Tightly Secure Non-Interactive BLS Multi-Signatures

Renas Bacho [1,2] Benedikt Wagner [3]

August 30, 2024

[1] CISPA Helmholtz Center for Information Security
renas.bacho@cispa.de
[2] Saarland University

[3] Ethereum Foundation
benedikt.wagner@ethereum.org

Abstract

Due to their simplicity, compactness, and algebraic structure, BLS signatures are among the most widely used signatures in practice. For example, used as multi-signatures, they are integral in Ethereum's proof-of-stake consensus. From the perspective of concrete security, however, BLS (multi-)signatures suffer from a security loss linear in the number of signing queries. It is well-known that this loss can not be avoided using current proof techniques.

In this paper, we introduce a new variant of BLS multi-signatures that achieves tight security while remaining fully compatible with regular BLS. In particular, our signatures can be seamlessly combined with regular BLS signatures, resulting in regular BLS signatures. Moreover, it can easily be implemented using existing BLS implementations in a black-box way. Our scheme is also one of the most efficient non-interactive multi-signatures, and in particular more efficient than previous tightly secure schemes. We demonstrate the practical applicability of our scheme by showing how proof-of-stake protocols that currently use BLS can adopt our variant for fully compatible opt-in tight security.

Keywords: Non-Interactive, Multi-Signatures, BLS Signatures, Tightness, Pairings

©IACR 2024. This is the full version of an article that will be published in the proceedings of ASIACRYPT 2024.

Contents

1 Introduction

One of the most widely used digital signature schemes is due to Boneh, Lynn, and Shacham (BLS) [BLS01]. BLS signatures play a crucial role in many decentralized applications such as Chia [con22, Inc24], randomness beacons [BL22, Org20], lotteries [BLL+23, GHM+17], and Ethereum's proof-of-stake (PoS) consensus [Edg23]. They are not only simple and efficient, but they also possess several attractive algebraic properties. A particularly useful property is their support for efficient *non-interactive multi-signatures* [Bol03, BDN18]. For instance, suppose Alice, Bob, and Charlie each have a BLS secret key and use it to sign a message m individually. These individual signatures can be aggregated into a single BLS signature for m, which can be verified against a combination of their public keys. Informally, this aggregated signature certifies that all three parties have signed m. This multi-signature feature is central to Ethereum's PoS mechanism, and it is also the subject of this work. In particular, we focus on the concrete security of BLS multi-signatures, as explained next.

Concrete Security of BLS. The security proof for BLS (multi-signatures) follows a straightforward reduction approach: assuming an efficient adversary $\mathcal{A}$ breaking BLS, one can construct an efficient reduction $\mathcal{R}$ that breaks the Computational Diffie-Hellman (CDH) assumption. Specifically, if $\mathcal{A}$ breaks BLS with probability ϵ, then $\mathcal{R}$ breaks CDH with probability at least $\epsilon' = \epsilon/\Theta(q_s)$, where q_s denotes the number of signatures that $\mathcal{A}$ learns. For practical values of q_s, this results in a relatively *loose* security bound: if $q_s = 2^{30}$ and CDH is 128-bit hard, this proof only guarantees 98 bits of security for BLS.

Tightness and Impossibility. It would be highly desirable to have a *tight* security proof, meaning a proof where $\epsilon' \approx \epsilon$. Unfortunately, such a tight proof is not possible for BLS multi-signatures. This is because a tight proof for BLS multi-signatures would imply a tight proof for single-signer BLS signatures, and existing impossibility results rule this out for unique signatures like BLS [BJLS16, Cor02, KK12]. In contrast, a certain non-unique variant of BLS signatures can be proven tightly secure [GJKW07, KW03]. However, this variant sacrifices many of the desirable algebraic properties of the original BLS scheme.

Our Goal. While BLS cannot achieve tight security, we can still explore the following question:

Is there a tightly secure and non-interactive multi-signature compatible with standard BLS signatures?

Here, we should clarify what we mean by *compatibility*. Clearly, it cannot mean that signature verification is exactly the same as in BLS, due to the aforementioned impossibility results [BJLS16, Cor02, KK12]. Instead, the minimal requirement for compatibility should be: (1) verification and signing algorithms can easily be obtained by invoking BLS signing and verification in a black-box manner, and (2) signers using the new scheme should be able to combine their signatures with those of legacy signers using plain BLS, without requiring significant changes in the verification process.

1.1 Our Contribution

We affirmatively address this question by constructing a variant of BLS multi-signatures that is efficient, tightly secure, and compatible with standard BLS signatures, as outlined next.

Security. Our scheme achieves tight security based on the CDH assumption in the random oracle model (ROM). In particular, we do not rely on the algebraic group model (AGM) [FKL18] or the knowledge of secret key model (KOSK) [Bol03]. We compare the security guarantees of our scheme with existing non-interactive multi-signatures in Table 1.

Efficiency. We compare the efficiency of non-interactive multi-signatures in Table 2. Our scheme is (almost) as efficient as regular BLS multi-signatures: signing involves computing one hash followed by calling regular BLS signing, while verification and aggregation maintain the same efficiency as regular BLS. Notably, our scheme outperforms the previous tightly secure schemes, $\mathsf{BNN07}$ [BNN07] and $\mathsf{QLH12}$ [QLH12], in terms of efficiency.

Compatibility and Applications. The core of our result is a new tightly secure signature scheme. Intuitively, a signer randomly uses one of two BLS keys for each message. Consequently, our signatures can be aggregated with regular BLS signatures, resulting in a standard BLS signature[1]. Our scheme can be implemented on top of existing BLS implementations in a black-box manner. As an application, we consider proof-of-stake (PoS) blockchains utilizing BLS, such as Ethereum [Edg23]. In this context,

[1]The signature is not unique because it is valid for one of multiple possible keys.

Scheme	Assumption	Loss	Idealization
BLS [Bol03]	CDH	$\Theta(q_s)$	ROM
RY07 [RY07]	CDH	$\Theta(q_s)$	ROM
BDN18 [BDN18]	CDH	$\Theta(q_h^2/\epsilon)$	ROM
LOSSW06 [LOS+06]	CDH	$\Theta(\ell q_s)$	KOSK
QX10 [QX10]	CDH	$\Theta(q_s^2 q_h/\epsilon)$	ROM
DGNW20 [DGNW20]	wBDHI	$\Theta(q_h)$	ROM
BNN07 [BNN07]	CDH	$\Theta(1)$	ROM
QLH12 [QLH12]	CDH	$\Theta(1)$	ROM
BLSMS_2	CDH	$\Theta(1)$	ROM

Table 1: Comparison of non-interactive multi-signature schemes in the pairing setting. We compare under which hardness assumption the scheme is proven secure, the asymptotic tightness loss of the security proof, and under which idealized model the scheme is proven secure. Here, we do not consider proofs in the algebraic group model (AGM). We denote the number of random oracle and signing queries by q_h and q_s, respectively, and the advantage of an adversary against the scheme by ϵ. For LOSSW06 [LOS+06], ℓ denotes the bit-length of messages. Further, wBDHI denotes the weak bilinear Diffie-Hellman inversion assumption [BBG05], ROM denotes the random oracle model, and KOSK denotes the knowledge of secret key model [Bol03].

our results demonstrate how to operate a validator with tight security while remaining compatible with existing validators. For further details, we refer to Section 5.

Remark 1 (Tightness and Compatibility). We argue that compatibility issues are often a reason why many tight schemes are not even being considered for deployment in practice. As an example, note that a long line of research has focused on Schnorr-compatible multi-party signatures, e.g. [AB21, CKM21, NRS21] and references therein. Such schemes are currently being implemented in Bitcoin and even about to be standardized. On the contrary, tightly secure variants which are not compatible (e.g., [PW23, PW24]) are not even considered for deployment and purely academic. Our scheme stands out as being compatible and tightly secure at the same time, which means that it is much more likely that this will find applications in practice.

1.2 More on Related Work

In this section, we discuss related work in more detail. Especially, we discuss previous results on non-interactive multi-signatures in general, and results specifically on BLS multi-signatures.

Multi-Signatures. Multi-signatures have been introduced by Itakura and Nakamura [IN83] and later formalized in the *plain public key model* by Bellare and Neven [BN06]. In this model, each signer independently generates its own public-secret key pair. A major concern in this setting are rogue-key attacks, in which the adversary would choose its public key as a function of an honest user's key, allowing him to create forgeries easily. Such attacks have hindered progress in early stages [Lan96, LHL95, MOR01, MH96, OO93]. In order to prevent rogue-key attacks, Boldyreva [Bol03] has introduced the *knowledge of secret key (KOSK) model* for multi-signature schemes which was adopted by many subsequent works [CKM21, DEF+19, LOS+06]. In this model, it is assumed that the adversary must reveal its secret keys at public key registration directly. For a discussion on this model with its drawbacks, we refer to [BN06, RY07]. Many multi-signature schemes with several rounds of communication per signature have been proposed. Three-round multi-signature schemes have been constructed in [BCJ08, BN06, BDN18, FH20, MPSW19, MOR01], all of which base their security on standard assumptions, specifically the Decisional Diffie-Hellman (DDH) assumption and the Discrete Logarithm (DLOG) assumption. Further, two-round multi-signature schemes have been constructed [AB21, BD21, BTT22, CKM21, DOTT21, NRS21, NRSW20, PW23, PW24, TZ23], some of which are partially non-interactive (i.e., the first signing round is message-independent and can be preprocessed) [NRS21, TZ23], while others achieve tight security [PW23, PW24]. In this work, we focus specifically on *non-interactive* multi-signatures.

Scheme	Public Key	Sig Share	Signature	Cost (Sig)	Cost (Ver)
BLS [Bol03]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr
RY07 [RY07]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr
BDN18 [BDN18]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr
LOSSW06 [LOS+06]	$1\langle\mathbb{G}_T\rangle$	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle$	$2\text{ex} + 1\text{ex}^{\ell}$	$2\text{pr} + 1\text{ex}^{\ell}$
QX10 [QX10]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	$2\text{pr} + 1\text{ex}^{N+1}$
DGNW20 [DGNW20]	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle$	4ex	$3\text{pr} + 1\text{ex}$
BNN07 [BNN07]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 1$	$1\langle\mathbb{G}\rangle + N$	1ex	$(N+1)\text{pr}$
QLH12 [QLH12]	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1$	$4\langle\mathbb{G}\rangle$	2ex	4pr
BLSMS_2	$2\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 1$	$1\langle\mathbb{G}\rangle + N$	1ex	2pr

Table 2: Comparison of non-interactive multi-signature schemes in the pairing setting. We assume that all constructions are instantiated with a symmetric pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and compare the size of a public key, signature share, the size of the signature, the computational cost per signer, and the computational cost for verification. We denote the size of a group element by $\langle\mathbb{G}\rangle$ (respectively $\langle\mathbb{G}_T\rangle$), the number of signers by N, and the number of exponentiations, pairings, and k-multi-exponentiations for $k \in \mathbb{N}$ by ex, pr, and ex^k, respectively. For LOSSW06 [LOS+06], ℓ denotes the bit-length of messages.

Non-Interactive Multi-Signatures. A non-interactive multi-signature scheme is a multi-signature scheme which requires only a single round of communication among a set of n parties to produce a signature. Namely, each party outputs a signature share, and then the n signature shares can be (publicly) combined into a single short signature. Despite its practical relevance, there are only a few non-interactive multi-signatures in the literature, which we briefly discuss next. The first non-interactive multi-signature scheme is the BLS multi-signature proposed by Boldyreva [Bol03]. As for single-signer BLS, its security is based on the Computational Diffie-Hellman (CDH) assumption and has a security loss of $\Theta(q_s)$ where q_s denotes an upper bound on the number of allowed signing queries. Further, the security proof relies on the KOSK model. Several follow-up works [BCG+23, BDN18, RY07] have proposed variants of the BLS multi-signature, eliminating the KOSK assumption. We will later elaborate in more detail on these schemes. Subsequently, several other schemes have been proposed [BNN07, LOS+06]. The scheme proposed by Lu et al. [LOS+06] is based on the Waters signature scheme [Wat05]. Its security is based on the CDH assumption and it has a security loss of $\Theta(\ell q_s)$ where ℓ denotes the bit-length of messages. The security proof relies on the KOSK model. The scheme proposed by Bellare et al. [BNN07] is based on the aggregate signature scheme of Boneh et al. [BGLS03]. Here, a user i's signature σ_i is a key-prefixed BLS signature $\mathsf{H}(\mathsf{pk}_i, \mathsf{m})^{\mathsf{sk}_i}$ with the multi-signature being simply the product of individual signatures. Its security is based on the CDH assumption and comes with a security loss of $\Theta(q_s)$. Further, by using the Katz-Wang technique [GJKW07], the authors obtain a tight multi-signature. Notably, their schemes do not rely on the KOSK model. However, the final signatures require $n+1$ pairing evaluations for verification. Later, Qian and Xu [QX10] have proposed a multi-signature scheme that only requires two pairing evaluations (and one multi-exponentiation) for verification. Its security is based on the CDH assumption and it has a large security loss of $\Theta(q_s^2 q_h/\epsilon)$ where q_h denotes an upper bound on the number of allowed hash queries. A follow-up work by Qian et al. [QLH12] improves upon this by proposing the first non-interactive multi-signature scheme with tight security and efficient verification. Their scheme is based on the Waters signature scheme and uses the Katz-Wang technique to obtain a tight security reduction from the CDH assumption. Notably, their scheme does not rely on the KOSK model. Finally, Drijvers et al. [DGNW20] have proposed a multi-signature scheme based on the Boneh-Boyen-Goh hierarchical identity-based encryption (HIBE) scheme [BBG05]. Its security is based on the weak bilinear Diffie-Hellman inversion (wBDHI) assumption [BBG05] for type-3 pairings and has a security loss of $\Theta(q_h)$. Notably, their scheme does not rely on the KOSK model.

BLS Multi-Signatures. As stated above, the proof of security for the original BLS multi-signature by Boldyreva [Bol03] relies on the KOSK model. This was improved upon by the scheme of Ristenpart and Yilek [RY07] leveraging proofs of possession (POPs) of secret keys to prevent rogue-key attacks without relying on the KOSK model. Still, the security is based on the CDH assumption and has a security loss of $\Theta(q_s)$. Later, Boneh et al. [BDN18] have proposed another variant of the BLS multi-signature without relying on the KOSK model. Essentially, this is achieved by rerandomization of public keys of users as

$\mathsf{pk}_i' := \mathsf{pk}_i^{a_i}$ where $a_i := \mathsf{H}_{\mathrm{rand}}(\mathsf{pk}_i, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\})$ for a random oracle $\mathsf{H}_{\mathrm{rand}}$. Their security proof is still based on the CDH assumption, but now additionally relies on rewinding which results in a very loose bound of $\Theta(q_h^2/\epsilon)$. More recently, Baldimtsi et al. [BCG+23] gave a tight security reduction for the BLS multi-signature with rerandomization of public keys based on the DLOG assumption. However, their security proof relies on the algebraic group model (AGM) [FKL18]. The recent Internet Engineering Task Force (IETF) draft [BGW+22] specifies BLS signatures with proofs of possession for use in practical deployments. In fact, this is how BLS signatures on the Ethereum blockchain are used [Edg23]. As such, none of the proposed variants for BLS multi-signatures has a tight security proof without relying on the AGM.

1.3 Paper Organization

In Section 2, we give an informal but detailed technical overview of our constructions and proof techniques. In Section 3, we formally recall relevant cryptographic background and definitions. Then, in Section 4, we formally present our construction and its analysis. We conclude in Section 5, where we discuss an application to proof-of-stake blockchains.

2 Technical Overview

In this section, we give an informal overview of our constructions and our proof techniques. We do so in two steps: first, we introduce a new tightly secure variant of standard BLS signatures. While there is already a tightly secure variant of BLS [KW03], our variant is structurally more compatible with standard BLS as we will see. In the second step, we then show how to lift this construction to the multi-signature setting while preserving tight security.

2.1 Tightly Secure and Structured BLS Signatures

Let us first review BLS signatures and existing ways to construct tightly secure variants thereof. For simplicity, most of this overview will be written assuming a symmetric pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ is a cyclic group of prime order p with generator g.

BLS Signatures. A regular BLS signature for a message m with respect to public key $\mathsf{pk} = g^{\mathsf{sk}}$ is given as $\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}}$, where $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$ is a random oracle. It will be instructive to review the non-tight security proof of BLS [BLS01]: the goal is to give a reduction from the CDH assumption. This reduction gets as input two group elements $X = g^x$ and $Y = g^y$, and its goal is to compute g^{xy}. To this end, the reduction simulates the $\mathsf{EUF\text{-}CMA}$ security game with public key $\mathsf{pk} = X$ for the adversary. While doing so, it splits the message space into two partitions: (1) for most messages m, it will program $\mathsf{H}(\mathsf{m}) := g^{\gamma_{\mathsf{m}}}$, where $\gamma_{\mathsf{m}} \in \mathbb{Z}_p$ is a random exponent known to the reduction. Note that for these messages, the reduction can efficiently provide signatures to the adversary by returning $\sigma = X^{\gamma_{\mathsf{m}}}$, i.e., γ_{m} serves as a trapdoor; (2) for all other messages, it will embed the challenge Y into the hash: $\mathsf{H}(\mathsf{m}) := Y \cdot g^{\gamma_{\mathsf{m}}}$. For these messages, the reduction can efficiently obtain a CDH solution from a valid signature output by the adversary. As long as the adversary only queries signatures for messages from the first partition, and forges for a message in the second partition, the reduction succeeds. Unfortunately, this partitioning leads to a security loss linear in the number of signing queries. Indeed, it is known that such a loss is inherent for unique signatures like BLS [BJLS16, Cor02].

Random Bits for Tight Security. It is well-known that with a minimal change, BLS signatures can be made tightly secure: to sign a message m, a signer would pseudorandomly derive a bit $\beta_{\mathsf{m}} \in \{0,1\}$ from the message, and then compute the signature as $\sigma = \mathsf{H}(\mathsf{m}, \beta_{\mathsf{m}})^{\mathsf{sk}}$. This is often called the Katz-Wang technique [GJKW07], and it enables the following tight security proof: for each message m, the reduction programs $\mathsf{H}(\mathsf{m}, \beta_{\mathsf{m}}) := g^{\gamma_{\mathsf{m}}}$ and $\mathsf{H}(\mathsf{m}, 1-\beta_{\mathsf{m}}) := Y \cdot g^{\gamma'_{\mathsf{m}}}$. That is, the reduction embeds a trapdoor in one branch, and the challenge in the second branch for each message. Obviously, the reduction can now compute σ using the trapdoor. On the other hand, the bit β_{m^*} for the forgery message m^* remains pseudorandom for the adversary, and so with probability $1/2$, the $(1-\beta_{\mathsf{m}^*})$-branch is used in the forgery, which ultimately allows the reduction to solve CDH. Observe that this proof does not partition the message space.

Algebraic Structure Lost. While the Katz-Wang technique gives an elegant way to obtain tight security, we pay a price: BLS signatures have many desirable algebraic features, which the random bit variant does not. For instance, assume we have two BLS signatures $\sigma_A = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_A}$ and $\sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_B}$ under different keys for the same message m. Then, the product $\sigma_A \cdot \sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_A + \mathsf{sk}_B}$ is a valid signature under the product of the keys $\mathsf{pk}_A \cdot \mathsf{pk}_B$. This observation underlies the design of BLS multi-signatures. Now, consider the same setting for the Katz-Wang variant: as each signer has to compute its bit pseudorandomly, the two signatures are likely of the form $\sigma_A = \mathsf{H}(\mathsf{m}, 0)^{\mathsf{sk}_A}$ and $\sigma_B = \mathsf{H}(\mathsf{m}, 1)^{\mathsf{sk}_B}$. When we multiply them, we do not get a BLS signature under the product of keys.

Towards a Solution. The above example shows that, in order to preserve the algebraic structure of BLS signatures, it is essential to ensure every signer uses the same basis $\mathsf{H}(\mathsf{m})$ for a given message m. Still, resorting to standard BLS can not lead to tight security [BJLS16, Cor02], as already explained. To make progress towards a solution, let us assume that we still have a pseudorandom bit β_{m}, but use it differently. Concretely, say a signer now holds two BLS public keys $\mathsf{pk}_0 = g^{\mathsf{sk}_0}$ and $\mathsf{pk}_1 = g^{\mathsf{sk}_1}$. Then, we could define the signature to be $\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{\beta_{\mathsf{m}}}}$. For now, it is not clear at all that this leads to a tight security proof, but we can already see that this is much more compatible with BLS than the Katz-Wang variant: suppose Alice uses this variant, but Bob still uses regular BLS. Now say we have their two signatures $\sigma_A = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{A,\beta_{\mathsf{m}}}}$ and $\sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_B}$. Then, the product $\sigma_A \cdot \sigma_B$ is a *regular BLS signature* for m with respect to the key $\mathsf{pk}_{A,\beta_{\mathsf{m}}} \cdot \mathsf{pk}_B$.

Proving Security. As the previous example shows, the variant above is structurally compatible with regular BLS signatures. The question is if this variant is also tightly secure. Indeed reusing the Katz-Wang proof technique does not work: we only have one branch for each message. This means that for each message m, we have to decide whether we would embed the challenge or a trapdoor, i.e., we have to partition the message space. Fortunately, it turns out that we can still get a tight security proof. Say our reduction gets as input the CDH challenge $X = g^x$ and $Y = g^y$. As in the proof for regular BLS signatures, we want to embed X in the key and Y in some of the random oracle outputs. Of course, embedding X in a fixed key, say in pk_0, is not a good idea. This is because an adversary could potentially always choose to use pk_1 in its forgery and the scheme degenerates to regular BLS. Instead, say we embed X randomly: we sample a bit $\hat{\beta} \xleftarrow{\$} \{0,1\}$ at random and define $\mathsf{pk}_{\hat{\beta}} := X$, and make sure that the reduction knows $\mathsf{sk}_{1-\hat{\beta}}$. Next, the reduction partitions the message space. This has to be done in a way that ensures that the reduction can always simulate signatures, namely:

- *Trapdoor Partition.* If $\beta_{\mathsf{m}} = \hat{\beta}$, i.e., the signature is $\sigma = \mathsf{H}(\mathsf{m})^x$, the reduction embeds a trapdoor into $\mathsf{H}(\mathsf{m})$.
- *Challenge Partition.* Otherwise, the signature is $\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{1-\hat{\beta}}}$, and the reduction can embed Y into $\mathsf{H}(\mathsf{m})$ because it can always simulate such signatures using $\mathsf{sk}_{1-\hat{\beta}}$.

Now, consider the adversary's forgery (m^*, σ^*). We can argue that the bit b_{m^*} is hidden from the adversary, so with probability $1/2$ over the choice of this bit, the message m^* is in the challenge partition. Similarly, with probability $1/2$ over the choice of $\hat{\beta}$ the forgery is with respect to $\mathsf{pk}_{\hat{\beta}}$. This means that with probability $1/4$, the forgery contains g^{xy}, which the reduction can use to solve CDH. In this way, we get a tight security proof.

2.2 Tightly Secure BLS Multi-Signatures

Equipped with our tightly secure variant of BLS signatures, we now turn our focus to multi-signatures. We will first recall common techniques to securely turn BLS signatures into multi-signatures. Throughout, we consider the simplified setting of two parties, Alice and Bob, signing a message m. Alice will generally be assumed to be our honest party, whereas Bob is assumed to be adversarial.

BLS Multi-Signatures. As we have mentioned above, we can combine the two BLS signatures $\sigma_A = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_A}$ of Alice and $\sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_B}$ of Bob into a single signature $\sigma = \sigma_A \cdot \sigma_B$ for the key $\mathsf{pk}_A \cdot \mathsf{pk}_B$. It is a well-established fact that without further modification, this naive BLS multi-signature is insecure due to so-called *rogue-key attacks*: Bob could choose its key as $\mathsf{pk}_B := \mathsf{pk}_A^{-1} \cdot g^\delta$, which allows him to create valid multi-signatures for public key $\mathsf{pk}_A \cdot \mathsf{pk}_B = g^\delta$ without talking to Alice. Prominently, there are two ways to solve this issue:

- *Random Linear Combinations.* Instead of defining the multi-signature as $\sigma_A \cdot \sigma_B$, we define it as $\sigma_A^{a_A} \cdot \sigma_B^{a_B}$ and the aggregate public key as $\mathsf{pk}_A^{a_A} \cdot \mathsf{pk}_B^{a_B}$, where $(a_A, a_B) \in \mathbb{Z}_p$ are random coefficients derived using a random oracle.
- *Proofs of Possession.* A public key is only considered valid if it comes with a proof of possession[2] of the secret key [RY07]. Concretely, this proof is usually implemented as $\mathsf{G}(\mathsf{pk}_B)^{\mathsf{sk}_B}$ (for Bob), i.e., as a BLS signature using a different random oracle.

The latter approach is often used in practice, e.g., in Ethereum [Edg23], and it will also be our focus. Ristenpart and Yilek [RY07] have shown that this approach is provably secure for regular BLS. Interestingly, their proof does not add any additional security loss: the security loss for the multi-signature is the same as the one for regular BLS. Luckily, their proof technique also applies to our tightly secure variant, except for some complications in the asymmetric pairing setting. This holds even if Alice uses our variant, and Bob uses regular BLS. In the following, we review the challenge, the proof strategy by Ristenpart and Yilek [RY07], and explain the complications when using asymmetric pairings.

A Closer Look at the Proof. When we want to prove the security of the BLS multi-signature with proofs of possession, we have to simulate our honest signer Alice for the adversary, and we have to turn the adversary's forgery into a CDH solution. While the former task did not change compared to the case of standard signatures, the latter task is more challenging in the multi-signature setting. This is because a forgery σ^* is now valid for a message m^* and *some* combined key $\mathsf{pk}_A \cdot \mathsf{pk}_B$, where pk_B is made up by Bob. More concretely, say we have embedded our CDH challenge $(X, Y) = (g^x, g^y)$ in Alice's key, such that $\mathsf{sk}_A = x$, and say we have managed that $\mathsf{H}(\mathsf{m}^*)$ contains the challenge Y, i.e., $\mathsf{H}(\mathsf{m}^*) = Y \cdot g^{\gamma_{\mathsf{m}^*}}$ for some γ_{m^*} known to the reduction. For simplicity, assume $\gamma_{\mathsf{m}^*} = 0$ for now. Then, the forgery has the form

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\mathsf{sk}_A + \mathsf{sk}_B} = g^{xy} \cdot Y^{\mathsf{sk}_B}.$$

So, if the reduction wants to compute the CDH solution g^{xy} from σ^*, it has to compute $Y^{\mathsf{sk}_B} = g^{y\mathsf{sk}_B}$. But the reduction does not know y and $\mathsf{pk}_B = g^{\mathsf{sk}_B}$ is made up by the adversary!

The Adversary Solves Our Problem. As already observed by Ristenpart and Yilek [RY07], we can let the adversary solve our problem via proofs of possession: The reduction would embed Y into random oracle G as well. In simplified terms, assume that $\mathsf{G}(\mathsf{pk}_B) = Y$. In this case, the proof of possession $\mathsf{G}(\mathsf{pk}_B)^{\mathsf{sk}_B}$ is exactly the desired term Y^{sk_B}, and the reduction can use it to compute g^{xy}.

Complications in the Asymmetric Setting. So far, we have simplified notation by using the symmetric pairing setting. Indeed, the technique by Ristenpart and Yilek [RY07] for regular BLS multi-signatures only works in the symmetric pairing setting and the type-2 pairing setting where there is an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$. Making the analysis work for an asymmetric pairing without such an isomorphism, also known as the type-3 setting[3], leads to subtle challenges, which we outline next. Namely, recall that above we have assumed $\gamma_{\mathsf{m}^*} = 0$. In general, this will of course not be the case and the forgery will have the form

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\mathsf{sk}_A + \mathsf{sk}_B} = (Y \cdot g^{\gamma_{\mathsf{m}^*}})^{\mathsf{sk}_A + \mathsf{sk}_B} = g^{xy} \cdot X^{\gamma_{\mathsf{m}^*}} \cdot Y^{\mathsf{sk}_B} \cdot g^{\gamma_{\mathsf{m}^*}\mathsf{sk}_B}.$$

We have already discussed how the reduction can eliminate the term Y^{sk_B} using the proofs of possession. It can of course also eliminate the term $X^{\gamma_{\mathsf{m}^*}}$ using knowledge of X and γ_{m^*}. In the symmetric pairing setting, the reduction can also compute and remove the final term $g^{\gamma_{\mathsf{m}^*}\mathsf{sk}_B} = \mathsf{pk}_B^{\gamma_{\mathsf{m}^*}}$. In the asymmetric setting, however, the reduction has the adversarially chosen keys pk_B only in one group, say $\mathbb{G}_2$, but the signature σ^* is in the other group $\mathbb{G}_1$, so the reduction needs to compute $g_1^{\gamma_{\mathsf{m}^*}\mathsf{sk}_B}$ over $\mathbb{G}_1$. It is not clear how to do that[4]. Our solution is to define the random oracles multiplicatively, namely, we set $\mathsf{H}(\mathsf{m}) := \mathsf{pk}_{1-b_{\mathsf{m}}}^{\gamma_{\mathsf{m}}}$ for each message m. It turns out that this enables a tight proof in the type-3 setting.

[2] Enforcing such a valid proof of possession can of course be modeled as just another check in the signature verification algorithm.

[3] The type-3 setting is indeed the most common setting in practice.

[4] We could of course force Bob to additionally output its public key pk_B over $\mathbb{G}_1$, but this is generally not what happens in practice, so we refrain from doing so.

3 Preliminaries

Here, we define our notation and recall relevant cryptographic primitives and assumptions.

Notation. We denote the security parameter by λ and assume that all algorithms get 1^λ implicitly as input. We use standard cryptographic terminology like *negligible, overwhelming, PPT*. To sample an element x uniformly at random from a set W, we write $x \xleftarrow{\$} W$. We write $x \leftarrow \mathcal{D}$ if $\mathcal{D}$ is a distribution or a probabilistic algorithm. That is, writing $y \leftarrow \mathcal{A}(x)$ means that algorithm $\mathcal{A}$ is run with uniformly sampled random coins on input x and y is the output. If $\mathcal{A}$ is known to be deterministic, we write $y := \mathcal{A}(x)$ instead. We write $\mathbf{T}(\mathcal{A})$ to denote the running time of $\mathcal{A}$. We define $[N] := \{1, \ldots, N\} \subseteq \mathbb{N}$.

Computational Assumptions. Throughout this paper, we assume an algorithm $\mathsf{PGGen}(1^\lambda)$ that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a non-degenerate[5] pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$.

Definition 1 (CDH Assumption)**.** We say that the CDH assumption holds relative to PGGen, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathsf{PGGen}}(\lambda) := \Pr\left[z = xy \;\middle|\; \begin{array}{l} (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda), \\ x, y \xleftarrow{\$} \mathbb{Z}_p,\ X_1 := g_1^x,\ X_2 := g_2^x,\ Y := g_1^y \\ g_1^z \leftarrow \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e, X_1, Y, X_2) \end{array} \right]$$

Digital Signatures. We define the syntax of digital signatures and the standard notion of unforgeability under chosen-message attacks [GMR88].

Definition 2 (Signature Scheme)**.** A signature scheme is a tuple of PPT algorithms $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ takes as input the security parameter 1^λ and outputs global system parameters par. We assume that par implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to SIG take par at least implicitly as input.
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ takes as input system parameters par, and outputs a public key pk and a secret key sk.
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$ takes as input a secret key sk and a message m and outputs a signature σ.
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$ is deterministic, takes as input a public key pk, a message m, and a signature σ, and outputs a bit $b \in \{0, 1\}$.

We require that SIG is complete in the following sense. For all $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$, all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\mathsf{par})$, and all messages m, we have

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) = 1 \mid \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})\right] = 1.$$

Definition 3 (EUF-CMA Security for Signatures)**.** Let $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a signature scheme. Consider an adversary $\mathcal{A}$. Further, consider the game $\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}(\lambda)$ defined as follows:

1. Run $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$.
2. Initialize $\mathcal{Q} := \emptyset$ and let O be the following oracle:
   - $\mathrm{O}(\mathsf{m})$: Take as input m, set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$, and return $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})$.
3. Run $\mathcal{A}$ on input $(\mathsf{par}, \mathsf{pk})$ and with access to oracle O. Obtain (m^*, σ^*) from $\mathcal{A}$.
4. Output 1 if and only if $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}^*, \sigma^*) = 1$ and $\mathsf{m}^* \notin \mathcal{Q}$.

We say that SIG is EUF-CMA secure, if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{SIG}}(\lambda) := \Pr\left[\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}(\lambda) \Rightarrow 1\right].$$

[5] Non-degenerate means that $e(g_1, g_2)$ is a generator of the group $\mathbb{G}_T$.

Next, we recall the signature scheme due to Boneh, Lynn, and Shacham [BLS01]. It is well-known that it is (non-tightly) EUF-CMA secure based on the CDH assumption [BLS01, Cor00]. Throughout, we assume that signatures are in $\mathbb{G}_1$ and public keys are in $\mathbb{G}_2$. By symmetry, all of our results apply if the roles are reversed.

Definition 4 (BLS Signatures [BLS01])**.** Consider a random oracle $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$. The signature scheme[6] $\mathsf{BLS} = (\mathsf{BLS.Setup}, \mathsf{BLS.Gen}, \mathsf{BLS.Sig}^{\mathsf{H}}, \mathsf{BLS.Ver}^{\mathsf{H}})$ is defined via the following algorithms:

- $\mathsf{BLS.Setup}(1^\lambda) \to \mathsf{par}$: Define $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
- $\mathsf{BLS.Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$: Sample $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{pk} := g_2^{\mathsf{sk}}$.
- $\mathsf{BLS.Sig}^{\mathsf{H}}(\mathsf{sk}, \mathsf{m}) \to \sigma$: Set $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}}$.
- $\mathsf{BLS.Ver}^{\mathsf{H}}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$: Return $b = 1$ if $e(\mathsf{H}(\mathsf{m}), \mathsf{pk}) = e(\sigma, g_2)$. Otherwise, return $b = 0$.

Lemma 1. *If the* CDH *assumption holds relative to* PGGen *and* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$ *is modeled as a random oracle, then* BLS *is* EUF-CMA *secure. More precisely, for every PPT algorithm* $\mathcal{A}$ *that makes at most* Q *queries to* O*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *and*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLS}}(\lambda) \leq \Theta(Q) \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

Multi-Signatures. Next, we give a definition for multi-signatures, specifically, non-interactive multi-signatures. In such a scheme, each signer independently generates its key pair via a key generation algorithm Gen. To sign a message, each signer locally uses its secret key to compute a signature via an algorithm Sig. A list of such signatures for the same message can then be combined into a single signature via an algorithm Combine. As a result, one obtains a signature that verifies for the given message and with respect to the list of public keys. Note that trivially, we obtain a multi-signature scheme by letting Combine concatenate the signatures. However, the goal should always be to construct non-trivial multi-signatures in a sense that Combine outputs a signature much smaller than the concatenation.

Definition 5 (Multi-Signature Scheme)**.** A multi-signature scheme is a tuple of PPT algorithms $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Combine}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ takes as input the security parameter 1^λ and outputs global system parameters par. We assume that par implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to MS take par at least implicitly as input.
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ takes as input system parameters par, and outputs a public key pk and a secret key sk.
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$ takes as input a secret key sk and a message m and outputs a signature σ.
- $\mathsf{Combine}((\mathsf{pk}_1, \sigma_1), \ldots, (\mathsf{pk}_N, \sigma_N), \mathsf{m}) \to \sigma$ is deterministic, takes as input a list of keys and signatures $(\mathsf{pk}_1, \sigma_1), \ldots, (\mathsf{pk}_N, \sigma_N)$, and a message m, and outputs a signature σ.
- $\mathsf{Ver}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathsf{m}, \sigma) \to b$ is deterministic, takes as input a list of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, a message m, and a signature σ, and outputs a bit $b \in \{0,1\}$.

We require that MS is complete in the following sense. For all $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$, all $N \in \mathbb{N}$, all $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathsf{Gen}(\mathsf{par})$ for every $i \in [N]$, and all messages m, we have

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathsf{m}, \sigma) = 1 \;\middle|\; \begin{array}{l} \forall i \in [N]:\ \sigma_i \leftarrow \mathsf{Sig}(\mathsf{sk}_i, \mathsf{m}), \\ \sigma := \mathsf{Combine}((\mathsf{pk}_1, \sigma_1), \ldots, (\mathsf{pk}_N, \sigma_N), \mathsf{m}) \end{array}\right] = 1.$$

[6] For BLS signatures, we make the hash function H an explicit parameter, which will simplify notation later. Concretely, the proofs of possession in BLS multi-signatures are implemented using BLS on a different hash function. For our constructions, we omit adding every hash function as an explicit parameter to avoid clutter.

Below, we define unforgeability for multi-signatures following our syntax. As in the unforgeability game for signatures, the adversary gets access to a target public key and to a signing oracle. The only difference to the game for signatures is that the forgery is now a combined signature, and is expected to come with a list of public keys that includes the target public key.

Definition 6 (EUF-CMA Security for Multi-Signatures)**.** Let $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Combine}, \mathsf{Ver})$ be a multi-signature scheme. Consider an adversary $\mathcal{A}$ and the game $\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}(\lambda)$ defined as follows:

1. Run $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$.
2. Initialize $\mathcal{Q} := \emptyset$ and let O be the following oracle:
   - $\mathrm{O}(\mathsf{m})$: Take as input m, set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$, and return $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})$.
3. Run $\mathcal{A}$ on input $(\mathsf{par}, \mathsf{pk})$ and with access to oracle O. Obtain a list of keys $(\mathsf{pk}^*_1, \ldots, \mathsf{pk}^*_N)$ and a pair (m^*, σ^*) from $\mathcal{A}$.
4. Output 1 if and only if there is an index $i \in [N]$ such that $\mathsf{pk} = \mathsf{pk}^*_i$, and it holds that $\mathsf{Ver}(\mathsf{pk}^*_1, \ldots, \mathsf{pk}^*_N, \mathsf{m}^*, \sigma^*) = 1$, and that $\mathsf{m}^* \notin \mathcal{Q}$.

We say that MS is EUF-CMA secure, if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible:

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}}(\lambda) := \Pr\left[\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}(\lambda) \Rightarrow 1\right].$$

4 Variants of BLS Multi-Signatures

In this section, we present our new tightly secure variant of BLS multi-signatures. To avoid repetition and to highlight the similarity to BLS, we do not only define a single scheme, but rather a class of schemes BLSMS_L, where $L \in \mathbb{N}$ is a parameter. Informally, this parameter specifies how many instances of BLS are combined. The interesting cases for this paper are as follows:

- $L = 1$: This corresponds to standard BLS multi-signatures with proofs of possession as used for example in Ethereum [Edg23]. Here, we give the to this date tightest known proof without the algebraic group model. Essentially, the security is not larger than for (single-signer) BLS signatures.
- $L = 2$: In this case, we obtain a tightly secure multi-signature based on CDH.

Interestingly, these constructions are fully compatible, i.e., one signer may use $L = 1$ whereas a different signer may choose to use $L = 2$ or even[7] $L = 3$. As corollaries of the case $L = 2$, we obtain new tightly secure variants of (single-signer) BLS signatures.

4.1 Parameterized Construction

Let $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$ be a random oracle which informally takes the role of the random oracle in BLS signatures. We first define helper algorithms KeyProve and KeyVer. Roughly, these are used to prove possession of secret keys, which is a common method to prevent rogue-key attacks [Edg23]. The way they are commonly implemented is as BLS signatures on the public key itself, using a different random oracle $\mathsf{G}\colon \{0,1\}^* \to \mathbb{G}_1$:

- $\mathsf{KeyProve}(\mathsf{pk}, \mathsf{sk}) \to \pi$: Set $\pi := \mathsf{BLS.Sig}^{\mathsf{G}}(\mathsf{sk}, \mathsf{pk})$
- $\mathsf{KeyVer}(\mathsf{pk}, \pi) \to \{0,1\}$: Return $b := \mathsf{BLS.Ver}^{\mathsf{G}}(\mathsf{pk}, \mathsf{pk}, \pi)$.

Finally, we use a third random oracle $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0, \ldots, L-1\}$ that will be used to randomly decide which key to use for signing. With these algorithms at hand, we now define BLSMS_L for $L \in \mathbb{N}$.

- $\mathsf{BLSMS}_L.\mathsf{Setup}(1^\lambda) \to \mathsf{par}$:

[7] One can also use our technique to prove tight security from CDH for $L = 3$, but we do not see a good reason to use this scheme, so we omit presenting this proof.

1. $\mathsf{par} \leftarrow \mathsf{BLS.Setup}(1^\lambda)$

- $\mathsf{BLSMS}_L.\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$:
  1. $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
  2. For all $\beta \in \{0, \dots, L-1\}$: $(\mathsf{pk}_\beta, \mathsf{sk}_\beta) \leftarrow \mathsf{BLS.Gen}(\mathsf{par})$
  3. $\mathsf{sk} := (\mathsf{sk}_0, \dots, \mathsf{sk}_{L-1}, \mathsf{seed})$
  4. For all $\beta \in \{0, \dots, L-1\}$: $\pi_\beta := \mathsf{KeyProve}(\mathsf{pk}_\beta, \mathsf{sk}_\beta)$
  5. $\pi := (\pi_0, \dots, \pi_{L-1}), \quad \mathsf{pk} := \big((\mathsf{pk}_0, \pi_0), \dots, (\mathsf{pk}_{L-1}, \pi_{L-1})\big)$
- $\mathsf{BLSMS}_L.\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$:
  1. $\beta_\mathsf{m} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$
  2. $\sigma := \mathsf{BLS.Sig}^\mathsf{H}(\mathsf{sk}_{\beta_\mathsf{m}}, \mathsf{m})$
- $\mathsf{BLSMS}_L.\mathsf{Combine}((\mathsf{pk}_1, \sigma_1), \dots, (\mathsf{pk}_N, \sigma_N), \mathsf{m}) \to \sigma$:
  1. For all $i \in [N]$: parse $\mathsf{pk}_i = \big((\mathsf{pk}_{i,0}, \dots, \mathsf{pk}_{i,L-1}), (\pi_{i,0}, \dots, \pi_{i,L-1})\big)$
  2. For all $i \in [N]$: $\beta_i := \min\{\beta \in \{0, \dots, L-1\} \mid \mathsf{BLS.Ver}^\mathsf{H}(\mathsf{pk}_{i,\beta}, \mathsf{m}, \sigma_i) = 1\}$
  3. $\bar{\sigma} := \prod_{i=1}^N \sigma_i, \quad \sigma := (\bar{\sigma}, \beta_1, \dots, \beta_N)$
- $\mathsf{BLSMS}_L.\mathsf{Ver}(\mathsf{pk}_1, \dots, \mathsf{pk}_N, \mathsf{m}, \sigma) \to b$:
  1. For all $i \in [N]$: parse $\mathsf{pk}_i = \big((\mathsf{pk}_{i,0}, \dots, \mathsf{pk}_{i,L-1}), (\pi_{i,0}, \dots, \pi_{i,L-1})\big)$
  2. Parse $\sigma = (\bar{\sigma}, \beta_1, \dots, \beta_N)$
  3. $\bar{\mathsf{pk}} := \prod_{i=1}^N \mathsf{pk}_{i,\beta_i}$
  4. $b := \mathsf{BLS.Ver}^\mathsf{H}(\bar{\mathsf{pk}}, \mathsf{m}, \bar{\sigma}) \wedge \bigwedge_{i \in [N]} \mathsf{KeyVer}(\mathsf{pk}_{i,\beta_i}, \pi_{i,\beta_i})$

Remark 2. In many applications, one would verify the proofs $\pi_{i,\beta}$ contained in public keys once when a party registers.

Remark 3. For the case $L = 1$, this scheme is exactly the standard BLS multi-signature scheme with proofs of possesion, noting that the step $\beta_\mathsf{m} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$ can safely omitted as β_m is fixed in this case in the signing algorithm. Similarly, the bits β_i can be omitted from the combined signature.

Remark 4. A combined signature has size $\mathsf{size}(\mathbb{G}_1) + N \log L$, where $\mathsf{size}(\mathbb{G}_1)$ denotes the size of a single group element. That is, the size of the signature still scales linearly with the number of signers N. However, for the interesting parameters $L \in \{1, 2\}$, this means at most one bit per signer. In practice, this can be ignored, as this only exceeds a small number of group elements for a very large number of signers. Constructions with similar efficiency have been proposed before [PW23, PW24].

Remark 5. One interesting feature of these multi-signatures is that they are interoperable: one signer may decide to keep using standard BLS signatures BLSMS_1, and another signer may choose to use BLSMS_2 or BLSMS_L for arbitrary $L \in \mathbb{N}$. The signatures can still be combined using obvious adaptations of algorithm $\mathsf{Combine}$. It is also clear that from the perspective of a single signer using BLSMS_L to sign, the security of BLSMS_L applies even if the adversary may use a different L. For example, a signer using $L = 2$ has tight security from CDH even when other users use standard BLS multi-signatures.

4.2 Security with One Key: BLS Multi-Signatures

Regular BLS multi-signatures correspond to the case $L = 1$. Ristenpart and Yilek [RY07] gave a proof for this variant with a security loss similar to standard BLS signatures (cf. Lemma 1). Their proof is in the type-2 pairing setting (following the well-known classification in [GPS06]), i.e., it relies on the existence of an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$. As outlined in the technical overview, we give a proof with the same security loss without this assumption. In this way, our proof also applies to the type-3 pairing setting. We postpone the proof to Appendix A and note that the proof is a simplified version of the proof of Theorem 2.

Theorem 1. *Assume that* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$, *and* $\mathsf{G}\colon \{0,1\}^* \to \mathbb{G}_1$ *are random oracles. If the* CDH *assumption holds relative to* PGGen, *then* BLSMS_1 *is* EUF-CMA *secure. More precisely, for every PPT algorithm* $\mathcal{A}$, *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *and*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_1}(\lambda) \leq \frac{8Q_S + 4Q_S Q_\mathsf{H} + 4Q_S Q_\mathsf{G}}{p} + 4Q_S \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda),$$

where $Q_\mathsf{H}, Q_\mathsf{G}, Q_S$ *denote the number of queries to* H, G, *and* O, *respectively.*

4.3 Two Keys and Tight Security

With $L = 2$, we get tight security from CDH, which is stated in the following theorem.

Theorem 2. *Assume that* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$, $\mathsf{G}\colon \{0,1\}^* \to \mathbb{G}_1$, *and* $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}$ *are random oracles. If the* CDH *assumption holds relative to* PGGen, *then* BLSMS_2 *is* EUF-CMA *secure. More precisely, for every PPT algorithm* $\mathcal{A}$, *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *and*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_2}(\lambda) \leq \frac{Q_{\hat{\mathsf{H}}}}{2^\lambda} + \frac{8 + 4Q_\mathsf{H} + 4Q_\mathsf{G}}{p} + 4 \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda),$$

where $Q_\mathsf{H}, Q_\mathsf{G}, Q_{\hat{\mathsf{H}}}$ *denote the number of queries to* H, G, *and* $\hat{\mathsf{H}}$, *respectively.*

Proof. We present our proof as a sequence of games, where the first game $\mathbf{G}_0$ is the EUF-CMA game. In games $\mathbf{G}_1$ to $\mathbf{G}_2$, we guess whether the adversary forges with respect to public key pk_0 or pk_1. Say our guess is $\hat{\beta} \in \{0,1\}$, meaning we now assume that the forgery is with respect to $\mathsf{pk}_{\hat{\beta}}$. Similarly, we ensure that $\mathsf{pk}_{\hat{\beta}}$ is not the key that the honest signer would have used for the forgery message. In games $\mathbf{G}_3$ and $\mathbf{G}_4$, we embed a random group element v into random oracle outputs for oracle H and establish that we can simulate the signing oracle efficiently without using $\mathsf{sk}_{\hat{\beta}}$. Intuitively, v and $\mathsf{pk}_{\hat{\beta}}$ will correspond to the CDH instance. At this point, the proof for the single signer setting would end with a reduction to the CDH assumption. As we are in the multi-signature setting, the following steps are needed: in game $\mathbf{G}_5$, we establish that the proof of possession for $\mathsf{pk}_{\hat{\beta}}$ can be simulated without using $\mathsf{sk}_{\hat{\beta}}$. In game $\mathbf{G}_6$, we then embed v into the adversary's proofs of possession. This is essential for eliminating terms related to adversarially chosen keys in the forgery when we then reduce to CDH. Let us now make all of this more precise.

Game $\mathbf{G}_0$: This is the EUF-CMA game for scheme BLSMS_2 and adversary $\mathcal{A}$, with a conceptual modification in the winning condition. We recall this game to fix notation. The game does the following to generate parameters and keys:

1. Set $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
2. Set $\mathcal{Q} := \emptyset$ and initialize empty maps $h[\cdot], \hat{h}[\cdot]$, and $g[\cdot]$.
3. Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{sk}_0, \mathsf{sk}_1 \xleftarrow{\$} \mathbb{Z}_p$.
4. Set $\mathsf{pk}_0 := g_2^{\mathsf{sk}_0}$, $\mathsf{pk}_1 := g_2^{\mathsf{sk}_1}$ and $\tilde{\mathsf{pk}}_0 := g_1^{\mathsf{sk}_0}$, $\tilde{\mathsf{pk}}_1 := g_1^{\mathsf{sk}_1}$.
5. Set $\pi_0 := \mathsf{G}(\mathsf{pk}_0)^{\mathsf{sk}_0}$ and $\pi_1 := \mathsf{G}(\mathsf{pk}_1)^{\mathsf{sk}_1}$.
6. Set $\mathsf{pk} := ((\mathsf{pk}_0, \pi_0), (\mathsf{pk}_1, \pi_1))$

Note that $\tilde{\mathsf{pk}}_0$ and $\tilde{\mathsf{pk}}_1$ are not used yet, but will be used in the following games. The game gives par and pk to the adversary $\mathcal{A}$. In addition, $\mathcal{A}$ gets access to random oracles H, $\hat{\mathsf{H}}$, G, and a signing oracle O. For the proof it will be useful that $\hat{\mathsf{H}}(\mathsf{seed}, \cdot)$ is implemented indirectly via a random oracle $\mathsf{B}\colon \{0,1\}^* \to \{0,1\}$ that is implemented lazily and only known to the game. In this way, the game will be able to distinguish queries made by $\mathcal{A}$ from queries it made itself. With this in mind, the oracles are implemented as follows:

- $\mathsf{H}(\mathsf{m})$: If $h[\mathsf{m}] = \bot$, sample $h[\mathsf{m}] \xleftarrow{\$} \mathbb{G}_1$. Return $h[\mathsf{m}]$.
- $\hat{\mathsf{H}}(\mathsf{seed}', \mathsf{m})$: If $\hat{h}[\mathsf{seed}', \mathsf{m}] = \bot$, do:
  1. If $\mathsf{seed}' = \mathsf{seed}$, set $\hat{h}[\mathsf{seed}', \mathsf{m}] := \mathsf{B}(\mathsf{m})$.

2. Else, sample $\hat{h}[\mathsf{seed}', \mathsf{m}] \xleftarrow{\$} \{0,1\}$.

Return $\hat{h}[\mathsf{seed}', \mathsf{m}]$.

- $\mathsf{G}(\mathsf{pk})$: If $g[\mathsf{pk}] = \bot$, sample $g[\mathsf{pk}] \xleftarrow{\$} \mathbb{G}_1$. Return $g[\mathsf{pk}]$.
- $\mathrm{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$, set $\beta_{\mathsf{m}} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$, return $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{\beta_{\mathsf{m}}}}$.

Finally, the adversary $\mathcal{A}$ outputs a list of public keys and a forgery. In the actual EUF-CMA game for this scheme, each such public key is a pair of BLS keys with associated proofs of possession, and the signature would contain one bit for each such pair indicating which key is used. Without loss of generality[8], we simplify the game here by assuming that $\mathcal{A}$ directly outputs a set of BLS keys with their proofs of possession. More precisely, we assume that $\mathcal{A}$ outputs a list of N pairs $(\mathsf{pk}_i^*, \pi_i^*) \in \mathbb{G}_2 \times \mathbb{G}_1$, $i \in [N]$ and a forgery $(\mathsf{m}^*, \sigma^*) \in \{0,1\}^* \times \mathbb{G}_1$. The game does the following to determine if it outputs 0 or 1:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.
2. Set $\mathcal{V}_0 := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}_0\}$ and $\mathcal{V}_1 := \{i \in [N] \setminus \mathcal{V}_0 \mid \mathsf{pk}_i^* = \mathsf{pk}_1\}$.
3. If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$, terminate with output 0.
4. If there is an $i \in [N]$ with $e\left(\mathsf{G}(\mathsf{pk}_i^*), \mathsf{pk}_i^*\right) \neq e\left(\pi_i^*, g_2\right)$, terminate with output 0.
5. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^{N} \mathsf{pk}_i^*$.
6. If $e\left(h^*, \bar{\mathsf{pk}}^*\right) \neq e\left(\sigma^*, g_2\right)$ terminate with output 0. Otherwise, terminate with output 1.

We have

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_2}(\lambda) \leq \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Game $\mathbf{G}_1$: In this game, we change the winning condition, such that the game now additionally outputs 0 if the adversary ever queried $\hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m}^*)$. More precisely, the new check to evaluate the winning condition is as follows:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.
2. If $\hat{h}[\mathsf{seed}, \mathsf{m}^*] \neq \bot$, terminate with output 0.
3. Set $\mathcal{V}_0 := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}_0\}$ and $\mathcal{V}_1 := \{i \in [N] \setminus \mathcal{V}_0 \mid \mathsf{pk}_i^* = \mathsf{pk}_1\}$.
4. If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$, terminate with output 0.
5. If there is an $i \in [N]$ with $e\left(\mathsf{G}(\mathsf{pk}_i^*), \mathsf{pk}_i^*\right) \neq e\left(\pi_i^*, g_2\right)$, terminate with output 0.
6. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^{N} \mathsf{pk}_i^*$.
7. If $e\left(h^*, \bar{\mathsf{pk}}^*\right) \neq e\left(\sigma^*, g_2\right)$ terminate with output 0. Otherwise, terminate with output 1.

Here, the highlighted line is what we added. If $\mathsf{m}^* \in \mathcal{Q}$, the change has no effect. Otherwise, if $\mathbf{G}_0$ and $\mathbf{G}_1$ differ in their output, then $\mathcal{A}$ must have queried $\hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$ for some m, concretely, for m^*. As $\mathcal{A}$ obtains no information about seed, and seed is uniform over $\{0,1\}^\lambda$, the probability that this happens is at most $1/2^\lambda$ for each fixed random oracle query. With a union bound over all random oracle queries, we get

$$\left|\Pr\left[\mathbf{G}_0 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1 \Rightarrow 1\right]\right| \leq \frac{Q_{\hat{\mathsf{H}}}}{2^\lambda}.$$

Game $\mathbf{G}_2$: We let the game sample a bit $\hat{\beta} \xleftarrow{\$} \{0,1\}$ at the beginning, and again change the winning condition. Now, it is as follows:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.

[8] A reduction can just drop the unused keys that it got from the adversary, and remove the bits from the forgery.

2. If $\hat{h}[\mathsf{seed}, \mathsf{m}^*] \neq \bot$, terminate with output 0.
3. Set $\mathcal{V}_0 := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}_0\}$ and $\mathcal{V}_1 := \{i \in [N] \setminus \mathcal{V}_0 \mid \mathsf{pk}_i^* = \mathsf{pk}_1\}$.
4. If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$, terminate with output 0.
5. Set $\beta_{\mathsf{m}^*} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m}^*)$. If $\mathcal{V}_{\hat{\beta}} = \emptyset$ or $\hat{\beta} \neq 1 - \beta_{\mathsf{m}^*}$, terminate with output 0.
6. If there is an $i \in [N]$ with $e\left(\mathsf{G}(\mathsf{pk}_i^*), \mathsf{pk}_i^*\right) \neq e\left(\pi_i^*, g_2\right)$, terminate with output 0.
7. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^{N} \mathsf{pk}_i^*$.
8. If $e\left(h^*, \bar{\mathsf{pk}}^*\right) \neq e\left(\sigma^*, g_2\right)$ terminate with output 0. Otherwise, terminate with output 1.

If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$ or $\hat{h}[\mathsf{seed}, \mathsf{m}^*] \neq \bot$, the change has no effect. Otherwise, note that $\mathcal{A}$'s view is independent of $\hat{\beta}$ and β_{m^*}. Therefore, we get

$$\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] \geq \Pr\left[1 - \beta_{\mathsf{m}^*} = \hat{\beta} \wedge \mathcal{V}_{\hat{\beta}} \neq \emptyset\right] \cdot \Pr\left[\mathbf{G}_1 \Rightarrow 1\right].$$

As $\mathcal{V}_0 \cup \mathcal{V}_1 \neq \emptyset$, the probability of the event $\mathcal{V}_{\hat{\beta}} \neq \emptyset$ is at least $1/2$. Also, even conditioned on a fixed $\hat{\beta}$, the probability of $1 - \beta_{\mathsf{m}^*} = \hat{\beta}$ is $1/2$, as $\hat{\beta}$ and β_{m^*} are independent random variables. So we get

$$\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] \geq \frac{1}{4} \cdot \Pr\left[\mathbf{G}_1 \Rightarrow 1\right].$$

Game $\mathbf{G}_3$: We change how oracle H is implemented. For this, we let the game sample an element $v \xleftarrow{\$} \mathbb{G}_1$ at the beginning. Further, the game now internally holds a random oracle $\Gamma\colon \{0,1\}^* \to \mathbb{Z}_p$, implemented lazily in the standard way, and implements H as follows:

- $\mathsf{H}(\mathsf{m})$: If $h[\mathsf{m}] = \bot$, do:
  1. If $1 - \mathsf{B}(\mathsf{m}) = \hat{\beta}$, set $h[\mathsf{m}] := v^{\Gamma(\mathsf{m})}$.
  2. Else, set $h[\mathsf{m}] := {g_1}^{\Gamma(\mathsf{m})}$.

  Return $h[\mathsf{m}]$.

Assuming $v \neq 1 \in \mathbb{G}_1$, the outputs of H are still uniform and independent of the rest of the game and each other. The probability that $v = 1$ is at most $1/p$, so that we get[9]

$$\left|\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_3 \Rightarrow 1\right]\right| \leq \frac{1}{p}.$$

Game $\mathbf{G}_4$: We change how the signing oracle O is implemented. After this change, the secret key $\mathsf{sk}_{\hat{\beta}}$ will no longer be used by the signing oracle:

- $\mathrm{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$, set $\beta_{\mathsf{m}} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$, and do:
  1. If $\beta_{\mathsf{m}} = \hat{\beta}$, return $\sigma := \tilde{\mathsf{pk}}_{\hat{\beta}}^{\Gamma(\mathsf{m})}$.
  2. Else, return $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{1-\beta}}$.

Clearly, we did not change the signing oracle for the case that $\beta_{\mathsf{m}} = 1 - \hat{\beta}$. For the other case, i.e., $\beta_{\mathsf{m}} = \hat{\beta}$, we have

$$\mathsf{H}(\mathsf{m})^{\mathsf{sk}_{\beta_{\mathsf{m}}}} = \left({g_1}^{\Gamma(\mathsf{m})}\right)^{\mathsf{sk}_{\beta_{\mathsf{m}}}} = g_1^{\Gamma(\mathsf{m}) \cdot \mathsf{sk}_{\hat{\beta}}} = \tilde{\mathsf{pk}}_{\hat{\beta}}^{\Gamma(\mathsf{m})}.$$

Therefore, we get

$$\Pr\left[\mathbf{G}_3 \Rightarrow 1\right] = \Pr\left[\mathbf{G}_4 \Rightarrow 1\right].$$

Game $\mathbf{G}_5$: We change how the game computes $\pi_{\hat{\beta}}$. Namely, from now on, the initial steps of the game are:

[9] For the interested reader, this is where we use the indirection given by implementing $\hat{\mathsf{H}}(\mathsf{seed}, \cdot)$ via B. If we would have queried $\hat{\mathsf{H}}$ in the implementation of H, then $\mathcal{A}$ could easily make the output of $\mathbf{G}_2$ and $\mathbf{G}_3$ differ by querying $\mathsf{H}(\mathsf{m}^*)$.

1. Sample $\hat{\beta} \xleftarrow{\$} \{0,1\}$ and $v \xleftarrow{\$} \mathbb{G}_1$ and set $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
2. Set $\mathcal{Q} := \emptyset$ and initialize empty maps $h[\cdot], \hat{h}[\cdot]$, and $g[\cdot]$.
3. Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{sk}_0, \mathsf{sk}_1 \xleftarrow{\$} \mathbb{Z}_p$.
4. Set $\mathsf{pk}_0 := g_2^{\mathsf{sk}_2}$, $\mathsf{pk}_1 := g_2^{\mathsf{sk}_1}$ and $\tilde{\mathsf{pk}}_0 := g_1^{\mathsf{sk}_0}$, $\tilde{\mathsf{pk}}_1 := g_1^{\mathsf{sk}_1}$.
5. Sample $\delta \xleftarrow{\$} \mathbb{Z}_p$ and set $g[\mathsf{pk}_{\hat{\beta}}] := g_1^{\delta}$.
6. Set $\pi_{1-\hat{\beta}} := \mathsf{G}(\mathsf{pk}_{1-\hat{\beta}})^{\mathsf{sk}_{1-\hat{\beta}}}$ and $\pi_{\hat{\beta}} := \tilde{\mathsf{pk}}_{\hat{\beta}}^{\delta}$.
7. Set $\mathsf{pk} := ((\mathsf{pk}_0, \pi_0), (\mathsf{pk}_1, \pi_1))$

It is easy to see that the distribution of the outputs of G remains unchanged, as δ is sampled uniformly. Also, the distribution of $\pi_{\hat{\beta}}$ is not changed because

$$\mathsf{G}(\mathsf{pk}_{\hat{\beta}})^{\mathsf{sk}_{\hat{\beta}}} = g_1^{\delta \mathsf{sk}_{\hat{\beta}}} = \tilde{\mathsf{pk}}_{\hat{\beta}}^{\delta}.$$

We get

$$\Pr[\mathbf{G}_4 \Rightarrow 1] = \Pr[\mathbf{G}_5 \Rightarrow 1].$$

Note that the game can now be simulated without ever using $\mathsf{sk}_{\hat{\beta}}$, assuming $\mathsf{pk}_{\hat{\beta}}$ and $\tilde{\mathsf{pk}}_{\hat{\beta}}$ are given.

Game $\mathbf{G}_6$: We change oracle G. Namely, the game now internally holds a random oracle $\Delta\colon \{0,1\}^* \to \mathbb{Z}_p$, implemented lazily in the standard way, and implements G as follows:

- $\mathsf{G}(\mathsf{pk})$: If $g[\mathsf{pk}] = \perp$, set $g[\mathsf{pk}] := v^{\Delta(\mathsf{pk})}$. Return $g[\mathsf{pk}]$.

If $v \neq 1 \in \mathbb{G}_1$, the distribution remains unchanged, and the probability that $v = 1$ is at most $1/p$, so

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{1}{p}.$$

Final Reduction: Before we give the final reduction breaking CDH, we examine the forgery signature σ^* after the changes we have made and provide intuition how the reduction can solve CDH. To this end, assume $\mathbf{G}_6$ outputs 1. For ease of notation, set $x := \mathsf{sk}_{\hat{\beta}}$ and let $y \in \mathbb{Z}_p$ be such that $v = g_1^y$. We assume that a reduction is given g_1^x, g_2^x, and g_1^y, and its goal is to output g_1^{xy}. By the changes we have made, such a reduction never explicitly needs x or y over $\mathbb{Z}_p$ to simulate $\mathbf{G}_6$. Recall that the final forgery (m^*, σ^*) of the adversary comes with a list of pairs $(\mathsf{pk}_i^*, \pi_i^*)$ for $i \in [N]$. Denote by $\mathcal{C} \subseteq [N]$ the set of indices i such that $\mathsf{pk}_i^* \neq \mathsf{pk}_{\hat{\beta}}$. Intuitively, this corresponds to the set of indices for which the list contains adversarially chosen public keys. For each $i \in \mathcal{C}$, let $\mathsf{sk}_i^* \in \mathbb{Z}_p$ be such that $\mathsf{pk}_i^* = g_2^{\mathsf{sk}_i^*}$. We set $\ell := |[N] \setminus \mathcal{C}|$. Due to the change in $\mathbf{G}_2$, we know that $\ell \neq 0$. By the verification equation, we know that

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\bar{\mathsf{sk}}^*} \text{ for } \bar{\mathsf{sk}}^* = \ell x + \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*.$$

By definition of $\mathsf{H}(\mathsf{m}^*)$ (see $\mathbf{G}_3$) and recalling that $1 - \beta_{\mathsf{m}^*} = \hat{\beta}$ (see $\mathbf{G}_2$), we know that the discrete logarithm of σ^* with respect to g_1 is

$$\Gamma(\mathsf{m}^*) \cdot y \cdot \bar{\mathsf{sk}}^* = \Gamma(\mathsf{m}^*) \cdot y \cdot \left(\ell x + \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*\right) = \Gamma(\mathsf{m}^*) \cdot \left(\ell x y + y \cdot \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*\right).$$

Rearranging and taking to the exponent yields

$$g_1^{xy} = \left(\sigma^{*1/\Gamma(\mathsf{m}^*)} \cdot \prod_{i \in \mathcal{C}} g_1^{-y \mathsf{sk}_i^*}\right)^{1/\ell}, \tag{1}$$

assuming $\Gamma(\mathsf{m}^*) \neq 0$ for now. Further, for every $i \in \mathcal{C}$, we get from the definition of G (see $\mathbf{G}_6$), from $v = g_1^y$, and from the verification equation that

$$\pi_i^* = \mathsf{G}(\mathsf{pk}_i^*)^{\mathsf{sk}_i^*} = g_1^{y\Delta(\mathsf{pk}_i^*)\mathsf{sk}_i^*}. \tag{2}$$

Assuming $\Delta(\mathsf{pk}_i^*) \neq 0$, rearranging Equation (2) and plugging it into Equation (1), we get

$$g_1^{xy} = \left(\sigma^{*1/\Gamma(\mathsf{m}^*)} \cdot \prod_{i\in\mathcal{C}} \pi_i^{*-1/\Delta(\mathsf{pk}_i^*)}\right)^{1/\ell}. \tag{3}$$

Now our main observation is that the right-hand side of Equation (3) can efficiently be computed. We now turn this into a reduction $\mathcal{B}$ solving the CDH problem if $\mathbf{G}_6$ outputs 1 and assuming $\Gamma(\mathsf{m}^*) \neq 0$ and $\Delta(\mathsf{pk}_i^*) \neq 0$:

1. The reduction $\mathcal{B}$ gets as input parameters $\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e$ and elements $X_1 = g_1^x$, $X_2 = g_2^x$, and $Y = g_1^y$. Its goal is to output g_1^{xy}.

2. The reduction sets $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$, samples $\hat{\beta} \xleftarrow{\$} \{0,1\}$ as in $\mathbf{G}_6$, and defines $\mathsf{pk}_{\hat{\beta}} := X_2$, $\mathsf{pk}_{\hat{\beta}} := X_1$, and $v := Y$. It sets up the remaining parameters and then simulates $\mathbf{G}_6$ for $\mathcal{A}$, which is possible efficiently without the knowledge of x and y.

3. When $\mathcal{A}$ outputs a forgery and $\mathbf{G}_6$ would output 1, the reduction aborts if $\Gamma(\mathsf{m}^*) = 0$ or $\Delta(\mathsf{pk}_i^*) = 0$ for any $i \in \mathcal{C}$. Otherwise, it computes g_1^{xy} as in Equation (3) and outputs it.

We see that $\mathcal{B}$ perfectly simulates game $\mathbf{G}_6$ for $\mathcal{A}$ and the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. The probability that $\mathcal{B}$ has to abort before using Equation (3) is at most $(Q_\mathsf{H} + Q_\mathsf{G})/p$. By the discussion above, we get

$$\Pr[\mathbf{G}_6 \Rightarrow 1] \leq \frac{Q_\mathsf{H} + Q_\mathsf{G}}{p} + \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

□

Corollary 1. *Consider the digital signature scheme obtained by fixing $N = 1$ signer in the multi-signature scheme BLSMS_2. If the CDH assumption holds relative to PGGen, then this scheme is $\mathsf{EUF\text{-}CMA}$ secure, with a tight proof.*

5 Application: PoS Blockchains with Opt-In Tightness

We anticipate that the insights presented in this paper will prove valuable in the context of proof-of-stake (PoS) blockchain systems utilizing BLS multi-signatures, such as Ethereum [Edg23]. Within this domain, the interoperability of our construction with regular BLS (see Remark 5) makes it particularly advantageous.

PoS Blockchains and Multi-Signatures. Let us begin by revisiting, in simplified terms, the relevant aspects of a proof-of-stake blockchain. In such a system, participants can lock (aka stake) a designated quantity of coins and publicly declare a BLS public key to register as *validators*. When a block is proposed, it has to be attested by enough validators to be deemed valid. These attestations consist of BLS signatures of the block with respect to the validators' keys. What ends up on chain is the combined BLS multi-signature.

Our Multi-Signatures for Opt-In Tightness. Now envision a system in which each validator can register a (small) number of BLS keys, such that signing with one of these keys means the validator attested the block. A natural motivation to use this setup is to ensure that validators can continue functioning even if access to some keys is lost. We argue that such a system allows for opt-in tightness without mandating a departure from BLS: Namely, consider Alice taking the role of a validator. As she prioritizes concrete security, she would register two BLS keys. Whenever she had to sign a block, she would pseudorandomly decide which key to use, thereby implementing our scheme. Note that she can do so by implementing minimal logic on top of existing BLS implementations. Conversely, if Bob prefers

to adhere to regular BLS signatures, he can simply register a single key as usual. Importantly, Alice's signatures and Bob's remain compatible and can be aggregated seamlessly, even without awareness of Alice's adoption of our scheme. Thus, each validator retains the autonomy to *independently* select their preferred variant.

We remark that while some proof-of-stake chains indeed allow multiple BLS keys [Blo22] and can directly implement the setting above, Ethereum currently does not[10]. Consequently, our research can be viewed as an argument for the integration of such functionality in the future.

References

[AB21] Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 4.)

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005. (Cited on page 4, 5.)

[BCG+23] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Francois Garillot, Jonas Lindstrom, Ben Riva, Arnab Roy, Mahdi Sedaghat, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. Subset-optimized bls multi-signature with key aggregation. Cryptology ePrint Archive, Paper 2023/498, 2023. https://eprint.iacr.org/2023/498. (Cited on page 5, 6.)

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008. (Cited on page 4.)

[BD21] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 650–678. Springer, Heidelberg, December 2021. (Cited on page 4.)

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2018. (Cited on page 3, 4, 5.)

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003. (Cited on page 5.)

[BGW+22] Dan Boneh, Sergey Gorbunov, Riad S. Wahby, Hoeteck Wee, Christopher A. Wood, and Zhenfei Zhang. BLS Signatures. Internet-Draft draft-irtf-cfrg-bls-signature-05, Internet Engineering Task Force, June 2022. Work in Progress. (Cited on page 6.)

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Heidelberg, May 2016. (Cited on page 3, 6, 7.)

[BL22] Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 193–207. ACM Press, November 2022. (Cited on page 3.)

[10] One potential workaround would involve registering multiple validators. However, this approach necessitates locking twice the amount of funds as previously required, underscoring the potential benefits of a native support for multiple keys.

[BLL+23] Erica Blum, Derek Leung, Julian Loss, Jonathan Katz, and Tal Rabin. Analyzing the real-world security of the algorand blockchain. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 830–844. ACM Press, November 2023. (Cited on page 3.)

[Blo22] Harmony Blockchain. Harmony – Creating A Validator. https://docs.harmony.one/home/network/validators/creating-a-validator, 2022. Accessed: 2024-05-07. (Cited on page 18.)

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001. (Cited on page 3, 6, 10.)

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. (Cited on page 4.)

[BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Heidelberg, July 2007. (Cited on page 3, 4, 5.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003. (Cited on page 3, 4, 5.)

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 276–305. Springer, Heidelberg, August 2022. (Cited on page 4.)

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021. https://eprint.iacr.org/2021/1375. (Cited on page 4.)

[con22] Chia contributors. Chia network: Implementation of bls signatures. GitHub repository, November 2022. The green cryptocurrency with Chialisp. (Cited on page 3.)

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000. (Cited on page 10.)

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Heidelberg, April / May 2002. (Cited on page 3, 6, 7.)

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019. (Cited on page 4.)

[DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2093–2110. USENIX Association, August 2020. (Cited on page 4, 5.)

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Heidelberg, May 2021. (Cited on page 4.)

[Edg23] Ben Edgington. *Upgrading Ethereum - A technical handbook on Ethereum's move to proof of stake and beyond*. Edition 0.3: Capella [wip] edition, 2023. (Cited on page 3, 6, 8, 11, 17.)

[FH20] Masayuki Fukumitsu and Shingo Hasegawa. A tightly secure ddh-based multisignature with public-key aggregation. In *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 321–327, 2020. (Cited on page 4.)

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. (Cited on page 3, 6.)

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery. (Cited on page 3.)

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007. (Cited on page 3, 5, 6.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 9.)

[GPS06] S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. https://eprint.iacr.org/2006/165. (Cited on page 12.)

[IN83] Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983. (Cited on page 4.)

[Inc24] Chia Network Inc. Chialisp primer: 5. bls signatures, 2024. (Cited on page 3.)

[KK12] Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 537–553. Springer, Heidelberg, April 2012. (Cited on page 3.)

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003. (Cited on page 3, 6.)

[Lan96] Susan K. Langford. Weakness in some threshold cryptosystems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 74–82. Springer, Heidelberg, August 1996. (Cited on page 4.)

[LHL95] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 194–204. Springer, Heidelberg, May 1995. (Cited on page 4.)

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Heidelberg, May / June 2006. (Cited on page 4, 5.)

[MH96] Markus Michels and Patrick Horster. On the risk of disruption in several multiparty signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, pages 334–345, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. (Cited on page 4.)

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 245–254. ACM Press, November 2001. (Cited on page 4.)

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87:2139 – 2164, 2019. (Cited on page 4.)

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 4.)

[NRSW20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020. (Cited on page 4.)

[OO93] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *ASIACRYPT'91*, volume 739 of *LNCS*, pages 139–148. Springer, Heidelberg, November 1993. (Cited on page 4.)

[Org20] Drand Organization. Drand - a distributed randomness beacon daemon. GitHub repository, 2020. (Cited on page 3.)

[PW23] Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 597–627. Springer, Heidelberg, April 2023. (Cited on page 4, 12.)

[PW24] Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 460–489, Zurich, Switherland, May 26–30, 2024. Springer, Heidelberg. (Cited on page 4, 12.)

[QLH12] Haifeng Qian, Xiangxue Li, and Xinli Huang. Tightly secure non-interactive multisignatures in the plain public key model. *Informatica (Vilnius)*, 3, 01 2012. (Cited on page 3, 4, 5.)

[QX10] Haifeng Qian and Shouhuai Xu. Non-interactive multisignatures in the plain public-key model with efficient verification. *Information Processing Letters*, 111(2):82–89, 2010. (Cited on page 4, 5.)

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 228–245. Springer, Heidelberg, May 2007. (Cited on page 4, 5, 8, 12.)

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 628–658. Springer, Heidelberg, April 2023. (Cited on page 4.)

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005. (Cited on page 5.)

Appendix

A Postponed Security Proofs

Proof of Theorem 1. Essentially, the proof is a simplification of the proof of Theorem 2, and readers familiar with this proof can simply note that the main difference is that signatures are simulated via a guessing argument. On the other hand, readers familiar with the standard proof for BLS signatures may note that games $\mathbf{G}_0$ to $\mathbf{G}_3$ are used to simulate signing as in the standard proof. These readers may especially be interested in the steps starting from $\mathbf{G}_4$, which deal with extracting a solution from a combined signature. This part differs from standard BLS signatures and uses the techniques developed in this paper.

Game $\mathbf{G}_0$: We start with $\mathbf{G}_0$, which is the $\mathsf{EUF\text{-}CMA}$ game for BLSMS_1 and adversary $\mathcal{A}$. To fix notation, we recall this game here. Note that the random oracle $\hat{\mathsf{H}}$ can be omitted, see Remark 3. Initially, the game generates parameters and a pair of keys, and sets up maps to simulate random oracles. This is done as follows:

1. Set $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
2. Set $\mathcal{Q} := \emptyset$ and initialize empty maps $h[\cdot]$ and $g[\cdot]$.
3. Sample $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{pk} := g_2^{\mathsf{sk}}$ and $\tilde{\mathsf{pk}} := g_1^{\mathsf{sk}}$.
4. Set $\pi := \mathsf{G}(\mathsf{pk})^{\mathsf{sk}}$.

The key $\tilde{\mathsf{pk}}$ will be used in the following games. It gives par and (pk, π) to the adversary $\mathcal{A}$. In addition, $\mathcal{A}$ gets access to random oracles H, G, and a signing oracle O, implemented by the game as follows:

- $\mathsf{H}(\mathsf{m})$: If $h[\mathsf{m}] = \bot$, sample $h[\mathsf{m}] \xleftarrow{\$} \mathbb{G}_1$. Return $h[\mathsf{m}]$.
- $\mathsf{G}(\mathsf{pk}')$: If $g[\mathsf{pk}'] = \bot$, sample $g[\mathsf{pk}'] \xleftarrow{\$} \mathbb{G}_1$. Return $g[\mathsf{pk}']$.
- $\mathrm{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$ and return $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}}$.

When $\mathcal{A}$ terminates, it outputs a list of public keys and a forgery. More precisely, $\mathcal{A}$ outputs a list of N pairs $(\mathsf{pk}_i^*, \pi_i^*) \in \mathbb{G}_2 \times \mathbb{G}_1$, $i \in [N]$ and a forgery $(\mathsf{m}^*, \sigma^*) \in \{0,1\}^* \times \mathbb{G}_1$. The game does the following to determine its output:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.
2. Set $\mathcal{V} := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}\}$. If $\mathcal{V} = \emptyset$, terminate with output 0.
3. If there is an $i \in [N]$ with $e\left(\mathsf{G}(\mathsf{pk}_i^*), \mathsf{pk}_i^*\right) \neq e\left(\pi_i^*, g_2\right)$, terminate with output 0.
4. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^N \mathsf{pk}_i^*$.
5. If $e\left(h^*, \bar{\mathsf{pk}}^*\right) \neq e\left(\sigma^*, g_2\right)$ terminate with output 0. Otherwise, terminate with output 1.

By definition, we have

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_1}(\lambda) = \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Game $\mathbf{G}_1$: Starting from $\mathbf{G}_1$, the game internally holds a function $\mathsf{B}\colon \{0,1\}^* \to \{0,1\}$ such that each $\mathsf{B}(\mathsf{m})$ is set to 1 independently with probability $1/(Q_S + 1)$. The game can efficiently implement this function in a standard lazy way. We further change the signing oracle O as follows:

- $\mathrm{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$. If $\mathsf{B}(\mathsf{m}) = 1$, terminate with output 0. Otherwise, return $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}}$.

Also, we change how the game evaluates the winning condition:

1. If $\mathsf{m}^* \in \mathcal{Q}$ or $\mathsf{B}(\mathsf{m}^*) = 0$, terminate with output 0.
2. Set $\mathcal{V} := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}\}$. If $\mathcal{V} = \emptyset$, terminate with output 0.

3. If there is an $i \in [N]$ with $e\left(\mathsf{G}(\mathsf{pk}_i^*), \mathsf{pk}_i^*\right) \neq e\left(\pi_i^*, g_2\right)$, terminate with output 0.
4. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^{N} \mathsf{pk}_i^*$.
5. If $e\left(h^*, \bar{\mathsf{pk}}^*\right) \neq e\left(\sigma^*, g_2\right)$ terminate with output 0. Otherwise, terminate with output 1.

In other words, the game aborts if for any signing query m we have $\mathsf{B}(\mathsf{m}) = 1$ or for the forgery m^* we have $\mathsf{B}(\mathsf{m}^*) = 0$. As $\mathcal{A}$'s view is independent of B, we get

$$\begin{aligned}\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] &\geq \frac{1}{Q_S+1}\left(1-\frac{1}{Q_S+1}\right)^{Q_S} \cdot \Pr\left[\mathbf{G}_0 \Rightarrow 1\right] \\ &= \frac{1}{Q_S}\left(1-\frac{1}{Q_S+1}\right)^{Q_S+1} \cdot \Pr\left[\mathbf{G}_0 \Rightarrow 1\right] \geq \frac{1}{4Q_S} \cdot \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].\end{aligned}$$

via a standard calculation and the fact that $(1-1/x)^x \geq 1/4$ for all[11] $x \geq 2$.

Game $\mathbf{G}_2$: Let $Y \xleftarrow{\$} \mathbb{G}_1$ be a random group element that $\mathbf{G}_2$ samples when it starts, and let $\Gamma\colon \{0,1\}^* \to \mathbb{Z}_p$ be a random oracle that $\mathbf{G}_2$ holds internally, implemented lazily. The game now simulates random oracle H differently:

- $\mathsf{H}(\mathsf{m})$: If $h[\mathsf{m}] = \perp$, set $h[\mathsf{m}] := Y^{\Gamma(\mathsf{m})}$ if $\mathsf{B}(\mathsf{m}) = 1$ and $h[\mathsf{m}] := g_1^{\Gamma(\mathsf{m})}$ if $\mathsf{B}(\mathsf{m}) = 0$. Return $h[\mathsf{m}]$.

Clearly, unless $Y = g_1^0$, which happens with probability $1/p$, the adversary's view does not change. Hence we get

$$\left|\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2 \Rightarrow 1\right]\right| \leq \frac{1}{p}.$$

Game $\mathbf{G}_3$: We change how the game implements the signing oracle:

- $\mathrm{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$. If $\mathsf{B}(\mathsf{m}) = 1$, terminate with output 0. Otherwise, return $\sigma := \tilde{\mathsf{pk}}^{\Gamma(\mathsf{m})}$.

Note that in $\mathbf{G}_2$, if O outputs a signature σ on a message m, then $\mathsf{B}(\mathsf{m}) = 1$, and therefore

$$\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}} = \left(g_1^{\Gamma(\mathsf{m})}\right)^{\mathsf{sk}} = \tilde{\mathsf{pk}}^{\Gamma(\mathsf{m})}.$$

Therefore, the view in $\mathbf{G}_3$ is the same, and we get

$$\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] = \Pr\left[\mathbf{G}_3 \Rightarrow 1\right].$$

Note that we have now removed the secret key sk from the signing oracle, and it is only used for computing the proof of possesion π. In the next game, we eliminate this use as well.

Game $\mathbf{G}_4$: We change how π is computed in the initil setup of the game:

1. Set $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
2. Set $\mathcal{Q} := \emptyset$ and initialize empty maps $h[\cdot]$ and $g[\cdot]$.
3. Sample $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{pk} := g_2^{\mathsf{sk}}$ and $\tilde{\mathsf{pk}} := g_1^{\mathsf{sk}}$.
4. Sample $\delta \xleftarrow{\$} \mathbb{Z}_p$ and set $g[\mathsf{pk}] := g_1^{\delta}$ and $\pi := \tilde{\mathsf{pk}}^{\delta}$.

The distribution of the random oracle outputs of G remain unchanged because δ is sampled uniformly at random. The distribution of π also did not change, because $\mathsf{G}(\mathsf{pk})^{\mathsf{sk}} = g_1^{\delta \mathsf{sk}} = \tilde{\mathsf{pk}}^{\delta}$. Hence, we get

$$\Pr\left[\mathbf{G}_3 \Rightarrow 1\right] = \Pr\left[\mathbf{G}_4 \Rightarrow 1\right].$$

Game $\mathbf{G}_5$: We change how the random oracle is implemented G. To this end, recall the definition of element $Y \in \mathbb{G}_1$ from $\mathbf{G}_2$ and let $\Delta\colon \{0,1\}^* \to \mathbb{Z}_p$ be a random oracle that the game implements lazily and keeps to itself. The random oracle G is now implemented as follows:

[11] Without loss of generality, we can assume $\mathcal{A}$ makes at least one signing query.

1. $\mathsf{G}(\mathsf{pk}')$: If $g[\mathsf{pk}'] = \perp$, set $g[\mathsf{pk}'] := Y^{\Delta(\mathsf{pk}')}$. Return $g[\mathsf{pk}']$.

Note that as long as $Y \neq g_1^0$, the distribution did not change. The probability that $Y = g_1^0$ is at most $1/p$, and so we get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{1}{p}.$$

Final Reduction: Recall that when $\mathcal{A}$ terminates, it outputs a list of pairs $(\mathsf{pk}_i^*, \pi_i^*)$ and a pair (m^*, σ^*). Let us now examine the structure of this forgery, assuming that $\mathbf{G}_5$ outputs 1. We use the notation $x := \mathsf{sk}$ and let $y \in \mathbb{Z}_p$ be such that $Y = g_1^y$. Denote by $\mathcal{C} \subseteq [N]$ the set of indices i such that $\mathsf{pk}_i^* \neq \mathsf{pk}$, i.e., $\mathcal{C} := [N] \setminus \mathcal{V}$. Further, for each $i \in \mathcal{C}$, let $\mathsf{sk}_i^* \in \mathbb{Z}_p$ be such that $\mathsf{pk}_i^* = g_2^{\mathsf{sk}_i^*}$. Also, set $\ell := |\mathcal{V}|$. We know that if $\mathbf{G}_5$ outputs 1, then $\ell > 0$. With this notation, the verification equation implies that By the verification equation, we know that

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\bar{\mathsf{sk}}^*} \text{ for } \bar{\mathsf{sk}}^* = \ell \cdot x + \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*.$$

Now, we know that $\mathsf{B}(\mathsf{m}^*) = 1$ (cf. $\mathbf{G}_1$) and therefore

$$\mathsf{H}(\mathsf{m}^*) = Y^{\Gamma(\mathsf{m}^*)} = g_1^{y \cdot \Gamma(\mathsf{m}^*)}.$$

In combination, we get that the discrete logarithm of σ^* with respect to g_1 is

$$y \cdot \Gamma(\mathsf{m}^*) \cdot \bar{\mathsf{sk}}^* = \Gamma(\mathsf{m}^*) \cdot \left(\ell x y + y \cdot \sum_{i \in \mathcal{C}} \mathsf{sk}_i^* \right).$$

By rearranging and assuming $\Gamma(\mathsf{m}^*) \neq 0$, we obtain

$$g_1^{xy} = \left({\sigma^*}^{1/\Gamma(\mathsf{m}^*)} \cdot \prod_{i \in \mathcal{C}} g_1^{-y\mathsf{sk}_i^*} \right)^{1/\ell}. \tag{4}$$

For every $i \in \mathcal{C}$, we get (cf. $\mathbf{G}_5$) and the verification equation that

$$\pi_i^* = \mathsf{G}(\mathsf{pk}_i^*)^{\mathsf{sk}_i^*} = g_1^{\Delta(\mathsf{pk}_i^*) y \mathsf{sk}_i^*}. \tag{5}$$

Assuming $\Delta(\mathsf{pk}_i^*) \neq 0$, rearranging Equation (5) and plugging it into Equation (4), we get

$$g_1^{xy} = \left({\sigma^*}^{1/\Gamma(\mathsf{m}^*)} \cdot \prod_{i \in \mathcal{C}} {\pi_i^*}^{-1/\Delta(\mathsf{pk}_i^*)} \right)^{1/\ell}. \tag{6}$$

Now, observe that we can bound the probability that $\Gamma(\mathsf{m}^*) = 0$ or $\Delta(\mathsf{pk}_i^*) = 0$ for some $i \in [N]$ by $(Q_\mathsf{H} + Q_\mathsf{G})/p$. Assuming this does not happen, we now observe that a reduction that gets as input $X_1 = g_1^x$, $X_2 := g_2^x$, and $Y = g_1^y$ can efficiently simulate $\mathbf{G}_5$ with $\mathsf{pk} := X_2$ and $\tilde{\mathsf{pk}} := X_1$ for $\mathcal{A}$. It can then use Equation (6) to efficiently compute g_1^{xy} and solve CDH. We omit the details of the reduction, but refer the reader to the proof of Theorem 2 for a similar reduction. We get

$$\Pr[\mathbf{G}_5 \Rightarrow 1] \leq \frac{Q_\mathsf{H} + Q_\mathsf{G}}{p} + \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

□