Short Randomizable Signatures

David Pointcheval[1] and Olivier Sanders[1,2]

[1] École normale supérieure, CNRS & INRIA, Paris, France
[2] Orange Labs, Applied Crypto Group, Caen, 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]. 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 [34, 13] 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 [25, 9], 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 r and computing (a^r, b^r, c^r). 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 [25, 10, 9]. 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 [28]).

One might have thought that the seminal work of Groth and Sahai [31], 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 [30] 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, 16, 20, 5] 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 Figure 1 (see Section 7), our signature and verification algorithms are much more efficient.

Our work proceeds from the observation that most of the recent protocols [25, 10, 9] using CL-signatures require type 3 pairings for efficiency and security reasons (see [28]). 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 shows 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 was proved in the generic group model, which does not provide the same guarantees. However, as illustrated by [19, 11, 2], 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 [33, 27], where authentication must be performed in less than 300 ms, 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 proposed a MAC scheme, proven secure in the generic group model, with useful properties allowing to construct keyed-verification anonymous credentials (the secret-key analogue of standard anonymous credentials). Although our signature shares similarities with their scheme, it offers much more flexibility. Indeed, the construction from [22] 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 Section 2 and present new computational assumptions in Section 3. Section 4 describes our signature scheme whose conversion into a sequential aggregate signature scheme is described in Section 5. Section 6 describes a variant of our scheme allowing to sign committed values along with a protocol for proving knowledge of a signature. Finally, Section 7 provides a comparison with related works.

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 [28] 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 exist 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 favor 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 `Setup` algorithm which, on input a security parameter k, outputs $p.p.$, a description of the public parameters;
- the key generation algorithm `Keygen` which, on input $p.p$, outputs a pair of signing and verification keys $(\mathsf{sk}, \mathsf{pk})$ – we assume that sk contains pk, and that pk contains $p.p.$;

– 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) [29] 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 $p.p.$, a description of the public parameters;
– the key generation algorithm $\texttt{AS.Keygen}$ which, on input $p.p$, outputs a pair of signing and verification keys $(\mathsf{sk}, \mathsf{pk})$ – we assume that sk contains pk, and that pk contains $p.p.$;
– 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 signing key sk and a message m, 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 $p.p.$ 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 [41].

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) this 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 $6(q + q_G)^2/p$.*

Proof. Let g and $\widetilde{g}$ be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, x and y be the secret scalars that define (X, Y) and $(\widetilde{X}, \widetilde{Y})$, and $r_i \in \mathbb{Z}_p^*$ be the scalar such that the i^{th} oracle answer on scalar m_i is answered by $(h_i, t_i = h_i^{(x+y \cdot m_i)})$ with $h_i = g^{r_i}$.

In the following, we associate group elements with polynomials whose formal variables are the above unknown scalars: $x, y, r_1, \ldots, r_q$, with first all the inputs available to the adversary: $\widetilde{X} = x$ and $\widetilde{Y} = y$ in $\mathbb{G}_2$, $Y = y$ and $h_i = r_i$, $t_i = r_i(x + y \cdot m_i)$, for $i = 1, \ldots, q$, in $\mathbb{G}_1$. We must first prove that an adversary $\mathcal{A}$ is unable to symbolically produce a new valid tuple, and then that an accidental validity is quite unlikely.

For the output tuple (h^*, t^*) on a scalar m^*, since h^* and t^* are elements in $\mathbb{G}_1$, they can just be combinations of previous tuples (h_i, t_i), g, and Y (without any help from elements in $\mathbb{G}_2$): they have been built with queries to the oracle of internal law in $\mathbb{G}_1$, and so we know $((u_{i,1}, v_{i,1}, u_{i,2}, v_{i,2})_i, (w_1, w_2), (w'_1, w'_2)) \in \mathbb{Z}_p^{4q+4}$ such that:

$$g^{r^*} = h^* = g^{w_1} \cdot Y^{w'_1} \cdot \prod_{i=1}^{q} h_i^{u_{i,1}} \cdot t_i^{v_{i,1}} \quad \text{and} \quad g^{z^*} = t^* = g^{w_2} \cdot Y^{w'_2} \cdot \prod_{i=1}^{q} h_i^{u_{i,2}} \cdot t_i^{v_{i,2}},$$

and thus

$$r^* = w_1 + w_1' \cdot y + \sum_{i=1}^{q} (u_{i,1} \cdot r_i + v_{i,1}(x + y \cdot m_i) \cdot r_i)$$

$$\text{and} \quad z^* = w_2 + w_2' \cdot y + \sum_{i=1}^{q} (u_{i,2} \cdot r_i + v_{i,2}(x + y \cdot m_i) \cdot r_i).$$

The validity of the new tuple implies that $z^* = r^*(x + y \cdot m^*)$, which leads to:

$$\begin{aligned} w_2 + w_2' \cdot y + \sum_{i=1}^{q} (u_{i,2} \cdot r_i + v_{i,2}(xr_i + m_i \cdot yr_i)) \\ = w_1 \cdot x + w_1' \cdot xy + \sum_{i=1}^{q} (u_{i,1} \cdot xr_i + v_{i,1}(x^2 r_i + m_i \cdot xyr_i)) \\ + m^* \cdot (w_1 \cdot y + w_1' \cdot y^2 + \sum_{i=1}^{q} (u_{i,1} \cdot yr_i + v_{i,1}(xyr_i + m_i \cdot y^2 r_i))). \end{aligned}$$

For the two multivariable polynomials to be equal, the same monomials should appear on both sides:

- no monomials of degree 3 on the left, so $v_{i,1} = 0$ for all i;
- no term in r_i on the right, so $u_{i,2} = 0$ for all i;
- no constant term on the right, so $w_2 = 0$;
- no term in x nor xy on the left, so $w_1 = 0$ and w_1'=0:

$$w_2' \cdot y + \sum_{i=1}^{q} v_{i,2}(xr_i + m_i \cdot yr_i) = \sum_{i=1}^{q} u_{i,1} \cdot xr_i + m^* \cdot \sum_{i=1}^{q} u_{i,1} \cdot yr_i.$$

- no more term in y on the right, so w_2'=0:

$$\sum_{i=1}^{q} v_{i,2}(xr_i + m_i \cdot yr_i) = \sum_{i=1}^{q} u_{i,1} \cdot xr_i + m^* \cdot \sum_{i=1}^{q} u_{i,1} \cdot yr_i.$$

The monomials xr_i imply $v_{i,2} = u_{i,1}$ for all i, while the monomials yr_i imply $u_{i,1} \cdot m_i = u_{i,1} \cdot m^*$ for all i. Since $r^* \neq 0$ (otherwise $h^* = 1_{\mathbb{G}_1}$), there is at least one $u_{i,1} = v_{i,2} \neq 0$, and then $m^* = m_i$: the pair is not for a new scalar! An adversary is then unable to symbolically produce a valid tuple for a new scalar.

Now, it remains to evaluate the probability for an accidental validity: when two different polynomials involved in the answers to the oracles evaluate to the same value. Since the elements provided by the oracle are associated with polynomials of degree at most 2 and since the public elements are associated with polynomials of degree at most 1, the polynomials resulting from queries to the different group oracles are of degree at most 3 (because of pairing queries). Let q_G be the maximum number of group-oracle queries, there are thus at most $3 + 2q + q_G$ polynomials, and thus at most $(3 + 2q + q_G)^2/2$ pairs of distinct polynomials that could evaluate to the same value. By the Schwartz-Zippel lemma, the probability that such an event occurs is then upper-bounded by $3(3 + 2q + q_G)^2/2p \leq 6(q + q_G)^2/p$ which is negligible. □

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 consists of the following algorithms:

- $\mathtt{Setup}(1^k)$: Given a security parameter k, this algorithm outputs $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. These bilinear groups must be of type 3. In the following, we denote $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$;
- $\mathtt{Keygen}(p.p.)$: This algorithm selects $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ and $(x, y) \xleftarrow{\$} \mathbb{Z}_p^2$, 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 $h \xleftarrow{\$} \mathbb{G}_1^*$ 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 $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computing $\sigma' \leftarrow (\sigma_1^r, \sigma_2^r)$ which is still a valid signature on m: it corresponds to replace $h \in \mathbb{G}_1^*$ by $h' = h^r \in \mathbb{G}_1^*$.

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:

- $\texttt{Setup}(1^k)$: Given a security parameter k, this algorithm outputs $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. These bilinear groups must be of type 3. In the following, we denote $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$;
- $\texttt{Keygen}(p.p.)$: This algorithm selects $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ and $(x, y_1, \ldots, y_r) \xleftarrow{\$} \mathbb{Z}_p^{r+1}$, 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 $(\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$.
- $\texttt{Sign}(\mathsf{sk}, m_1, \ldots, m_r)$: This algorithm selects a random $h \xleftarrow{\$} \mathbb{G}_1^*$ and outputs $\sigma \leftarrow (h, h^{(x+\sum y_j \cdot m_j)})$.
- $\texttt{Verify}(\mathsf{pk}, (m_1, \ldots, m_r), \sigma)$: 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.

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 $\varepsilon - q/p$.*

Proof. Let $\mathcal{A}$ be an adversary against the EUF-CMA security of this scheme. We construct a reduction $\mathcal{R}$ using $\mathcal{A}$ against the EUF-CMA security of the previous single-message signature scheme. The challenger of the latter game will be denoted by $\mathcal{C}$.

- **Setup:** $\mathcal{R}$ receives from $\mathcal{C}$ a public key pk^* which contains the public parameters of the signature scheme $p.p.$ along with $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$. Next, it selects $\alpha_j, \beta_j \xleftarrow{\$} \mathbb{Z}_p$, for $j = 1, \ldots, r$, and sets $\widetilde{Y}_j \leftarrow \widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j}$. It outputs $\mathsf{pk} \leftarrow (\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_r)$.
- **Queries:** When $\mathcal{A}$ queries a signature on a vector $M_i = (m_{i,1}, \ldots, m_{i,r})$, $\mathcal{R}$ first requests from $\mathcal{C}$ a signature on $m_i = \sum \alpha_j m_{i,j}$ and so receives $\sigma = (\sigma_1, \sigma_2)$ such that $e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\sum \alpha_j m_{i,j}}) = e(\sigma_2, \widetilde{g})$. Next, it computes $\sigma_2' \leftarrow$

$\sigma_2 \cdot \sigma_1^{\sum \beta_j \cdot m_{i,j}}$ and returns (σ_1, σ_2') to $\mathcal{A}$:

$$\begin{aligned} e(\sigma_2', \widetilde{g}) &= e(\sigma_2 \cdot \sigma_1^{\sum \beta_j \cdot m_{i,j}}, \widetilde{g}) = e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{\sum \beta_j \cdot m_{i,j}}, \widetilde{g}) \\ &= e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\sum \alpha_j m_{i,j}}) \cdot e(\sigma_1, \widetilde{g}^{\sum \beta_j \cdot m_{i,j}}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}^{\alpha_j m_{i,j}} \widetilde{g}^{\beta_j \cdot m_{i,j}}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod (\widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j})^{m_{i,j}}) = e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_{i,j}}), \end{aligned}$$

which is a valid signature on M_i.

– **Output:** Eventually, $\mathcal{A}$ outputs a signature $\sigma = (\sigma_1, \sigma_2)$ on a vector $M^* = (m_1^*, \ldots, m_r^*)$. The signature σ is a valid forgery if

- $e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j^*}) = e(\sigma_2, \widetilde{g})$;
- for $i = 1, \ldots, q$, $M^* \neq M_i$.

If $\sum \alpha_j m_j^* = \sum \alpha_j m_{i,j}$, for some $i \in \{1, \ldots, q\}$, then $\mathcal{R}$ aborts. Otherwise, it outputs $\sigma^* = (\sigma_1^*, \sigma_2^*)$, with $\sigma_1^* \leftarrow \sigma_1$ and $\sigma_2^* \leftarrow \sigma_2 \cdot \sigma_1^{-\sum \beta_j \cdot m_j^*}$, together with $m^* \leftarrow \sum \alpha_j m_j^*$:

$$\begin{aligned} e(\sigma_2^*, \widetilde{g}) &= e(\sigma_2 \cdot \sigma_1^{-\sum \beta_j \cdot m_j^*}, \widetilde{g}) = e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1^{-\sum \beta_j \cdot m_j^*}, \widetilde{g}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}_j^{m_j^*}) \cdot e(\sigma_1, \widetilde{g}^{-\sum \beta_j \cdot m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod (\widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j})^{m_j^*}) \cdot e(\sigma_1, \prod \widetilde{g}^{-\beta_j \cdot m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \widetilde{Y}^{\alpha_j m_j^*}) = e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\sum \alpha_j m_j^*}) = e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{m^*}). \end{aligned}$$

Since m^* has never been asked to the signing oracle, this is a valid forgery under the public key pk^*.

We remark that, unless that adversary outputs a vector $M^* = (m_1^*, \ldots, m_r^*)$ such that $\sum \alpha_j m_j^* = \sum \alpha_j m_{i,j}$ for some vector $M_i = (m_{i,1}, \ldots, m_{i,r})$, a valid forgery makes $\mathcal{R}$ outputs a valid forgery against the single-message signature scheme. We thus have to prove that such linear relations are quite unlikely.

Let us denote y the scalar such that $\widetilde{Y} = \widetilde{g}^y$, and let us select $\gamma_j \xleftarrow{\$} \mathbb{Z}_p$, for $j = 1, \ldots, r$. We now set $\alpha_j' \leftarrow \alpha_j - \gamma_j$ and $\beta_j' \leftarrow \beta_j + y\gamma_j$, for $j = 1, \ldots, r$. One can remark that

$$\widetilde{Y}^{\alpha_j'} \widetilde{g}^{\beta_j'} = \widetilde{Y}^{\alpha_j - \gamma_j} \widetilde{g}^{\beta_j + y\gamma_j} = \widetilde{Y}^{\alpha_j} \widetilde{Y}^{-\gamma_j} \widetilde{g}^{\beta_j} \widetilde{Y}^{\gamma_j} = \widetilde{Y}^{\alpha_j} \widetilde{g}^{\beta_j} = \widetilde{Y}_j.$$

Hence, the public key is independent of the actual γ_j's, and thus reveals no information about the α_j's, and this is the same for the signatures that just depends on σ_1 chosen by the oracle and the public key.

As a consequence, the complete view of the adversary is totally independent of the α_j's. Hence, the probability that $\mathcal{R}$ aborts is upper-bounded by q/p. □

5 A Sequential Aggregate Signature

5.1 Our Construction

It is possible to slightly modify the scheme from section 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 section 4.2.

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

- $\mathtt{AS.Setup}(1^k)$: Given a security parameter k, this algorithm selects a random $x \in \mathbb{Z}_p$ and outputs $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, X, \widetilde{g}, \widetilde{X})$, where $X = g^x$ and $\widetilde{X} = \widetilde{g}^x$ for some generators $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$.
- $\mathtt{AS.Keygen}(p.p.)$: 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 $\widetilde{Y}$.
- $\mathtt{AS.Sign}(\mathsf{sk}, \sigma, (m_1, \ldots, m_r), (\mathsf{pk}_1, \ldots, \mathsf{pk}_r), m)$ proceeds as follows:
  - If $r = 0$, then $\sigma \leftarrow (g, X)$;
  - If $r > 0$ but $\mathtt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), \sigma, (m_1, \ldots, m_r)) = 0$, 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'.
- $\mathtt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), (m_1, \ldots, m_r), \sigma)$ 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 `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 Section 4.2.

5.2 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.*

Proof. Let $\mathcal{A}$ be an adversary against the existential unforgeability of our aggregate signature scheme, in the certified public key setting. We construct a reduction $\mathcal{R}$ using $\mathcal{A}$ against the existential unforgeability of the single-message signature scheme described in Section 4.1. The challenger of the latter game will be denoted by $\mathcal{C}$.

- **Setup:** $\mathcal{R}$ first initializes a key list `KeyList` as empty. Next, it gets from $\mathcal{C}$ a public key pk which contains the public parameters of the signature scheme $p.p.$ along with $(\widetilde{g}, \widetilde{X}, \widetilde{Y})$. It then requests a signature on 0 from $\mathcal{C}$ which returns $\tau = (\tau_1, \tau_2)$ such that $e(\tau_1, \widetilde{X}) = e(\tau_2, \widetilde{g})$. One can set $g \leftarrow \tau_1$ and $X \leftarrow \tau_2$. $\mathcal{R}$ then sets the public parameters of the aggregate signature scheme as $(p.p., g, X, \widetilde{g}, \widetilde{X})$ and sends $\mathsf{pk}^* \leftarrow \widetilde{Y}$ to $\mathcal{A}$. In the following, x^* and y^* will denote the (unknown) scalars such that $\widetilde{X} = \widetilde{g}^{x^*}$ (as well as $X = g^{x^*}$) and $\widetilde{Y} = \widetilde{g}^{y^*}$. The scalar y^* is thus the unknown secret key sk^* associated to pk^*.
- **Join Query:** When $\mathcal{A}$ asks for adding a public key pk_i, the certification process that includes a proof of knowledge of the associated secret keys sk_i allows $\mathcal{R}$ to extracts it: it thus stores pk_i in `KeyList`, and stores $(\mathsf{sk}_i, \mathsf{pk}_i)$ in its own list of signing/verification keys for future simulations.
- **Signature Query:** When $\mathcal{A}$ requests for an aggregation of a message m_i to the aggregate signature σ_i on messages $(m_{i,1}, \ldots, m_{i,r_i})$ under public keys $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i})$, if $r_i > 0$, $\mathcal{R}$ first checks the validity of σ_i, and aborts in the negative case. Then, it requests a signature on m_i from $\mathcal{C}$, which returns $\sigma = (\sigma_1, \sigma_2)$. All the public keys involved in aggregate signatures must have been previously certified, then $\mathcal{R}$ knows the associated secret keys $(\mathsf{sk}_{i,1}, \ldots, \mathsf{sk}_{i,r_i}) = (y_{i,1}, \ldots, y_{i,r_i})$: it selects a random $t \xleftarrow{\$} \mathbb{Z}_p$ and returns

$\sigma' \leftarrow (\sigma_1^t, (\sigma_2 \cdot \sigma_1^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}})^t)$. This signature satisfies

$$\begin{aligned} e(\sigma'_2, \widetilde{g}) &= e((\sigma_2 \cdot \sigma_1^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}})^t, \widetilde{g}) = e(\sigma_2^t, \widetilde{g}) \cdot e(\sigma_1^t, \widetilde{g})^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}} \\ &= e(\sigma_2, \widetilde{g})^t \cdot \prod_{j=1}^{r_i} e(\sigma_1^t, \widetilde{g}^{y_{i,j} \cdot m_{i,j}}) = e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{m_i})^t \cdot \prod_{j=1}^{r_i} e(\sigma_1^t, \mathsf{pk}_{i,j}^{m_{i,j}}) \\ &= e(\sigma_1^t, \widetilde{X} \cdot \mathsf{pk}^{m_i} \cdot \prod_{j=1}^{r_i} \mathsf{pk}_{i,j}^{m_{i,j}}) = e(\sigma'_1, \widetilde{X} \cdot \prod_{j=1}^{r_i} \mathsf{pk}_{i,j}^{m_{i,j}} \cdot \mathsf{pk}^{m_i}) \end{aligned}$$

which is thus a valid signature on the vector $((m_{i,j})_i, m_i)$ under the public keys $((\mathsf{pk}_{i,j})_i, \mathsf{pk}^*)$.

– **Output:** $\mathcal{A}$ eventually outputs an aggregate signature $\sigma = (\sigma_1, \sigma_2)$ on messages $(m_1^*, \ldots, m_r^*)$ under the public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_r)$. The aggregate signature σ is a valid forgery if the following conditions are satisfied:
  1. $\mathtt{AS.Verify}((\mathsf{pk}_1, \ldots, \mathsf{pk}_r), \sigma, (m_1^*, \ldots, m_r^*)) = 1$;
  2. For all $\mathsf{pk}_j \neq \mathsf{pk}^*$, $\mathsf{pk}_j \in \mathtt{KeyList}$;
  3. 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 $i = 1, \ldots, q$.

  The first condition implies that $e(\sigma_1, \widetilde{X} \prod \mathsf{pk}_j^{m_j^*}) = e(\sigma_2, \widetilde{g})$, while the second implies that $\mathcal{R}$ knows y_j such that $\mathsf{pk}_j = \widetilde{g}^{y_j}$, for $j = 1, \ldots, r$, when $\mathsf{pk}_j \neq \mathsf{pk}^*$. The third one implies that there exists (a unique, since the public keys are distinct) $j^* \in [1, r]$ such that $\mathsf{pk}^* = \mathsf{pk}_{j^*}$: $\mathcal{R}$ can compute $\sigma^* = (\sigma_1^* \leftarrow \sigma_1, \sigma_2^* \leftarrow \sigma_2 \cdot \prod_{j \neq j^*} \sigma_1^{-y_j \cdot m_j^*})$ which satisfies

$$\begin{aligned} e(\sigma_2^*, \widetilde{g}) &= e(\sigma_2 \cdot \prod_{j \neq j^*} \sigma_1^{-y_j \cdot m_j^*}, \widetilde{g}) = e(\sigma_2, \widetilde{g}) \cdot \prod_{j \neq j^*} e(\sigma_1^{-y_j \cdot m_j^*}, \widetilde{g}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \mathsf{pk}_j^{m_j^*}) \cdot \prod_{j \neq j^*} e(\sigma_1, \widetilde{g}^{-y_j \cdot m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \prod \mathsf{pk}_j^{m_j^*}) \cdot \prod_{j \neq j^*} e(\sigma_1, \widetilde{Y}_j^{-m_j^*}) \\ &= e(\sigma_1, \widetilde{X} \cdot \mathsf{pk}_{j^*}^{m_{j^*}^*}) = e(\sigma_1^*, \widetilde{X} \cdot \mathsf{pk}_{j^*}^{m_{j^*}^*}) \end{aligned}$$

  Since $\mathsf{pk}_{j^*} = \mathsf{pk}^*$ and $m^* = m_{j^*}^*$ was not query to $\mathcal{C}$, this last equation shows that (m^*, σ^*) is a valid forgery (for the single-message signature scheme described in Section 4.1) under pk^*.

But one has to additionally show that signature queries are correctly simulated with $\sigma' \leftarrow (\sigma_1^t, (\sigma_2 \cdot \sigma_1^{\sum_{j=1}^{r_i} y_{i,j} \cdot m_{i,j}})^t)$, with $\sigma = (\sigma_1, \sigma_2)$ a signature of m_1 under pk^*, and t a random scalar, whereas the real signature should be $\sigma' = (\sigma_{i1}^t, (\sigma_{i2} \cdot \sigma_{i1}^{y \cdot m})^t)$, where $\sigma_i = (\sigma_{i1}, \sigma_{i2})$ was a valid signature on messages $(m_{i,1}, \ldots, m_{i,r_i})$ under public keys $(\mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,r_i})$ (or $\sigma_i = (g, X)$ when $r = 0$), and t a random scalar.

But one can note that in both cases σ'_1 is a random element in $\mathbb{G}_1^*$, while σ'_2 is the unique element that satisfies $e(\sigma'_2, \widetilde{g}) = e(\sigma'_1, \widetilde{X} \cdot \prod_{j=1}^{r_i} \mathsf{pk}_{i,j}^{m_{i,j}} \cdot \mathsf{pk}^{m_i})$. Hence, the perfect simulation. □

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 [12, 26, 10], 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 Section 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:

- $\mathtt{Setup}(1^k)$: Given a security parameter k, this algorithm outputs $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. 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.
- $\mathtt{Keygen}(p.p.)$: 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 $\mathsf{pk} \leftarrow (g, Y, \widetilde{g}, \widetilde{X}, \widetilde{Y})$.
- $\mathtt{Protocol}$: 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 Section 4.1. However, because of the additional elements in the public key, the EUF-CMA security of this single-message signature scheme relies on the Assumption 1.

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

- $\texttt{Setup}(1^k)$: Given a security parameter k, this algorithm outputs $p.p. \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. 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.
- $\texttt{Keygen}(p.p.)$: 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 knowkedge 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 Section 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 Section 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 [40]) 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 $(m_1, \ldots, m_r)$.*

Proof. The completeness follows from the one of π and from the fact that σ' is a valid signature on the block $(m_1, \ldots, m_r, t)$ under the public key $(\widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_{r+1})$, where $\widetilde{Y}_{r+1} = \widetilde{g}$.

To prove the zero-knowledge property we construct a valid simulator $\mathcal{S}$. First, $\mathcal{S}$ generates two random elements σ'_1 and σ'_2 of $\mathbb{G}_1$ and sets $\sigma' \leftarrow (\sigma'_1, \sigma'_2)$. The pair σ' is then correctly distributed because r and t were randomly generated in Step 1. Next, $\mathcal{S}$ runs the simulator of the proof π to simulate Step 2. The zero-knowledge property of π implies then the one of our protocol.

Finally, let us consider a prover $\mathcal{P}$ such that the verifier's acceptance is non-negligible. We construct an extractor $\mathcal{E}$ using $\mathcal{P}$ to output a valid signature σ on a block of message. Since π is a proof of knowledge, $\mathcal{E}$ can run the associated extractor to get a block $(m_1, \ldots, m_r)$ along with t such that:

$$e(\sigma'_1, \widetilde{X}) \cdot \prod e(\sigma'_1, \widetilde{Y}_j)^{m_j} \cdot e(\sigma'_1, \widetilde{g})^t = e(\sigma'_2, \widetilde{g})$$

$\mathcal{E}$ can then compute $\sigma = (\sigma_1, \sigma_2) \leftarrow (\sigma'_1, \sigma'_2 \cdot (\sigma'_1)^{-t})$ which is a valid signature on the block $(m_1, \ldots, m_r)$ since:

$$\begin{aligned} e(\sigma_1, \widetilde{X}) \cdot \prod e(\sigma_1, \widetilde{Y}_j)^{m_j} &= e(\sigma'_1, \widetilde{X}) \cdot \prod e(\sigma'_1, \widetilde{Y}_j)^{m_j} \cdot e(\sigma'_1, \widetilde{g})^t \cdot e(\sigma'_1, \widetilde{g})^{-t} \\ &= e(\sigma'_2, \widetilde{g}) \cdot e(\sigma'_1, \widetilde{g})^{-t} \\ &= e(\sigma_2, \widetilde{g}) \end{aligned}$$

For any prover $\mathcal{P}$ accepted by a verifier with non-negligible probability, we can construct $\mathcal{E}$ that outputs a valid signature by interacting with $\mathcal{P}$. Our protocol is therefore a valid proof of knowledge.

7 Efficiency and Applications

We compare in Figure 1 the efficiency of our scheme with the ones of CL-signatures [19] and BBS-signatures [12, 4] 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 Figure 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 Section 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.

In section 6, we have shown that our signatures share the same features than CL-signatures and so can replace them in many applications. We provide below some examples and describe the performance gains in Figures 2 and 3.

7.1 Group Signature

Let us consider the shortest group signature (in the ROM) proposed in [10]. A user of this system first gets a CL-signature σ on a secret value sk and then uses it to prove membership in the group. The main point for anonymity is that he can provide randomized versions of σ when generating group signatures. They are unlinkable. Indeed, linking (a, b, c) with (a^t, b^t, c^t) for some $t \in \mathbb{Z}_p$ is equivalent to breaking the DDH assumption in $\mathbb{G}_1$.

Therefore, the construction of [10] does not specifically requires CL-signatures but simply a signature scheme which allows (1) to sign a committed message and (2) to randomize the signatures. Both properties are achieved by our scheme which can thus be used to instantiate this group signature. For completeness we describe below the resulting algorithms (we borrow the notations of [10]).

- $\mathtt{GSetup}(1^k)$: The Group Manager runs the `Setup` and the `Keygen` algorithms of the single-message signature scheme described in Section 4.1 to get $\mathsf{sk} = (x, y)$ and $\mathsf{pk} = (\widetilde{g}, \widetilde{X}, \widetilde{Y})$. He then sets the group public key gpk as pk along with some generator $g \in \mathbb{G}_1$, and sets his secret key gmsk as sk.
- $\mathtt{PKIJoin}(i, 1^k)$: The user i generates $(\mathsf{usk}[i], \mathsf{upk}[i]) \leftarrow \Sigma.\mathtt{Keygen}(1^k)$ for some digital signature scheme Σ and sends $\mathsf{upk}[i]$ to a Certification Authority. We assume then that $\mathsf{upk}[i]$ is publicly available such that anyone can get an authentic copy of it.
- `GJoin` : To join the group, a user i starts an interactive protocol with the group manager. He first generates a secret $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$ and sends the pair $(\tau, \widetilde{\tau}) \leftarrow (g^{\mathsf{sk}_i}, \widetilde{Y}^{\mathsf{sk}_i})$ along with a signature $\eta \leftarrow \Sigma.\mathtt{Sign}(\mathsf{usk}[i], \tau)$ to the group manager. The latter then checks the validity of η and the one of the pair $(\tau, \widetilde{\tau})$ by testing whether $e(\tau, \widetilde{Y}) = e(g, \widetilde{\tau})$ or not. Next, the user starts an interactive proof of knowledge of sk_i, such as the Schnorr's protocol [40]. If everything is correct, the group manager generates a random $u \xleftarrow{\$} \mathbb{Z}_p$ and computes $\sigma \leftarrow (\sigma_1, \sigma_2) \leftarrow (g^u, (g^x \cdot (\tau)^y)^u)$ which is a valid signature on sk_i, as explained in Section 6. Finally, the group manager stores $(i, \tau, \eta, \widetilde{\tau})$ in a secret register and sends σ to the user who sets gsk_i as $(\mathsf{sk}_i, \sigma, e(\sigma_1, \widetilde{Y}))$. Actually, the user does not need to store $e(\sigma_1, \widetilde{Y})$ but this will allow him to avoid pairing computations during the `GSign` algorithm.
- $\mathtt{GSign}(\mathsf{gsk}_i, m)$: To sign a message m the user first randomizes σ by generating a random t and computing $(\sigma'_1, \sigma'_2) \leftarrow (\sigma_1^t, \sigma_2^t)$ and then computes a signature of knowledge of sk_i. To do so, he selects a random $k \xleftarrow{\$} \mathbb{Z}_p$ and computes $e(\sigma'_1, \widetilde{Y})^k \leftarrow e(\sigma_1, \widetilde{Y})^{k \cdot t}$ and $c \leftarrow \mathcal{H}(\sigma'_1, \sigma'_2, e(\sigma_1, \widetilde{Y})^{k \cdot t}, m)$ for some hash function $\mathcal{H}$ which will be modelled as a random oracle in the security proof. Finally, he computes $s \leftarrow k + c \cdot \mathsf{sk}_i$ and outputs $(\sigma'_1, \sigma'_2, c, s) \in \mathbb{G}_1^2 \times \mathbb{Z}_p^2$ as the group signature μ on m.
- $\mathtt{GVerify}(\mathsf{gpk}_i, m)$: To verify a signature $\mu = (\sigma_1, \sigma_2, c, s)$ on m the verifier computes $R \leftarrow (e(\sigma_1^{-1}, \widetilde{X}) \cdot e(\sigma_2, \widetilde{g}))^{-c} \cdot e(\sigma_1^s, \widetilde{Y})$ and then checks whether $c = \mathcal{H}(\sigma_1, \sigma_2, R, m)$. This actually corresponds to the verification of the signature of knowledge. If it is valid then he outputs 1. Otherwise, he outputs

0. Correctness follows from the fact that, if (σ_1, σ_2) is a valid signature on sk_i, then:

$$\begin{aligned}(e(\sigma_1^{-1}, \widetilde{X}) \cdot e(\sigma_2, \widetilde{g}))^{-c} \cdot e(\sigma_1^s, \widetilde{Y}) \\ &= e(\sigma_1, \widetilde{Y})^k \cdot [e(\sigma_1, \widetilde{Y})^{\mathsf{sk}_i} \cdot e(\sigma_1, \widetilde{X}) \cdot e(\sigma_2^{-1}, \widetilde{g})]^c \\ &= e(\sigma_1, \widetilde{Y})^k \cdot [e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{\mathsf{sk}_i}) \cdot e(\sigma_2, \widetilde{g})^{-1}]^c \\ &= e(\sigma_1, \widetilde{Y})^k\end{aligned}$$

– $\mathtt{GOpen}(\mathsf{gmsk}, m, \mu)$: To open a signature μ, the group manager tests, for all entries $(i, \tau_i, \eta_i, \widetilde{\tau}_i)$, whether $e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1, \widetilde{X})^{-1} = e(\sigma_1, \widetilde{\tau})$ holds until he gets a match. He then outputs the corresponding (i, τ_i, η_i) along with a proof of knowledge of a valid $\widetilde{\tau}_i$. This proof can then be checked by anyone to verify the validity of the opening.

This example shows that rewriting the algorithms of existing constructions in this new context is quite obvious and so that our signature allows efficiency gains without the need for designing a new scheme. In particular, the security analysis can be directly derived from the original one. The only change is that the security properties will now rely on the unforgeability of our scheme instead of the CL one. Here, the use of type 3 pairings is not even a restriction since it was already required by the original construction. Figure 2 shows the performance improvements we achieve compared to the latter.

Group Signature	Size of Sig.	Sig. Cost	Verif. Cost
Bichsel *et al* [10]	3 $\mathbb{G}_1$ + 2 $\mathbb{Z}_p$	3 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$ + 1 $\mathtt{H}$	5 $\mathtt{P}$ + 1 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$
Section 7.1	2 $\mathbb{G}_1$ + 2 $\mathbb{Z}_p$	2 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$ + 1 $\mathtt{H}$	3 $\mathtt{P}$ + 1 $\mathtt{E}_{\mathbb{G}_1}$ + 1 $\mathtt{E}_{\mathbb{G}_T}$

Fig. 2. Efficiency comparison between the original version of the group signature scheme described in [10] and the one instantiated with our signature scheme, as described in Section 7.1. Here, $\mathtt{E}_{\mathbb{G}_1}$ (resp. $\mathtt{E}_{\mathbb{G}_T}$) refers to the cost of an exponentiation in $\mathbb{G}_1$ (resp. $\mathbb{G}_T$), $\mathtt{P}$ to the cost of a pairing computation and $\mathtt{H}$ to the cost of hashing elements to $\mathbb{Z}_p$. We do not consider operations in $\mathbb{Z}_p$ since their cost is negligible compared to the other ones.

7.2 Anonymous Credentials

Anonymous credentials allow users to prove possession of credentials without revealing any other information about themselves. Ideally, different uses of the same credential should be unlinkable. Moreover, users should be able to privately obtain credentials and then prove various statements about them without revealing them. In the bilinear setting, the Camenisch-Lysyanskaya scheme [19] and the ones from [12, 4] fulfill all these requirements. Some other schemes (such

as the one of [32]) achieve a remarkable efficiency but at the cost of loosing at least one of these properties.

The constructions of [19, 4] follow the same framework. Issuance of a credential consists in a protocol where the user sends a commitment of its attributes (which may be information-theoretically hidden) and then proves knowledge of them to the issuer. If the latter is convinced, he returns a signature σ on the block of committed valued. Once the user has received σ, he can prove possession of the credential by providing a proof of knowledge of this signature on its attributes.

This framework can therefore be instantiated with our signature scheme. The issuance protocol is then the one for signing committed messages we described in section 6.1 while presentation of a credential consists in producing a proof of knowledge of a signature, as in section 6.2.

Anonymous Credentials	Issuing		Showing		
	User	Issuer	User	Verifier	Data Sent
CL [19]	$(r+1)\,\mathtt{E}_{\mathbb{G}_1}$ + $PK\{\mathtt{E}_{\mathbb{G}_1}[r+1]\}$	$(2r+4)\,\mathtt{E}_{\mathbb{G}_1}$ + $\mathtt{Ver}(PK)$	$(2r+4)\,\mathtt{E}_{\mathbb{G}_1}$ + $PK\{\mathtt{P}[r+2]\}$	$(4r+2)\,\mathtt{P}$ + $\mathtt{Ver}(PK)$	$(2r+3)\,\mathbb{G}_1$ + $\lvert PK\rvert$
BBS+ [4]	$(r+1)\,\mathtt{E}_{\mathbb{G}_1}$ + $PK\{\mathtt{E}_{\mathbb{G}_1}[r+1]\}$	$2\,\mathtt{E}_{\mathbb{G}_1}$ + $\mathtt{Ver}(PK)$	$3\,\mathtt{E}_{\mathbb{G}_1}$ + $PK\{\mathtt{P}[r+4]$ + $\mathtt{E}_{\mathbb{G}_1}[2]+\mathtt{E}_{\mathbb{G}_1}[3]\}$	$\mathtt{Ver}(PK)$	$2\,\mathbb{G}_1 + \lvert PK\rvert$
Section 7.2	$(r+1)\,\mathtt{E}_{\mathbb{G}_1}$ + $PK\{\mathtt{E}_{\mathbb{G}_1}[r+1]\}$	$2\,\mathtt{E}_{\mathbb{G}_1}$ + $\mathtt{Ver}(PK)$	$2\,\mathtt{E}_{\mathbb{G}_1}$ + $PK\{\mathtt{P}[r+1]\}$	$\mathtt{Ver}(PK)$	$2\mathbb{G}_1 + \lvert PK\rvert$

Fig. 3. Efficiency comparison between related works. Here, r refers to the number of attributes to be certified, $\mathtt{E}_{\mathbb{G}_1}$ to the cost of an exponentiation in $\mathbb{G}_1$ and $\mathtt{P}$ to the cost of a pairing computation. $PK\{\mathtt{E}_{\mathbb{G}_1}[n]\}$ (resp. $PK\{\mathtt{P}[n]\}$) denotes the cost of proving knowledge of n secret scalars involved in a multi-exponentiation (resp. pairing-product) equation, $\mathtt{Ver}(PK)$ the cost of verifying this proof and $\lvert PK\rvert$ the size of the proof transcript.

Figure 3 shows that the anonymous credentials from [19] suffer from the linear size of the CL-signatures but profits from the randomizability of the latter. On the contrary, those from [4] profits from the constant size of BBS-signatures but require to prove more complex statements since these signatures cannot be revealed. Using our signatures for anonymous credentials combines the advantages of both solutions since they offer both constant-size and randomizability.

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.

References

1. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, August 2010.
2. Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, August 2011.
3. Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Structure-preserving signatures from type II pairings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 390–407. Springer, 2014.
4. Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125. Springer, September 2006.
5. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 1087–1098. ACM Press, November 2013.
6. Niko Bari and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, May 1997.
7. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.
8. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, February 2005.
9. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Sec.*, 12(3):219–249, 2013.
10. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, September 2010.

11. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.
12. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.
13. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, August 2001.
14. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, May 2003.
15. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04: 11th Conference on Computer and Communications Security*, pages 132–145. ACM Press, October 2004.
16. Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. *ACM Trans. Inf. Syst. Secur.*, 15(1):4, 2012.
17. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, May 2001.
18. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, September 2002.
19. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, August 2004.
20. Sébastien Canard and Roch Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13: 8th Conference on Computer and Communications Security*, pages 381–392. ACM Press, May 2013.
21. Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible e-cash made practical. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 77–100. Springer, 2015.
22. Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 1205–1216. ACM Press, November 2014.
23. Sanjit Chatterjee and Alfred Menezes. Type 2 structure-preserving signature schemes revisited. *IACR Cryptology ePrint Archive*, 2014:635, 2014.
24. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, April 1991.

25. Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient DAA scheme. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, Passau, Germany, April 14-16, 2010. Proceedings*, volume 6035 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2010.
26. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, September 2006.
27. Nicolas Desmoulins, Roch Lescuyer, Olivier Sanders, and Jacques Traoré. Direct anonymous attestations with dependent basename opening. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, volume 8813 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2014.
28. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
29. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
30. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, December 2007.
31. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.
32. Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511. Springer, December 2014.
33. Gesine Hinterwälder, Christian T. Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. Efficient e-cash in practice: Nfc-based payments for public transportation systems. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, volume 7981 of *Lecture Notes in Computer Science*, pages 40–59. Springer, 2013.
34. Antoine Joux. A one round protocol for tripartite diffie-hellman. In Wieb Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.
35. Kwangsu Lee, Dong Hoon Lee, and Moti Yung. Aggregating CL-signatures revisited: Extended functionality and better efficiency. In Ahmad-Reza Sadeghi, editor, *FC 2013: 17th International Conference on Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 171–188. Springer, April 2013.
36. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, May / June 2006.

37. Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90. Springer, May 2004.
38. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999: 6th Annual International Workshop on Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, August 1999.
39. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, August 1991.
40. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, August 1989.
41. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, May 1997.