O}$.

4.2 A Multi-message Signature Scheme

Description. We now present a variant of the previous scheme to sign r-message vectors $(m_1, \ldots, m_r) \in \mathbb{Z}_p^r$ at once. Our signature scheme consists of the following algorithms, where all the sums and products are on j between 1 and r:

- $\mathtt{Setup}(1^k)$: Given a security parameter k, this algorithm outputs $pp \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. These bilinear groups must be of type 3. In the following, we denote $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$;
- $\mathtt{Keygen}(pp)$: This algorithm selects $\widetilde{g} \overset{\$}{\leftarrow} \mathbb{G}_2$ and $(x, y_1, \ldots, y_r) \overset{\$}{\leftarrow} \mathbb{Z}_p^{r+1}$, computes $(\widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r) \leftarrow (\widetilde{g}^x, \widetilde{g}^{y_1}, \ldots, \widetilde{g}^{y_r})$ and sets sk as $(x, y_1, \ldots, y_r)$ and pk as $(\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$.
- $\mathtt{Sign}(\mathsf{sk}, m_1, \ldots, m_r)$: This algorithm selects a random $h \overset{\$}{\leftarrow} \mathbb{G}_1^*$ and outputs $\sigma \leftarrow (h, h^{(x+\sum y_j \cdot m_j)})$.
- $\mathtt{Verify}(\mathsf{pk}, (m_1, \ldots, m_r), \sigma)$: This algorithm parses σ as (σ_1, σ_2) and checks whether $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(\sigma_2, \widetilde{g})$ are both satisfied. In the positive case, it outputs 1, and 0 otherwise.

Correctness: If $\sigma = (\sigma_1 = h, \sigma_2 = h^{(x+\sum y_j \cdot m_j)})$, then

$$\begin{aligned} e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) &= e(h, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(h, \widetilde{g})^{x+\sum y_j \cdot m_j} \\ &= e(h^{x+\sum y_j \cdot m_j}, \widetilde{g}) = e(\sigma_2, \widetilde{g}). \end{aligned}$$

Security Analysis. We now rely the security of this multiple-message signature scheme to the security of the single-message signature scheme, and so on Assumption 2. Due to space limitations, the proof of the following theorem is provided in in the full version [40].

Theorem 6. *The multiple-message signature scheme achieves the EUF-CMA security level under the above Assumption 2. More precisely, if an adversary can break the EUF-CMA of the multiple-message signature scheme with probability ε, then there exists an adversary against the EUF-CMA security of the single-message signature scheme, within the same running time and the same number of signing queries, succeeding with probability greater than $\varepsilon - q/p$.*

5 A Sequential Aggregate Signature

Our Construction. It is possible to slightly modify the scheme from Sect. 4.2 to convert it into a sequential aggregate signature scheme. The signer's secret key of the original scheme to sign r-message vector was $(x, y_1, \ldots, y_r)$. But now, let us assume one publishes a signature on the r-vector $(0, \ldots, 0)$: $(g, X) = (g, g^x) \in \mathbb{G}_1^2$ for some $g \in \mathbb{G}_1$. This additional knowledge does not help an adversary to produce forgeries on non-zero vectors, but the scalar value x is no longer useful in the secret key since one can sign a vector $(m_1, \ldots, m_r)$ by selecting a random $t \xleftarrow{\$} \mathbb{Z}_p$ and computing $(g^t, (X)^t \cdot (g^t)^{\sum y_j \cdot m_j})$. The correctness follows from the one of the original scheme.

On the other hand, we can use the public key sharing technique from [35] to construct an efficient sequential aggregate signature scheme in the standard model: each signer j (from 1 to r) generates his own signing and verification keys $(y_j, \widetilde{Y}_j)$ but uses the same element X from the public parameters. To sign a message $m_1 \in \mathbb{Z}_p^*$, the first selects a random $t_1 \xleftarrow{\$} \mathbb{Z}_p$ and outputs $(\sigma_1, \sigma_2) \leftarrow (g^{t_1}, (X)^{t_1} \cdot (g^{t_1})^{y_1 \cdot m_1})$. A subsequent signer 2 can generate an aggregate signature on m_2 by selecting a random t_2 and computing $(\sigma_1', \sigma_2') \leftarrow (\sigma_1^{t_2}, (\sigma_2 \cdot \sigma_1^{y_2 \cdot m_2})^{t_2})$. Therefore, $(\sigma_1', \sigma_2') = (g^{t_1 \cdot t_2}, g^{t_1 \cdot t_2 (x + m_1 \cdot y_1 + m_2 \cdot y_2)}) = (g^t, g^{t(x + m_1 \cdot y_1 + m_2 \cdot y_2)})$, for $t = t_1 t_2$, and so its validity can be verified using the `Verify` algorithm described in Sect. 4.2.

More formally, our sequential aggregate signature scheme is defined by the following algorithms.

- `AS.Setup`(1^k): Given a security parameter k, this algorithm selects a random $x \in \mathbb{Z}_p$ and outputs $pp \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, X, \widetilde{g}, \widetilde{X})$, where $X = g^x$ and $\widetilde{X} = \widetilde{g}^x$ for some generators $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$.

- $\texttt{AS.Keygen}(pp)$: This algorithm selects a random $y \xleftarrow{\$} \mathbb{Z}_p$, computes $\widetilde{Y} \leftarrow \widetilde{g}^y$ and sets sk as y and pk as $\widetilde{Y}$.
- $\texttt{AS.Sign}(\mathsf{sk}, \sigma, (m_1, \ldots, m_r), (\mathsf{pk}_1, \ldots, \mathsf{pk}_r), m)$ proceeds as follows:
  - If $r = 0$, then $\sigma \leftarrow (g, X)$;
  - If $r > 0$ but $\texttt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), \sigma, (m_1, \ldots, m_r)) = 0$, then it halts;
  - If $m = 0$, then it halts;
  - If for some $j \in \{1, \ldots, r\}$ $\mathsf{pk}_j = \mathsf{pk}$, then it halts.

  If the algorithm did not halt, then it parses sk as y and σ as (σ_1, σ_2), selects $t \xleftarrow{\$} \mathbb{Z}_p$ and computes $\sigma' = (\sigma'_1, \sigma'_2) \leftarrow (\sigma_1^t, (\sigma_2 \cdot \sigma_1^{y \cdot m})^t)$. It eventually outputs σ'.
- $\texttt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), (m_1, \ldots, m_r), \sigma)$ parses σ as (σ_1, σ_2) and pk_j as $\widetilde{Y}_j$, for $j = 1, \ldots, r$, and checks whether $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j}) = e(\sigma_2, \widetilde{g})$ are both satisfied. In the positive case, it outputs 1, and 0 otherwise.

Correctness. If $r = 0$, then the algorithm $\texttt{AS.Sign}$ outputs $(g^t, (X \cdot g^{y \cdot m})^t) = (g^t, g^{t(x + y \cdot m)})$. By induction, let us now assume that $\sigma = (g^s, g^{s(x + \sum y_j \cdot m_j)})$, then an aggregate signature σ' on m is equal to $(g^{t \cdot s}, g^{t \cdot s(x + m \cdot y + \sum y_j \cdot m_j)})$, which is equal to $(h, h^{x + \sum y_j \cdot m_j + y \cdot m})$ for some $h \in \mathbb{G}_1$. The correctness of our sequential aggregate signature scheme follows then from the signature scheme described in Sect. 4.2.

Security Analysis. We now rely the security of this aggregate signature scheme, in the certified public key setting, to the security of the single-message signature scheme, and so on Assumption 2:

Theorem 7. *The aggregate signature scheme achieves the EUF-CMA security level, in the certified public-key setting, under the above Assumption 2. More precisely, if an adversary can break the EUF-CMA of the aggregate signature scheme, then there exists an adversary against the EUF-CMA security of the single-message signature scheme, within the same running time and the same number of signing queries, succeeding with the same probability.*

The proof can be found in the in the full version [40].

6 Useful Features

6.1 Signing Committed Messages

Many cryptographic primitives require efficient protocols to obtain signatures on *committed* (or transformed) values. For example, in some group signature schemes [10,12,26], users must get a certificate on their secret key $m \in \mathbb{Z}_p$ to join the group. The non-frameability property [8] expected from such a primitive prevents the users to directly send the value m to the group manager. Instead, they rather send a public value g^m, for some public $g \in \mathbb{G}_1$, and start a protocol with the latter to get a signature on the secret value m.

Our signature scheme can be slightly modified to handle such a protocol: one can submit g^m to the signer and prove knowledge of m. If the proof is valid, the signer can return $\sigma = (\sigma_1, \sigma_2) \leftarrow (g^u, (g^x \cdot (g^m)^y)^u)$, for some $u \xleftarrow{\$} \mathbb{Z}_p$, which is a valid signature on m.

However, g^m is not hiding enough in some applications, and namely if information-theoretical security is required. For example, in anonymous credentials [17], the elements $g^{m_1}, \ldots, g^{m_r}$ may provide too much information on the attributes $(m_1, \ldots, m_r)$, if they belong to small sets.

The modified BBS signature scheme [12] described in [4] enables the signer to sign messages $(m_1, \ldots, m_r)$ from a Pedersen commitment [39] $C = g_0^t \cdot g_1^{m_1} \cdots g_r^{m_r}$ (where t is a random scalar). We need to slightly modify the scheme described in Sect. 4.2 to add such a feature. Indeed, the latter does not provide any element of $\mathbb{G}_1$ in the public key. The resulting protocol is described below, in the multi-message setting. But we first start with the single-message protocol.

A Single-Message Protocol. The signature scheme for signing one information-theoretically hidden message consists of the following algorithms:

- `Setup`(1^k): Given a security parameter k, this algorithm outputs $pp \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. These bilinear groups must be of type 3. In the following, we denote $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$ and $\mathbb{G}_2^* = \mathbb{G}_2 \backslash \{1_{\mathbb{G}_2}\}$, which are the sets of the generators.
- `Keygen`(pp): This algorithm selects $g \xleftarrow{\$} \mathbb{G}_1^*$, $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2^*$ and $(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$, computes $(X, Y) \leftarrow (g^x, g^y)$ and $(\widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^x, \widetilde{g}^y)$, and sets $\mathsf{sk} \leftarrow X$ and $\mathsf{pk} \leftarrow (g, Y, \widetilde{g}, \widetilde{X}, \widetilde{Y})$.
- `Protocol`: A user who wishes to obtain a signature on the message $m \in \mathbb{Z}_p$ first selects a random $t \xleftarrow{\$} \mathbb{Z}_p$ and computes $C \leftarrow g^t Y^m$. He then sends C to the signer. They both run a proof of knowledge of the opening of the commitment. If the signer is convinced, he selects a random $u \xleftarrow{\$} \mathbb{Z}_p$ and returns $\sigma' \leftarrow (g^u, (XC)^u)$. The user can now unblind the signature by computing $\sigma \leftarrow (\sigma_1', \sigma_2'/{\sigma_1'}^t)$.

The element σ then satisfies $\sigma_1 = g^u$ and $\sigma_2 = (XC)^u/g^{ut} = (Xg^tY^m/g^t)^u = (XY^m)^u$, which is a valid signature on m for the single-message signature scheme described in Sect. 4.1. However, because of the additional elements in the public key, the EUF-CMA security of the underlying signature scheme now relies on the Assumption 1.

A Multi-message Protocol. The signature scheme for signing information-theoretically hidden messages consists of the following algorithms:

- `Setup`(1^k): Given a security parameter k, this algorithm outputs $pp \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. These bilinear groups must be of type 3. In the following, we denote $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$ and $\mathbb{G}_2^* = \mathbb{G}_2 \backslash \{1_{\mathbb{G}_2}\}$, which are the sets of the generators.

– `Keygen`(pp): This algorithm selects $g \xleftarrow{\$} \mathbb{G}_1^*$, $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2^*$ and $(x, y_1, \ldots, y_r) \xleftarrow{\$} \mathbb{Z}_p^{r+1}$, computes $(X, Y_1, \ldots, Y_r) \leftarrow (g^x, g^{y_1}, \ldots, g^{y_r})$ and $(\widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r) \leftarrow (\widetilde{g}^x, \widetilde{g}^{y_1}, \ldots, \widetilde{g}^{y_r})$, and sets $\mathsf{sk} \leftarrow X$ and $\mathsf{pk} \leftarrow (g, Y_1, \ldots, Y_r, \widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$.
– `Protocol`: A user who wishes to obtain a signature on $(m_1, \ldots, m_r)$ first selects a random $t \xleftarrow{\$} \mathbb{Z}_p$ and computes $C \leftarrow g^t \prod_{i=1}^r Y_i^{m_i}$. He then sends C to the signer. They both run a proof of knowledge of the opening of the commitment. If the signer is convinced, he selects a random $u \xleftarrow{\$} \mathbb{Z}_p$ and returns $\sigma' \leftarrow (g^u, (XC)^u)$. The user can now unblind the signature by computing $\sigma \leftarrow (\sigma'_1, \sigma'_2/{\sigma'_1}^t)$.

Again, the element σ satisfies $\sigma_1 = g^u$ and $\sigma_2 = (XC)^u/g^{ut}$. If one develops, $\sigma_2 = (Xg^t \prod_{i=1}^r Y_i^{m_i}/g^t)^u = (X \prod_{i=1}^r Y_i^{m_i})^u$, which is a valid signature on $(m_1, \ldots, m_r)$ for the multi-message signature scheme described in Sect. 4.2, but with additional elements in the public key: the EUF-CMA security of this multi-message signature scheme can also be shown equivalent to the one of the single-message signature scheme, with a similar proof as the one for Theorem 6, and thus relies on the Assumption 1.

6.2 Proving Knowledge of a Signature

If we still consider the example of anonymous credentials, the previous protocols have addressed the problem of their issuance. However, once a user has obtained his credential, he must also be able to use it to prove that its attributes are certified, while remaining anonymous. To do so, the protocols usually follow the framework described in [19] and so need an efficient way to prove knowledge of a signature.

Our scheme offers such functionality thanks to the ability of our signatures to be sequentially aggregated. Informally, to prove knowledge of a signature $\sigma = (\sigma_1, \sigma_2)$ on a message m, the user will aggregate a signature on some random message t under a dummy public key $\widetilde{g}$ (which is part of the public parameters). The resulting signature σ' is then valid on the block (m, t) and does not reveal any information on m.

More formally, let $\mathsf{pk} \leftarrow (\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$ be a public key for the signature scheme of Sect. 4.2 and $\sigma = (\sigma_1, \sigma_2)$ be a valid signature on a block $(m_1, \ldots, m_r)$ under it. To prove knowledge of σ, the prover does the following:

1. He selects random $r, t \xleftarrow{\$} \mathbb{Z}_p$ and computes $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$.
2. He sends $\sigma' = (\sigma'_1, \sigma'_2)$ to the verifier and carries out a zero-knowledge proof of knowledge π (such as the Schnorr's interactive protocol [41]) of $(m_1, \ldots, m_r)$ and t such that:
$$e(\sigma'_1, \widetilde{X}) \cdot \prod e(\sigma'_1, \widetilde{Y}_j)^{m_j} \cdot e(\sigma'_1, \widetilde{g})^t = e(\sigma'_2, \widetilde{g})$$
The verifier accepts if π is valid.

Theorem 8. *The protocol above is a zero-knowledge proof of knowledge of a signature σ on the block $(m_1, \ldots, m_r)$.*

The proof is provided in the in the full version [40].

7 Efficiency

We compare in Fig. 1 the efficiency of our scheme with the ones of CL-signatures [19] and BBS-signatures [4,12] since they are the most popular schemes used as building blocks for pairing-based protocols. As described in [4], to compute a BBS signature on a block of r messages $(m_1, \ldots, m_r)$, a signer whose secret key is $\gamma \in \mathbb{Z}_p$ first selects two random scalars e and s and then computes $A \leftarrow (g_0 g_1^s g_2^{m_1} \ldots g_{r+1}^{m_r})^{\frac{1}{e+\gamma}}$ for some public parameters $g_0, \ldots, g_{r+1}$. The signature is defined as (A, e, s). For proper comparison, we consider a variant of this scheme where the signer has generated the elements $g_i \leftarrow g_0^{y_i}$ for $i \in [1, r+1]$. Therefore, he can compute the element A more efficiently since $A = g_0^{\frac{1+\sum_{i=1}^{r+1} y_i \cdot m_i}{\gamma+e}}$.

	Size of Sig.	Sig. Cost	Verif. Cost	Rand.	Pairings
Sign. Schemes					
BBS [12, 4]	$1\,\mathbb{G}_1 + 2\,\mathbb{Z}_p$	$2\,\mathtt{R}_{\mathbb{Z}_p} + 1\,\mathtt{E}_{\mathbb{G}_1}$	$2\,\mathtt{P} + 1\,\mathtt{E}_{\mathbb{G}_2} + (r+1)\,\mathtt{E}_{\mathbb{G}_1}$	No	All
CL [19]	$(1+2r)\,\mathbb{G}_1$	$1\,\mathtt{R}_{\mathbb{G}_1} + 2r\,\mathtt{E}_{\mathbb{G}_1}$	$4r\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Yes	All
Ours [sect. 4.2]	$2\,\mathbb{G}_1$	$1\,\mathtt{R}_{\mathbb{G}_1} + 1\,\mathtt{E}_{\mathbb{G}_1}$	$2\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Yes	type 3
Seq. Aggregate Sign. Schemes					
LLY [35]	$3\,\mathbb{G}_1$	$1\,\mathtt{Ver.} + 5\,\mathtt{E}_{\mathbb{G}_1}$	$5\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Yes	All
Ours [sec. 5]	$2\,\mathbb{G}_1$	$1\mathtt{Ver.} + 3\,\mathtt{E}_{\mathbb{G}_1}$	$2\,\mathtt{P} + r\,\mathtt{E}_{\mathbb{G}_2}$	Yes	type 3

Fig. 1. Efficiency comparison between related works. Here, r refers to the number of messages, $\mathtt{R}_{\mathbb{G}_1}$ (resp. $\mathtt{R}_{\mathbb{Z}_p}$) to the cost of generating a random element of $\mathbb{G}_1$ (resp. $\mathbb{Z}_p$), $\mathtt{E}_{\mathbb{G}_i}$ to the cost of an exponentiation in $\mathbb{G}_i$ ($i \in \{1, 2\}$), $\mathtt{P}$ to the cost of a pairing computation and $\mathtt{Ver}$ to the cost of verifying an aggregate signature.

As illustrated in Fig. 1, our signature scheme (resp. sequential aggregate signature scheme) compares favourably with the one from [19] (resp. [35]). However, our scheme is only compatible with type 3 pairings but we argue that this is not a strong restriction since most of the recent cryptographic protocols already use them for efficiency and security reasons.

Although the efficiency of our scheme is similar to the one of BBS, we stress that the ability of our signatures to be randomized improves the efficiency of protocols using them. Indeed, as explained in Sect. 1.1, one cannot show several times a BBS signature while being unlinkable. One must then commit to the signature and then prove in a zero-knowledge way that the resulting commitment opens to a valid signature. This is not the case with our scheme since one can simply randomize the signature between each show. To illustrate this point, we provide some examples in in the full version [40].

8 Conclusion

In this work we have proposed a new signature scheme, suited for type 3 pairings, which achieves a remarkable efficiency. As CL-signatures, our signatures can be randomized and can be used as building blocks for many cryptographic primitives. In particular, they support efficient protocols for obtaining a signature on committed elements and can be efficiently combined with zero-knowledge proofs in the ROM. As illustrated in this paper, instantiating cryptographic constructions with our solution improves their efficiency and may therefore contribute to make them more accessible for real-life applications.

Acknowledgments. This work was supported in part by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
2. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg (2011)
3. Abe, M., Groth, J., Ohkubo, M., Tango, T.: Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 241–260. Springer, Heidelberg (2014)
4. 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)
5. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 1087–1098. ACM Press (2013)
6. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press (1993)
8. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: the case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
9. Bernhard, D., Fuchsbauer, G., Ghadafi, E., Smart, N.P., Warinschi, B.: Anonymous attestation with user-controlled linkability. Int. J. Inf. Sec. **12**(3), 219–249 (2013)
10. Bichsel, P., Camenisch, J., Neven, G., Smart, N.P., Warinschi, B.: Get shorty via group signatures without encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 381–398. Springer, Heidelberg (2010)
11. Boneh, D., Boyen, X.: Short signatures without random Oracles and the SDH assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008)

12. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
13. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
14. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
15. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 132–145. ACM Press (2004)
16. Camenisch, J., Groß, T.: Efficient attributes for anonymous credentials. ACM Trans. Inf. Syst. Secur. **15**(1), 4 (2012)
17. Camenisch, J.L., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
18. Camenisch, J.L., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
19. Camenisch, J.L., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
20. Canard, S., Lescuyer, R.: Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W.G. (eds.) ASIACCS 2013, pp. 381–392. ACM Press (2013)
21. Canard, S., Pointcheval, D., Sanders, O., Traoré, J.: Divisible e-cash made practical. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 77–100. Springer, Heidelberg (2015)
22. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014, pp. 1205–1216. ACM Press (2014)
23. Chatterjee, S., Menezes, A.: Typpe 2 structure-preserving signature schemes revisited. Cryptology ePrint Archive, Report 2014/635 (2014). http://eprint.iacr.org/2014/635
24. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
25. Chen, L., Page, D., Smart, N.P.: On the design and implementation of an efficient DAA scheme. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 223–237. Springer, Heidelberg (2010)
26. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
27. Desmoulins, N., Lescuyer, R., Sanders, O., Traoré, J.: Direct anonymous attestations with dependent basename opening. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 206–221. Springer, Heidelberg (2014)
28. Dodis, Y., Kiltz, E., Pietrzak, K., Wichs, D.: Message authentication, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 355–374. Springer, Heidelberg (2012)
29. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. **156**(16), 3113–3121 (2008)

30. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
31. Groth, J.: Fully anonymous group signatures without random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
32. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
33. Hinterwälder, G., Zenger, C.T., Baldimtsi, F., Lysyanskaya, A., Paar, C., Burleson, W.P.: Efficient e-cash in practice: NFC-based payments for public transportation systems. In: De Cristofaro, E., Wright, M. (eds.) PETS 2013. LNCS, vol. 7981, pp. 40–59. Springer, Heidelberg (2013)
34. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838. Springer, Heidelberg (2000)
35. Lee, K., Lee, D.H., Yung, M.: Aggregating CL-signatures revisited: extended functionality and better efficiency. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 171–188. Springer, Heidelberg (2013)
36. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
37. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
38. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems (extended abstract). In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
39. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
40. Pointcheval, D., Sanders, O.: Short randomizable signatures. Cryptology ePrint Archive, Report 2015/525 (2015). http://eprint.iacr.org/2015/525
41. Schnorr, C.-P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
42. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)