Abstract

This paper presents efficient structure-preserving signature schemes based on simple assumptions such as Decisional-Linear. We first give two general frameworks for constructing fully secure signature schemes from weaker building blocks such as variations of one-time signatures and random-message secure signatures. They can be seen as refinements of the Even-Goldreich-Micali framework, and preserve many desirable properties of the underlying schemes such as constant signature size and structure preservation. We then instantiate them based on simple (i.e., not q-type) assumptions over symmetric and asymmetric bilinear groups. The resulting schemes are structure-preserving and yield constant-size signatures consisting of 11 to 14 group elements, which compares favorably to existing schemes whose security relies on q-type assumptions.

Keywords: Structure-preserving signatures, Tagged one-time signatures, Partially one-time signatures, Extended random message attacks
Contents

1 Introduction 1
   1.1 Our contribution 1
   1.2 Related Works 2

2 Preliminaries 3
   2.1 Notation 3
   2.2 Bilinear groups 3
   2.3 Assumptions 3

3 Definitions 4
   3.1 Common setup 4
   3.2 Signature schemes 5
   3.3 Partial one-time and tagged one-time signatures 6
   3.4 Structure-preserving signatures 7

4 Generic Constructions 7
   4.1 SIG1: Combining tagged one-time and RMA-secure signatures 7
   4.2 SIG2: Combining partial one-time and XRMA-secure signatures 9

5 Instantiating SIG1 9
   5.1 Setup for Type-I groups 10
   5.2 Tagged one-time signature scheme 10
   5.3 RMA-secure signature scheme 12
   5.4 Security and efficiency of resulting SIG1 19

6 Instantiating SIG2 20
   6.1 Setup for Type-III groups 20
   6.2 Partial one-time signatures for unilateral messages 21
   6.3 Partial one-time signatures for bilateral messages 23
   6.4 XRMA-secure signature scheme 24
   6.5 Security and efficiency of resulting SIG2 28

7 Applications 30

8 Conclusions and Open Questions 31

A Waters’ Dual System Signature Scheme 33
1 Introduction

A structure-preserving signature (SPS) scheme \[4\] is a digital signature scheme with two structural properties: (i) the verification keys, messages, and signatures are all elements of a bilinear group; and (ii) the verification algorithm checks a conjunction of pairing product equations over the key, the message and the signature. This makes them compatible with the efficient non-interactive proof system for pairing-product equations by Groth and Sahai (GS) \[37\]. Structure-preserving cryptographic primitives promise to combine the advantages of optimized number theoretic non-blackbox constructions with the modularity and insight of protocols that use only generic cryptographic building blocks.

Indeed the instantiation of known generic constructions with an SPS scheme and the GS proof system has led to many new and more efficient schemes: Groth \[36\] showed how to construct an efficient simulation-sound zero-knowledge proof system (ss-NIZK) building on generic constructions of \[24, 47, 42\]. Abe et al. \[4, 8\] show how to obtain efficient round-optimal blind signatures by instantiating a framework by Fischlin \[27\]. SPS are also important building blocks for a wide range of cryptographic functionalities such as anonymous proxy signatures \[29\], delegatable anonymous credentials \[10\], transferable e-cash \[30\] and compact verifiable shuffles \[21\]. Most recently, \[38\] show how to construct a structure preserving tree-based signature scheme with a tight security reduction following the approach of \[33, 25\]. This signature scheme is then used to build a ss-NIZK which in turn is used with the Naor-Yung-Sahai \[43, 46\] paradigm to build the first CCA secure public-key encryption scheme with a tight security reduction. Examples for other schemes that benefit from efficient SPS are \[11, 15, 12, 40, 34, 9, 45, 31, 28, 35\].

Because properties (i) and (ii) are the only dependencies on the SPS scheme made by these constructions, any structure-preserving signature scheme can be used as a drop-in replacement. Unfortunately, all known efficient instantiations of SPS \[8, 4, 5\] are based on so-called \(q\)-type or interactive assumptions. An open question since Groth’s seminal work \[36\] (only partially answered by \[20\]) is to construct a SPS scheme that is both efficient – in particular constant-size in the number of signed group elements – and that is based on assumptions that are as weak as those required by the GS proof system itself.

1.1 Our contribution

We begin by presenting two new generic constructions of signature schemes that are secure against chosen message attacks (CMA) from variations of one-time signatures and signatures secure against random message attacks (RMA). Both constructions inherit the structure-preserving and constant-size properties from the underlying components. We then instantiate the building blocks with the desired properties over bilinear groups. They yield constant-size structure-preserving signature schemes whose signatures consist of only 11 to 14 group elements and whose security can be proven based on simple assumptions such as Decisional-Linear (DLIN) for symmetric bilinear groups and analogues of DDH and DLIN for asymmetric bilinear groups. These are the first constant-size structure-preserving signature schemes that eliminate the use of interactive or \(q\)-type assumptions while achieving reasonable efficiency. We give more details on our generic constructions and their instantiations:

- The first generic construction (SIG1, Section 4.1) combines a new variation of one-time signatures which we call tagged one-time signatures (TOS) and signatures secure against random message attacks (RMA). A TOS is a signature scheme that attaches a fresh tag to each signature. It is unforgeable with respect to tags used only once. In our construction, a message is signed with our TOS using a fresh random tag, and then the tag is signed with the second signature scheme, denoted by rSIG. Since rSIG only signs random tags, RMA-security is sufficient. In Section 5 we construct structure-preserving TOS and rSIG based on DLIN over symmetric (Type-I) bilinear groups. Our TOS yields constant-size signatures and optimally small tags that consists of only one group element. The resulting structure-preserving signature scheme produces signatures consisting of 14 group elements, and relies solely on the DLIN assumption.\[3\]

- The second generic construction (SIG2, Section 4.2) combines partial one-time signatures and signatures secure against extended random message attacks (XRMA). The latter is a new notion that we explain below. A partial one-time signature scheme, denoted by POS, is a one-time signature scheme

\[1\] The optimal TOS proposed in this paper was first presented in \[3\]. We included it here as it saves one group element in a tag compared to the original construction in \[2\], and reduces the resulting signature size from 17 in \[2\] to 14.
in which only a part of the key is renewed for every signing operation. The notion was first introduced by Bellare and Shoup \cite{BellareShoup99} under the name of two-tier signatures. In our construction, a message is signed with POS and then the one-time portion of the public-key is certified by the second signature scheme, denoted by \textit{xSIG}. The difference between a TOS and POS is that a one-time public-key is associated with a one-time secret-key. Since the one-time secret-key is needed for signing, it must be known to the reduction in the security proof. XRMA-security guarantees that \textit{xSIG} is unforgeable even if the adversary is given auxiliary information associated with the randomly chosen messages (e.g. the random coins used for selecting the message). The auxiliary information allows the reduction algorithm to secure the second scheme to use the one-time secret key to generate the POS component correctly.

In Section \ref{sec:construction} we construct structure-preserving POS and \textit{xSIG} signature schemes based on assumptions that are analogues of DDH and DLIN in Type-III bilinear groups. The resulting \textit{SIG2} is structure-preserving and produces signatures consisting of 11 or 14 group elements depending on whether messages belong to either or both source groups.

The role of TOS and POS is to compress a message into a constant number of random group elements. This observation is interesting in light of \cite{Bar-IoffeGoldreich95} that implies the impossibility of constructing collision resistant and shrinking structure-preserving hash functions, which could immediately yield constant-size signatures. Our (extended) RMA-secure signature schemes are structure-preserving variants of Waters’ dual-signature scheme \cite{Waters09}. In general, the difficulty of constructing CMA-secure SPS arises from the fact that the exponents of the group elements chosen by the adversary as a message are not known to the reduction in the security proof. On the other hand, for RMA security, it is the challenger that chooses the message and therefore the exponents can be known in reductions. This is the crucial advantage for constructing (extended) RMA-secure structure-preserving signature schemes based on Waters’ dual-signature scheme.

As our SPSs can be drop-in replacements for existing SPS, we only briefly introduce recent applications in Section \ref{sec:applications}. They include group signatures, tightly-secure structure-preserving signatures and public-key encryption, and efficient adaptive oblivious transfer.

1.2 Related Works

On Generic Constructions: Even, Goldreich and Micali \cite{EvenGoldreichMicali93} proposed a generic framework (the EGM framework) that combines a one-time signature scheme and a signature scheme that is secure against non-adaptive chosen message attacks (NACMA) to construct a signature scheme that is secure against adaptive chosen message attacks (CMA).

In fact, our generic constructions can be seen as refinements of the EGM framework. There are two reasons why the original framework falls short for our purpose. The first is that relaxing to NACMA does not seem to help much in constructing efficient structure-preserving signatures since the messages are still under the control of the adversary and the exponents of the messages are not known to the reduction algorithm in the security proof. As mentioned above, resorting to (extended) RMA is a great help in this regard. In \cite{EvenGoldreichMicali93}, they also showed that CMA-secure signatures exist \textit{i.e.} RMA-secure signatures exist. The proof, however, does not follow their framework and their impractical construction is mainly a feasibility result. In fact, we argue that RMA-security alone is not sufficient for the original EGM framework. As mentioned above, the necessity of XRMA security arises in the reduction that uses RMA-security to argue security of the ordinary signature scheme, as the reduction not only needs to know the random one-time public-keys, but also their corresponding one-time secret keys in order to generate the one-time signature components of the signatures. The auxiliary information in the XRMA definition facilitates access to these secret keys. Similarly, tagged one-time signatures avoid this problem as tags do not have associated secret values. The second reason is that the EGM approach is not quite suited to our task is that the EGM framework produces signatures that are linear in the size of one-time public-keys of the one-time signature scheme, and known structure-preserving one-time signature schemes have one-time public-keys that scale linearly with the number of group elements to be signed. Here, tagged or partial one-time signature schemes come in handy as they have one-time public-keys separated from long-term public-keys. Thus, to obtain constant-size signatures, we only require the one-time keys to be constant-size while allowing the long-term part to scale in the size of the message.
On Efficient Instantiations: All previous constructions of structure-preserving signature schemes are either inefficient, or use strong assumptions, or do not yield constant-size signatures. In particular, there are few schemes that base on simple assumptions. Hofheinz and Jager [38] constructed an SPS scheme by following the EGM framework. The resulting scheme allows a tight security reduction to DLIN but the size of signatures depends logarithmically on the number of signing operations as their NACMA-secure scheme is tree-based (like the Goldwasser-Micali-Rivest signature scheme [33]). Chase and Kohlweiss [20] and Camenisch, Dubovitskaya, and Haralambiev [18] constructed SPS schemes with security based on DLIN that improve the performance of Groth’s scheme [36] by several orders of magnitude. The size of the resulting signatures, however, is still linear in the number of signed group elements, and an order of magnitude larger than in our constructions. Finally, Camenisch, Dubovitskaya, and Haralambiev constructed a constant-size SPS scheme based on simple assumptions over composite-order groups [17].

2 Preliminaries

2.1 Notation

By \( X := Y \), we denote that object \( Y \) is referred to as \( X \). For set \( X \), notation \( a \leftarrow X \) denotes a uniform sampling from \( X \). Multiple independent samples from the same set \( X \) are denoted by \( a_1, a_2, a_3, \ldots \leftarrow X \). By \( Y \leftarrow A(X) \), we denote the process where algorithm \( A \) is executed with \( X \) as input and its output is labeled as \( Y \). When \( A \) is an oracle algorithm that interacts with oracle \( O \), it is denoted as \( Y \leftarrow A^O(X) \). By \( \Pr_{X \mid A_1, A_2, \ldots, A_k}[Y] \) we denote the probability that event \( X \) happens after executing the sequence of algorithms \( A_1, \ldots, A_k \). The probability is taken over all coin flips observed in \( A_1, \ldots, A_k \) unless otherwise noted. We say that a function \( \epsilon \) is negligible in security parameter \( \lambda \) if \( \epsilon \approx \lambda^{-c} \) holds for all constant \( c > 0 \) and all sufficiently large \( \lambda \). We refer to probabilistic polynomial time algorithms as p.p.t. algorithms. Unless stated otherwise, we assume that all algorithms are potentially probabilistic.

2.2 Bilinear groups

Let \( \mathcal{G} \) be a bilinear group generator that takes security parameter \( 1^\lambda \) and outputs a description of bilinear groups \( \Lambda := (p, G_1, G_2, G_T, e) \), where \( G_1, G_2 \) and \( G_T \) are groups of prime order \( p \), and \( e \) is an efficient and non-degenerate bilinear map \( G_1 \times G_2 \rightarrow G_T \). In this paper, generators for \( G_1 \) and \( G_2 \) are implicit in \( \Lambda \), and default random generators \( G \) and \( \hat{G} \) are chosen explicitly and independently. Groups \( G_1 \) and \( G_2 \) are called the source groups and \( G_T \) is called the target group. We use multiplicative notation for \( G_1, G_2 \) and \( G_T \). By \( G_1^{\ast} \), we denote \( G_1 \setminus \{1\} \), which is the set of all elements in \( G_1 \) except the identity. The same applies to \( G_2 \) and \( G_T \) as well. Following the terminology in [32] we say that \( \Lambda \) is Type-III when there is no efficient mapping between \( G_1 \) and \( G_2 \) in either direction.

In the Type-III setting, we denote elements in \( G_2 \) by putting a tilde on a variable like \( \hat{X} \) for visual aid. By using the same letter for elements in \( G_2 \) and \( G_1 \) with a hat on the \( G_2 \) element, e.g., \( X \) and \( \hat{X} \), we denote a pair of elements in relation \( \log_G X = \log_{\hat{G}} \hat{X} \). Should their relation be explicitly stated, we write \( X \sim \hat{X} \). Note that default random generators \( G \) and \( \hat{G} \) are independent each other but notational consistency retains.

We count the number of group elements to measure the size of cryptographic objects such as keys, messages, and signatures. For Type-III groups, we denote the size by \((x, y)\) when it consists of \( x \) and \( y \) elements from \( G_1 \) and \( G_2 \), respectively. We refer to the setting as Type-I when \( G_1 = G_2 \) (i.e., there are efficient mappings in both directions). This is also called the symmetric setting. In this case, we define \( \Lambda := (p, G, G_T, e) \). When we need to be specific, the group description yielded by \( \mathcal{G} \) will be written as \( \Lambda_{\text{asym}} \) or \( \Lambda_{\text{sym}} \).

2.3 Assumptions

Let \( \mathcal{G} \) be a generator of bilinear groups. All hardness assumptions we deal with are defined relative to \( \mathcal{G} \). We first define the computational and decisional Diffie-Hellman assumptions (CDH\(_1\), DDH\(_1\)) and decisional linear assumption (DLIN\(_1\)) for Type-III bilinear groups. The corresponding more standard assumptions, CDH, DDH, and DLIN, in Type-I groups are obtained by setting \( G_1 = G_2 \) and \( G = \hat{G} \) in the respective definitions.
Definition 1 (Computation co-Diffie-Hellman Assumption: CDH). Given \( \Lambda \leftarrow \mathcal{G}^I \), \( G \leftarrow \mathcal{G}^{I}_1 \), \( \hat{G} \leftarrow \mathcal{G}^{I}_2 \), \( G^x \), \( \hat{G}^x \), and \( G^y \) for \( x, y \leftarrow \mathbb{Z}_p \), any p.p.t. algorithm \( A \) outputs \( G^{xy} \) with negligible probability \( \text{Adv}^{\text{GCDH}}_{\mathcal{G},A}(\lambda) \) in \( \lambda \).

Definition 2 (Decisional Diffie-Hellman Assumption in \( \mathbb{G}_1 \): DDH). Given \( \Lambda \leftarrow \mathcal{G}^I \), \( G \leftarrow \mathcal{G}^{I}_1 \), and \( (G^2, G^y, Z_0) \) where \( Z_1 = G^{xy} \) and \( Z_0 = G^z \) for random \( x, y, z \leftarrow \mathbb{Z}_p \) and random bit \( b \), any p.p.t. algorithm \( A \) decides whether \( b = 1 \) or \( 0 \) with negligible advantage \( \text{Adv}^{\text{DDH1}}_{\mathcal{G},A}(\lambda) \) in \( \lambda \).

Definition 3 (Decisional Linear Assumption in \( \mathbb{G}_1 \): DLIN). Given \( \Lambda \leftarrow \mathcal{G}^I \), \( (G_1, G_2, G_3) \leftarrow (\mathcal{G}^{I}_1)^3 \) and \( (G_1^x, G_2^y, Z_0) \) where \( Z_1 = G_3^{x+y} \) and \( Z_0 = G_3^z \) for random \( x, y, z \leftarrow \mathbb{Z}_p \) and random bit \( b \), any p.p.t. algorithm \( A \) decides whether \( b = 1 \) or \( 0 \) with negligible advantage \( \text{Adv}^{\text{DLIN1}}_{\mathcal{G},A}(\lambda) \) in \( \lambda \).

For DDH1 and DLIN1, we define an analogous assumption in \( \mathbb{G}_2 \) (DDH2) by swapping \( \mathbb{G}_1 \) and \( \mathbb{G}_2 \) in the respective definitions. In Type-III bilinear groups, it is assumed that both DDH1 and DDH2 hold simultaneously. The assumption is called the symmetric external Diffie-Hellman assumption (SXDH), and we define advantage \( \text{Adv}^{\text{SXDH}}_{\mathcal{G},A}(\lambda) := \text{Adv}^{\text{DDH1}}_{\mathcal{G},A}(\lambda) + \text{Adv}^{\text{DDH2}}_{\mathcal{G},A}(\lambda) \). We extend DLIN in a similar manner:

Definition 4 (External Decision Linear Assumption in \( \mathbb{G}_1 \): XDLIN). Given \( \Lambda \leftarrow \mathcal{G}^I \), \( (G_1, G_2, G_3) \leftarrow (\mathcal{G}^{I}_1)^3 \) and \( (G_1^x, G_2^y, G_1^z, G_2^w, Z_0) \) where \( G_1, G_2, G_3 \sim (G_1, G_2, G_3), Z_1 = G_3^{x+y}, \) and \( Z_0 = G_3^z \) random bit \( x, y, z \leftarrow \mathbb{Z}_p \) and random bit \( b \), any p.p.t. algorithm \( A \) decides whether \( b = 1 \) or \( 0 \) with negligible advantage \( \text{Adv}^{\text{XDLIN1}}_{\mathcal{G},A}(\lambda) \) in \( \lambda \).

The XDLIN1 assumption is equivalent to the DLIN1 assumption in the generic bilinear group model [50, 4] where one can simulate the extra elements, \( G_1, G_2, G_3, G_1^z, G_2^w, \) in XDLIN1 from \( G_1, G_2, G_3, G_1^z, G_2^w, \) in DLIN1. We define the XDLIN2 assumption analogously by giving \( G_3^{x+y} \) or \( G_3^z \) as \( Z_0 \), to \( A \) instead. Then we define the simultaneous external DLIN assumption, SXDLIN, that assumes that both XDLIN1 and XDLIN2 hold at the same time. By \( \text{Adv}^{\text{XDLIN1}}_{\mathcal{G},A}(\lambda) \) (resp.), we denote the advantage function for XDLIN2 (and SXDLIN, resp.).

Definition 5 (Double Pairing Assumption in \( \mathbb{G}_1 \): DBP). Given \( \Lambda \leftarrow \mathcal{G}^I \) and \( (G_z, G_r) \leftarrow (\mathcal{G}^I)^2 \), any p.p.t. algorithm \( A \) outputs \( (Z, R) \in (G^*_2)^2 \) that satisfies \( 1 = e(G_z, Z) \cdot e(G_r, R) \) with negligible probability \( \text{Adv}^{\text{DBP1}}_{\mathcal{G},A}(\lambda) \) in \( \lambda \).

The double pairing assumption in \( \mathbb{G}_2 \) (DBP2) is defined in the same manner by swapping \( \mathbb{G}_1 \) and \( \mathbb{G}_2 \). It is known that DBP1 (DBP2, resp.) is implied by DDH1 (DDH2, resp.) and the reduction is tight [8]. Note that the double pairing assumption does not hold in Type-I groups since \( Z = G_r, R = G_z^{-1} \) is a trivial solution. Thus in Type-I groups we will instead use the following extension:

Definition 6 (Simultaneous Double Pairing Assumption [19]: SDP). Given \( \Lambda \leftarrow \mathcal{G}^I \) and \( (G_z, G_r, H_z, H_r) \leftarrow (G^*_3)^4 \), any p.p.t. algorithm \( A \) outputs \( (Z, R, S) \in (G^*_3)^3 \) that satisfies \( 1 = e(G_z, Z) \cdot e(G_r, R) \wedge 1 = e(H_z, Z) \cdot e(H_r, S) \) with negligible probability \( \text{Adv}^{\text{SDP}}_{\mathcal{G},A}(\lambda) \) in \( \lambda \).

As shown in [19], for the Type-I setting the simultaneous double pairing assumption holds relative to \( \mathcal{G} \) if the decisional linear assumption holds for \( \mathcal{G} \).

3 Definitions

3.1 Common setup

All building blocks make use of a common setup algorithm Setup that takes the security parameter \( 1^\lambda \) and outputs a global parameter \( gk \) that is given to all other algorithms. Usually \( gk \) consists of a description \( \Lambda \).
of a bilinear group setup and a default generator for each group. In this paper, we include several additional
generators in \( g_k \) for technical reasons. Note that when the resulting signature scheme is used in multi-user
applications different additional generators need to be assigned to individual users or one needs to fall back
on the common reference string model, whereas \( \lambda \) and the default generators can be shared. Thus we count
the size of \( g_k \) when we assess the efficiency of concrete instantiations. For ease of notation, we make \( g_k \)
implicit except w.r.t. key generation algorithms.

### 3.2 Signature schemes

We use the following syntax for signature schemes suitable for the multi-user and multi-algorithm setting. We
follow standard syntax with the following modifications: the key generation function takes as input
global parameter \( g_k \) generated by Setup (instead of security parameter \( 1^\lambda \)), and the message space \( \mathcal{M} \) is
determined solely by \( g_k \) (instead of being determined by the public-key).

**Definition 7 (Signature Scheme).** A signature scheme \( \Sigma \) is a triple of polynomial-time algorithms \((\text{Key}, \text{Sign}, \text{Vrf})\):

- \( \text{SIG.Key}(g_k) \) generates a public-key \( vk \) and a secret-key \( sk \).
- \( \text{SIG.Sign}(sk, msg) \) takes \( sk \) and message \( msg \) and outputs a signature \( \sigma \).
- \( \text{SIG.Vrf}(vk, msg, \sigma) \) outputs 1 for acceptance or 0 for rejection.

Correctness requires that

\[
\forall j \in [1,n], \, \text{SIG.Vrf}(vk, sk, msg_j) = \text{SIG.Sign}(vk, \sigma_j, msg_j)
\]

It is implicit that \( A \) in the first run hands over an internal state to that in the second run.

**Definition 8 (Unforgeability against Adaptive Chosen-Message Attacks).** A signature scheme is unforgeable
against adaptive chosen message attacks (UF-CMA) if for any probabilistic polynomial-time oracle algorithms \( A \) the following advantage function \( \text{Adv}_{\Sigma,A}^{\text{uf-cma}}(\lambda) \) is bound by a negligible function in \( \lambda \).

\[
\text{Adv}_{\Sigma,A}^{\text{uf-cma}}(\lambda) := \Pr \left[ \begin{array}{c}
\text{msg}^1 \not\in Q_m \land \text{SIG.Vrf}(vk, \sigma^1, msg^1)
\end{array} \right] \begin{array}{c}
g_k \leftarrow \text{Setup}(1^\lambda),
(vk, sk) \leftarrow \text{SIG.Key}(g_k),
(\sigma^1, msg^1) \leftarrow A^O(vk)
\end{array}
\]

\( O_s \) is a signing oracle that, on receiving message \( msg_j \), performs \( \sigma_j \leftarrow \text{SIG.Sign}(sk, msg_j) \), returns \( \sigma_j \) to \( A \), and records \( msg_j \) to \( Q_m \), which is an initially empty list.

**Definition 9 (Unforgeability against Non-Adaptive Chosen-Message Attacks).** A signature scheme is
unforgeable against non-adaptive chosen message attacks (UF-NACMA) if for any probabilistic polynomial-time
algorithms \( A \) and any polynomial \( n \) in \( \lambda \), the following advantage function \( \text{Adv}_{\Sigma,A}^{\text{uf-nacma}}(\lambda) \) is bound by a negligible function in \( \lambda \).

\[
\text{Adv}_{\Sigma,A}^{\text{uf-nacma}}(\lambda, n) := \Pr \left[ \begin{array}{c}
\forall j \in [1,n], \, \text{msg}^1 \neq msg_j \land 1 = \text{SIG.Vrf}(vk, \sigma^1, msg^1)
\end{array} \right] \begin{array}{c}
g_k \leftarrow \text{Setup}(1^\lambda),
(\sigma_1, \ldots, \sigma_n) \leftarrow A(g_k),
(vk, sk) \leftarrow \text{SIG.Key}(g_k),
\forall j \in [1,n], \, \sigma_j \leftarrow \text{SIG.Sign}(sk, msg_j),
(\sigma^1, msg^1) \leftarrow A(vk, \sigma_1, \ldots, \sigma_n)
\end{array}
\]

It is implicit that \( A \) in the first run hands over an internal state to that in the second run.

**Definition 10 (Unforgeability against Random Message Attacks (UF-RMA)).** A signature scheme is
unforgeable against random message attacks (UF-RMA) if for any probabilistic polynomial-time algorithms \( A \) and any positive integer \( n \) bound by a polynomial in \( \lambda \), the following advantage function \( \text{Adv}_{\Sigma,A}^{\text{uf-rma}}(\lambda) \) is negligible in \( \lambda \).

\[
\text{Adv}_{\Sigma,A}^{\text{uf-rma}}(\lambda, n) := \Pr \left[ \begin{array}{c}
\forall j \in [1,n], \, \text{msg}^1 \neq msg_j \land 1 = \text{SIG.Vrf}(vk, \sigma^1, msg^1)
\end{array} \right] \begin{array}{c}
g_k \leftarrow \text{Setup}(1^\lambda),
(vk, sk) \leftarrow \text{SIG.Key}(g_k),
(\sigma_1, \ldots, \sigma_n) \leftarrow \mathcal{M}^n,
\forall j \in [1,n], \, \sigma_j \leftarrow \text{SIG.Sign}(sk, msg_j),
(\sigma^1, msg^1) \leftarrow A(vk, \sigma_1, \sigma_2, \ldots, \sigma_n)
\end{array}
\]


We consider a variation of random message attacks where the adversary is given, for example, the random coin used to sample the random message. Our formal definition covers more general idea of auxiliary information about the message generator as follows. Let MSGGen be a message generation algorithm that takes \( gk \) (and random coins as well) as input and outputs \( msg \in M \). Furthermore, MSGGen outputs auxiliary information \( \omega \), which may give some hint about the random coins used for selecting \( msg \). The extended random message attack is defined relative to message generator MSGGen as follows.

The above syntax and security notions can be applied to one-time signature schemes by restricting the oracle access only once or parameter \( n \) to 1.

**Definition 11 (Unforgeability against Extended Random Message Attacks (UF-XRMA)).** A signature scheme is unforgeable against extended random message attacks (UF-XRMA) with respect to message oracle access only once or parameter \( n \) bound by a polynomial in \( \lambda \), the following advantage function \( \text{Adv}_{\text{SIG}, A}^{uf\text{-xrma}}(\lambda) \) is bound by a negligible function in \( \lambda \).

\[
\text{Adv}_{\text{SIG}, A}^{uf\text{-xrma}}(\lambda) := \Pr \left[ \forall j \in [1, n], \text{msg} \downarrow \neq \text{msg} \downarrow \wedge
\right.
\]

\[
1 = \text{SIG.Vrf}(vk, \sigma_1, \text{msg} \downarrow)
\]

\[
\left| \begin{array}{c}
gk \leftarrow \text{Setup}(1^\lambda),
(vk, sk) \leftarrow \text{SIG.Key}(gk),
\forall j \in [1, n],
\left( \text{msg} \downarrow, \omega_j \right) \leftarrow \text{MSGGen}(gk),
\sigma_j \leftarrow \text{SIG.Sign}(sk, \text{msg} \downarrow),
\left( \sigma_1, \text{msg} \downarrow \right) \leftarrow A(vk, \sigma_1, \text{msg} \downarrow, \omega_1, \ldots, \sigma_n, \text{msg} \downarrow, \omega_n)
\end{array} \right.
\]

For the above security notions, UF-CMA ⇒ UF-XRMA ⇒ UF-RMA holds. More precisely, for any signature scheme SIG, for any \( A' \) there exists \( A \) such that \( \text{Adv}_{\text{SIG}, A}^{uf\text{-xrma}}(\lambda) \geq \text{Adv}_{\text{SIG}, A'}^{uf\text{-xrma}}(\lambda) \), and for any \( A'' \) there exists \( A' \) such that \( \text{Adv}_{\text{SIG}, A''}^{uf\text{-xrma}}(\lambda) \geq \text{Adv}_{\text{SIG}, A'}^{uf\text{-xrma}}(\lambda) \).

### 3.3 Partial one-time and tagged one-time signatures

Partial one-time signatures, also known as two-tier signatures [13], are a variation of one-time signatures where only part of the public-key and secret-key must be updated for every signing, while the remaining part can be persistent.

**Definition 12 (Partial One-Time Signature Scheme [13]).** A partial one-time signatures scheme POS is a set of polynomial-time algorithms \( \{\text{Key, Update, Sign, Vrf}\} \).

- \( \text{POS.Key}(gk) \) generates a long-term public-key \( pk \) and secret-key \( sk \), and sets the associated message space to be \( M_a \) as defined by \( gk \). (Recall that we require that \( M_a \) be completely defined by \( gk \).)
- \( \text{POS.Update}(gk) \) takes \( gk \) as input, and outputs a one-time key pair \( \{opk, osk\} \). We denote the space for \( opk \) by \( K_{opk} \).
- \( \text{POS.Sign}(sk, msg, osk) \) outputs a signature \( \sigma \) on message \( msg \), based on \( sk \) and \( osk \).
- \( \text{POS.Vrf}(pk, opk, msg, \sigma) \) outputs 1 for acceptance, or 0 for rejection.

Correctness requires that \( 1 = \text{POS.Vrf}(pk, opk, msg, \sigma) \) holds except for negligible probability for any \( gk, pk, opk, \sigma \), and \( msg \in M_a \), such that \( gk \leftarrow \text{Setup}(1^\lambda) \), \( (pk, sk) \leftarrow \text{POS.Key}(gk) \), \( (opk, osk) \leftarrow \text{POS.Update}(gk) \), \( \sigma \leftarrow \text{POS.Sign}(sk, msg, osk) \).

A tagged one-time signature scheme is a signature scheme whose signing function in addition to the long-term secret key takes a tag as input. A tag is one-time, i.e., it must be different for every signing.

**Definition 13 (Tagged One-Time Signature Scheme).** A tagged one-time signature scheme TOS is a set of polynomial-time algorithms \( \{\text{Key, Tag, Sign, Vrf}\} \).

- \( \text{TOS.Key}(gk) \) generates a long-term public-key \( pk \) and secret-key \( sk \), and sets the associated message space to be \( M_t \) as defined by \( gk \).
• TOS.Tag\((gk)\) takes \(gk\) as input and outputs \(tag\). By \(T\), we denote the space for \(tag\).

• TOS.Sign\((sk, msg, tag)\) outputs signature \(\sigma\) for message \(msg\) based on \(sk\) and \(tag\).

• TOS.Vrf\((pk, tag, msg, \sigma)\) outputs 1 for acceptance, or 0 for rejection.

Correctness requires that \(1 = TOS.Vrf(pk, tag, msg, \sigma)\) holds except for negligible probability for any \(gk, pk, tag, \sigma\), and \(msg \in \Sigma\), such that \(gk \leftarrow \text{Setup}(1^\lambda)\), \((pk, sk) \leftarrow TOS.Key(gk)\), \(tag \leftarrow TOS.Tag(gk)\), \(\sigma \leftarrow TOS.Sign(sk, msg, tag)\).

A TOS scheme is a POS scheme for which \(tag = osk = opk\). We can thus give a security notion for POS schemes that also applies to TOS schemes by reading Update = Tag and \(tag = osk = opk\).

**Definition 14 (Unforgeability against One-Time Adaptive Chosen-Message Attacks).** A partial one-time signature scheme is unforgeable against one-time adaptive chosen message attacks (OT-CMA) if for any probabilistic polynomial-time oracle algorithms \(A\) the following advantage function \(Adv_{\text{POS}, A}^{\text{ota}}\) is negligible in \(\lambda\).

\[
Adv_{\text{POS}, A}^{\text{ota}}(\lambda) := \Pr \left[ \exists (opk, msg, \sigma) \in Q_m \text{ s.t.} \right.
\begin{align*}
opk^\dagger &= opk \land msg^\dagger \neq msg \land \\
1 &= TOS.Vrf(pk, opk^\dagger, \sigma^\dagger, msg^\dagger) \right. \left. gk \leftarrow \text{Setup}(1^\lambda),
\begin{align*}
(pk, sk) &\leftarrow TOS.Key(gk), \\
(opk^\dagger, \sigma^\dagger, msg^\dagger) &\leftarrow A^{(\Omega, \Omega)}(pk) \right. \right]
\end{align*}
\]

\(Q_m\) is initially an empty list. \(\Omega\) is the one-time key generation oracle that on receiving a request invokes a fresh session \(j\), performs \((opk_j, osk_j) \leftarrow \text{POS.Update}(gk)\), and returns \(opk_j\). \(\Omega\) is the signing oracle that, on receiving a message \(msg_j\) for session \(j\), performs \(\sigma_j \leftarrow \text{POS.Sign}(sk, msg_j, osk_j)\), returns \(\sigma_j\) to \(A\), and records \((opk_j, msg_j, \sigma_j)\) to the list \(Q_m\). \(\Omega\) works only once for each session. Strong unforgeability is defined by replacing condition \(msg^\dagger \neq msg\) with \((msg^\dagger, \sigma^\dagger) \neq (msg, \sigma)\).

We define a non-adaptive variant (OT-NACMA) of the above notion by integrating \(\Omega\) into \(\Omega\), so that \(opk_j\) and \(\sigma_j\) are returned to \(A\) at the same time. Namely, \(A\) must submit \(msg_j\) before seeing \(opk_j\). If a scheme is secure in the sense of OT-CMA, the scheme is also secure in the sense of OT-NACMA. By \(Adv_{\text{POS}, A}^{\text{otnacma}}(\lambda)\) we denote the advantage of \(A\) in this non-adaptive case. For TOS, we use the same notation, OT-CMA and OT-NACMA, and define advantage functions \(Adv_{\text{TOS}, A}^{\text{otnacma}}\) and \(Adv_{\text{TOS}, A}^{\text{otcma}}\) accordingly. We will also consider strong unforgeability, for which we use labels sot-nacma and sot-cma. Recall that if a scheme is strongly unforgeable, it is unforgeable as well.

We define a condition that is relevant for coupling random message secure signature schemes with partial one-time and tagged one-time signature schemes in later sections.

**Definition 15 (Tag/One-time Public-Key Uniformity).** A TOS is called uniform-tag if TOS.Tag outputs \(tag\) that uniformly distributes over \(tag\) space \(T\). Similarly, a POS is called uniform-key if POS.Update outputs \(opk\) that distributes uniformly over key space \(K_{opk}\).

### 3.4 Structure-preserving signatures

A signature scheme is structure-preserving over a bilinear group \(\Lambda\), if public-keys, signatures, and messages are all source group elements of \(\Lambda\), and the verification only evaluates pairing product equations. Similarly, POS and TOS schemes are structure-preserving if their public-keys, signatures, messages, and tags or one-time public-keys consist of source group elements and the verification only evaluates pairing product equations.

## 4 Generic Constructions

### 4.1 SIG1: Combining tagged one-time and RMA-secure signatures

Let \(rSIG\) be a signature scheme with message space \(M_m\), and TOS be a tagged one-time signature scheme with tag space \(T\) such that \(M_m = T\) and both schemes use the same Setup. We construct a signature scheme \(SIG1\) from \(rSIG\) and TOS. Let \(gk\) be the global parameter generated by \(\text{Setup}(1^\lambda)\). It is assumed that a secret-key of \(rSIG\) includes \(gk\).

[Generic Construction 1: SIG1]
The real security game except that the winning condition is changed to no longer accept repetition
of tags. Departing from the UF-CMA game for SIG1, the security proof is done by evaluating two game transitions. The first transition is based on the OT-NACMA security of TOS. This part is rather simple as we can construct a simulator in a straightforward manner by following the key generation and signing of rSIG. The second transition is based on UF-RMA of rSIG. We construct a simulator that, given signatures of rSIG on uniformly chosen tags as messages, simulates signatures of SIG1 for messages provided by the adversary. For this to be done, the simulator needs to compute one-time signatures of TOS for the given uniform tags. This, however, can be done without any problem since the simulator has legitimate signing keys that are sufficient to run the signing function of TOS with uniform tags.

Proof. Any signature that is accepted as a successful forgery must either reuse an existing tag, or sign a new tag. We show that former case reduces to attacking TOS and the latter case reduces to attacking rSIG. Thus the success probability \( \text{Adv}_{\text{SIG1,A}}^{\text{uf,cma}}(\lambda) \) of an attacker on SIG1 will be bounded by the sum of the success probabilities \( \text{Adv}_{\text{TOS,B}}^{\text{ot-nacma}}(\lambda) \) of an attacker on TOS and the success probability \( \text{Adv}_{\text{SIG,C}}^{\text{uf,nacma}}(\lambda) \) of an attacker on rSIG.

Game 0: The actual Unforgeability game. \( \Pr[\text{Game 0}] = \text{Adv}_{\text{SIG1,A}}^{\text{uf,cma}}(\lambda) \).

Game 1: The real security game except that the winning condition is changed to no longer accept repetition of tags.

Lemma 1. \( |\Pr[\text{Game 0}] - \Pr[\text{Game 1}]| \leq \text{Adv}_{\text{TOS,B}}^{\text{ot-nacma}}(\lambda) \)

Proof. Attacker \( A \) wins in Game 0, but loses in Game 1, iff it produces a forgery that reuses a tag from a signing query. We describe a reduction \( B \) that uses such an attacker to break the OT-NACMA-security of TOS. The reduction \( B \) receives \( gk \) and \( pk_t \) from the challenger of TOS, sets up \( vk_r \) and \( sk_r \), honestly by running rSIG.Key\((gk)\), and provides \( gk \) and \( vk = (vk_r, pk_t) \) to \( A \).

To answer a signing query, \( B \) uses the signing oracle of TOS to get \( tag \) and \( \sigma_t \), signs \( \text{tag} \) using \( sk_r \) to produce \( rSig \), and returns \( (tag, \sigma_t, \sigma_r) \). When \( A \) produces a forgery \( (tag^1, \sigma^1_t, \sigma^1_r) \) on message \( msg^1 \), \( B \) outputs \( (msg^1, tag^1, \sigma^1) \) as a forgery for TOS.

Game 2: The fully idealized game. The winning condition is changed to reject all signatures.

Lemma 2. \( |\Pr[\text{Game 1}] - \Pr[\text{Game 2}]| \leq \text{Adv}_{\text{SIG,C}}^{\text{uf,nacma}}(\lambda) \)
Theorem 2. If resulting SIG1 is ρ-preserving, so is SIG2.

Thus Adv_{SIG1,A}^\text{cma}\lambda = \Pr[\text{Game 0}] \leq \text{Adv}_{\text{TOS,B}}^{\text{ot,nacma}}\lambda + \text{Adv}_{\text{SIG,C}}^{\text{af,rma}}\lambda as claimed.

The following theorem is immediately obtained from the construction.

Theorem 2. If TOS.Tag produces constant-size tags and signatures in the size of input messages, the resulting SIG1 produces constant-size signatures as well. Furthermore, if TOS and rSIG are structure-preserving, so is SIG1.

4.2 SIG2: Combining partial one-time and XRMA-secure signatures

Let xSIG be a signature scheme with message space \( M_x \), and POS be a partial one-time signature scheme with one-time public-key space \( K_{\text{opk}} \) such that \( M_x = K_{\text{opk}} \) and both schemes use the same Setup. We construct a signature scheme SIG2 from xSIG and POS. Let \( gk \) be a global parameter generated by Setup(1^\lambda). It is assumed that a secret key for xSIG contains \( gk \).

Theorem 3. SIG2 is UF-CMA if POS is uniform-key and OT-NACMA, and xSIG is UF-XRMA relative to POS.Update as a message generator. In particular, for any p.p.t. algorithm \( A \), there exist p.p.t. algorithms B and C such that Adv_{SIG2,A}^{\text{cma}}\lambda \leq \text{Adv}_{\text{TOS,B}}^{\text{ot,nacma}}\lambda + \text{Adv}_{\text{SIG,C}}^{\text{af,rma}}\lambda.

Theorem 4. If POS produces constant-size one-time public-keys and signatures in the size of input messages, the resulting SIG2 produces constant-size signatures as well. Furthermore, if POS and xSIG are structure-preserving, so is SIG2.

5 Instantiating SIG1

We instantiate the building blocks TOS and rSIG of our first generic construction to obtain our first SPS scheme. We do so in the Type-I bilinear group setting. The resulting SIG1 scheme is an efficient structure-preserving signature scheme based only on the DLIN assumption.
5.1 Setup for Type-I groups

The following setup procedure is common for all instantiations in this section. The global parameter \( gk \) is given to all functions implicitly.

- Setup(1\(^3 \)): Run \( \Lambda = (\lambda, G, G_T, e) \leftarrow G(1\(^3 \)) \) and pick random generators \((G, C, F, U) \leftarrow (G^*)^4 \).

Output \( gk := (\lambda, G, C, F, U) \).

The parameter \( gk \) fixes the message space \( \mathcal{M}_k := \{(C^m, F^m, U^m) \in G^3 \mid m \in \mathbb{Z}_p \} \) for the RMA-secure signature scheme presented in Section 5.3. For our generic framework to work, the tagged one-time signature schemes should have the same tag space.

5.2 Tagged one-time signature scheme

Our scheme generates tags consisting of only one group element, \( C^i \), which is optimally efficient in its size. However, as mentioned above, we need to adjust the tag space to match the message space of \( rSIG \). We thus describe the scheme with a tag in the extended form of \((C^t, F^t, U^t)\). The extended elements \( F^t \) and \( U^t \) can be dropped when unnecessary as it is done in its direct application shown in Section 5.3.

Our concrete construction of TOS can be seen as an adaptation of a one-time signature scheme in [8] so that it enjoys optimally short one-time public-key (i.e., a tag) with no corresponding one-time secret-key. We note that, given TOS, one can construct a one-time signature scheme. But the reverse is not known in general.

[Scheme TOS]

TOS.Key(\( gk \)): Parse \( gk := (\lambda, G, C, F, U) \). Choose \( w_z, w_r, \mu_z, \mu_s, \tau \) uniformly from \( \mathbb{Z}_p^* \) and compute \( G_z := G^{w_z}, G_r := G^{w_r}, H_z := G^{\mu_z}, H_s := G^{\mu_s}, G_t := G^\tau \) and for \( i = 1, \ldots, k \), uniformly choose \( \chi_i, \gamma_i, \delta_i \) from \( \mathbb{Z}_p^* \) and compute

\[
G_i := G_z^{\chi_i} G^{\gamma_i}, \quad \text{and} \quad H_i := H_z^{\chi_i} H_s^{\delta_i}.
\]

Output \( pk := (G_z, G_r, H_z, H_s, G_t, G_1, \ldots, G_k, H_1, \ldots, H_k) \in G^{2k+5} \) and \( sk := (w_z, \mu_z, \tau, \chi_1, \gamma_1, \delta_1, \ldots, \chi_k, \gamma_k, \delta_k) \in \mathbb{Z}_p^{2k+5} \).

TOS.Tag(\( gk \)): Choose \( t \leftarrow \mathbb{Z}_p^* \), compute \( T := C^t \). Output \( tag := (T, T', T'') := (C^t, F^t, U^t) \in G^3 \).

TOS.Sign(\( sk, msg, tag \)): Parse \( msg \) as \((M_1, \ldots, M_k)\) and \( tag \) as \((T, T', T'')\). Parse \( sk \) accordingly. Choose \( \zeta \leftarrow \mathbb{Z}_p^* \) and output \( \sigma := (Z, R, S) \in G^3 \) where

\[
Z := G_z^{\sum_{i=1}^k M_i^{-\chi_i}}, \quad R := (T^r G_z^{-\zeta} \prod_{i=1}^k M_i^{-\gamma_i}), \quad \text{and} \quad S := (H_z^{-\zeta} \prod_{i=1}^k M_i^{-\delta_i}).
\]

TOS.Vrf(\( pk, tag, msg, \sigma \)): Parse \( \sigma \) as \((Z, R, S) \in G^3\), \( msg \) as \((M_1, \ldots, M_k) \in G^k \), and \( tag \) as \((T, T', T'')\). Return 1 if the following equations hold. Return 0, otherwise.

\[
e(T, G_i) = e(G_z, Z) e(G_r, R) \prod_{i=1}^k e(G_i, M_i) \tag{2}
\]

\[
1 = e(H_z, Z) e(H_s, S) \prod_{i=1}^k e(H_i, M_i) \tag{3}
\]

Correctness is verified by inspecting the following relations.

For (2):

\[
e(G_z, G_z^{\sum_{i=1}^k M_i^{-\chi_i}}) e(G_r, (T^r G_z^{-\zeta} \prod_{i=1}^k M_i^{-\gamma_i}) \prod_{i=1}^k e(G_i^{\sum_{i=1}^k M_i^{-\chi_i}})) = e(G_z, G_z^{\sum_{i=1}^k M_i^{-\chi_i}}) e(G_r, T^r) e(G, G_z^{-\zeta}) = e(G, T^r) = e(T, G_t)
\]

For (3):

\[
e(H_z, G_z^{\sum_{i=1}^k M_i^{-\chi_i}}) e(H_s, (H_z^{-\zeta} \prod_{i=1}^k M_i^{-\delta_i}) \prod_{i=1}^k e(H_z, H_z^{\sum_{i=1}^k M_i^{-\chi_i}})) = e(H_z, G_z^{\sum_{i=1}^k M_i^{-\chi_i}}) e(G, H_z^{-\zeta}) = 1
\]
We state the following theorems, of which the first one is immediate from the construction.

**Theorem 5.** The above TOS is structure-preserving, and yields uniform tags and constant-size signatures.

**Theorem 6.** The above TOS is strongly unforgeable against one-time tag adaptive chosen message attacks (SOT-CMA) if the SDP assumption holds. In particular, for all p.p.t. algorithms $A$, there exists p.p.t. algorithm $B$ such that $\text{Adv}_{\text{TOS},A}(\lambda) \leq \text{Adv}_{\text{SDP}}(\lambda) + 1/\lambda$, where $\lambda$ is the size of the groups produced by $G$. Moreover, the run-time overhead of the reduction $B$ is a small number of multi-exponentiations per signing or tag query.

**Proof.** Given successful forger $A$ against TOS as a black-box, we construct $B$ that breaks SDP as follows. Let $I_{\text{sd}} = (A, G_z, G_r, H_z, H_s)$ be an instance of SDP. Algorithm $B$ simulates the attack game against TOS as follows. It first builds $g_k := (A, G, C, F, U)$ by choosing $G$ randomly from $G^*$, choosing $c, f, u \leftarrow Z_p$, and computing $C = G^c$, $F = G^f$, and $U = G^u$. This yields a $g_k$ in the same distribution as produced by Setup. Next $B$ simulates TOS.Key by taking $(G_z, G_r, H_z, H_s)$ from $I_{\text{sd}}$ and computing $G_t := H_z^*$ for random $\tau$ in $Z_p^*$. It then generates $G_i$ and $H_i$ according to $\mathbb{T}$. This perfectly simulates TOS.Key.

On receiving the $j$-th query to $O_1$, algorithm $B$ computes

$$T := (G_z^j G_r^j) \frac{1}{\lambda}$$

for $\zeta, \rho \leftarrow Z_p^*$. If $T = 1$, $B$ sets $Z^* := H_s$, $S^* := H_z^{-1}$, and $R^* := (\zeta^\rho)^\zeta$, outputs $(Z^*, R^*, S^*)$ and stops. Otherwise, $B$ stores $(\zeta, \rho)$ and returns $\text{tag}_j := (T, T^{1/\zeta}, T^{1/\zeta})$ to $A$.

On receiving signing query $\text{msg}_j = (M_1, \ldots, M_k)$, algorithm $B$ takes $\zeta$ and $\rho$ used for computing $\text{tag}_j$ (if $\text{tag}_j$ is not yet defined, execute the above procedure for generating $\text{tag}_j$ and take new $\zeta$ and $\rho$) and computes

$$Z := H_z^\zeta \prod_{i=1}^{k} M_i^{-\zeta}, \quad R := H_z^\rho \prod_{i=1}^{k} M_i^{-\rho}, \quad S := H_z^{-\zeta} \prod_{i=1}^{k} M_i^{-\rho}.$$  

Then $B$ returns $\text{tag}_j := (Z, R, S)$ to $A$ and records $(\text{tag}_j, \sigma, \text{msg}_j)$.

When $A$ outputs a forgery $(\text{tag}^1, \sigma^1, \text{msg}^1)$, algorithm $B$ searches the records for $(\text{tag}, \sigma, \text{msg})$ such that $\text{tag}^1 = \text{tag}$ and $(\text{msg}^1, \sigma^1) \neq (\text{msg}, \sigma)$. If no such entry exists, $B$ aborts. Otherwise, $B$ computes

$$Z^* := Z^\zeta \prod_{i=1}^{k} \left( \frac{M_i^{\zeta}}{M_i} \right)^{\gamma_i}, \quad R^* := R^\rho \prod_{i=1}^{k} \left( \frac{M_i^{\rho}}{M_i} \right)^{\gamma_i}, \quad S^* := S^\zeta \prod_{i=1}^{k} \left( \frac{M_i^{\zeta}}{M_i} \right)^{\delta_i},$$

where $(Z, R, S)$, $(M_1, \ldots, M_k)$ and their dagger counterparts are taken from $(\sigma, \text{msg})$ and $(\sigma^1, \text{msg}^1)$, respectively. $B$ finally outputs $(Z^*, R^*, S^*)$ and stops. This completes the description of $B$.

We claim that $B$ solves the problem by itself or the view of $A$ is perfectly simulated. The correctness of key generation has been already inspected. In the simulation of $O_1$, there is a case of $T = 1$ that happens with probability $1/\lambda$. If it happens, $B$ outputs a correct answer to $I_{\text{sd}}$, which is clearly by observing $G_z = G_r^{\rho/\zeta}$, $Z^* \neq H_s$, $e(G_z, Z^*) = e(G_r, R^*) = e(G_r^{\rho/\zeta}, Z^*) = 1$ and $e(H_z, Z^*) = e(H_s, S^*) = 1$. Otherwise, tag $T$ is uniformly distributed over $G^*$ and the simulation is perfect.

Oracle $O_3$ is simulated perfectly as well. Correctness of simulated $\text{tag}_j := (Z, R, S)$ can be verified by inspecting the following relations.

(Right-hand of 2) \[ e(G_z, H_z^\zeta \prod_{i=1}^{k} M_i^{-\zeta}) \cdot e(G_r, H_z^\rho \prod_{i=1}^{k} M_i^{-\rho}) \prod_{i=1}^{k} e(G_z^\gamma M_i, \lambda_i) = e(G_z^\zeta^\gamma, H_s) = e(T_1, G_i) \]

(Right-hand of 3) \[ e(H_z, H_z^\zeta \prod_{i=1}^{k} M_i^{-\zeta}) \cdot e(H_s, H_z^{-\zeta} \prod_{i=1}^{k} M_i^{-\delta}) \prod_{i=1}^{k} e(H_z^\zeta H_s, M_i) = e(H_z, H_s^\zeta) = 1 \]
Every $Z$ is uniformly distributed over $G$ due to the uniform choice of $\zeta$. Then $R$ and $S$ are uniquely determined by following the distribution of $Z$.

Accordingly, $\mathcal{A}$ outputs a successful forgery with non-negligible probability and $\mathcal{B}$ finds a corresponding record $(\text{tag}, \sigma, \text{msg})$. We show that output $(Z^*, R^*, S^*)$ from $\mathcal{B}$ is a valid solution to $I_{\text{adv}}$. First, equation (2) is satisfied because

$$1 = e \left( G_z, \frac{Z^+}{Z} \right) e \left( G_r, \frac{R^+}{R} \right) \prod_{i=1}^k e \left( G_z^{\gamma_i} G_r^{\delta_i}, \frac{M_i^1}{M_i} \right)$$

$$= e \left( G_z, \frac{Z^+}{Z} \prod_{i=1}^k \left( \frac{M_i^1}{M_i} \right)^{\chi_i} \right) e \left( G_r, \frac{R^+}{R} \prod_{i=1}^k \left( \frac{M_i^1}{M_i} \right)^{\gamma_i} \right)$$

$$= e \left( G_z, Z^* \right) e \left( G_r, R^* \right),$$

holds. Equation (3) can be verified similarly.

It remains to prove that $Z^* \neq 1$. Note that, if $\text{msg} = \text{msg}^\dagger$ but this is still a valid forgery then it must be the case that $(Z, R) \neq (Z^\dagger, R^\dagger)$. Since $R$ (resp. $R^\dagger$) is uniquely determined by $Z$ and $\text{msg}$ (resp. $Z^\dagger$, $\text{msg}^\dagger$), that would mean that $Z^* \neq 1$. Alternatively, if $\text{msg}^\dagger \neq \text{msg}$, then there exists $\ell \in \{1, \ldots, k\}$ such that $M_i^\dagger / M_i \neq 1$. We claim that parameters $\chi_1, \ldots, \chi_k$ are independent of the view of $\mathcal{A}$. We prove it by showing that, for every possible assignment to $\chi_1, \ldots, \chi_k$, there exists an assignment to other coins, i.e., $(\gamma_1, \ldots, \gamma_k, \delta_1, \ldots, \delta_k)$ and $(\zeta(1), \rho(1), \ldots, \zeta(q_0), \rho(q_0))$ for $q_0$ queries, that is consistent with the view of $\mathcal{A}$. (By $\zeta(j)$, we denote $\zeta$ with respect to the $j$-th query. We follow this convention hereafter. Without loss of generality, we assume that $\mathcal{A}$ makes $q_0$ tag queries and the same number of signing queries.) Observe that the view of $\mathcal{A}$ consists of independent group elements $(G, G_z, G_r, H_z, H_s, \gamma_1, \gamma_2, \ldots, \gamma_k)$ and $(T^{(j)}_1, Z^{(j)}, M_i^{(j)}, M_k^{(j)})$ for $j = 1, \ldots, q_0$. (Note that we omit $R^{(j)}$ and $S^{(j)}$ from the view since they are uniquely determined by the other components.) We represent the view by the discrete-logarithms of these group elements with respect to base $G$. Namely, the view is represented by $(1, w_z, w_r, \mu_z, \mu_\delta, w_1, \mu_1, \ldots, w_k, \mu_k)$ and $(T^{(j)}_1, \zeta^{(j)}, m_i^{(j)}, \ldots, m_k^{(j)})$ for $j = 1, \ldots, q_0$. The view and the random coins follow relations from (1), (4), and (5), which translate to

\[ w_i = w_z \chi_i + w_r \gamma_i, \quad \mu_i = \mu_z \chi_i + \mu_\delta \delta_i \quad \text{for } i = 1, \ldots, k, \]

\[ \tau t^{(j)} = w_z \zeta^{(j)} + w_r \rho^{(j)}, \quad \text{and} \]

\[ z^{(j)} = \mu_\chi \zeta^{(j)} - \sum_{i=1}^k m_i^{(j)} \chi_i \quad \text{for } j = 1, \ldots, q_0. \]

For any $\ell \in \{1, \ldots, k\}$, fix $\chi_1, \ldots, \chi_{\ell-1}, \chi_{\ell+1}, \ldots, \chi_k$, and consider $\chi_\ell$. For every value of $\chi_\ell$ in $\mathbb{Z}_p$, the linear equations in (6) determine $\gamma_\ell$ and $\delta_\ell$. Then, if $m_\ell^{(j)} \neq 0$, equation (8) determines $\zeta^{(j)}$, and $\rho^{(j)}$ follows from equation (7). If $m_\ell^{(j)} = 0$, then $\zeta^{(j)}$, $\rho^{(j)}$ can be assigned independently from $\chi_\ell$. The above holds for every $\ell \in \{1, \ldots, k\}$. Thus, if $(\chi_1, \ldots, \chi_k)$ is distributed uniformly over $\mathbb{Z}_p^k$, then other coins are distributed uniformly as well and the view of $\mathcal{A}$ is still consistent.

Now we see that given $\mathcal{A}$’s view, $\left( M_i^1 / M_i \right)^{\chi_i}$ is distributed uniformly over $G$ and independent of the other $\{\chi_i\}_{i \neq \ell}$. Therefore $Z^* = 1$ happens only with probability $1/p$. Thus, $\mathcal{B}$ outputs a valid $(Z^*, R^*, S^*)$ with probability $\text{Adv}_{G, B}^{\text{adv}} = 1/p + (1 - 1/p)(1 - 1/p)\text{Adv}_{G, A}^{\text{adv}}$, which leads to $\text{Adv}_{G, B}^{\text{adv}} \leq 1/p$ as claimed.

\[ \square \]

**Remark 1.** The above TOS does not trivially work in the Type-III setting since computing $R$ from $T$ in signing, simulating $T$ using $G_r$ in the reduction, and computing pairing $e(G_r, R)$ in the verification cannot be consistent. In a very recent paper [7], however, it is claimed that it can work if some extra group elements are given in public-keys and the underlying assumption. But details are not published yet.

**Remark 2.** The TOS can be used to sign messages of unbounded length by chaining the signatures. Every message block except for the last one is followed by a tag used to sign the next block. The signature
consists of all internal signatures and tags. The initial tag is considered as the tag for the entire signature. For a message consisting of \( m \) group elements, it repeats \( \tau := 1 + \max(0, \lfloor \frac{m-k}{p} \rfloor) \) times and the resulting signature consists of \( 4\tau - 1 \) elements.

### 5.3 RMA-secure signature scheme

To sign random group elements we will use a construction based on the dual system signature scheme of Waters [51]. For readers unfamiliar with Waters' scheme we recall it in Appendix A. Our intuition for making the original scheme structure-preserving is as follows. While the original scheme is CMA-secure under the DLIN assumption, the security proof makes use of a trapdoor commitment to elements in \( Z_p \) and consequently messages are elements in \( Z_p \) rather than \( G \). Our construction below resorts to RMA-security and removes this commitment to allow messages to be a sequence of random group elements satisfying a particular relation. Concretely, the message space \( M_k := \{(C^m, F^m, U^m) \in G^k \mid m \in Z_p \} \) is defined by generators \((C, F, U)\) in \( g_k \). Moreover, the tag elements of Waters’ scheme are removed in our RMA-secure scheme as they were primarily required for (adaptive) CMA-security.

Other minor modifications are needed for the structure-preserving property. We modify the verification algorithm. Our verification algorithm is deterministic and uses five verification equations. Two equations are for signature elements that are not related to the message part—this is a consequence of deterministic verification. Three equations are for the (extended) message part. We also slightly modify the verification key. One element in \( G_T \) is divided into two elements of \( G \) via randomization due to the requirement of SPS.

**[Scheme rSIG]**

**rSIG.Key(\( gk \)):** Given \( gk := (A, G, C, F, U) \) as input, uniformly select \( V, V_1, V_2, H \) from \( G^* \) and \( a_1, a_2, b, \alpha, \) and \( p \) from \( Z_p^* \). Then compute and output \( vk := (B, A_1, A_2, B_1, B_2, R_1, R_2, W_1, W_2, H, X_1, X_2) \) and \( sk := (vk, K_1, K_2, V, V_1, V_2) \) where

\[
\begin{align*}
B & := G^b, & A_1 & := G^{a_1}, & A_2 & := G^{a_2}, & B_1 & := G^{b-a_1}, & B_2 & := G^{b-a_2} \\
R_1 & := VV_1^{a_1}, & R_2 & := VV_2^{a_2}, & W_1 & := R_1^b, & W_2 & := R_2^b, \\
X_1 & := G^\rho, & X_2 & := G^{a_1-b/\rho}, & K_1 & := G^\alpha, & K_2 & := G^{-a_1}.
\end{align*}
\]

**rSIG.Sign(\( sk, msg \)):** Parse \( msg \) into \((M_1, M_2, M_3)\). Pick random \( r_1, r_2, z_1, z_2 \in Z_p \). Let \( r = r_1 + r_2 \). Compute and output signature \( \sigma := (S_0, S_1, \ldots, S_7) \) where

\[
\begin{align*}
S_0 & := (M_3 H)^{r_1}, & S_1 & := K_2 V^r, & S_2 & := K_1^{-1} V_1^r G^{z_1}, & S_3 & := B^{-z_1}, \\
S_4 & := V_2^r G^{z_2}, & S_5 & := B^{-z_2}, & S_6 & := B^r, & S_7 & := G^r.
\end{align*}
\]

**rSIG.Vrf(\( vk, \sigma, msg \)):** Parse \( msg \) into \((M_1, M_2, M_3)\) and \( \sigma \) into \((S_0, S_1, \ldots, S_7)\). Also parse \( vk \) accordingly. Verify the following pairing product equations:

\[
\begin{align*}
e(S_1, B) e(S_2, B_1) e(S_3, A_1) & = e(S_6, R_1) e(S_7, W_1), & (9) \\
e(S_1, B) e(S_4, B_2) e(S_5, A_2) & = e(S_6, R_2) e(S_7, W_2) e(X_1, X_2), & (10) \\
e(S_7, M_3 H) & = e(G, S_0), & (11) \\
e(F, M_1) & = e(C, M_2), & (12) \\
e(U, M_1) & = e(C, M_3). & (13)
\end{align*}
\]
The scheme is structure-preserving by construction and the correctness is easily verified as follows.

\[ (\text{Left-hand of } 9) = e(G^{\alpha_1}, V^r, G^b) e(G^{-\alpha_1} V_1^r G^{z_1}, G^{ba_1}) e(G^{-b_2}, G^{a_1}) \]
\[ = e(G, V)^{br} e(G, V_1^{ba_1 r}) \]
\[ = e(G, V)^{b(r_1 + r_2)} e(G, V_1^{ba_1 (r_1 + r_2)}) \]
\[ = e(G^{br_2}, V V_1^{a_1}) e(G^{r_1}, V_1^{b_2}) \]
\[ = (\text{Right-hand of } 9) \]

\[ (\text{Left-hand of } 10) = e(G^{\alpha_1}, V^r, G^b) e(V_2^r G^{z_2}, G^{ba_2}) e(G^{-b_2}, G^{a_2}) \]
\[ = e(G, V)^{br} e(G, V_2^{ba_2 r}) \]
\[ = e(G, V)^{b(r_1 + r_2)} e(G, V_2^{ba_2 (r_1 + r_2)}) e(G, G^{ba_1}) \]
\[ = e(G^{br_2}, V_2^{a_2}) e(G^{r_1}, V_2^{b_2}) e(G^p, G^{ba_1 \rho}) \]
\[ = (\text{Right-hand of } 10) \]

Equation (9) and (10) hold since \( r = r_1 + r_2 \). The followings also hold.

\[ (\text{Left-hand of } 11) = e(G^{\alpha_1}, U^m H) = e(G, U^m H)^{s_1} = e(G, (U^m H)^{s_1}) = (\text{Right-hand of } 11), \]
\[ (\text{Left-hand of } 12) = e(F, C^m) = e(F, C)^m = e(C, F^m) = (\text{Right-hand of } 12), \]
\[ (\text{Left-hand of } 13) = e(U, C^m) = e(U, C)^m = e(C, U^m) = (\text{Right-hand of } 13). \]

**Theorem 7.** The above rSIG scheme is UF-RMA under the DLIN assumption. In particular, for any p.p.t. algorithm \( A \) against rSIG that makes at most \( q_s(\lambda) \) signing queries, there exists p.p.t. algorithm \( B \) for DLIN such that \( \text{Adv}_{\text{SIG}_{\lambda}}^{\text{uf-rma}}(\lambda) \leq (q_s(\lambda) + 2) \cdot \text{Adv}_{G, B}^{\text{dlin}}(\lambda) \).

**Proof.** We refer to the signatures output by the signing algorithm as normal signatures. In the proof we will consider an additional type of signatures which we refer to as simulation-type signatures; these will be computationally indistinguishable but easier to simulate. For \( \gamma \in \mathbb{Z}_\rho \), simulation-type signatures are of the form \( \sigma = (S_0, S_1 = S_1 \cdot G^{-\alpha_1 \cdot \rho \gamma}, S_2 = S_2 \cdot G^{\alpha_2 \gamma}, S_3, S_4 = S_4 \cdot G^{\alpha_1 \gamma}, S_5, \ldots, S_7) \) where \((S_0, \ldots, S_7)\) is a normal signature. We give the outline of the proof using some lemmas. Proofs for the lemmas are given after the outline.

**Lemma 3.** Any signature that is accepted by the verification algorithm must be either a normal signature or a simulation-type signature.

To prove this lemma, we introduced two verification equations for signature elements that are not related to a message. We consider a sequence of games. Let \( p_i \) be the probability that the adversary succeeds in

**Game i:** The real security game except that the first \( i \) signatures that are given by the oracle are simulation-type signatures.

**Lemma 5.** There exists an adversary \( B_2 \) such that \( |p^i_{\text{norm}}(\lambda) - p^i_{\text{norm}}(\lambda)| \leq \text{Adv}_{G, B_2}^{\text{dlin}}(\lambda) \).

**Game q:** All signatures given by the oracle are simulation-type signatures.

14
Lemma 6. There exists an adversary $B_3$ such that $p^\text{norm}(\lambda) \leq \text{Adv}^\text{cdh}_{G,B_3}(\lambda)$.

We have shown that in Game $q$, $A$ can output a normal-type forgery with at most negligible probability. Thus, by Lemma 5 we can conclude that the same is true in Game 0 and it holds that

$$
\text{Adv}^\text{lf-ema}_{G,B_0,A}(\lambda) = p_0(\lambda) = p_0^\text{sim}(\lambda) + p_0^\text{norm}(\lambda) \leq p_0^\text{sim}(\lambda) + \sum_{i=1}^{q} |p_i^\text{norm}(\lambda) - p_i^\text{norm}(\lambda)| + p_q^\text{norm}(\lambda)
$$

$$
\leq \text{Adv}^\text{dlin}_{G,B_3}(\lambda) + q\text{Adv}^\text{dlin}_{G,B_5}(\lambda) + \text{Adv}^\text{cdh}_{G,B_5}(\lambda) \leq (q + 2) \cdot \text{Adv}^\text{dlin}_{G,B}(\lambda).
$$

Proof. (of Lemma 6)

We have to show that only normal and simulation-type signatures can fulfil these equations. We ignore verification equations (12) and (13) that establish that $\text{msg}$ is well-formed. A signature has 4 random exponents, $r_1, r_2, s_1, s_2$. A simulation-type signature has additional exponent $\gamma$.

We interpret $S_7$ as $G^{r_1}$, and it follows from verification equation (11) that $S_0$ is $(M_3, H)^{r_1}$. We interpret $S_3$ as $G^{-b_{21}}, S_5$ as $G^{-b_{22}}, S_0$ as $G^{r_2}$. Now we have fixed all exponents of a normal signature. The remaining two verification equations tell us that

$$
e(G^b, S_1) \cdot e(G^{d_{a_1}}, S_2) = e(VV_1^{r_1}, G^{r_2}) \cdot e((VV_1^{r_1})^b, G^r) \cdot e(G^{a_1}, G^{b_{21}}),
$$

$$
e(G^b, S_1) \cdot e(G^{d_{a_2}}, S_4) = e(VV_2^{r_2}, G^{b_{21}}) \cdot e((VV_2^{r_2})^b, G^r) \cdot e(G^{a_2}, G^{b_{22}}) \cdot e(G, G)^{a_1 b}.
$$

We interpret $S_1$ as $G^a V^{\gamma} G^{-a a_1}$. Now we have two equations and two unknowns that fix $S_2$ to $G^{-a} V_1^{a_1} G^{b_{21}}$ and $S_4$ to $V_2^{a_2} G^{b_{22}}$ respectively. If $\gamma = 0$ we have a normal signature, otherwise we have a simulation-type signature.

Proof. (of Lemma 6)

Suppose for contradiction that there is an adversary $A$, which, when playing Game 0 (and thus receiving only normal signatures), produces forgeries which are formed like simulation-type signatures. Then we can construct an adversary $B_1$ for DLIN as follows.

Let $I_{\text{lin}} = (\Lambda, G_1, G_2, G_3, X, Y, Z)$ be an instance of DLIN where $\Lambda = (\rho, \zeta, \zeta, \zeta, \zeta)$ is a Type-I bilinear group setting and $G_1$, $G_2$, $G_3$ are randomly taken from $\zeta^*$ and there exist random $x, y, z, \in \zeta^r$ such that $X = G^x$, $Y = G^y$ and $Z = G^z$ or $G^{\epsilon x y}$. Given $I_{\text{lin}}$, adversary $B_1$ works as follows. It first sets $G := G_3$ and chooses $C, F, U$ at random from $\zeta^*$, and then sets them into $g k$. Next, it chooses $v, v_1, v_2 \in \zeta^r$ and computes $V := G^v$, $V_1 := G^{v_1}$, and $V_2 := G^{v_2}$. (This way we know the discrete log of these values w.r.t. $G_3$.) Then it chooses random $H \in \zeta^*$, $b, \alpha, \rho \in \zeta^r$ and compute:

$$
B := G^\rho,
$$

$$
A_1 := G_1, \quad A_2 := G_2, \quad B_1 := G_1^b, \quad B_2 := G_2^b,
$$

$$
R_1 := V V_1^{a_1} = G_3^{a_1} G_1^c, \quad R_2 := V V_2^{a_2} = G_3^{a_2} G_2^c, \quad W_1 := R_1^{\alpha} = (G_3^{a_1} G_1^c)^b, \quad W_2 := R_2^{\alpha} = (G_3^{a_2} G_2^c)^b,
$$

$$
X_1 := G_3^\rho, \quad X_2 := G^{\alpha a_1 b/\rho} = G_1^{ab/\rho}, \quad K_1 := G_3^\alpha, \quad K_2 := G^{a a_1} = G_3^a.
$$

and sets them into $v k$ and $s k$, accordingly. Note that both the distribution of the public and secret keys are statistically close to that in the real DLIN game. Moreover, to sign random messages, $B_1$ can follow the real signing algorithm by using $s k$.

Suppose that $A$ produces a valid forgery $\sigma^\dagger$ and $\text{msg}^\dagger$. Then $B_1$ proceeds as follows. It parses $\sigma^\dagger$ as $(S_0, \ldots, S_7)$. By Lemma 1 it is shown that if the verification equations hold, then it must hold that $S_1 = G^{a_1} V^{\gamma} G^{-a a_1} \gamma$, $S_2 = G^{-a} V_1^{a_1} G^{b_{21}}$, and $S_4 = V_2^{a_2} G^{b_{22}}$. If this is a simulation-type signature, it holds that $\gamma \neq 0$. According to our choice of public-key, we can rewrite $S_1 = G_1^{a_1} V^{\gamma} G^{b_{21}}$, $S_2 = G_3^{a_1} V_1^{a_1} G_2^{b_{21}}$, and $S_4 = V_2^{a_2} G_3^{b_{22}}$, where $f$ is the discrete log of $G_1$ w.r.t. $G_3$. Thus, if $B_1$
can extract $G_2^{-f_1}, G_2^1, G_1^1$, it can easily break the DLIN instance by testing whether $1 = e(Z, G_2^{-f_1}) \cdot e(G_2^1, X)e(G_1^1, Y)$. $B_1$ can extract such values because the signature includes $S_3 = G_3^{b_1}, S_5 = G_3^{b_2}$, $S_6 = G_3^1$, and $S_7 = G_3^2$, and it has $b_1, b_2$ and the discrete logarithms of $V, V_1, V_2$ w.r.t. $G_3$. Thus, it will be straightforward to extract the above values.

Proof. (of Lemma 5).

Suppose for contradiction that there exists an adversary $\mathcal{A}$ such that the probabilities that $\mathcal{A}$ outputs a normal-type forgery in Game $i$ and Game $i + 1$ differ by a non-negligible amount. Then we will use $\mathcal{A}$ to construct an algorithm $B_2$ that breaks the DLIN assumption.

$B_2$ is given an instance of DLIN: $I_{\text{DLIN}} = (A, G_1, G_2, G_3, X, Y, Z)$.

Note that determining whether a signature is of normal-type or simulation-type naturally corresponds to a DLIN problem: each signature contains $S_7 = G^{r_1}, S_6 = (G^b)^{r_2}$, and $S_1$ which will include $V^{r_3 + r_2}$ or $V^{r_1 + r_2}G^{-a_1a_2\gamma}$ depending on whether this is a normal- or simulation-type signature. (Recall that we define $r = r_1 + r_2$.) If $B_2$ sets $G = G_2, G^i = G_1$, and $V = G_3$, then it seems fairly straightforward to argue based on the DLIN assumption that it will be impossible for the adversary to distinguish normal and simulation-type signatures. However, $B_2$ cannot tell whether $\mathcal{A}$’s forgery is normal- or simulation-type in this simulation. Thus, there will be no way for $B_2$ to take advantage of a change in $\mathcal{A}$’s success probability to solve the DLIN challenge.

The solution is to set things up so that, with high probability $B_2$ can take $S_0$ from the adversary’s forgery and extract something that looks like $G_0^3$ (which will allow $B_2$ to distinguish DLIN tuples and consequently detect simulation-type signatures), but at the same time it is guaranteed that for the $i$-th message, the $G_3$ component of $S_0$ will cancel out, leaving only an $G_2^3$ component which will not allow the challenger itself to know whether a simulated signature is normal-type or simulation-type.

More specifically, the idea will be to choose some secret values $\xi, \beta, \chi, \eta$ and embed them in the parameters so that for message $(C^w, F^w, U^w)$ we get $U^w H = G_2^{\chi w + \eta} G_3^{\beta w + \beta}$. Then $S_0 = (U^w H)^{r_1} = G_2^{(\chi w + \eta) r_1} G_3^{(\beta w + \beta) r_1}$. If $\xi w + \beta \neq 0$, this gives useful information on $G_3^3$ (in particular it will allow $B_2$ to test candidate values), while if $\xi w + \beta = 0$, this has no $G_3$ component and thus doesn’t help at all with finding $G_0^3$. $B_2$ chooses $\xi, \beta$ so that $\xi w + \beta = 0$ for the $w$ used to generate the $i$th message. Furthermore, it will be guaranteed that $\xi, \beta$ are information theoretically hidden even given $w$, so the adversary has only negligible chance of producing another message with $U^w$ such that $\xi w^* + \beta = 0$ as well.

Now we show details of the algorithm for $B_2$. First of all, $B_2$ sets up the message space and generates the public-key in the following manner. $B_2$ sets $(C, F)$, used to define message space $M$, to $(G_2^3, G_3)$ by choosing random $\varphi \leftarrow \mathbb{Z}_p^*$. It chooses random $\xi, \beta, \chi, \eta \leftarrow \mathbb{Z}_p^*$, and computes $U := G_2^\chi G_3^\beta$, and $H := G_2^\beta G_3^\beta$. These values will be uniformly distributed, and independent of $\xi, \beta$. $B_2$ then sets

$$gk = (A, G, C, F, U) := (A, G_2, G_1^\beta, G_3, G_2 G_3^\beta)$$

$B_2$ also sets $B := G_1$, and chooses $V, V_1, V_2$. It must choose these values carefully so that it can compute both $R_i$ and $R_i^w$, and at the same time so that the component $V^+$ of a signature-value $S_i$ gives $B_2$ some useful information (in particular it will allow $B_2$ to derive $G_0^3$). It does this by choosing $\nu_1, \nu_2, \delta \leftarrow \mathbb{Z}_p^*$, and computing $V := G_3^{-a_1a_2\delta}, V_1 := G_2^{\nu_1} G_3^{a_2\delta}$, and $V_2 := G_2^{\nu_2} G_3^{a_1\delta}$.

Next, $B_2$ chooses $a_1, a_2, \alpha, \rho \leftarrow \mathbb{Z}_p^*$ and computes

$$B := G_1, \quad A_1 := G_2^{a_1}, \quad A_2 := G_2^{a_2}, \quad B_1 := G_2^{a_1}, \quad B_2 := G_1^{a_2}$$

$$R_1 := V V_1^{a_1} = G_2^{a_1^{a_1}}, \quad R_2 := V V_2^{a_2} = G_2^{a_2^{a_2}}, \quad W_1 := R_1^b = G_1^{a_1^{a_1}}, \quad W_2 := R_2^b = G_1^{a_2^{a_2}},$$

$$X_1 := G_0^\rho, \quad X_2 := G_3^{a_1^{a_1}}, \quad K_1 := G_2^\rho, \quad K_2 := G_2^{a_1^{a_1}},$$

and sets them into $vk$ and $sk$, accordingly. Note that both of these tuples are distributed statistically close to those produced by Setup and rSIG.Key.

Next $B_2$ simulates signatures for the $j$-th random message as follows.

**Case** $j < i$: It chooses $w_j$ at random and computes $(M_1, M_2, M_3) = (C^{w_j}, F^{w_j}, U^{w_j})$. It can compute a simulation-type signatures for this message since it has $sk$ and $G_0^{a_1a_2} = G_2^{a_1^{a_2}}$. 16
Case \( j = i \): It chooses \( w \) such that \( \xi w + \beta = 0 \) and computes \((M_1, M_2, M_3) = (C^w, F^w, U^w)\). Note that since no information about \( \xi, \beta \) is revealed this message will look appropriately random to the adversary. It will implicitly hold that \( r_1 = y \) and \( r_2 = x \). \( B_2 \) computes \( S_0 = G_{\delta x} = G_1^\gamma = X \) and \( S_2 = G^{r_1} = G_2^{r_1} = Y \). Recall that it chose \( U, H \) such that \( U^w H = G_2^{\nu_1 + \eta} \). Thus, \( B_2 \) can compute \( S_0 = (M_3 H)^{r_1} = Y^{\nu_1 + \eta} \).

What remains is to compute \( S_1, S_2, S_4 \). Note that this involves computing \( V^r, V_1^r \), and \( V_2^r \) respectively. This is where \( B_2 \) will embed its challenge. Recall that \( V = G_3^{\alpha_1 a_2 \delta} \). Thus, it will compute \( V^r = (G_3^{r_1 + r_2})^{-a_1 a_2 \delta} \). It follows that if \( Z = G_3^{\delta y} \) this will be correct; if \( Z = G_3^{\delta z} \) then there will be an extra factor of \( G_3^{-\alpha a_2 \delta} \). If \( B_2 \) lets \( G_3^{\delta z} = G_3^{\delta (z-x+y)} \) (which is uniformly random from the adversary’s point of view), then this is distributed exactly as it should be in a simulation-type signature. Thus, \( B_2 \) computes \( S_1 \) which should be either \( G_3^{\alpha_1 V^r} \) or \( G^{\alpha_1 V^r} G^{-\alpha a_2 \gamma} \).

\( B_2 \) can try to apply the same approach to compute \( V_1^r \) to get \( S_2 \). However, recall that \( B_2 \) sets \( V_1 = G_2^{\alpha_2 \delta} \). Thus, computing \( V_1^r \) involves computing \( G_2^r \), which \( B_2 \) cannot do. (If it could it could use that to break the DLIN assumption.) To get around this, \( B_2 \) uses \( z_1, z_2 \). It chooses random \( s_1, s_2 \) and implicitly sets \( G_2^{z_1} = G_2^{r_1 + r_2 + s_1} \) and \( G_2^{z_2} = G_2^{r_2 + r_2 + s_2} \). While it cannot compute these values, it can compute \( G^{-z_1} = G_1^{s_1} G_1^{-s_2} \) and \( G^{-z_1} = G_2^{s_2} G_2^{-s_1} \). Then to generate \( S_2, B_2 \) can compute \( G_2^{s_1} G_2^{s_2} G_1^{-s_1} G_1^{-s_2} G_2^{r_1 + r_2 + s_1} G_2^{r_2 + r_2 + s_2} \) as \( G_2^{s_1} G_2^{s_2} G_1^{-s_1} G_1^{-s_2} G_2^{r_1 + r_2 + s_1} G_2^{r_2 + r_2 + s_2} \) and \( G^{\alpha_2 V_1^r} G^{\gamma} = G^{\alpha_2 V_1^r} G^{\gamma} \).

If \( Z = G_3^{\delta y} = G_3^{\gamma} \), then this will be \( G_2^{\alpha_2 \delta} G_3^{\gamma} G_2^{r_1 + r_2 + s_1} G_2^{r_2 + r_2 + s_2} \). Thus, computing \( V_1^r \) involves computing \( G_2^r \), which \( B_2 \) cannot do. (If it could it could use that to break the DLIN assumption.) To get around this, \( B_2 \) uses \( z_1, z_2 \). It chooses random \( s_1, s_2 \) and implicitly sets \( G_2^{z_1} = G_2^{r_1 + r_2 + s_1} \) and \( G_2^{z_2} = G_2^{r_2 + r_2 + s_2} \). While it cannot compute these values, it can compute \( G^{-z_1} = G_1^{s_1} G_1^{-s_2} \) and \( G^{-z_1} = G_2^{s_2} G_2^{-s_1} \). Then to generate \( S_2, B_2 \) can compute \( G_2^{s_1} G_2^{s_2} G_1^{-s_1} G_1^{-s_2} G_2^{r_1 + r_2 + s_1} G_2^{r_2 + r_2 + s_2} \) as \( G_2^{s_1} G_2^{s_2} G_1^{-s_1} G_1^{-s_2} G_2^{r_1 + r_2 + s_1} G_2^{r_2 + r_2 + s_2} \) and \( G^{\alpha_2 V_1^r} G^{\gamma} = G^{\alpha_2 V_1^r} G^{\gamma} \).

Case \( j > i \): It chooses \( w \) and computes \( m_j = (M_1, M_2, M_3) = (C^w, F^w, U^w) \) and a signature \( \sigma \) according to \( \sigma \). It outputs \( \sigma, m_j \).

On receiving forgery \( S = (S_0, S_1, \ldots, S_7) \) and \((M_1, M_2, M_3) = (C^w, F^w, U^w) \) for some message \( w \), \( B_2 \) outputs 1 if and only if
\[
e(S_0, G_1) \cdot (M_2^r G_3^\delta, S_6) = e(S_1 G_2^{\alpha_1})^{1/-\alpha a_2 \delta}, (M_1^{1/\delta})^\delta G_1^\gamma \cdot e(S_7, (M_1^{1/\delta})^\delta G_1^\gamma).
\]
By Lemma 3, we are guaranteed that if the signature \( S \) verifies, then there must exist \( w, r_1, r_2, \gamma \) such that \( S_0 = (U^w H)^r, S_1 = G^{\alpha w_1} V^{r} G^{-a_1 a_2 \gamma}, S_6 = G^{br_2} \), and \( S_7 = G^{r_1} \) where \( r = r_1 + r_2 \). We are also guaranteed that \( M_1 = (G_1^w)^w \) and \( M_2 = G_3^{w_2} \).

Rephrased in terms of our parameters, this means

\[
S_0 = (G_2^{w+\eta} G_3^{\xi w+\beta})^{r_1} \\
S_1 = G_2^{\alpha w_1} G_3^{-a_1 a_2 \delta r_1} G_2^{-a_1 a_2 \gamma} \\
S_7 = G_2^{r_1}.
\]

Plugging this into the above computation we get that \( B_2 \) will output 1 if and only if

\[
e((G_2^{w+\eta} G_3^{\xi w+\beta})^{r_1}, G_1) \cdot e((G_3^{w+\beta}, G_1^{2 r_2})
\]

\[
= e((G_2^{\alpha w_1} G_3^{-a_1 a_2 \delta r} G_2^{-a_1 a_2 \gamma} G_2^{-\alpha w_1})^{1/(-a_1 a_2 \delta)}, (G_1^w)^{2 \beta} G_1^{r_2}) \cdot e(G_2^{r_1}, (G_1^w) \cdot (G_1^w)^r_2).
\]

Simplifying the left side to

\[
e((G_2^{\alpha w_1} G_3^{\xi w+\beta})^{r_1}, G_1) \cdot e(G_3^{w+\beta}, G_1^{r_2})
\]

\[
= e(G_2, G_1)^{(\xi w+\eta)r_1} \cdot e(G_3, G_1)^{(\xi w+\beta)r_1} \cdot e(G_3, G_1)^{(\xi w+\beta)r_2}
\]

and the right side to

\[
e((G_3^{-a_1 a_2 \delta r} G_2^{-a_1 a_2 \gamma} G_2^{-\alpha w_1})^{1/(-a_1 a_2 \delta)}, (G_1^{w+\beta})r_2
\]

\[
= e(G_3 G_2^{\beta}, G_1^{w+\beta}) \cdot e(G_2, G_1)^{(\xi w+\beta)r_1} \cdot e(G_3, G_1)^{(\xi w+\beta)r_2}
\]

and by dividing out all the pairings of the left side we obtain the simplified equation

\[
1 = e(G_2, G_1)^{(\gamma/\delta)(\xi w+\beta)}
\]

which is true if and only if either \( \xi w + \beta = 0 \) or \( \gamma = 0 \). Since \( \xi w_1 + \beta \) is a pairwise-independent function, we are guaranteed that \( \xi w + \beta = 0 \) happens with negligible probability. Thus, we conclude that \( B_2 \) outputs 1 iff \( \gamma = 0 \) and this was a normal-type signature, and \( B_2 \) outputs 0 iff \( \gamma \neq 0 \) and this was a simulation-type signature.

**Proof.** (of Lemma 6).

Suppose that there exists an adversary \( A \) that outputs normal-type forgeries with non-negligible probability in Game 3. Then we construct an adversary \( B_3 \) for the CDH problem as follows.

\( B_3 \) is given \( X = G^r, Y = G^y \) and must compute \( G^{xy} \). \( B_3 \) will proceed as follows.

**Message space setup and key generation:** \( B_3 \) will implicitly set \( a := xy \) and \( a_1 := y \). It chooses \( b, a_1 \) at random from \( Z_p^* \). \( B_3 \) needs to be able to compute \( V^{a_1^2} \), so it chooses random \( v_2 \in Z_p^* \) and sets \( V_2 := G^{v_2} \). It also wants to have the discrete logarithm of \( V_1 \), so it will choose random \( v_1 \in Z_p^* \) and set \( V_1 := G^{v_1} \). \( B_3 \) chooses \( U, C, F \in G \) and \( H, V \in G^* \) at random, sets \( G^{a_2} := Y \), and computes \( \forall V^{a_2}_i = V Y^{v_2} \). It chooses random \( \rho' \in Z_p^* \) and sets \( X_1 := X^{\rho'} \) and \( X_2 := Y^{a/b\rho'} \). The rest of the parameters can be constructed honestly.

**Signature queries:** On a signature query, \( B_3 \) chooses \( w \) at random, computes \( \{M_1, M_2, M_3\} = (C^w, F^w, U^w) \), and generates a simulation-type signature as follows. It chooses random \( r_1, r_2, z_1, z_2 \in Z_p \) and random \( s \in Z_p \) and implicitly sets \( \gamma := (x - s) \). \( B_3 \) computes

\[
S_1 := Y^{s a_1 V^r} = G^{y a_1 V^r} = G^{xy a_1 + xy a_1 - x y a_1 V^r} = G^{xy a_1 V^r} G^{(x-s) y a_1} = G^{xy a_1 V^r} G^{-\gamma a_2 a_1} ;
\]

\[
S_2 := Y^{x y V^r} G_2 = G^{x y V^r} G_2 = G^{x y + x y - x y V^r} G_2 = G^{x y V^r} G_2 G^{(x-s) y} = G^{-a_1} V_1 G_2 G^{a_2} ;
\]

\[
S_4 := V_1^{v_2} G^{2 a_1} G^{-s a_1} = V_2^{v_2} G_2 G^{x a_1} G^{-s a_1} = V_2^{v_2} G_2 G^{(x-s) a_1} = V_2^{v_2} G^{s a_1} ;
\]

The rest of the signature can be computed honestly.
**Adversary’s forgery:** When the adversary outputs a normal-type forgery, there exists \( r_1, r_2, z_1 \) such that \( S_2 = G^{-\alpha} V_1^{r_1+z_1} G^{z_1}, S_3 = (G^b)^{-z_1}, S_6 = G^{r_2b}, \) and \( S_7 = G^{r_1} \). Thus, \( B_3 \) can compute

\[
S_2^{-1} \cdot S_7^{-1} S_6^{1/b} S_3^{-1/b} = G^\alpha V_1^{-r_1} G^{-z_1} \cdot (G^{r_1})^{v_1} (G^{r_2b})^{v_1/b} ((G^b)^{-z_1})^{-1/b} \\
= G^\alpha V_1^{-r_1-r_2} G^{-z_1} \cdot (G^{v_1})^{v_1} (G^b)^2 G^{z_1} \\
= G^\alpha V_1^{-r_1-r_2} G^{-z_1} \cdot V_1^{v_1} V_1^2 G^{z_1} \\
= G^\alpha.
\]

\( B_3 \) will output this value. By our choice of parameters, recall that \( \alpha = xy \), so it holds that \( G^\alpha = G^{xy} \) as desired.

That is, \( B_3 \) can solve the CDH problem.

Let \( \text{MSGGen} \) be an extended random message generator that first chooses \( \omega = m \) randomly from \( \mathbb{Z}_p \) and then computes \( \text{msg} = (C^m, F^m, U^m) \). Note that this is what the reduction algorithm does in the proof of Theorem 7. Therefore, the same reduction algorithm works for the case of extended random message attacks with respect to message generator \( \text{MSGGen} \). We thus have the following.

**Corollary 1.** Under the DLIN assumption, \( r\text{SIG} \) scheme is UF-XRMA w.r.t. the message generator that provides \( \omega = m \) for every message \( \text{msg} = (C^m, F^m, U^m) \). In particular, for any p.p.t. algorithm \( A \) against \( r\text{SIG} \) that is given at most \( q_s(\lambda) \) signatures, there exists p.p.t. algorithm \( B \) such that \( \text{Adv}_{r\text{SIG}, A}(\lambda) \leq (q_s(\lambda) + 2) \cdot \text{Adv}_{G_2, B}(\lambda) \).

### 5.4 Security and efficiency of resulting SIG1

Let SIG1 be the signature scheme obtained from TOS and \( r\text{SIG} \) by following the first generic construction in Section 4. From Theorems 1, 5.2, and 7 the following is immediate.

**Theorem 8.** SIG1 is a structure-preserving signature scheme that yields constant-size signatures, and is UF-CMA under the DLIN assumption. In particular, for any p.p.t. algorithm \( A \) for SIG1 making at most \( q_s(\lambda) \) signing queries, there exists p.p.t. algorithm \( B \) such that \( \text{Adv}_{\text{SIG1}, A}^{\text{uf-cma}}(\lambda) \leq (q_s(\lambda) + 3) \cdot \text{Adv}_{G_2, B}(\lambda) + 1/p(\lambda) \), where \( p(\lambda) \) is the size of the groups produced by \( G \).

The efficiency is summarized in Table 1. It is compared to an existing efficient structure-preserving scheme in [4, Section 5.2]. (The original scheme is presented over asymmetric bilinear groups. It is translated to the symmetric setting for our purpose.) We measure the efficiency by counting the number of group elements and the number of pairing product equations for verifying a signature.

| Scheme | \( |\text{msg}| \) | \( |g^k| + |v^k| \) | \( |\sigma| \) | \#(PPE) | Assumption |
|--------|-----------------|-----------------|---------|----------|------------|
| \[4\] | \( k \) | \( 2k + 13 \) | \( 7 \) | \( 2 \) | \( q\text{-SFP} \) |
| SIG1 | \( k \) | \( 2k + 21 \) | \( 14 \) | \( 7 \) | \( \text{DLIN} \) |

**Table 1:** Efficiency Comparison of constant-size SPS over symmetric bilinear groups.

In Table 2 we also assess the cost of proving possession of valid signatures and messages by using Groth-Sahai NIWI and NIZK proof system. Columns "\( \sigma \)" indicate the case where a witness is a valid signature. (Regarding the signature scheme from [4], we optimize by putting randomizable parts of a signature in the clear.) The message is put in the clear. Similarly, columns "\( (\sigma, \text{msg}) \)" show the case where a witness consists of a valid signature and a message. Details of each assessment are as follows.

For NIWI, the cost of proving valid \( \sigma \) is counted by

\[
|\text{NIWI}(\sigma)| = |\text{com}| \times |\sigma_{\text{wit}}| + |\sigma_{\text{rad}}| + |\pi_{\text{NL}}| \times \#(\text{NLPPPE}) + |\pi_{L}| \times \#(\text{LPPE}) \tag{14}
\]

and the cost of proving valid \( (\sigma, \text{msg}) \) is counted by

\[
|\text{NIWI}(\sigma, \text{msg})| = |\text{com}| \times (|\sigma_{\text{wit}}| + |\text{msg}|) + |\sigma_{\text{rad}}| + |\pi_{\text{NL}}| \times \#(\text{NLPPPE}) + |\pi_{L}| \times \#(\text{LPPE}) \tag{15}
\]
where \( |\sigma_{NL}|, |\sigma_{rd}|, |\sigma_{wl}|, |com| \) are the size of a proof for a linear/non-linear relation, randomizable parts of a signature, rest of the parts in the signature, and commitment per witness, respectively. Also, LPPE and NLPPE denotes the linear and non-linear PPEs in the verification predicate of the signature scheme. For NIZK, we need to turn either of input constants in every constant pairing into a witness, and prove that it is committed correctly. Those proof of correct commitment of public constants are done by proving a relation in multiscalar multiplication equations, whose size is denoted by \( |\pi_{MS}| \). Let \( |\text{CONST}| \) denote the number of constant pairings in the verification PPE. The costs for ZK are estimated by

\[
|\text{NIZK}(\sigma)| = |com| \times (|\sigma_{wl}| + |\text{CONST}|) + |\sigma_{rd}| + |\pi_{NL}| \times (#\text{NLPPE} + |\text{CONST}|) + |\pi_{L}| \times (#\text{LPPE}) + |\pi_{MS}| \times |\text{CONST}|
\]

(16)

and the cost of proving valid \((\sigma, \text{msg})\) is counted by

\[
|\text{NIZK}(\sigma, \text{msg})| = |com| \times (|\sigma_{wl}| + |\text{msg}| + |\text{CONST}|) + |\sigma_{rd}|
\]

(17)

According to [37], we have \((|\text{com}|, |\pi_{L}|, |\pi_{NL}|) = (3, 3, 9)\) in \(G\), and \(|\pi_{MS}| = 3\) in \(\mathbb{Z}_p\). Proof \(\pi_{MS}\) can consist of elements in \(G\) by describing the relation of correct commitment of public value with a pairing product equation. It turns entire proof to be structure-preserving with increased proof size.

For [4], we have \(|\sigma_{wl}| = 3, |\sigma_{rd}| = 4\). Since the verification consists of 2 non-linear equations, we have \(#\text{NLPPE} = 0\) and \(#\text{LPPE} = 2\). This results in \(|\text{NIWI}(\sigma)| = 3 \cdot 3 + 4 + 9 \cdot 0 + 3 \cdot 2 = 19\) and \(|\text{NIWI}(\sigma, \text{msg})| = 3 \cdot (3 + k) + 4 + 9 \cdot 0 + 3 \cdot 2 = 3k + 19\). For NIZK(\(\sigma\)), we have \(#\text{CONST} = 6 + k\) constant pairings in the signature verification. (In detail, \(k\) comes from the pairings that involve the message, 4 is from the pairings that only involves public-key, and 2 is from the pairings that involves the randomizable part of the signature.) Thus \(3 \cdot (6 + k)\) group elements and \(Zp\) elements are needed on top of NIWI(\(\sigma\)). For NIZK(\(\sigma, \text{msg}\)), on the other hand, the message is hidden as a witness. Thus we can set \(#\text{CONST} = 6\) and the additional cost on NIWI(\(\sigma, \text{msg}\)) is \(3 \cdot 6\) group elements and \(Zp\) elements.

Regarding to SIG1, whole signature is considered as a witness. Thus we have \(|\sigma_{wl}| = 14\) and \(|\sigma_{rd}| = 0\). And the verification consists of 6 linear equations and 1 non-linear equation; \(#\text{NLPPE} = 1\) and \(#\text{LPPE} = 6\). We thus have \(|\text{NIWI}(\sigma)| = 3 \cdot 14 + 0 + 9 \cdot 1 + 3 \cdot 6 = 69\) and \(|\text{NIWI}(\sigma, \text{msg})| = 3 \cdot (14 + k) + 0 + 9 \cdot 1 + 3 \cdot 6 = 3k + 69\). For NIZK(\(\sigma\)), we have \(#\text{CONST} = 1 + k\) constant pairings in the signature verification, which results in adding \(3 + 3k\) elements in both \(G\) and \(\mathbb{Z}_p\) to NIWI(\(\sigma\)). Finally, for NIZK(\(\sigma, \text{msg}\)), we have \(#\text{CONST} = 1\), which adds 3 elements in \(G\) and \(\mathbb{Z}_p\) to NIWI(\(\sigma, \text{msg}\)).

<table>
<thead>
<tr>
<th>Scheme</th>
<th>NIWI((\sigma))</th>
<th>NIWI((\sigma, \text{msg}))</th>
<th>NIZK((\sigma))</th>
<th>NIZK((\sigma, \text{msg}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG1</td>
<td>19</td>
<td>3k + 19</td>
<td>(3k + 37, 3k + 18)</td>
<td>(3k + 37, 18)</td>
</tr>
<tr>
<td></td>
<td>69</td>
<td>3k + 69</td>
<td>(3k + 72, 3k + 3)</td>
<td>(3k + 72, 3)</td>
</tr>
</tbody>
</table>

Table 2: Size of GS proofs and commitments for proving possession of a valid signature and message in WI or ZK. Numbers count elements in \(G\). For ZK, \((x, y)\) denotes \(x\) elements in \(G\) and \(y\) elements in \(\mathbb{Z}_p\).

6 Instantiating SIG2

We instantiate the POS and xSIG building blocks of our second generic construction to obtain our second SPS scheme. Here we choose the Type-III bilinear group setting. The resulting SIG2 scheme is an efficient structure-preserving signature scheme based on SXDH and XDLIN.

6.1 Setup for Type-III groups

The following setup procedure is common for all building blocks in this section. The global parameter \(gk\) is given to all functions implicitly.

- Setup(\(1^λ\)): Run \(\Lambda = (p, G_1, G_2, G_T, e) \leftarrow G(1^λ)\) and choose generators \(G \in G_1^∗\) and \(\hat{G} \in G_2^∗\). Also choose \(u, f_1, f_2\) randomly from \(\mathbb{Z}_p^∗\), compute \(F_1 := G^{f_1}, \hat{F}_1 := \hat{G}^{f_1}, F_2 := G^{f_2}, \hat{F}_2 := \hat{G}^{f_2}, U := G^u, \hat{U} := \hat{G}^u\), and output \(gk := (\Lambda, G, \hat{G}, F_1, F_2, \hat{F}_2, U, \hat{U})\).
A $gk$ defines a message space $\mathcal{M}_k = \{(\hat{F}_1^m, \hat{F}_2^m, \hat{M}^m) \in (G^*_2)^3 \mid m \in \mathbb{Z}_p\}$ for the XRMA-secure signature scheme in this section. For our generic construction to work, the partial one-time signature scheme must have the same key space.

### 6.2 Partial one-time signatures for unilateral messages

We first construct a partial one-time signature scheme, POSu2, for messages in $G^*_2$ for $k > 0$. The suffix "u2" indicates that the scheme is unilateral and messages are taken from $G_2$. Correspondingly, POSu1 refers to the scheme whose messages belong to $G_1$, which is obtained by swapping $G_2$ and $G_1$ in the following description. In the following section we will show how to combine POSu2 and POSu1 to obtain signatures on bilateral messages consisting of elements from both $G_1$ and $G_2$.

Our POSu2 scheme is a minor refinement of the one-time signature scheme introduced in [8]. It comes, however, with a security proof for the new security model. Basically, a one-time public-key in our scheme consists of one element in the source group $G_1$, the opposite group from the one to which the messages belong. This property is very useful when we move on to construct a POS scheme for signing bilateral messages.

Like the tags in the TOS of Section 5.2 the one-time public-keys of POSu2 will have to be in an extended form, $(F_1^u, F_2^u, U^u)$, to meet the constraint from xSIG presented in the sequel. The extended part $(F_1^u, F_2^u)$ can be dropped if unnecessary.

#### Scheme POSu2

**POSu2.Key(gk):** Take generators $U$ and $\hat{U}$ from $gk$. Choose $w_r$ uniformly from $\mathbb{Z}_p^*$ and compute $G_r := U^{w_r}$. For $i = 1, \ldots, k$, uniformly choose $\chi_i$ and $\gamma_i$ from $\mathbb{Z}_p$ and compute $G_i := U^{\chi_i}G_r^{\gamma_i}$. Output $pk := (G_r, G_1, \ldots, G_k) \in \mathbb{G}^{k+1}_1$ and $sk := (\chi_1, \gamma_1, \ldots, \chi_k, \gamma_k, w_r)$.

**POSu2.Update(gk):** Take $F_1, F_2, U$ from $gk$. Choose $a \leftarrow \mathbb{Z}_p$ and output $opk := (F_1^u, F_2^u, U^u) \in \mathbb{G}^3_1$ and $osk := a$.

**POSu2.Sign(sk, msg, osk):** Parse $msg$ into $(\hat{M}_1, \ldots, \hat{M}_k) \in \mathbb{G}_2^k$. Take $a$ and $w_r$ from $osk$ and $sk$, respectively. Choose $\rho$ randomly from $\mathbb{Z}_p$ and compute $\zeta := a - \rho w_r \mod p$. Then compute and output $\sigma := (\hat{Z}, \hat{R}) \in \mathbb{G}^2_2$ as the signature, where

$$
\hat{Z} := \hat{U}^\zeta \prod_{i=1}^k \hat{M}_i^{-\chi_i} \quad \text{and} \quad \hat{R} := \hat{U}^p \prod_{i=1}^k \hat{M}_i^{-\gamma_i}. \tag{18}
$$

**POSu2.Vrf(pk, opk, msg, \sigma):** Parse $\sigma$ as $(\hat{Z}, \hat{R}) \in \mathbb{G}^2_2$, $msg$ as $(\hat{M}_1, \ldots, \hat{M}_k) \in \mathbb{G}^k_2$, and $opk$ as $(A_1, A_2, A)$. Return 1, if

$$
e(A, \hat{U}) = e(U, \hat{Z}) e(G_r, \hat{R}) \prod_{i=1}^k e(G_i, \hat{M}_i) \tag{19}
$$

holds. Return 0, otherwise.

Scheme POSu2 is structure-preserving and has uniform one-time public-keys by construction. It is correct as the following relation holds for the verification equation and the computed signatures:

$$
e(U, \hat{Z}) e(G_r, \hat{R}) \prod_{i=1}^k e(G_i, \hat{M}_i) = e(U, \hat{U}^\zeta \prod_{i=1}^k \hat{M}_i^{-\chi_i}) e(G_r, \hat{U}^p \prod_{i=1}^k \hat{M}_i^{-\gamma_i}) \prod_{i=1}^k e(U^{\chi_i}G_r^{\gamma_i}, \hat{M}_i) = e(U, \hat{U}^\zeta) e(U^{w_r}, \hat{U}^p) = e(U^{\zeta+w_r}, \hat{U}) = e(A, \hat{U}).
$$

**Theorem 9.** POSu2 is strongly unforgeable against OT-CMA if DBP$_1$ holds. In particular, for all p.p.t. algorithms $A$ there exists a p.p.t. algorithm $B$ such that $\text{Adv}_{\text{POSu2}_A}^\text{OT-CMA}(\lambda) \leq \text{Adv}_{B}^\text{DBP}_1(\lambda) + 1/p(\lambda)$, where $p(\lambda)$ is the size of the groups produced by $G$. Moreover, the run-time overhead of the reduction $B$ is a small number of multi-exponentiations per signing or key query.
Proof. Using a successful forger $A$ against POSu2 as a black-box, we construct $B$ that is successful in breaking DBP$_1$. Given instance $I_{dbp1} = (\Lambda, G_z, G_r)$ of DBP$_1$, algorithm $B$ simulates the attack game against POSu2 as follows.

Key Generation: Set $U := G_z, \hat{U} \leftarrow G_z^*$, and $g_k := (\Lambda, U^g, \hat{U}^g, U^f_1, \hat{U}^f_1, U^f_2, \hat{U}^f_2, U, \hat{U})$ for $g, f_1, f_2 \leftarrow Z_p$. Then generate $pk$ by following POSu2.Key$(gk)$ except that $G_r$ is taken from $I_{dbp1}$.

One-time key query to $O_1$: On receiving a one-time key query, generate $\zeta, \rho \leftarrow Z_p$, compute $A := U^\zeta G_r^\rho$, $A_1 := A_1^\zeta$, $A_2 := A_2^\zeta$ with $f_1$ and $f_2$ generated in Setup, and return $opk := (A_1, A_2, A)$.

Signature query to $O_2$: On receiving a signing query, $msg(j)$, compute $\hat{Z}$ and $\hat{R}$ as described in (18) taking $\chi_i$ and $\gamma_i$ from those used in key generation and $\zeta$ and $\rho$ from those used in simulating $O_1$. Then output $\sigma := (\hat{Z}, \hat{R})$. For each signing, transcript $(opk, \sigma, msg)$ is recorded.

When $A$ outputs a forgery $(opk^\dagger, \sigma^\dagger, msg^\dagger)$, algorithm $B$ searches the records for $(opk, \sigma, msg)$ such that $opk^\dagger = opk$ and $(\sigma^\dagger, msg^\dagger) \neq (msg, \sigma)$. If no such entry exists, $B$ aborts. Otherwise, $B$ computes

$$
\hat{Z}^* := \frac{\hat{Z}^+}{Z} \prod_{i=1}^k \left( \frac{M_i^1}{M_i} \right)^{\chi_i}, \quad \text{and} \quad \hat{R}^* := \frac{\hat{R}^1}{R} \prod_{i=1}^k \left( \frac{M_i^1}{M_i} \right)^{\gamma_i},
$$

(20)

where $(\hat{Z}, \hat{R}, \hat{M}_1, \ldots, \hat{M}_k)$ and its dagger counterpart are taken from $(\sigma, msg)$ and $(\sigma^\dagger, msg^\dagger)$, respectively. $B$ finally outputs $(\hat{Z}^*, \hat{R}^*)$. This completes the description of $B$.

We first claim that the simulation by $B$ is perfect; keys distribute uniformly due to the randomness of $G_z$ and $G_r$ in the given instance, and signaures are computed following the legitimate procedure. It is noted that $f_1^g$ and $f_2^g$ corresponds to $f_1$ and $f_2$ in the real execution. Accordingly, $A$ outputs a successful forgery with noticeable probability and $B$ finds a corresponding record $(opk, \sigma, msg)$.

We next claim that each $\chi_i$ is independent of the view of $A$. Concretely, we show that, if coins $\chi_1, \ldots, \chi_k$ are distributed uniformly over $(Z_p)^k$, other coins $\gamma_1, \ldots, \gamma_k, \zeta(1), \rho(1), \ldots, \zeta(k), \rho(k)$ are distributed uniformly and $A$'s view is consistent. Observe that the view of $A$ making $q$ signing queries consists of independent group elements $(U, \hat{U}), (G, F_1, F_2), (G_r, G_1, \ldots, G_k)$ and $(A(j), \hat{Z}(j), \hat{M}_1(j), \ldots, \hat{M}_k(j))$ for $j = 1, \ldots, q_s$. (Note that $\hat{G}, \hat{F}_1, \hat{F}_2$, and $A_1(j), A_2(j)$, and $\hat{R}(j)$ for all $j$ are uniquely determined by the other group elements.) We represent the view by the discrete-logarithms of these group elements with respect to bases $U$ and $\hat{U}$ in each group. Namely, the view is represented by $(g, f_1^g, f_2^g, w_r, w_1, \ldots, w_k)$ and $(a(j), z(j), m_1^{(j)}, \ldots, m_k^{(j)})$ for $j = 1, \ldots, q_s$. To be consistent, the view and the coins must satisfy the following relations:

$$
w_i = \chi_i + w_r \gamma_i \quad \text{for } i = 1, \ldots, k, \quad \text{and} \quad \begin{align*}
a(j) &= \zeta(j) + w_r \rho(j), \quad \text{and} \quad z(j) = \zeta(j) - \sum_{i=1}^k m_i^{(j)} \chi_i \quad \text{for } j = 1, \ldots, q_s. \quad (21) \end{align*}
$$

From relation (21), $(\gamma_1, \ldots, \gamma_k)$ is distributed uniformly according to the uniform distribution of $(\chi_1, \ldots, \chi_k)$. From the second relation in (22) for every $j$, if $(m_1, \ldots, m_k) \neq (0, \ldots, 0)$ then $\zeta(j)$ is distributed uniformly according to the uniform distribution of $(\chi_1, \ldots, \chi_k)$. Then, from the first relation of (22), $\zeta(j)$ is distributed uniformly, too. If $(m_1, \ldots, m_k) = (0, \ldots, 0)$, then $\zeta(j)$ and $\rho(j)$ are independent of $(\chi_1, \ldots, \chi_k)$ and can be uniformly assigned by following the first relation in (22).

Finally, we claim that $(\hat{Z}^*, \hat{R}^*)$ is a valid solution to the given instance of DBP$_1$. Since both forged and recorded signatures fulfill the verification equation, dividing the equations results in

$$
1 = e \left( U, \frac{\hat{Z}^*}{Z} \right) e \left( G_r, \frac{\hat{R}^*}{R} \right) \prod_{i=1}^k e \left( U^{\chi_i} G_r^{\gamma_i}, \frac{M_i^1}{M_i} \right) = e \left( U, \frac{\hat{Z}^*}{Z} \right) \prod_{i=1}^k \left( \frac{M_i^1}{M_i} \right)^{\chi_i} e \left( G_r, \frac{\hat{R}^*}{R} \prod_{i=1}^k \left( \frac{M_i^1}{M_i} \right)^{\gamma_i} \right) = e \left( U, \hat{Z}^* \right) e \left( G_r, \hat{R}^* \right).
$$

22
What remains is to prove that $\tilde{Z}^* \neq 1$. If $msg^1 \neq msg^{(i)}$, there exists $\ell \in \{1, \ldots, k\}$ such that $\frac{M^i}{M^\ell} \neq 1$. As already proven, $x_\ell$ is independent of the view of $A$ and of the other $x_i$ values. Thus $\left(\frac{M^i}{M^\ell}\right)^{x_\ell}$ is distributed uniformly over $\mathbb{G}_2$ and so is $\tilde{Z}^*$. Accordingly, $Z^* = 1$ holds only if $Z^\ell = \tilde{Z} \prod (M^i/M^\ell)^{-x_i}$, which happens only with probability $1/p$ over the choice of $x_\ell$. Otherwise, if $msg^1 = msg^{(i)}$ and $(Z^i, R^i) \neq (Z, R)$, then, we have $Z^i = Z$ to fulfill $Z^* = 1$. However, if $Z^i = Z$, then $R^i = R$ holds since the verification equation uniquely determines such $R^i$ and $R$. Thus $msg^1 = msg^{(i)}$ and $(Z^i, R^i) \neq (Z, R)$ can never happen. We thus have $Adv_{\text{PoS}u1, A}(\lambda) \leq Adv_{\text{PoS}d1, G, B}(\lambda) + 1/p$ as stated.

### 6.3 Partial one-time signatures for bilateral messages

Using $\text{PoS}u1$ for $msg \in \mathbb{G}_1^{k_1+1}$ and $\text{PoS}u2$ for $msg \in \mathbb{G}_2^{k_2}$, we construct a PoSb scheme for signing bilateral messages $(msg_1, msg_2) \in \mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$. The scheme is a simple two-story construction where $msg_2$ is signed by $\text{PoS}u2$ with one-time secret-key $osk_2 \in \mathbb{G}_1$ and then the one-time public-key $opk_2$ is attached to $msg_1$ and signed by $\text{PoS}u1$. Public-key $opk_2$ is included in the signature, and $opk_1$ is output as a one-time public-key for $\text{PoS}b$.

**[Scheme PoSb]**

**PoSb.Key($gk$):** Run $(pk_1, sk_1) \leftarrow \text{PoS}u1.\text{Key}(gk)$ for message size $k_1+1$ and $(pk_2, sk_2) \leftarrow \text{PoS}u2.\text{Key}(gk)$ for message size $k_2$. Set $pk := (pk_1, pk_2)$ and $sk := (sk_1, sk_2)$, and output $(pk, sk)$.

**PoSb.Update($gk$):** Run $(opk, osk) \leftarrow \text{PoS}u1.\text{Update}(gk)$ and output $(opk, osk)$.

**PoSb.Sign($sk$, $msg$, $osk$):** Parse $msg$ into $(msg_1, msg_2) \in \mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$, and $sk$ into $(sk_1, sk_2)$. Run $(opk_2, osk_2) \leftarrow \text{PoS}u2.\text{Update}(gk)$, and compute $\sigma_2 \leftarrow \text{PoS}u2.\text{Sign}(sk_2, msg_2, osk_2)$ and $\sigma_1 \leftarrow \text{PoS}u1.\text{Sign}(sk_1, msg_1, opk_2, osk_2)$). Output $\sigma := (\sigma_1, \sigma_2, opk_2)$.

**PoSb.Vrf($pk$, $opk$, $msg$, $\sigma$):** Parse $msg$ into $(msg_1, msg_2) \in \mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$, and $\sigma$ into $(\sigma_1, \sigma_2, opk_2)$. If $1 = \text{PoS}u1.\text{Vrf}(pk_1, opk, (msg_1, opk_1, \sigma_1)) = \text{PoS}u2.\text{Vrf}(pk_2, opk_2, msg_2, \sigma_2)$, output 1. Otherwise, output 0.

We consider dropping unnecessary extended part from $opk_2$ so that it consists of only one group element. Then, for a message in $\mathbb{G}_1^{k_1} \times \mathbb{G}_2^{k_2}$, the above PoSb uses a public-key of size $(k_2 + 1, k_1 + 2)$, yields a one-time public-key of size $(0, 3)$, and a signature of size $(3, 2)$. Verification requires 2 pairing product equations. A one-time public-key, which is treated as a message to $\text{xSIG}$ in this section, is of the form $opk = (F_1^2, F_2^2, U^\alpha) \in \mathbb{G}_2^{23}$. The structure-preservation and uniform public-key properties carry over from the underlying PoS1u and PoS2u.

**Theorem 10.** Scheme PoSb is strongly unforgeable against OT-CMA if SXDH holds. In particular, for all p.p.t. algorithms $A$ there exists a p.p.t. algorithm $B$ such that $Adv_{\text{PoS}b, A}(\lambda) \leq Adv_{\text{PoS}d, G, B}(\lambda) + 2/p(\lambda)$, where $p(\lambda)$ is the size of the groups produced by $G$. Moreover, the run-time overhead of the reduction $B$ is a small number of multi-exponentiations per signing or key query.

**Proof.** Suppose an adversary $A$ outputs a forgery $(opk^1, \sigma^1, msg^1)$. Then there exists a triple $(\sigma, opk, msg)$ observed by the signing oracle such that $opk^1 = opk$ and $(msg^1, \sigma^1) \neq (msg, \sigma)$. Let $msg^1 = (msg_1^1, msg_2^1)$ and $\sigma^1 = (\sigma_1^{1,1}, \sigma_2^{1,1}, opk_2^1)$. Similarly, let $msg = (msg_1, msg_2)$ and $\sigma = (\sigma_1, \sigma_2, opk_2)$. Then there are two cases: either $(msg_1, opk_2, \sigma_1) \neq ((msg_1^1, opk_2^1), \sigma_1^1)$, or $opk_2 = opk_2^1$ and $(msg_2, \sigma_2) \neq (msg_2^1, \sigma_2^1)$. In the first case we break the strong unforgeability of PoS1u and contradict the DBP2 assumption; in the second case we break the strong unforgeability of PoS2u and contradict the DBP1 assumption.

Accordingly, we have $Adv_{\text{PoS}b, A}(\lambda) \leq Adv_{\text{PoS}d, G, B}(\lambda) + 1/p + Adv_{\text{PoS}d, G, B}(\lambda) + 1/p \leq Adv_{\text{PoS}d, G, B}(\lambda) + 2/p$. 

23
6.4 XRMA-secure signature scheme

Our construction is based on a variant of Waters’ dual system encryption proposed by Ramanna, Chatterjee, and Sarkar [44]. An intuition behind our XRMA-secure scheme is the same as that of RMA-secure scheme in the previous section. Recall that $gk = (\Lambda, G, \hat{G}, F_1, \hat{F}_1, F_2, \hat{F}_2, U, \hat{U})$ with $\Lambda = (p, G_1, G_2, G_7, e)$ is generated by Setup$(1^\lambda)$ in advance (see Section 6.1).

[Scheme xSIG]

xSIG.Gen$(gk)$: Given $gk$ as input, uniformly select generators $V, V' \leftarrow G_1^*$, $\hat{V}, \hat{V}' \leftarrow G_2^*$ such that $V \sim V$, $V' \sim V'$, $H \leftarrow G_2^*$, and exponents $a, b, a, \beta \leftarrow Z_p^*$. Then compute and output $vk := (gk, \hat{B}, \tilde{A}, \tilde{B}_0, \tilde{R}, \tilde{W}, H, X_1, \tilde{X}_2)$ and $sk := (vk, K_1, K_2, V, V')$ where

$$\tilde{B} := G^b, \quad \tilde{A} := G^a, \quad \tilde{B}_0 := G^{ab}, \quad \tilde{R} := \hat{V}(V')^a, \quad \tilde{W} := \hat{R}^b$$

$$X_1 := G^\rho, \quad \tilde{X}_2 := \hat{G}^{ab/r}, \quad K_1 := G^\alpha, \quad K_2 := G^\beta.$$

xSIG.Sign$(sk, msg)$: Parse $msg$ into $(\tilde{M}_1, \tilde{M}_2, \tilde{M}_3) = (\tilde{F}_1^m, \tilde{F}_2^m, \tilde{U}^m) \in G_2^3$. Pick random $r_1, r_2, z \leftarrow Z_p$. Let $r := r_1 + r_2$. Compute and output signature $\sigma := (\tilde{S}_0, S_1, \ldots, S_5)$ where

$$\tilde{S}_0 := (\tilde{M}_3\tilde{H})^{r_1}, \quad S_1 := K_1V^r, \quad S_2 := (V')^bG^{-z}, \quad S_3 := K_2^3, \quad S_4 := K_2^2, \quad S_5 := G^r.$$  

xSIG.Vrfy$(vk, msg, \sigma)$: Parse $msg$ into $(\tilde{M}_1, \tilde{M}_2, \tilde{M}_3)$ and $\sigma$ into $(\tilde{S}_0, S_1, \ldots, S_5)$. Also parse $vk$ accordingly. Verify the following pairing product equations:

$$e(S_1, \tilde{B})e(S_2, \tilde{B}_0)e(S_3, \tilde{A}) = e(S_4, \tilde{R})e(S_5, \tilde{W})e(\bar{X}_1, \bar{X}_2), \quad (23)$$

$$e(S_0, \tilde{M}_3\tilde{H}) = e(G, S_0), \quad (24)$$

$$e(F_1, \tilde{M}_3) = e(U, \tilde{M}_1), \quad (25)$$

$$e(F_2, \tilde{M}_3) = e(U, \tilde{M}_2). \quad (26)$$

The scheme is structure-preserving by construction. We can easily verify the correctness as follows.

(Left-hand of (23)) = $e(G^\alpha V^r, \hat{G}^b)e((V')^bG^{-z}, \hat{G}^{ab})e(G^{b\rho}, \tilde{G}^a)$

= $e(G, \hat{G})^{ab}e(V, \hat{G})^br e(V', \hat{G})^{br}$

= $e(G, \tilde{V})^{b(r_1+r_2)}e(G, \tilde{V})^{ab(r_1+r_2)}e(G, \hat{G})^{ab}$

= $e(G^{br_1}, \tilde{V}(V')^a)e(G^{r_1}, \tilde{V}^b(V')^a)e(G, \hat{G})^{ab}$

= (Right-hand of (23))

Equation (23) holds since $r = r_1 + r_2$, $V \sim \tilde{V}$, and $V' \sim \tilde{V}'$. The followings also hold.

(Left-hand of (24)) = $e(G^{r_1}, \tilde{U}^m\tilde{H}) = e(G, \tilde{U}^m\tilde{H})^{r_1} = e(G, (\tilde{U}^m\tilde{H})^{r_1})$ = (Right-hand of (24)),

(Left-hand of (25)) = $e(F_1, \tilde{U}^m) = e(F_1, \tilde{U})^{m} = e(U, \tilde{F}_1^m)$ = (Right-hand of (25)),

(Left-hand of (26)) = $e(F_2, \tilde{U}^m) = e(F_2, \tilde{U})^{m} = e(U, \tilde{F}_2^m)$ = (Right-hand of (26)).

**Theorem 11.** The above xSIG scheme is UF-XRMA with respect to the message generator that returns $\omega = m$ for every random message $msg = (\tilde{F}_1^m, \tilde{F}_2^m, \tilde{U}^m)$ under the DDH2 and XDLIN1 assumptions. In particular for any p.p.t. algorithm $A$ for xSIG making at most $q(\lambda)$ signing queries, there exist p.p.t. algorithms $B_1, B$ such that $\text{Adv}_{\text{xSIG}, A}^\text{uf-xrma}(\lambda) \leq \text{Adv}_G^{\text{ddh2}}(\lambda) + (q(\lambda) + 1)\text{Adv}_G^{\text{xdlin1}}(\lambda)$.  

24
Proof. In this scheme, simulation-type signatures are of the form \( \sigma = (\tilde{S}_0, S'_1 = S_1 \cdot G^{-a \gamma}, S'_2 = S_2 \cdot G^{\gamma}, S_3, S_4, S_5) \) for \( \gamma \in \mathbb{Z}_p \). The outline of the proof follows that of Water’s dual signature scheme and is quite similar to the proof of Theorem 7. We start with the following lemma.

**Lemma 7.** Any signature that is accepted by the verification algorithm must be either a normal-type signature or a simulation-type signature.

**Proof.** (of Lemma 7) We ignore the last row of verification equations that establish that msg is well-formed. A signature has 3 random exponents, \( r_1, r_2, z \). A simulation-type signature has an additional exponent \( \gamma \). We interpret \( \gamma \) as \( r_1 \), so the first verification equation implies that \( \tilde{S}_0 = (U^m H)^{r_1} \). For fixed \( b \in \mathbb{Z}_p \) (\( G^b \) is included in \( vk \), there exists \( r_2, z \in \mathbb{Z}_p \) such that \( S_3 = G^{bz}, S_4 = G^{br_2} \). If we fix \( S_1 = G^a V^r G^{-a \gamma} \), then a remaining unknown value is \( S_2 \). The verification equation is

\[
e(S_1, \tilde{G}^b) e(S_2, \tilde{G}^{ba}) e(S_3, \tilde{G}^a) = e(S_4, \tilde{R}) e(S_5, \tilde{R}^b) e(G, \tilde{G})^{rb}
\]

so we can fix \( S_2 = (V^r) G^{-a \gamma} G^z \).

Based on the notion of simulation-type signatures, we consider a sequence of games. Let \( p_i \) be the probability that the adversary succeeds in Game \( i \), and \( p_i^{\text{norm}}(\lambda) \) and \( p_i^{\text{sim}}(\lambda) \) be the probability that he succeeds with a normal-type or simulation-type forgery respectively. Then by Lemma 7, \( p_i(\lambda) = p_i^{\text{norm}}(\lambda) + p_i^{\text{sim}}(\lambda) \) for all \( i \).

**Game 0:** The actual Unforgeability under Extended Random Message Attacks game.

**Lemma 8.** There exists an adversary \( B_1 \) such that \( p_0^{\text{sim}}(\lambda) \leq \text{Adv}^{\text{ddh2}}_{G,B_1}(\lambda) \).

**Game i:** The real security game except that the first \( i \) signatures that are given by the oracle are simulation-type signatures.

**Lemma 9.** There exists an adversary \( B_2 \) such that \( |p_i^{\text{norm}}(\lambda) - p_i^{\text{sim}}(\lambda)| \leq \text{Adv}^{\text{ddlin1}}_{G,B_2}(\lambda) \).

**Game q:** All signatures given by the oracle are simulation-type signatures.

**Lemma 10.** There exists an adversary \( B_3 \) such that \( p_q^{\text{norm}}(\lambda) \leq \text{Adv}^{\text{co-cdh}}_{G,B_3}(\lambda) \).

We have shown that in **Game q**, \( A \) can output a normal-type forgery with at most negligible probability. Thus, by Lemma 7, we can conclude that the same is true in **Game 0**. Since we have already shown that in **Game 0** the adversary can output simulation-type forgeries only with negligible probability, and that any signature that is accepted by the verification algorithm is either normal or simulation-type, we conclude that the adversary can produce valid forgeries with only negligible probability

\[
\text{Adv}^{\text{unorm}}_{\text{Sig},A}(\lambda) = p_0(\lambda) = p_0^{\text{sim}}(\lambda) + p_0^{\text{norm}}(\lambda)
\]

\[
\leq p_0^{\text{sim}}(\lambda) + \sum_{i=1}^{q} |p_i^{\text{norm}}(\lambda) - p_i^{\text{norm}}(\lambda)| + p_q^{\text{norm}}(\lambda)
\]

\[
\leq \text{Adv}^{\text{ddh2}}_{G,B_1}(\lambda) + q \text{Adv}^{\text{ddlin1}}_{G,B_2}(\lambda) + \text{Adv}^{\text{co-cdh}}_{G,B_3}(\lambda)
\]

\[
\leq \text{Adv}^{\text{ddh2}}_{G,B_1}(\lambda) + (q + 1) \text{Adv}^{\text{ddlin1}}_{G,B_2}(\lambda)
\]

as stated. The last inequality holds since the CDH1 assumption is implied by the XD LIN1 assumption.

**Proof.** (of Lemma 8) We show that, if the adversary outputs a simulation-type forgery, then we can construct algorithm \( B_1 \) that solves the DDH2 problem. Algorithm \( B_1 \) is given instance \( (\Lambda, \hat{G}, \hat{G}^a, \hat{Z} \in G_2) \) of DDH2, and simulates the verification key and the signing oracle for the signature scheme. \( B_1 \) does not have the values \( a, s \).
B₁ generates gk and vk as follows. It selects G ← G₁, and exponents u, f₁, f₂ ← Zₚ*, computes $F₁ := G^{f₁}$, $F₂ := G^{f₂}$, $U := G^{u}$, $\hat{U} := G^{u}$, and sets them into gk. It also selects exponents v, v' ← Zₚ*, computes $V := G^{v}$, $V' := G^{v'}$. Next, it selects exponents b, α, h, ρ ← Zₚ*, computes $H := G^{b}$, and

$$\hat{B} := G^{b}, \quad \hat{A} := G^{a}, \quad \hat{B}_a := (G^{α})^{b}, \quad \hat{R} := \hat{V}(V')^a = G^{a}(\hat{G}^{a})^v, \quad \hat{W} := \hat{R}^b = G^{bc}(\hat{G}^{a})^{bc}$$

$$X₁ := G^{α}, \quad \hat{X}_2 := G^{αβ}, \quad K₁ := \hat{G}^{α}, \quad K₂ := G^{b}$$

and sets them into vk and sk, accordingly.

B₁ can generate normal-type signatures by using the (normal) signing algorithm since B₁ has α, b and V, V'. For i-th signature, B₁ randomly selects $m_i \in Z_p$, generates normal-type signature $σ_i$ for message ($F₁^{m_i}$, $F₂^{m_i}$, $U^{m_i}$), and gives ((F₁, F₂, U), $σ_i$, m) to A.

If adversary A outputs a simulation-type forgery $S₁ := (G^{αV'}r), S₂ := ((V')^rG^{-z})G^γ$, $S₃ := (G^{b})^{-s}, S₄ := (G^{b})^s$, $S₅ := G^{r_1}$, and $S₀ := (M₁H)^r$, for some $r₁, r₂, z, γ \in Z_p (r = r₁ + r₂)$ for message $msg = (F₁^{m₁}, F₂^{m₁}, U^{m₁})$, then B₁ can compute ($G^{αV'γ}, G^{γ}$) from S₁, S₂ respectively. The reason is as follows:

B₁ has b, so it can compute $G^{α}, G^{αγ}, G^{bγ}$ from $S₃ = G^{bγ}, S₅ = G^{r_1}, S₄ = G^{br_2}$, respectively and obtain $G' := G^{α+γb}, V' := G^{αγ}, (V')^r := G^{rγ}$ (B₁ has v, v'). Thus, B₁ can extract ($G^{αV'γ}, G^{γ}$) from S₁ and S₂ since it has α. B₁ can solve the DDH₂ problem by checking whether

$$e(Gγ, \hat{Z}) = e(G^{αγ}, \hat{G}^γ)$$

or not because $e(G^{αγ}, \hat{G}^γ) = e(G, \hat{G})e^{γ} = e(G, \hat{G})$. If $\hat{Z} := \hat{G}^{α}$ (DDH tuple), then the equation holds. Thus, B₁ solves the DDH₂ problem whenever the adversary outputs a valid simulation-type forgery, i.e., $p^{\text{sim}}(λ) ≤ \text{Adv}_{\text{DDH₂}}(λ)$ as claimed.

Proof. (of Lemma 9) Given access to A playing $p^{\text{sim}}(λ)$ and $p^{\text{norm}}(λ)$, we construct algorithm B₂ that solves the XDLIN₁ problem with advantage $|p^{\text{norm}}(λ) - p^{\text{sim}}(λ)|$.

B₂ is given instance $(A, G₁, G₂, G₃, G₄, X, Y, X, Y, Z ∈ G₁)$ of the XDLIN₁ problem. It implicitly holds that $G₁ = G₃^{β}, G₁ = G₂^{β}, X = G₁^Y, Y = G₃^X, X = G₂^Y, Y = G₃^X$ for some $b, x, y \in Z_p$. B₂ generates the group elements in gk and vk as follows: It selects exponents $ξ, β, χ₁, χ₂, r ← Z_p$, such that $ξm + β = 0$ where $m \in Z_p$ is the exponent of the i-th random message. (If $ξm + β = 0$, then it holds that $(U^{mH}) = G^{α_1+γx^2G^{α_1+γx^2}}$. Note that $ξ$ and $β$ are information theoretically hidden even given m, so the adversary has negligible chance of producing another message $U^{mH}$ such that $ξm + β = 0$. ) It then computes $G := G₂, G := G₂, F₁ := G₁^Y, F₁ := G₁^Y, F₂ := G₃, F₂ := G₃, U := G₃^Y, \hat{U} := G₃^Y, \hat{U}$, sets into gk, and then compute $H := G₃^{β}, G^{β}$, it also chooses $a, δ, ν ← Z_p$ and computes $V := G₃^{αδ}, V' := G₃^{αδ}$, $V := G₃^{αδ}, V' := G₃^{αδ}$, $\hat{X}_2 := (G^{α})^{b}, \quad K₁ := G^{α}, \quad K₂ := G^{b}$, computes

$$\hat{B} := G₁, \quad \hat{A} := G₃^{α}, \quad \hat{B}_a := (G^{α})^{b}, \quad \hat{R} := \hat{V}(V')^a = G^{α}(\hat{G}^{α})^v, \quad \hat{W} := \hat{R}^b = G^{bc}(\hat{G}^{α})^{bc}$$

and them sets them into vk and sk, accordingly.

Since B₂ has a, it can compute $G^{α}$ and further generate simulation-type signatures. Now B₂ simulates signatures for j-th random message as follows.

Case $j > i$: B₂ randomly selects $m_j \in Z_p$, generates normal-type signature $σ_j$ for message ($F₁^{m_j}$, $F₂^{m_j}$, $U^{m_j}$) by using $sk = (vk, G₂, G₃^{α}, V, V')$, and gives ((F₁, F₂, U), $σ_j$, mₐ) to A.

Case $j = i$: B₂ embeds the instance as follows. For the i-th randomly chosen message $msg = (F₁^{m₁}, F₂^{m₂}, U^{m_i}) \in G₂$, B₂ implicitly sets $r₁ := y, r₂ := x$ and computes $S₁ := G^{br₁} = G₁^Y, S₅ := G^{r₁} = G₃^{β}$.

B₂ can compute $S₀ := (G₃^{β})^{α_1+γx^2} = (U^{mH})^r$. Next, in order to compute $V^r$ and $(V')^r$, B₂ computes $(G₃^{γ})^{r} \cdot α_{β}$ as $Z^{−αβ}$. If $Z = G₃^{γ}$, then this will be correct. If $Z = G₃$ for $ξ ← Z_p$,
then we let $G^\gamma := G_3^{\delta((x+y))}$ and this will be a simulation-type signature. $B_2$ chooses $s \leftarrow \mathbb{Z}_p$ and implicitly sets $G^{-z} := G_2^{1-\ell r_2 + s}$. These values are not computable but $B_2$ can compute $G^{2b} = G_1^{2x-s}$. $S_2 := (G_1^y)^{a}Z^dG_2^{2x-y}Z^dG_2^{2x-r_2^y} = G_2^{y^2}Z^dG^{-z}$. $B_2$ generates a signature $\sigma := (S_0, \ldots, S_5)$ as follows:

$$S_0 := (G_2^y)^{m+1}z \quad S_1 := G_2^{x-a\delta} \quad S_2 := (G_2^y)^{a}Z^dG_2^{2x}$$

$$S_3 := (G_1^y)^{a}G_1^{-s} \quad S_4 := G_1^{x} \quad S_5 := G_2^{y^2}.$$

$B_2$ can generate $S_0$ correctly since $B_2$ set $\xi m + \beta = 0$. $B_2$ gives $((F_1^{m_1}, F_2^{m_2}, U^{m_3}), (\sigma, m_3))$ to $A$.

- If $Z = G_3^{1+y}$ \((\text{linear})\), then $A$ is in $p_{\text{norm}}^{1}(\lambda)$, otherwise $A$ is in $p_{\text{norm}}^{2}(\lambda)$. For all messages, $B_2$ can return $\mu(M_i) = m_i$.

  At some point, $A$ outputs forgery $(S_0, S_1, \ldots, S_5)$ and message $msg^* = (\tilde{Q}_1, \tilde{Q}_2, \tilde{Q}_3) = (\tilde{F}_1^\ast, \tilde{F}_2^\ast, \tilde{U}^\ast)$. $B_2$ outputs 1 if and only if

$$e(G_1, S_0) \cdot e(S_4, \tilde{G}_2^{\frac{1}{\phi}x_3}) = e((S_1G_2^{-a\delta^1})^{\frac{1}{\phi}(-a\delta)}, (\tilde{G}_1^{\frac{1}{\phi}})\tilde{G}_1^{\delta}) \cdot e(S_5, (\tilde{C}_1^{\frac{1}{\phi}}x_1\tilde{G}_1^{x_3})).$$

By Lemma 7 there exist $m^*, r_1^*, r_2^*, \gamma^*, r^* = r_1^* + r_2^*$ such that $S_0 = (\tilde{U}^\ast \tilde{H})^{\gamma^*}$, $S_1 = G_2^{y^2}V^rG_2^{-a\gamma^*}$, $S_4 = G_1^{x^2}$, $S_5 = G_2^{y^2}$, $\tilde{Q}_1 = (\tilde{G}_2^\ast)^{m^*} \cdot \tilde{Q}_2 = \tilde{C}_3^{m^*}$. Rephrased in terms of our parameters, this means

$$S_0 = (G_2^{m^*+1+x_2}G_3^{m^*+\beta})^{\gamma^*} \quad S_1 = G_2^{x-a\delta^*}G_2^{-a\gamma^*}$$

$$S_4 = G_1^{x^2} \quad S_5 = G_2^{y^2}.$$

Plugging this into the above computation, we have the left hand side is

$$e(G_1, S_0) \cdot e(S_4, \tilde{G}_2^{\frac{1}{\phi}x_3}) = e(G_1, (G_2^{m^*+1+x_2}G_3^{m^*+\beta})^{\gamma^*}) \cdot e(G_1^{\delta^*}, (G_3^{m^*+\beta})^{\delta^*})$$

$$= e(G_1, \tilde{G}_2^{(m^*+x_2)r^*}) \cdot e(G_1, \tilde{G}_3^{(m^*+\beta)r^*}) \cdot e(G_1, \tilde{G}_1^{(m^*+\beta)r^*}).$$

and the right hand side is

$$e((S_1G_2^{-a\delta^1})^{\frac{1}{\phi}(-a\delta)}, (\tilde{G}_1^{\frac{1}{\phi}})\tilde{G}_1^{\delta}) \cdot e(S_5, (\tilde{C}_1^{\frac{1}{\phi}}x_1\tilde{G}_1^{x_3}))$$

$$= e(G_2^{x-a\delta^*}G_2^{-a\gamma^*}, (G_3^{m^*+\beta})^{\delta^*}) \cdot e(G_1^{\delta^*}, (G_3^{m^*+x_2})^{\delta^*})$$

$$= e(G_3, \tilde{G}_1^{(m^*+\beta)r^*}) \cdot e(G_2, \tilde{G}_1^{(m^*+x_2)r^*}).$$

A simplified equation is $1 = e(G_2, \tilde{G}_1)^{\gamma^*/\delta((m^*+\beta)}$.

Thus, the difference of $A$’s advantage in two games gives the advantage of $B_2$ in solving the XDLIN_1 problem as stated.

**Proof.** (of Lemma 10) Observe that, in $p_{\text{norm}}^{1}(\lambda)$, $A$ is given simulation-type signatures only. We show that if $A$ outputs a normal-type forgery in $p_{\text{norm}}^{2}(\lambda)$ then we can construct algorithm $B_3$ that solves the co-CDH problem.
\(B_3\) is given instance \((\Lambda, G, \hat{G}, \hat{G}^e, G^x, \hat{G}^x, \hat{G}^y)\) of the co-CDH problem. \(B_3\) generates the verification key as follows: \(B_3\) selects exponents \(u, h, f_1, f_2 \leftarrow \mathbb{Z}_p^*\), computes \(F_1 := \hat{G}^{f_1}, F_2 := \hat{G}^{f_2}, \hat{F}_2 := G^{f_2}, \hat{U} := \hat{G}^{u}, \hat{U} := \hat{G}^{u}\), and sets them into \(gk\). \(B_3\) also selects exponents \(v, \nu \leftarrow \mathbb{Z}_p^*\), computes \(V := G^v, V' := \hat{G}^v, \hat{V} := \hat{G}^{v}, \hat{V}' := \hat{G}^{v}\). Next, it also selects exponents \(h, b, \rho \leftarrow \mathbb{Z}_p^*\), computes \(\hat{H} := \hat{G}^h\) and

\[
\hat{B} := \hat{G}^b, \quad \hat{A} := \hat{G}^u, \quad \hat{B}_u := (\hat{G}^v)^b, \quad \hat{R} := \hat{V}(\hat{V}')^a = \hat{V}(G^v)^b, \quad \hat{W} := \hat{R} = (\hat{V}(G^y)^b)^b
\]

and sets them into \(vk\) and \(sk\), accordingly. Note that it means implicitly \(\rho = \rho' x\) and \(\alpha = xy\) though \(B_3\) does not have \(\alpha, \rho\). Therefore \(B_3\) does not have \(K_1 = G^a = G^{xy}\), and cannot compute normal-type signatures. For \(i\)-th message, \(B_3\) randomly select \(m_i \in \mathbb{Z}_p\) and outputs simulation-type signatures for each random message \(msg_i = (F_1^{m_i}, F_2^{m_i}, \hat{U}^{m_i})\) as follows:

\(B_3\) selects \(r_1, r_2, z, \gamma' \leftarrow \mathbb{Z}_p\), sets \(r := r_1 + r_2\) (we want to set \(\gamma := x + \gamma'\)), and computes:

\[
S_1 := (G^a)^{-r} \cdot V' = (G^a)^{r'} \cdot G^{-\gamma'} (a = y, xy = \alpha) \\
S_2 := G' \cdot G^z \cdot G^{-z} = (V')^z \cdot G^{-z} \cdot G^y \\
S_3 := (G^b)^z \\
S_4 := G^{\nu z} \\
S_5 := G^\nu r_1 \\
\hat{S}_0 := (\hat{U}^{m_i} \hat{H})^{-1}.
\]

\(B_3\) gives \((F_1^{m_i}, F_2^{m_i}, \hat{U}^{m_i}), \sigma_i, m_i\) where \(\sigma_i := (\hat{S}_0, S_1, \ldots, S_5)\) to \(A\). At some point, \(A\) outputs a normal-type forgery, \(S_1' = G^a V'^r, S_2' = (V')^r G^{-z}, S_3' = (G^b)^z, S_4' = G^{\nu z}, S_5' = G^\nu r_1, \hat{S}_0' = (\hat{U}^{m_i} \hat{H})^{-1}\), for some \(r_1', r_2', z', \alpha' \in \mathbb{Z}_p\) for message \(msg' = (F_1^{m_i}, F_2^{m_i}, \hat{U}^{m_i})\). By using these values, \(B_3\) can compute \(G^* = (S_1')^{1/b} \cdot G^r = S_2', \hat{G}^{z'} = (S_3')^{1/b} \cdot \hat{V}'^y = (G^y \cdot \hat{G}^x)^y\) since \(V = G^v\). Thus, \(B_3\) can compute \(S_1' / V'^r = G^a = G^{xy}\). That is, \(B_3\) can solve the co-CDH problem and it holds that \(\text{Adv}^{\text{sim}}_{\text{SIG}, B_3}(\lambda) \leq \text{Adv}^{\text{pos}}_{\text{cdh}}(\lambda)\) as claimed.

**Remark 3.** It is difficult to modify \(x\text{SIG}\) so as to rely on the DDH\(_1\) and DDH\(_2\) assumption, that is, only on the SXDH assumption because we are not given instances in group \(\mathbb{G}_2\) and cannot simulate verification keys in group \(\mathbb{G}_2\) under the DDH\(_1\) assumption when we prove a similar statement to Lemma 2 by using DDH\(_1\). Constructing XRAM-secure SPS scheme only from the SXDH assumption is an important open problem since it will save on the number of group elements in a signature and a verification key. Moreover, it is non-trivial to modify \(x\text{SIG}\) so as to rely on the DDH\(_1\) and XDLIN\(_1\) because if we use assumptions only over \(\mathbb{G}_1\), then all elements in a signature must be in \(\mathbb{G}_1\). It means that a message must consist of elements in both \(\mathbb{G}_1\) and \(\mathbb{G}_2\), which we would like to avoid.

### 6.5 Security and efficiency of resulting SIG2

Let SIG2 be the scheme obtained from POS\(_b\) and \(x\text{SIG}\). SIG2 is structure-preserving as \(vk, \sigma, \text{ and } msg\) consist of group elements from \(\mathbb{G}_1\) and \(\mathbb{G}_2\), and SIG2.Vrf evaluates pairing product equations. From Theorem 9 and 10 and 11 we obtain the following theorem.

**Theorem 12.** SIG2 is a structure-preserving signature scheme that is unforgeable against adaptive chosen message attacks if SXDH and XDLIN\(_1\) hold for \(\hat{G}\). In particular, for any p.p.t. algorithm \(A\) for SIG2 making at most \(q_s(\lambda)\) signing queries, there exist p.p.t. algorithms \(B, C\) such that \(\text{Adv}^{\text{sig}, \text{ema}}_{\text{SIG2}, A}(\lambda) \leq (q_s(\lambda) + 1) \cdot \text{Adv}^{\text{pos}}_{\mathbb{G}_R, \mathbb{E}_C}(\lambda) + 2 \cdot \text{Adv}^{\text{cdh}}_{\mathbb{G}_C, \mathbb{E}_C}(\lambda) + 2 / p(\lambda)\), where \(p(\lambda)\) is the size of the groups produced by \(\hat{G}\).

Table 3 summarizes the efficiency of SIG2 for both unilateral messages consisting of \(k\) elements and bilateral messages consisting of \(k_1\) and \(k_2\) elements in \(\mathbb{G}_1\) and \(\mathbb{G}_2\), respectively. We count the number of group elements in public components of SIG2. Note that the default generators in \(gk\) is not included in the count. For comparison, we also evaluate the efficiency of the schemes in [4] Section 5.2 and [5] Section 5.2. For bilateral messages, the scheme from [4] is combined with POS\(_b\) from Section 6.3. Since the scheme in
can sign a single group element, extended part of one-time verification key from POSb messages. Entry \( G \) as those in (14) to (17) except that witnesses and linear equations in \( G \) relations in total. (They are non-linear since one-time public-key for \( x \), \( y \), \( z \) are input to constant pairings. And \( \sigma \) need to include only one generator for each \( G \).)

Table 3: Efficiency of SIG2 and compared to other schemes with constant-size signatures. The upper half is for unilateral messages and the lower half is for bilateral messages. Notation \((x, y)\) represents \( x \) elements in \( G_1 \) and \( y \) in \( G_2 \).

### Table 3: Efficiency of SIG2 and compared to other schemes with constant-size signatures.

| Scheme | \(|msg|\) | \(|gk| + |vk|\) | \(|\sigma|\) | \(#(PPE)\) | Assumptions |
|--------|---------|---------------|-----------|-----------|-------------|
| SIG2 : POSu1 + xSIG | \((k_1, 0)\) | \((5, 2k_1 + 9)\) | \((5, 2)\) | 2 | q-SFP |
| SIG2 : POSb + xSIG | \((k_1, 0)\) | \((5, k_1 + 12)\) | \((7, 4)\) | 5 | SXDH, XDLIN_1 |
| POSb + [4] | \((k_1, k_2)\) | \((k_2 + 12, k_1 + 7)\) | \((8, 5)\) | 4 | q-SFP |
| SIG2 : POSb + xSIG | \((k_1, k_2)\) | \((k_2 + 3, k_1 + 4)\) | \((3, 3)\) | 2 | q-type |
| Table 4: Costs of WI proofs with the GS proof system of valid signature of \( \sigma \) for unilateral messages and the lower half is for bilateral messages. Notation \((x, y)\) represents \( x \) elements in \( G_1 \) and \( y \) in \( G_2 \). |

In Table 3 and Table 5, we assess the size of proofs for showing ones possession of a valid signature and message of SIG2 by using the GS-proof system as NIWI or NIZK. The general formulas are the same as those in (14) to (17) except that witnesses and linear equations in \( G_1 \) and \( G_2 \) are considered separately. (We say that an equation is linear in \( G_1 \) if all variables in the equation are in \( G_1 \).) By \((x, y)\), we denote \( x \) and \( y \) elements in \( G_1 \) and \( G_2 \), respectively. Similarly, by \((x, y, z)\), we denote additional element \( z \) in \( \mathbb{Z}_p \).

In this asymmetric setting, we have \(|com| = (2, 0, 0)\) for committing to \( G_1 \) elements, and \(|com| = (0, 2, 0)\) for \( G_2 \). Proof size for linear equation in \( G_1 \) and \( G_2 \) is \(|\pi_L| = (0, 2, 0)\) and \((2, 0, 0)\), respectively. We also have \(|\pi_{NL}| = (4, 4, 0)\) and \(|\pi_{MS}| = (0, 0, 2)\).

We first consider the cases of NIWI shown in Table 3. For unilateral messages, we have \(|\sigma_{nl}| = (7, 4)\) group elements and \(|\sigma_{nl}| = (0, 0)\). Verifying POSu1 consists of one non-linear relation (19), and verifying xSIG consists of one linear equation in \( G_1 \) (23), two linear equations in \( G_2 \) (25) and one non-linear equation (24). Thus, \(|NIWI(\sigma)| = ((2, 0, 0) \times 7 + (0, 2, 0) \times 4) + 0 + (4, 4, 0) \times 2 + ((0, 2, 0) \times 1 + (2, 0, 0) \times 2) = (26, 18, 0)\). For bilateral messages, we have \(|\sigma_{nl}| = (8, 6)\) group elements and \(|\sigma_{nl}| = (0, 0)\). Verifying POSb consists of verification for POSu1 and POSu2, which are two non-linear relations in total. (They are non-linear since one-time public-key \( A \) is in \( G_1 \) whereas signature \( \tilde{Z}, \tilde{R} \) are in \( G_2 \).) Equations for xSIG are the same as above. Thus \(|NIWI(\sigma, msg)| = ((2, 0, 0) \times 8 + (0, 2, 0) \times 6) + 0 + (4, 4, 0) \times 3 + ((0, 2, 0) \times 1 + (2, 0, 0) \times 2) = (32, 26, 0)\). For NIWI(\( \sigma, msg \)), we add \((2k_1, 0)\) and \((2k_1, 2k_2)\) elements for the commitment of the message in unilateral and bilateral case, respectively. Hence \(|NIWI(\sigma, msg)| = (2k_1 + 26, 18, 0)\) for unilateral case, and \(|NIWI(\sigma, msg)| = (2k_1 + 32, 2k_2 + 26, 0)\) for bilateral case.

We next consider the cases of NIZK. Additional elements come from public constants to commit to, and the proof of their correct commitment. For NIZK(\( \sigma \)), every element in a message are regarded as public constants that are input to constant pairings. And xSIG involves one constant pairing \( (X_1, \tilde{X}_2) \) where we commit to \( X_1 \) so that (23) remains a linear equation. We thus have \( k_1 + 1 \) constants to commit to in \( G_1 \) for the unilateral case, and \( k_1 + 1 \) and \( k_2 \) constants to commit to in \( G_1 \) and \( G_2 \) respectively in the bilateral case. By wrapping up, we have \(|NIZK(\sigma)| = |NIWI(\sigma)| + ((2, 0, 0) \times (k_1 + 1) + (0, 0, 2) \times (k_1 + 1) = (2k_1 + 28, 18, 2k_1 + 2)\) for the unilateral case, and \(|NIZK(\sigma)| = |NIWI(\sigma)| + ((2, 0, 0) \times (k_1 + 1) + (0, 0, 2) \times k_2 + (0, 0, 2) \times (k_1 + k_2 + 1) = (32, 26, 0) + (2k_1 + 2, 0, 0) \times (0, 2k_2, 0) + (0, 0, 2k_1 + 2k_2 + 2) = (2k_1 + 34, 2k_2, 26, 2k_1 + 2k_2 + 2)\) for the bilateral case. For NIZK(\( \sigma, msg \)) where messages are already committed, additional elements are from committing to \( X_1 \) compared to the case of NIWI(\( \sigma, msg \)). We thus have \(|NIZK(\sigma, msg)| = |NIWI(\sigma, msg)| + ((2, 0, 0) \times 1 + (0, 0, 2) \times 1 = (2k_1 + 28, 18, 2)\) for unilateral case, and \(|NIZK(\sigma, msg)| = |NIWI(\sigma, msg)| + ((2, 0, 0) \times 1 + (0, 0, 2) \times 1 = (2k_1 + 34, 2k_2 + 26, 2)\) for bilateral case.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>NIWI(( \sigma ))</th>
<th>NIWI(( \sigma, msg ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG2</td>
<td>(26, 18, 0)</td>
<td>(2k_1 + 26, 18, 0)</td>
</tr>
<tr>
<td>Unilateral</td>
<td>(32, 26, 0)</td>
<td>(2k_1 + 32, 2k_2 + 26, 0)</td>
</tr>
</tbody>
</table>

Table 4: Costs of WI proofs with the GS proof system of valid signature of SIG2 for unilateral and bilateral messages. Entry \((x, y, z)\) denotes \( x \), \( y \), and \( z \) elements in \( G_1 \), \( G_2 \), and \( \mathbb{Z}_p \), respectively.
Table 5: Costs for proving valid signature of SIG2 for unilateral and bilateral messages in ZK with the GS proof system.

7 Applications

We list a few recent examples of applications of SPS that benefit from our results.

- **Group Signatures with Efficient Revocation and Compact Verifiable Shuffles.** Using our SIG1 scheme from Section 5 both the construction of a group signature scheme with efficient revocation by Libert, Peters and Yung [41] and the construction of compact verifiable shuffles by Chase et al. [21] can be proven purely under the DLIN assumption. All other building blocks already have efficient instantiations based on DLIN.

- **Tightly-secure Structure-preserving Signatures.** Hofheinz and Jager [38] construct a tightly-secure one-time signature scheme and use it to construct a tightly-secure tree-based SPS scheme, say tSIG. Instead, we propose to use our partial one-time scheme to construct tSIG. As the resulting tSIG is secure against non-adaptive chosen message attacks, it is secure against extended random message attacks as well. We then combine the POSb scheme and the new tSIG scheme according to our second generic construction. The resulting signature scheme is significantly more efficient than [38] and is a SPS scheme with a tight security reduction to SXDH. As shown in [3], the same is possible in Type-I groups by using the tagged one-time signature scheme in Section 5.2 whose security tightly reduced to DLIN.

- **Simulation-sound and Simulation-extractable NIZK.** In [3], we also show how to construct more efficient simulation-sound and simulation-extractable non-interactive zero-knowledge (SS-NIZK & SE-NIZK) proof systems. While in [3] we were primarily interested in tightly-secure NIZK and thus used the tree-based tSIG scheme, RMA-security suffices for constructing unbounded SS-NIZK and SE-NIZK schemes. Our rSIG and xSIG schemes can thus be used directly to construct even more efficient unbounded SE-NIZK if one lifts the requirement of a tight reduction.

- **Tightly-secure Structure-preserving CCA-secure Public-key Encryption.** Following the approach of [38] and [3], tightly-secure SE-NIZK enables tightly-secure and structure-preserving CCA-secure public-key encryption under standard decisional assumptions.

- **Efficient Adaptive Oblivious Transfer.** Hohenberger and Green proposed a universally composable (UC) adaptive oblivious transfer (AOT) protocol by using an SPS scheme based on a q-type assumption [34]. Thus their protocol relies on a q-type assumptions and constructing an efficient UC AOT protocol from only standard assumptions was an open problem. As a corollary of our result, we can obtain a UC AOT protocol based on only standard assumptions by replacing their SPS scheme with ours.

As an application of our schemes, Abe, Camenisch, Dubovitskaya, and Nishimaki proposed a UC AOT with hidden access control protocol from standard assumptions by using our schemes [11]. Moreover, they proposed an XRMA-secure SPS scheme only from the SXDH assumption based on another (non-structure-preserving) signature scheme by Chen, Lim, Ling, Wang, and Wee [22]. However, their scheme is less efficient than ours since their construction technique is different from ours and their message space is large.
8 Conclusions and Open Questions

We showed how to construct constant-size SPS consisting of only 11 to 14 group elements based on simple assumptions such as DLIN for symmetric pairings and analogues of DDH and XDLIN for asymmetric pairings. Our approach is modular and divides the problem into the need to construct constant-size RMA/xRMA secure SPS and constant-size structure-preserving one-time signatures. This is in line with the promise of [8] that SPS enable modular protocol design. Indeed this modularity facilitates applications in which one can cherry pick primitives according to requirements.

A tight bound for the size of SPS under simple assumptions is an important open question, and would shed light on the overhead of such a modular approach. It is also still an open question to construct efficient RMS/xRMA secure SPS schemes from only the SXDH assumption. Similarly, constructing (X)RMA-secure schemes with a message space that is a simple Cartesian product of groups without sacrificing efficiency and constructing more efficient RMA-secure schemes, which may not necessarily be XRAMA-secure are interesting open problems. All RMA-secure signature schemes developed in this paper are in fact XRMA-secure.

References


We review Waters’ dual system signature scheme \[51\] in this section.

### Waters’ Dual System Signature Scheme

We review Waters’ dual system signature scheme \[51\] in this section.

#### Scheme WdSIG

**WdSIG.Key(gk):** Given \(gk := (\Lambda, G)\) as input, sample \(V, V_1, V_2, H, I, U\) uniformly from \(G^*\) and \(a_1, a_2, b, c\) from \(\mathbb{Z}_p^*\). Then compute

\[
B := G^b, \quad A_1 := G^{a_1}, \quad A_2 := G^{a_2}, \quad B_1 := G^{b_1}, \quad B_2 := G^{b_2}
\]

\[
R_1 := VV_1^{a_1}, \quad R_2 := VV_2^{a_2}, \quad W_1 := R_1^c, \quad W_2 := R_2^c,
\]

\[
T := e(G, G)^{a_1b_1c_1}, \quad K_1 := G^{α_1}, \quad K_2 := G^{α_2},
\]

and output \(vk := (B, A_1, A_2, B_1, B_2, R_1, R_2, W_1, W_2, H, I, U, T)\) and \(sk := (vk, K_1, K_2, V, V_1, V_2)\).

**WdSIG.Sign(sk, msg):** Parse \(sk\) into \((vk, K_1, K_2, V, V_1, V_2)\). Also parse \(vk\) accordingly. For \(msg \in \mathbb{Z}_p\), pick random \(r_1, r_2, s_1, s_2, tag_k \in \mathbb{Z}_p\). Let \(r = r_1 + r_2\). Compute and output signature \(σ := (S_1, \ldots, S_7, S_0, tag_k)\) where

\[
S_1 := K_2V^r, \quad S_2 := K_1^{-1}V_1^{a_1}G^{a_1}, \quad S_3 := B^{-z_1}, \quad S_4 := V_2'G^{z_2},
\]

\[
S_5 := B^{-z_2}, \quad S_6 := B^r, \quad S_7 := G^r, \quad S_0 := (U^{msg_1}I^{tag_k}H)^{r_1}.
\]

**WdSIG.Vrf(vk, σ, msg):** Parse \(σ\) into \((S_1, \ldots, S_7, S_0, tag_k)\). Also parse \(vk\) accordingly. Pick random \(s_1, s_2, t\) and \(tag_c\) from \(\mathbb{Z}_p\), compute

\[
C_1 := B^{s_1+s_2}, \quad C_2 := B_1^{s_1}, \quad C_3 := A_1^{s_1}, \quad C_4 := B_2^{s_2}, \quad C_5 := A_2^{s_2}, \quad C_6 := R_1^{s_1}R_2^{s_2}, \quad C_7 := W_1^{s_1}W_2^{s_2}, \quad E_1 := (U^{msg_1}I^{tag_k}H)^{r_1}, \quad E_2 := G^t,
\]

and if \(tag_c - tag_k \neq 0\), verify

\[
e(\sigma(C_1, S_1) \cdot e(C_2, S_2) \cdot e(C_3, S_3) \cdot e(C_4, S_4) \cdot e(C_5, S_5), \ e(C_6, S_0) \cdot e(C_7, S_7) \cdot (e(E_1, S_7)/e(E_2, S_0))^{1/(tag_c - tag_k)} \cdot T^{s_2}.
\]