On Key Substitution Attacks against Aggregate Signatures and Multi-Signatures

Yuuki Fujita[1], Yusuke Sakai[2], Kyosuke Yamashita[1,2], and Goichiro Hanaoka[2]

[1] Osaka University
[2] National Institute of Advanced Industrial Science and Technology

Abstract. When we use signature schemes in practice, we sometimes should consider security beyond unforgeability. This paper considers security against key substitution attacks of multi-signer signatures (i.e., aggregate signatures and multi-signatures). Intuitively, this security property ensures that a malicious party cannot claim the ownership of a signature that is created by an honest signer. We investigate security against key substitution attacks of a wide range of aggregate signature schemes and multi-signature schemes: the Boneh-Gentry-Lynn-Shacham aggregate signature scheme, the sequential aggregate signature scheme by Lysyanskaya et al., the multi-signature scheme by Bellare and Neven, MuSig2, and the ordered multi-signature scheme by Boldyreva et al. Furthermore, if the scheme does not provide security against key substitution attacks, then we modify the scheme to become secure against the attacks.

1 Introduction

The most standard security requirement for signature schemes is unforgeability, which guarantees it is infeasible to forge a signature without the corresponding secret key. When proposing signature schemes theoretically, it is sufficient to satisfy unforgeability in many cases.

However, if we want to use those schemes in practice, we should sometimes pay attention to properties other than unforgeability. In particular, the properties that ensure the ownership of a signature have garnered significant attention in recent years: security against key substitution attacks [5, 14] (a.k.a. duplicate-signature key selection attack [4]), exclusive ownership [10, 18], and claimability [17, 21]. Intuitively, these properties mean that no malicious party can claim the ownership of a signature that an honest one creates. If these properties are violated, it becomes difficult for third parties (i.e., verifiers) to confirm who created the signature, compromising the most fundamental requirement for signature schemes.

Schemes not having security against key substitution attacks sometimes yield real-world attacks: on X.509 Mutual Authentication used in secure SOAP services [11] and an early version of Let's Encrypt's ACME protocol [19]. These applications are widely used in the real world. Therefore, investigating security

against key substitution attacks of signature schemes is meaningful not only from a theoretical perspective but also for practical applications.

If we want to use multi-signer schemes to improve the security of the above applications, the security against key substitution attacks for multi-signer schemes should be considered. In server authentication with Let's Encrypt, a signature is created on a server's public key. There might be the case that we let the certification be distributed to avoid the single point of failure, and multi-signer signature schemes can be used in such a scenario. Therefore, in this paper, we focus on security against key substitution attacks of aggregate signature schemes and multi-signature schemes.

1.1 Our Contribution

We investigate security against key substitution attacks of a wide range of aggregate signature schemes and multi-signer signature schemes, the Boneh-Gentry-Lynn-Shacham aggregate signature scheme [7] (BGLS), the sequential aggregate signature scheme from trapdoor permutation by Lysyanskaya et al. [13] (SAfromTDP for short), the multi-signature scheme by Bellare and Neven [3] (MS-BN), MuSig2 [16], and the ordered multi-signature scheme by Boldyreva et al. [6] (BGOY). All of them are among the most representative multi-signer schemes in theory, practice, or both.

We define security against key substitution attacks, named non-key substitutability (NKS), for each of the above schemes as they have different syntax and security definitions. We additionally introduce a weak variant of the security, named weak non-key substitutability (wNKS), if necessary. Then, we investigate if each scheme satisfies the security or not. If it does not, we modify the scheme so that it satisfies (w)NKS.

Table 1 summarizes our results. If the original scheme satisfies NKS, then we are done (SAfromTDP). If it only satisfies wNKS, then we upgrade the scheme to gain NKS (MS-BN and MuSig2). Otherwise, we modify the scheme to achieve (w)NKS (BGLS and BGOY).

Table 1. Summary of our results. "NKS" (resp., "wNKS") means that it satisfies NKS (resp., wNKS). "No" means that it does not satisfy even wNKS. "-" means that it is not necessary to modify the scheme.

Scheme	Original	Modified
BGLS [7]	No	NKS
SAfromTDP [13]	NKS	-
MS-BN [3]	wNKS	NKS
MuSig2 [16]	wNKS	NKS
BGOY [6]	No	wNKS

Some of our countermeasures against key substitution attacks highlight another usefulness of the protections against other attacks already implemented

in deployed implementations of the BGLS signatures and MuSig2. As explained in Section 1.4, our key-substitution attacks against these schemes are already avoided by forbidding the trivial identity-element public key (in the BGLS signatures and MuSig2) and also by forbidding repeated messages (in the BGLS signatures). While to the best of our knowledge, these protections were not implemented to avoid key-substitution attacks, these protections also manage to prevent key-substitution attacks. This fact highlights another importance of implementing these protections.

1.2 Overview of the Definition of NKS and wNKS

Here, we briefly explain the definition of security against (weak) key substitution attacks. We note that, while we define security against key substitution attacks for each scheme, the underlying concept remains the same.

We first explain wNKS. Given a set of public keys $\{\mathsf{pk}_i^*\}_i$, the adversary first chooses messages $\{m_i^*\}_i$ to be signed on by the secret keys that correspond to $\{\mathsf{pk}_i^*\}_i$. Given the signatures, the adversary is asked to output a tuple of messages and public keys, say $(m_i^{**}, \mathsf{pk}_i^{**})_i$, that differs from the aforementioned messages and public keys as a multiset but verifies with respect to the given aggregate signature. This definition captures the scenario where a malicious party sees an aggregate signature of other parties and claims that it was created by the malicious party.

Whereas the practical scenario is already captured by wNKS, we consider stronger security, NKS. In the experiment of NKS, the adversary is allowed to choose $(m_i^*, \mathsf{pk}_i^*)_i$ and an aggregate signature in addition to $(m_i^{**}, \mathsf{pk}_i^{**})_i$, and it wins if the aggregate signature is valid under both tuples. NKS ensures there is theoretically no room for performing key substitution attacks.

1.3 Related Work

Menezes and Smart [14] proposed key substitution attacks against signature schemes. They proved that existing signature schemes, such as the Schnorr signature scheme and ECDSA, are secure against key substitution attacks. Later, Bohli et al. [5] demonstrated that several signature schemes are insecure against key substitution attacks in a different setting. To the best of our knowledge, while key substitution attacks are proposed in [14], it is Bohli et al. [5] who formalized it. Further, it is known that key substitution attacks are possible on Ed25519 which is used in many applications such as TLS1.3 and ssh [9].

An et al. [2] considered key substitution attacks against lattice-based signature schemes. Namely, they demonstrated key substitution attacks against GPV, Lyubashevsky's signature scheme, and BLISS, and showed how to fix them.

Cremers et al. [10] claimed that three properties should be considered for signature schemes in practice beyond unforgeability; exclusive ownership, message-bound, and non re-signability. We note that security against key substitution

attacks is a generalization of exclusive ownership. While we only focus on security against key substitution attacks in this paper, we might be able to consider message-bound and non re-signability of multi-signer signature schemes.

Sakai et al. [20] pointed out that several group signature schemes are insecure against signature hijacking, which is similar to key substitution attacks. They formalized the notion of opening soundness, and demonstrated that some of the dynamic group signature schemes are secure against the attack, but several schemes should be fixed.

1.4 Ethical Consideration

We assess the effect of some of the deployed implementations of the cryptographic scheme which we found not satisfying our key substitution security. We found that (1) the BGLS aggregate signature does not satisfy weak non-key substitution security, (2) the MS-BS multi-signature scheme does not satisfy non-key substitution security, (3) the MuSig2 multi-signature scheme does not satisfy non-key substitution security, and (4) the BGOY ordered multi-signature scheme does not satisfy weak non-key substitution security. We describe how these facts affect some of the deployed implementations.

While the Internet Draft draft-irtf-cfrg-bls-signature-05 [8] defines BGLS aggregate signature algorithms, these algorithms are not affected by our algorithms for key substitution attacks. Our first algorithm uses the trivial public key of the identity element. This trivial public key is not allowed in the verification algorithm of the Internet Draft. The `AggregateVerify` of the Basic scheme and the `Verify` and `AggregateVerify` algorithms of the message augmentation scheme all check the validity of public keys by the `KeyValidate` algorithm. This `KeyValidate` algorithm rejects the identity element as a public key. Then our first algorithm does not affect the security of the schemes. Our second algorithm uses repeated messages in the scheme without key prefixing. The basic scheme, which is the scheme without key prefixing in the Internet Draft, does not accept repeated messages as valid (regardless of the given aggregate signature). Then our second algorithm does not affect the security of the basic scheme.

As for the MS-BN multi-signature scheme and the BGOY ordered multi-signature scheme, we are not aware of any deployment of the schemes. Hence we do not think that our findings of these schemes not satisfying non-key substitution security affect real-world deployments.

While BIP327 [15] defines MuSig2 multi-signature algorithms, these algorithms are again not affected by our algorithm for key substitution attacks. Our algorithm again uses the trivial public key of the identity element. This trivial public key is not allowed by `KeyAgg`. This `KeyAgg` algorithm uses `cpoint` algorithm to decode public keys as a byte string to an elliptic-curve point. This `cpoint` algorithm fails if the input byte string encodes the identity element. Therefore, our algorithm does not affect the security of the scheme defined in BIP 327.

2 Preliminary

We let λ denote a security parameter. A polynomial function and a negligible function are denoted by $\mathrm{poly}(\lambda)$ and $\mathrm{negl}(\lambda)$, respectively. For an integer n, we let $[n] := \{1, 2, \cdots, n\}$. We abbreviate probabilistic polynomial time as PPT. For two strings x and y, we denote their concatenation by $x||y$. We denote by $c \xleftarrow{U} S$ setting c to an element chosen uniformly randomly from a set S. We denote by G^* the set of the generators of a group G. A multiset is denoted by $\{\{\}\}$. For sequence of elements $x_1, \ldots, x_n$, we denote by $\mathbf{x}|_i^n$ the concatenation of $x_i, x_{i+1}, \cdots, x_{n-1}, x_n$. An interactive algorithm is denoted by $\langle \{A_i(s_i)\}_{i=1}^n \rangle(p)$ where s_i denotes the input given to each algorithm, and p denotes the common input given to all algorithms.

Security is defined by an experiment between a challenger and a (PPT) adversary. We let the experiment output 1 if the adversary breaks the security. In such a case, we say the adversary wins the experiment (or the game).

Next, we introduce a trapdoor one-way permutation (TDP). TDP is a permutation computable in polynomial time but the inverse of that is not efficiently computable without the trapdoor.

Definition 1 (Trapdoor One-way Permutation). *A trapdoor one-way permutation family Π over D consists of the three PPT algorithms $(Generate, Evaluate, Invert)$:*

- *$Generate(1^\lambda) \to (s, t)$: Given a security parameter 1^λ, it outputs the description s of a permutation and the corresponding trapdoor t.*
- *$Evaluate(s, x) \to y$: Given the description s of a permutation and a group element x, it outputs another group element y.*
- *$Invert(s, t, y) \to x$: Given the description s of a permutation, the corresponding trapdoor t, and a group element y, it computes the inverse x of the image y under s, and outputs x.*

where it holds that $\Pr[x = \mathcal{A}(s, Evaluate(s, x)) : (s, t) \leftarrow Generate(1^\lambda); x \xleftarrow{U} D] \leq \mathrm{negl}(\lambda)$ *for any PPT algorithm $\mathcal{A}$, and* $\Pr[x = Invert(s, t, Evaluate(s, x)) : (s, t) \leftarrow Generate(1^\lambda); x \xleftarrow{U} D] = 1$.

2.1 Bilinear Group and Assumptions

We here introduce bilinear groups. Let $\mathcal{G}(1^\lambda) \to (p, G_1, G_2, G_T, g_1, g_2, e, \psi)$ be a PPT algorithm that takes as input a security parameter 1^λ and outputs a tuple of a prime p, multiplicative groups G_1, G_2, and G_T, generators g_1 and g_2 of G_1 and G_2, respectively, a bilinear map $e : G_1 \times G_2 \to G_T$, and an isomorphism $\psi \colon G_2 \to G_1$ satisfying $\psi(g_2) = g_1$. We call this $\mathcal{G}$ a bilinear group generator. We assume that every algorithm is given $\mathsf{gk} = (p, G_1, G_2, G_T, g_1, g_2, e, \psi) \leftarrow \mathcal{G}(1^\lambda)$ even if it is not mentioned explicitly. A bilinear map $e : G_1 \times G_2 \to G_T$ is a efficiently computable map that satisfies the following two conditions:

- Bilinearity: $\forall u \in G_1, \forall v \in G_2$ and $\forall a, b \in \mathbb{Z}, e(u^a, v^b) = e(u, v)^{ab}$.

– Non-degenerateness: $\forall g_1 \in G_1^*$ and $\forall g_2 \in G_2^*$, $e(g_1, g_2) \neq 1_{G_T}$.

We note that if the isomorphism ψ is not efficiently computable, then we call such a setting the type-III setting. If the isomorphism ψ is efficiently computable, then the setting is called the type-II setting. If $G_1 = G_2$, then we refer to the setting as the type-I setting. For simplicity, we write G_1 and G_2 simply as G in type-I or type-II setting.

In this paper, we use several assumptions relative to bilinear groups, the CDH assumption, the DDH assumption, the co-CDH assumption, and the DBP assumption. However, due to the space limitation, we introduce these assumptions in Appendix A.

2.2 Aggregate Signatures

An aggregate signature is a cryptographic primitive in which multiple signatures can be aggregated into a single signature.

Definition 2 (Aggregate Signature Scheme). *An aggregate signature scheme consists of the six PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Agg}, \mathsf{AggVrf})$ *that work as follows:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$**:** Setup *is the setup algorithm that takes a security parameter* 1^λ *as input, and outputs a public parameter* pp*.*

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$**:** KeyGen *is the key generation algorithm that takes a public parameter* pp *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m) \to \sigma$**:** Sign *is the signing algorithm that takes a public parameter* pp*, a secret key* sk*, and a message* m *as input, and outputs a signature* σ*.*

$\mathsf{Vrf}(\mathsf{pp}, \mathsf{pk}, m, \sigma) \to 1/0$**:** Vrf *is the verification algorithm for individual signatures: It takes a public parameter* pp*, a public key* pk*, a message* m*, and a signature* σ *as input, and outputs 1 (valid) or 0 (invalid).*

$\mathsf{Agg}(\mathsf{pp}, ((\mathsf{pk}_i, m_i, \sigma_i))_i) \to \tau$**:** Agg *is the aggregation algorithm that takes as input a public parameter* pp*, a sequence of triplets of a public key, a message, and a signature* $((\mathsf{pk}_i, m_i, \sigma_i))_i$*, and outputs an aggregate signature* τ*.*

$\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i, m_i))_i, \tau) \to 1/0$**:** AggVrf *is the verification algorithm for aggregate signatures: It takes a public parameter* pp*, a sequence of pairs of a public key and a message* $((\mathsf{pk}_i, m_i))_i$*, and an aggregate signature* τ *as input, and outputs 1 (valid) or 0 (invalid).*

Aggregate signature schemes should satisfy correctness and unforgeability (EUF-CMA). However, due to the space limitation, we put their definitions in Appendix B.

2.3 Sequential Aggregate Signatures

A sequential aggregate signature is a cryptographic primitive where each signer receives the current cumulative signature as input and generates a new signature in sequence, progressively extending the aggregate.

Definition 3 (Sequential Aggregate Signatures). *A sequential aggregate signature consists of the four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{AggSign}, \mathsf{AggVrf})$ *that work as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*:* Setup *is the setup algorithm that takes a security parameter* 1^λ *as input, and outputs a public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$*:* KeyGen *is the key generation algorithm that takes a public parameter* pp *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{AggSign}(\mathsf{pp}, \mathsf{sk}_i, m_i, \sigma_{i-1}, (m_j)_{j\in[i-1]}, (\mathsf{pk}_j)_{j\in[i-1]}) \to \sigma_i$*:* AggSign *is the signing algorithm that takes a public parameter* pp*, a secret key* sk_i*, a message* m_i*, a signature* σ_{i-1}*, a sequence of messages* $(m_j)_{j\in[i-1]}$*, and a sequence of public keys* $(\mathsf{pk}_j)_{j\in[i-1]}$ *as input, and outputs a signature* σ_i*.*
- $\mathsf{AggVrf}(\mathsf{pp}, \sigma_i, (m_j)_{j\in[i]}, (\mathsf{pk}_j)_{j\in[i]}) \to \{0,1\}$*:* AggVrf *is the verification algorithm that takes a public parameter* pp*, a signature* σ_i*, a sequence of messages* $(m_j)_{j\in[i]}$*, and a sequence of public keys* $(\mathsf{pk}_j)_{j\in[i]}$ *as input, and outputs 1 (valid) or 0 (invalid).*

We do not introduce unforgeability of sequential aggregate signature schemes because it is not discussed in this paper.

2.4 Multi-signatures

A multi-signature is a cryptographic primitive in which multiple signers sign the same message and produce a single signature.

Definition 4 (Multi-signature). *A multi-signature consists of four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ *that work as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*:* Setup *is the setup algorithm that takes a security parameter* 1^λ *as input, and outputs a public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$*:* KeyGen *is the key generation algorithm that takes a public parameter* pp *as input, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\langle\{\mathsf{Sign}(\mathsf{sk}_i)\}_{i=1}^n\rangle(\mathsf{pp}, \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m) \to \sigma$: Sign *is an interactive signing algorithm that takes a public parameter* pp*, a secret key* sk_i*, a multiset of public keys* $\{\{pk_i\}\}_{i\in[n]}$ *and a message* m *as input, and interacts with the other signers and finally outputs a signature* σ*.*
- $\mathsf{Vrf}(\mathsf{pp}, \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m, \sigma) \to \{0,1\}$*:* Vrf *is the verification algorithm that takes a public parameter* pp*, a multiset of public keys* $\{\{\mathsf{pk}_i\}\}_{i\in[n]}$*, a message* m *and a signature* σ *as input, and outputs* 1 *(valid) or* 0 *(invalid).*

Multi-signatures should satisfy correctness and unforgeability (UF-MS). However, due to the space limitation, we introduce it in Appendix C.

2.5 Ordered Multi-signatures

An ordered multi-signature is a cryptographic primitive in which multiple users sign the same message and the order of signing is guaranteed.

Definition 5 (Ordered multi-signature). *An ordered multi-signature scheme consists of the four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{OSign}, \mathsf{OVrf})$ *that work as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*:* Setup *is the setup algorithm that takes a security parameter* λ *as input, and outputs a public parameter* pp*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}_i, \mathsf{sk}_i)$*:* KeyGen *is the key generation algorithm that takes a public parameter* pp *as input, and outputs a public/secret key pair* $(\mathsf{pk}_i, \mathsf{sk}_i)$*.*
- $\mathsf{OSign}(\mathsf{pp}, \mathsf{sk}_i, m, \sigma_{i-1}, (\mathsf{pk}_j)_{j\in[i-1]}) \to \sigma_i$*:* OSign *is the signing algorithm that takes a public parameter* pp*, a secret key* sk_i*, a message* m*, a signature* σ_{i-1} *and a sequence of public keys* $(\mathsf{pk}_j)_{j\in[i-1]}$*, and outputs* σ_i*.*
- $\mathsf{OVrf}(\mathsf{pp}, m, \sigma_i, (\mathsf{pk}_j)_{j\in[i]}) \to \{0,1\}$*:* OVrf *is the verification algorithm that takes a public parameter* pp*, a message* m*, a signature* σ_i*, and a sequence of public keys* $(\mathsf{pk}_j)_{j\in[i]}$ *as input, and it outputs* 1 *(accept) or* 0 *(reject).*

Ordered multi-signatures should satisfy correctness and unforgeability (UF-OMS). However, due to the space limitation, we put its formal definition in Appendix I.

3 Key Substitution Attacks on Aggregate Signatures

In this section, we first formalize security against key substitution attacks for (standard) aggregate signatures. Then, we recap the construction of the BGLS signature scheme [7], and demonstrate that it is not secure against key substitution attacks without the modification of the construction. Finally, we modify the BGLS signature scheme so that it becomes secure against the attacks.

3.1 Non-Key Substitutability

We define non-key substitutability (NKS) and weak non-key substitutability (wNKS) of aggregate signature schemes as follows.

Definition 6. *An aggregate signature scheme* $\Sigma_{\mathrm{AS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Agg}, \mathsf{AggVrf})$ *satisfies non-key substitutability (NKS) if for any PPT adversary* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{ExpAS}^{NKS}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$*, where* $\mathsf{ExpAS}^{NKS}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda)$ *is defined as follows:*

$\mathsf{ExpAS}^{NKS}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Setup}(1^\lambda)$;
$(((\mathsf{pk}_i^*, m_i^*))_{i\in[n^*]}, ((\mathsf{pk}_i^{**}, m_i^{**}))_{i\in[n^{**}]}, \tau) \leftarrow \mathcal{A}(\mathsf{pp})$:
Output 1 if $\Sigma_{\mathrm{AS}}.\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i^*, m_i^*))_{i\in[n^*]}, \tau) = 1$
$\wedge \Sigma_{\mathrm{AS}}.\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i^{**}, m_i^{**}))_{i\in[n^{**}]}, \tau) = 1$
$\wedge \{\{(\mathsf{pk}_i^*, m_i^*) \mid i \in [n^*]\}\} \neq \{\{(\mathsf{pk}_i^{**}, m_i^{**}) \mid i \in [n^{**}]\}\}$ *(as multisets)*;
Otherwise output 0

Definition 7. *An aggregate signature scheme* $\Sigma_{\mathrm{AS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Agg}, \mathsf{AggVrf})$ *satisfies weak non-key substitutability (wNKS) if for any PPT adversary* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{ExpAS}^{wNKS}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$*, where* $\mathsf{ExpAS}^{wNKS}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda)$ *is defined as follows:*

$\mathsf{ExpAS}^{wNKS}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda)$

$n^* \leftarrow \mathcal{A}(\mathsf{pp})$;
$(\mathsf{pk}_i^*, \mathsf{sk}_i^*) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for all $i \in [n^*]$;
$(m_i^*)_{i\in[n^*]} \leftarrow \mathcal{A}((\mathsf{pk}_i^*)_{i\in[n^*]})$;
$\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_i, m_i)$ for all $i \in [n^*]; \tau \leftarrow \mathsf{Agg}(\mathsf{pp}, ((\mathsf{pk}_i, m_i, \sigma_i))_{i\in[n^*]})$;
$(\mathsf{pk}_i^*, m_i^*)_{i\in[n^*]} \leftarrow \mathcal{A}(\tau)$:
Output 1 if $\Sigma_{\mathrm{AS}}.\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i^{**}, m_i^{**}))_{i\in[n^{**}]}, \tau) = 1$;
$\wedge\{\{(\mathsf{pk}_i^*, m_i^*) \mid i \in [n^*]\}\} \neq \{\{(\mathsf{pk}_i^{**}, m_i^{**}) \mid i \in [n^{**}]\}\}$ *(as multisets)*;
Otherwise, output 0;

3.2 The BGLS Signature Scheme

Let $H : \{0,1\}^* \to G_1$ be a random oracle. The BGLS signature scheme [7] is the following:

$\mathsf{Setup}(1^\lambda)$**:** Given a security parameter 1^λ, it generates $\mathsf{gk} = (p, G_1, G_2, G_T, g_1, g_2, e, \psi) \leftarrow \mathcal{G}(1^\lambda)$ and sets $\mathsf{pp} := \mathsf{gk}$. It outputs pp.

$\mathsf{KeyGen}(\mathsf{pp})$**:** Given a public parameter pp, it chooses $x \leftarrow \mathbb{Z}_p$, and outputs $\mathsf{pk} := g_2^x$ and $\mathsf{sk} := x$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$**:** Given a public parameter pp, a secret key sk, and a message m, it computes $h \leftarrow H(m)$ and outputs $\sigma := h^x$.

$\mathsf{Vrf}(\mathsf{pp}, \mathsf{pk}, m, \sigma)$**:** Given a public parameter pp, a public key pk, a message m, and a signature σ, it outputs 1 if $e(\sigma, g_2) = e(H(m), \mathsf{pk})$, otherwise 0.

$\mathsf{Agg}(\mathsf{pp}, ((\mathsf{pk}_i, m_i, \sigma_i))_{i\in[n]})$**:** Given a public parameter pp, a sequence of tuples of a public key, a message, and a signature $((\mathsf{pk}_i, m_i, \sigma_i))_{i\in[n]}$, it outputs an aggregate signature $\tau := \prod_{i=1}^n \sigma_i \in G_1$.

$\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i, m_i))_{i\in[n]}, \tau)$**:** Given a public parameter pp, a sequence of pairs of a public key and a message $((\mathsf{pk}_i, m_i))_{i\in[n]}$, and an aggregate signature, it outputs 1 if $e(\tau, g_2) = \prod_{i=1}^n e(H(m_i), \mathsf{pk}_i)$, otherwise 0.

Theorem 1 ([7]). *If the co-CDH assumption holds with respect to* $\mathcal{G}$*, then the BGLS signature scheme is EUF-CMA secure.*

Key Substitution Attacks on BGLS. We demonstrate that the BGLS signature scheme is insecure against weak key substitution attacks. Let Σ be the BGLS scheme. We first observe what will happen if an adversary wins the experiment $\mathsf{ExpAS}^{wNKS}_{\Sigma,\mathcal{A}}(\lambda)$ before demonstrating our adversaries. Let $((\mathsf{pk}_i^*, m^*))_{i\in[n^*]}$ and $((\mathsf{pk}_i^{**}, m^{**}))_{i\in[n^{**}]}$ be sequences of public keys and messages with which a signature τ is accepted in the verification. Then, it holds that $\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i^*, m^*))_{i\in[n^*]}, \tau) = \mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i^{**}, m^{**}))_{i\in[n^{**}]}) = 1$. By the construction of AggVrf, it holds that $e(\tau, g_2) = \prod_{i=1}^{n^*} e(H(m_i^*), \mathsf{pk}_i^*) = \prod_{i=1}^{n^{**}} e(H(m_i^{**}), \mathsf{pk}_i^{**})$.

Theorem 2. *The BGLS scheme does not satisfy wNKS.*

Proof. We construct an adversary $\mathcal{A}$ that breaks wNKS of the BGLS as follows: Given a public parameter pp as input, it outputs a random integer $n^* \leq \mathrm{poly}(\lambda)$. Next, it takes a sequence of public keys $(\mathsf{pk}_i^*)_{i\in[n^*]}$ and sets $m_i^* \xleftarrow{U} \mathcal{M}$ for all $i \in [n^*]$. Here $\mathcal{M}$ denotes a message space. Then, given a signature τ as input, it sets $(m_i^{**}, \mathsf{pk}_i^{**}) \leftarrow (m_i^*, \mathsf{pk}_i^*)$ for all $i \in [n^*]$ and $\mathsf{pk}_{n^*+1}^{**} \leftarrow 1_G$, chooses an arbitrary message $m_{n^*+1}^{**} \in \mathcal{M}$, and outputs $(m_i^{**}, \mathsf{pk}_i^{**})_{i\in[n^*+1]}$. Since we have that $e(H(m_{n^*+1}^{**}), \mathsf{pk}_{n^*+1}^{**}) = 1$, it holds that $\prod_{i=1}^{n^*+1} e(H(m_i^{**}), \mathsf{pk}_i^{**}) = e(H(m_{n^*+1}^{**}), \mathsf{pk}_{n^*+1}^{**}) \prod_{i=1}^{n^*} e(H(m_i^{**}), \mathsf{pk}_i^{**}) = \prod_{i=1}^{n^*} e(H(m_i^*), \mathsf{pk}_i^*) = e(\tau, g_2)$. Therefore, $\mathcal{A}$ breaks wNKS of the BGLS. □

In the case where signers are allowed to sign the same message, we can provide another proof of the theorem. This extra proof highlights the necessity of hashing a public key in our modification.

Proof. We construct an adversary $\mathcal{A}$ that breaks wNKS of the BGLS as follows: Given a public parameter pp as input, it outputs a random integer $n^* \leq \mathrm{poly}(\lambda)$. Next, it takes a sequence of public keys $(\mathsf{pk}_i^*)_{i\in[n^*]}$ and sets $m_i^* \xleftarrow{U} \mathcal{M}$ for all $i \in [n^*]$. Here $\mathcal{M}$ denotes a message space. Then, given a signature τ as input, it sets $(m_i^{**}, \mathsf{pk}_i^{**}) \leftarrow (m_i^*, \mathsf{pk}_i^*)$ for all $i \in [n^* - 1]$ and $m_{n^*}^{**}, m_{n^*+1}^{**} \leftarrow m_{n^*}^*$, computes $\mathsf{pk}_{n^*}^{**} \xleftarrow{U} G$ and $\mathsf{pk}_{n^*+1}^{**} \leftarrow \mathsf{pk}_{n^*}^* \cdot (\mathsf{pk}_{n^*}^{**})^{-1}$, and outputs $(m_i^{**}, \mathsf{pk}_i^{**})_{i\in[n^*+1]}$. Since we have that $e(H(m_{n^*+1}^{**}), \mathsf{pk}_{n^*+1}^{**}) e(H(m_{n^*}^{**}), \mathsf{pk}_{n^*}^{**}) = e(H(m_{n^*}^*), \mathsf{pk}_{n^*}^* \cdot (\mathsf{pk}_{n^*}^{**})^{-1}) e(H(m_{n^*}^*), \mathsf{pk}_{n^*}^{**}) = e(H(m_{n^*}^*), \mathsf{pk}_{n^*}^*)$, it holds that $\prod_{i=1}^{n^{**}} e(H(m_i^{**}), \mathsf{pk}_i^{**}) = \prod_{i=n^*}^{n^*+1} e(H(m_i^{**}), \mathsf{pk}_i^{**}) \prod_{i=1}^{n^*-1} e(H(m_i^*), \mathsf{pk}_i^*) = \prod_{i=1}^{n^*} e(H(m_i^*), \mathsf{pk}_i^*) = e(\tau, g_2)$. Thus, $\mathcal{A}$ breaks wNKS of the BGLS. □

3.3 Modified BGLS Signature Scheme

We show that the BGLS signature scheme becomes secure against key substitution attacks with slight modifications even in the case where multiple users sign the same message. That is, we prohibit $\mathsf{pk} = 1$, and introduce the key-prefixed BGLS signature scheme (i.e., we let the signing algorithm not only hash a message but also a public key). Further, the construction uses proof of possession to prove the security in the type-III pairing setting. Toward proving the security, we should introduce an additional computational hardness assumption: the double pairing (DBP) assumption. It is known that the DBP assumption is implied by the DDH assumption [1]. Therefore, the additional assumption is a mild one.

Definition 8 (The DBP Assumption). *The double pairing (DBP) assumption holds with respect to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ it holds that*

$$\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); h_r, h_z \xleftarrow{U} G_1 \setminus \{1\}; (r, z) \leftarrow \mathcal{A}(\mathsf{gk}, h_r, h_z) : e(h_z, z)e(h_r, r) = 1 \wedge (r, z) \neq (1, 1)] \leq \mathrm{negl}(\lambda).$$

We first provide the construction of the modified BGLS signature scheme. Namely, we do not let the bilinear group generator output an isomorphism, as we are considering the type-III setting. Then, we demonstrate that it is secure against key substitution attacks and unforgeable, respectively.

Let $H : \{0,1\}^* \to G_1$ and $H_P : \{0,1\}^* \to G_1$ be random oracles. The modified BGLS signature scheme Π_{BP} is constructed as follows.

$\mathsf{Setup}(1^\lambda)$**:** Given a security parameter 1^λ, it generates $\mathsf{gk} = (p, G_1, G_2, G_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda)$ and sets $\mathsf{pp} = \mathsf{gk}$. It outputs pp.

$\mathsf{KeyGen}(\mathsf{pp})$**:** Given a public parameter pp, it chooses $x \leftarrow \mathbb{Z}_p^*$, computes $\mathsf{pk} := g_2^x$ and $\sigma_{\mathsf{pop}} := (H_P(\mathsf{pk}))^x$, and outputs $\hat{\mathsf{pk}} := (\mathsf{pk}, \sigma_{\mathsf{pop}})$ and $\mathsf{sk} := x$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$**:** Given a public parameter pp, a secret key $\mathsf{sk} = x$, and a message m, it computes $h \leftarrow H(m \,||g_2^x)$ and outputs $\sigma := h^x$.

$\mathsf{Vrf}(\mathsf{pp}, \hat{\mathsf{pk}}, m, \sigma)$**:** Given a public parameter pp, a public key $\hat{\mathsf{pk}} = (\mathsf{pk}, \sigma_{\mathsf{pop}})$, a message m, and a signature σ, it outputs 1 if $e(\sigma, g_2) = e(H(m \,||\, \mathsf{pk}), \mathsf{pk})$, $\mathsf{pk} \neq 1$, and $e(\sigma_{\mathsf{pop}}, g_2) = e(H_P(\mathsf{pk}), \mathsf{pk})$, otherwise 0.

$\mathsf{Agg}(\mathsf{pp}, ((\hat{\mathsf{pk}}_i, m_i, \sigma_i))_{i\in[n]})$**:** Given a public parameter pp and a sequence of tuples of public key, message, and signature $(\hat{\mathsf{pk}}_i, m_i, \sigma_i))_{i\in[n]}$, it outputs an aggregate signature $\tau := \prod_{i=1}^n \sigma_i \in G_1$.

$\mathsf{AggVrf}(\mathsf{pp}, ((\hat{\mathsf{pk}}_i, m_i))_{i\in[n]}, \tau)$**:** Given a public parameter pp, a sequence of pairs of public key where $\hat{\mathsf{pk}}_i = (\mathsf{pk}_i, \sigma_{\mathsf{pop},i})$ and message $((\hat{\mathsf{pk}}_i, m_i))_{i\in[n]}$, and an aggregate signature τ, it outputs 1 if $e(\tau, g_2) = \prod_{i=1}^n e(H(m_i \,||\, \mathsf{pk}_i), \mathsf{pk}_i)$ and $\forall i \in [n], \mathsf{pk}_i \neq 1$ and $e(\sigma_{\mathsf{pop},i}, g_2) = e(H_P(\mathsf{pk}_i), \mathsf{pk}_i)$, otherwise 0.

Correctness is immediate. Thus, we focus on the security requirements.

Security against Key Substitution Attacks. We first argue $e(H(m \,||\, \mathsf{pk}), \mathsf{pk}) \cdot e(H(m \,||\, \mathsf{pk}'), \mathsf{pk}') \neq e(H(m \,||\, \mathsf{pk} \cdot \mathsf{pk}'), \mathsf{pk} \cdot \mathsf{pk}')$ with overwhelming probability. For the attack shown in the second proof of theorem 2 to work successfully, a polynomial time adversary should find a message m and public keys $\mathsf{pk} = g_2^x$ and $\mathsf{pk}' = g_2^{x'}$ such that $e(H(m \,||\, \mathsf{pk}), \mathsf{pk}) \cdot e(H(m \,||\, \mathsf{pk}'), \mathsf{pk}') = e(H(m \,||\, \mathsf{pk} \cdot \mathsf{pk}'), \mathsf{pk} \cdot \mathsf{pk}')$. As H is a random oracle, $H(m \,||\, \mathsf{pk}), H(m \,||\, \mathsf{pk}')$ and $H(m \,||\, \mathsf{pk} \cdot \mathsf{pk}')$ are mapped to distinct values, say $g_1^z, g_1^{z'}$ and $g_1^{z^*}$, respectively. In other words, it should hold that $e(g_1, g_2)^{zx} \cdot e(g_1, g_2)^{z'x'} = e(g_1, g_2)^{zx+z'x'} = e(g_1, g_2)^{z^*(x+x')}$ for the attack works well. However, the adversary can find such values with probability at most $\mathrm{poly}(\lambda)/p$, which is negligible in λ.

Now we prove that the modified BGLS signature scheme satisfies NKS.

Lemma 1. *The scheme* Π_{BP} *satisfies NKS.*

Proof. The proof proceeds as follows. We first introduce an intermediate problem, and show that if the problem can be solved by a PPT algorithm, then the DBP assumption is also solved by another PPT adversary. After that, we demonstrate that if there exists a PPT adversary that violates the security of the

modified BGLS signature scheme against key substitution attacks, then there is a PPT algorithm that solves the intermediate problem.

The intermediate problem is defined by the following experiment:

$$\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); h_1, \ldots, h_n \leftarrow G_1; (r_1, \ldots, r_n) \leftarrow \mathcal{A}(\mathsf{gk}, h_1, \ldots, h_n) : \prod_{i \in [n]} e(h_i, r_i) = 1 \wedge (r_1, \ldots, r_n) \neq (1, \ldots, 1)]. \quad (1)$$

Claim. If there is a PPT algorithm $\mathcal{A}$ that makes the probability in Eq. (1) non-negligible, then there exists a PPT adversary $\mathcal{A}'$ that breaks the DBP assumption with non-negligible probability.

Proof. We demonstrate how $\mathcal{A}'$ breaks the DBP assumption by simulating $\mathcal{A}$. Given $h_r, h_z \in G_1$, $\mathcal{A}'$ does the followings: For all $i \in [n]$, it chooses $a_i, b_i \in \mathbb{Z}_p$ uniformly, and computes $\hat{h}_i := h_r^{a_i} \cdot h_z^{b_i}$. Then, $\mathcal{A}'$ simulates the adversary $\mathcal{A}$ by giving $\hat{h}_1, \cdots, \hat{h}_n$ to obtain $(r_1, \cdots, r_n)$ s.t. $\prod_{i \in [n]} e(\hat{h}_i, r_i) = 1$ and $(r_1, \cdots, r_n) \neq (1, \cdots, 1)$. Finally, $\mathcal{A}'$ computes $r = \prod_{i \in [n]} r_i^{a_i}$ and $z = \prod_{i \in [n]} r_i^{b_i}$, and outputs (r, z).

We show that (r, z) satisfies that $e(h_r, r)e(h_z, z) = 1$. If $\prod_{i \in [n]} e(\hat{h}_i, r_i) = 1$, then it holds that $1 = \prod_{i \in [n]} e(\hat{h}_i, r_i) = \prod_{i \in [n]} e(h_r^{a_i} \cdot h_z^{b_i}, r_i) = e(h_r, \prod_{i \in [n]} r_i^{a_i}) \cdot e(h_z, \prod_{i \in [n]} r_i^{b_i})$. Thus, it is sufficient to show that $(\prod_{i \in [n]} r_i^{a_i}, \prod_{i \in [n]} r_i^{b_i}) \neq (1, 1)$ for our proof.

Observe that every $\hat{h}_i$ that is given to $\mathcal{A}$ is independent of a_i because b_i is chosen uniformly. Thus, $(r_1, \cdots, r_n)$ and $(a_1, \cdots, a_n)$ are independent. As there is at least one r_i s.t. $r_i \neq 1$ with non-negligible probability, $\prod_{i \in [n]} r_i^{a_i}$ is uniformly distributed over G_1. Thus, it holds that $\prod_{i \in [n]} r_i^{a_i} = 1$ with probability $1/p$, which implies $(\prod_{i \in [n]} r_i^{a_i}, \prod_{i \in [n]} r_i^{b_i}) \neq (1, 1)$ with non-negligible probability. This concludes the proof. □

Claim. If there exists a PPT adversary $\mathcal{A}$ that violates NKS of the modified BGLS signature scheme with non-negligible probability, then there is a PPT algorithm $\mathcal{A}'$ that makes the probability in Eq. (1) non-negligible.

Proof. Let q_H be the maximum number of queries that $\mathcal{A}$ can make to random oracle $H : \{0,1\}^* \to G_1$. Given $\hat{h}_1, \cdots, \hat{h}_{q_H}$, the algorithm $\mathcal{A}'$ simulates $\mathcal{A}$ as follows. When $\mathcal{A}$ queries $H(m_j \,||\, \mathsf{pk}_j)$ for the j-th time, $\mathcal{A}'$ gives $\hat{h}_j$ as an answer. Suppose that $\mathcal{A}$ outputs $((m_i^*, \hat{\mathsf{pk}}_i^*))_{i \in [n^*]}$, $((m_i^{**}, \hat{\mathsf{pk}}_i^{**}))_{i \in [n^{**}]}$ and an aggregate signature τ where $\{(m_i^*, \hat{\mathsf{pk}}_i^*)\}_{i \in [n^*]} \neq \{(m_i^{**}, \hat{\mathsf{pk}}_i^{**})\}_{i \in [n^{**}]}$ as multisets, $\mathsf{AggVrf}(\mathsf{pp}, ((m_i^*, \hat{\mathsf{pk}}_i^*))_{i \in [n^*]}, \tau) = 1$ and $\mathsf{AggVrf}(\mathsf{pp}, ((m_i^{**}, \hat{\mathsf{pk}}_i^{**}))_{i \in [n^{**}]}, \tau) = 1$. In other words, it holds that

$$e(\tau, g) \quad = \quad \prod_{i=1}^{n^*} e(H(m_i^* \,\|\, \mathsf{pk}_i^*), \mathsf{pk}_i^*) \quad = \quad \prod_{i=1}^{n^{**}} e(H(m_i^{**} \,\|\, \mathsf{pk}_i^{**}), \mathsf{pk}_i^{**}) \quad (2)$$

and

$$e(\sigma^*_{\mathsf{pop},i}, g_2) = e(H_P(\mathsf{pk}^*_i), \mathsf{pk}^*_i) \tag{3}$$

where $(\mathsf{pk}^*_i, \sigma^*_{\mathsf{pop},i} \leftarrow \hat{\mathsf{pk}}^*_i$ for all $i \in [n^*]$, and

$$e(\sigma^{**}_{\mathsf{pop},i}, g_2) = e(H_P(\mathsf{pk}^{**}_i), \mathsf{pk}^{**}_i) \tag{4}$$

where $(\mathsf{pk}^{**}_i, \sigma^{**}_{\mathsf{pop},i}) \leftarrow \hat{\mathsf{pk}}^{**}_i$ for all $i \in [n^{**}]$.

Let $t^*_j = \#\{i \in [n^*] \mid (m_j, \mathsf{pk}_j) = (m^*_i, \mathsf{pk}^*_i)\}$ and $t^{**}_j = \#\{i \in [n^{**}] \mid (m_j, \mathsf{pk}_j) = (m^{**}_i, \mathsf{pk}^{**}_i)\}$. Then $\mathcal{A}'$ outputs $(r_1, \ldots, r_{q_H})$ where $r_j = \mathsf{pk}_j^{t^*_j - t^{**}_j}$.

Due to Eq. (2), we have that $(r_1, \ldots, r_{q_H})$ satisfies $\prod_{i \in [q_H]} e(\hat{h}_i, r_i) = 1$.

Let

$$M^* = \{\{(m^*_i, (\mathsf{pk}^*_i, \sigma^*_{\mathsf{pop},i})) \mid i \in [n^*]\}\}, \tag{5}$$

$$M^{**} = \{\{(m^{**}_i, (\mathsf{pk}^{**}_i, \sigma^{**}_{\mathsf{pop},i})) \mid i \in [n^{**}]\}\}, \tag{6}$$

$$N^* = \{\{(m^*_i, \mathsf{pk}^*_i) \mid i \in [n^*]\}\}, \tag{7}$$

$$N^{**} = \{\{(m^{**}_i, \mathsf{pk}^{**}_i) \mid i \in [n^{**}]\}\} \tag{8}$$

as *multisets*. Then the winning condition ensures that $M^* \neq M^{**}$. Furthermore, this inequality ensures that $N^* \neq N^{**}$.

We now prove this. Let us assume that there is an element $(m, (\mathsf{pk}, \sigma_{\mathsf{pop}})) \in M^*$ whose multiplicity is m. Then we claim that the element (m, pk) belongs to N^* with the same multiplicity $n = m$. The element $(m, (\mathsf{pk}, \sigma_{\mathsf{pop}}))$ appears m times in M^* and then (m, pk) appears at least m times in N^*. Moreover, (m, pk) may appear more frequently, if there is an element $(m, (\mathsf{pk}, \sigma'_{\mathsf{pop}}))$ where $\sigma'_{\mathsf{pop}} \neq \sigma_{\mathsf{pop}}$ in M^*. However, the winning condition ensures that such a pair $(m, (\mathsf{pk}, \sigma'_{\mathsf{pop}}))$ never belongs to M^*. This is due to Eq. (3), which ensures that there is a unique σ_{pop} for each pk satisfying $e(\sigma_{\mathsf{pop}}, g_2) = e(H_P(\mathsf{pk}), \mathsf{pk})$. This ensures that $n = m$. The similar fact holds for M^{**} and N^{**}. Then, if there are elements $(m^*, (\mathsf{pk}^*, \sigma^*_{\mathsf{pop}})) \in M^*$ and $(m^{**}, (\mathsf{pk}^{**}, \sigma^{**}_{\mathsf{pop}})) \in M^{**}$ whose multiplicities differ, then the elements $(m^*, \mathsf{pk}^*) \in N^*$ and $(m^{**}, \mathsf{pk}^{**}) \in N^{**}$ have different multiplicities inherited from $(m^*, (\mathsf{pk}^*, \sigma^*_{\mathsf{pop}})) \in M^*$ and $(m^{**}, (\mathsf{pk}^{**}, \sigma^{**}_{\mathsf{pop}})) \in M^{**}$. This shows that if $M^* \neq M^{**}$, it holds that $N^* \neq N^{**}$.

We claim that at least for some $j \in [q_H]$, $t^*_j \neq t^{**}_j$. This is due to the fact that $N^* \neq N^{**}$. Since t^*_j and t^{**}_j are the multiplicities of $(m_j, \mathsf{pk}_j) \in N^*$ and $(m_j, \mathsf{pk}_j) \in N^{**}$, there are some (m_j, pk_j) whose multiplicities in N^* and N^{**} differ. We also have that $\mathsf{pk}_j \neq 1$ by the verification condition. Since we have that t^*_j, $t^{**}_j < p$ for sufficiently large security parameters λ, we have that $-p < t^*_j - t^{**}_j < p$ (and $t^*_j - t^{**}_j \neq 0$)). Then, for this j, we have that $\mathsf{pk}_j^{t^*_j - t^{**}_j} \neq 1$ for sufficiently large security parameters. Therefore, we can conclude that $\mathcal{A}$ outputs $(r_1, \ldots, r_{q_H}) \neq (1, \ldots, 1)$ for sufficiently large security parameters. □

Summarizing the above discussion, if the modified BGLS signature scheme is not NKS, then we can break the DBP assumption in polynomial time. □

Unforgeability. We introduce a mild assumption, the co-CDH′ assumption, and prove unforgeability of Π_{BP} under the assumption. Due to the space limitation, we put the definition of co-CDH′ assumption and the proof in Appendix B.

Lemma 2. Π_{BP} *is EUF-CMA secure under the co-CDH′ assumption.*

4 Key Substitution Attacks on Sequential Aggregate Signatures

We first define NKS for sequential aggregate signatures. We then prove that the sequential aggregate signature scheme from trapdoor permutation (SAfromTDP) by Lysyanskaya et al. [13] satisfies NKS without modification.

4.1 Non-Key Substitutability

We define NKS of sequential aggregate signature schemes as follows.

Definition 9 (Non-Key Substitutability). *A sequential aggregate signature* $\Sigma_{SAS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{AggSign}, \mathsf{AggVrf})$ *satisfies non-key substitutability if for all PPT algorithm* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{ExpSAS}^{NKS}_{\Sigma_{SAS},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ *where* $\mathsf{ExpSAS}^{NKS}_{\Sigma_{SAS},\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpSAS}^{NKS}_{\Sigma_{SAS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \Sigma_{SAS}.\mathsf{Setup}(1^\lambda)$;
$(((m_i^*, \mathsf{pk}_i^*))_{i\in[n^*]}, ((m_i^{**}, \mathsf{pk}_i^{**}))_{i\in[n^{**}]}, \sigma) \leftarrow \mathcal{A}(\mathsf{pp})$:
Output 1 if $\Sigma_{SAS}.\mathsf{AggVrf}(\mathsf{pp}, \sigma, ((m_i^*, \mathsf{pk}_i^*))_{i\in[n^*]}) = 1$
$\wedge \Sigma_{SAS}.\mathsf{AggVrf}(\mathsf{pp}, \sigma, ((m_i^{**}, \mathsf{pk}_i^{**}))_{i\in[n^{**}]}) = 1$
$\wedge ((m_i^*, \mathsf{pk}_i^*))_{i\in[n^*]} \neq ((m_i^{**}, \mathsf{pk}_i^{**}))_{i\in[n^{**}]}$ (as ordered set);
Otherwise output 0;

4.2 SAfromTDP

We introduce the SAfromTDP, and show that it satisfies NKS. SAfromTDP consists of the following algorithms (we here assume the domains of permutations are common among all signers.):

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Given a security parameter 1^λ, it chooses a trapdoor permutation family $\Pi = (Generate, Evaluate, Invert)$ over D and a hash function $H : \{0,1\}^* \to D$ and outputs a public parameter $\mathsf{pp} \leftarrow (H, \Pi, D)$.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Given a public parameter pp, it sets $(s,t) \xleftarrow{U} \Pi.Generate$, and outputs a public key $\mathsf{pk} \leftarrow s$ and a secret key $\mathsf{sk} \leftarrow (s,t)$.

$\mathsf{AggSign}(\mathsf{pp}, \mathsf{sk}_i, m_i, \sigma_{i-1}, (m_j)_{j\in[i-1]}, (\mathsf{pk}_j)_{j\in[i-1]}) \to \sigma_i$: Given a public parameter pp, a secret key sk_i, a message m_i, an aggregate so far σ_{i-1} ($\sigma_0 = 1$ for the first signing), a sequence of public keys $(\mathsf{pk}_j)_{j\in[i-1]}$ and a sequence of messages $(m_j)_{j\in[i-1]}$, it outputs $\perp$ if it holds that $\mathsf{AggVrf}(\mathsf{pp}, (m_j)_{j\in[i-1]}, \sigma_{i-1}, (\mathsf{pk}_j)_{j\in[i-1]}) = 0$. Otherwise, it computes $\sigma_i \leftarrow \Pi.Invert(s_i, t_i, \sigma_{i-1} \odot H(\mathbf{s}|_1^i, \mathbf{m}|_1^i)$ and outputs σ_i.

$\mathsf{AggVrf}(\mathsf{pp}, \sigma_i, (m_j)_{j\in[i]}, (\mathsf{pk}_j)_{j\in[i]}) \to \{0,1\}$: It takes a public parameter pp, a signature σ_i, a sequence of messages $(m_j)_{j\in[i]}$ and a sequence of public keys $(\mathsf{pk}_j)_{j\in[i]}$ as input. It outputs 0 if there exists j such that s_j does not describe a valid permutation, it holds that $\exists j \neq k \in [i], \mathsf{pk}_j = \mathsf{pk}_k$, or the number of messages differs from that of public keys. Otherwise, it computes $\sigma_{j-1} \leftarrow \Pi.Evaluate(s_j, \sigma_j) \odot H(\mathbf{s}|_1^j, \mathbf{m}|_1^j)^{-1}$ for $j = i, \cdots, 1$, and then it outputs 1 if it holds $\sigma_0 = 1$, otherwise outputs 0.

Theorem 3. *SAfromTDP satisfies NKS in the random oracle model.*

Proof. We show that there are collisions of random numbers if there exists a PPT adversary that breaks NKS of the SAfromTDP. Suppose that there exists a PPT algorithm $\mathcal{A}$ that breaks NKS of the SAfromTDP with non-negligible probability. Let Σ_{SAS} be the SAfromTDP scheme. Let $((\mathsf{pk}_i^*)_{i\in[n^*]}, (m_i^*)_{i\in[n^*]}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]}, (m_i^{**})_{i\in[n^{**}]}, \sigma)$ denote the output of $\mathcal{A}$ where it results in $\mathsf{ExpSAS}^{NKS}_{\Sigma_{SAS},\mathcal{A}}(\lambda) = 1$. Let $\sigma^*_{n^*-1}$ and $\sigma^{**}_{n^{**}-1}$ be $Evaluate(s^*_{n^*}, \sigma) \odot H(\mathbf{s}^*|_1^{n^*}, \mathbf{m}^*|_1^{n^*})^{-1}$ and $Evaluate(s^{**}_{n^{**}}, \sigma) \odot H(\mathbf{s}^{**}|_1^{n^{**}}, \mathbf{m}^{**}|_1^{n^{**}})^{-1}$ respectively. Since $Invert$ is the inverse map of $Evaluate$, we observe that $\sigma = Invert(s^*_{n^*}, t^*_{n^*}, \sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_1^{n^*}, \mathbf{m}^*|_1^{n^*})) = Invert(s^{**}_{n^{**}}, t^{**}_{n^{**}}, \sigma^{**}_{n^{**}-1} \odot H(\mathbf{s}^{**}|_1^{n^{**}}, \mathbf{m}^{**}|_1^{n^{**}}))$. We show that $Invert(s^*_{n^*}, t^*_{n^*}, \sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*}))$ and $Invert(s^{**}_{n^{**}}, t^{**}_{n^{**}}, \sigma^{**}_{n^{**}-1} \odot H(\mathbf{s}^{**}|_i^{n^{**}}, \mathbf{m}^{**}|_i^{n^{**}}))$ are uniformly random and independently distributed.

We first show that $Invert(s^*_{n^*}, t^*_{n^*}, \sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_1^{n^*}, \mathbf{m}^*|_1^{n^*}))$ is uniformly random. Without loss of generality, we assume that $\mathcal{A}$ queries $(\mathbf{s}^*|_1^k, \mathbf{m}^*|_1^k)$ for all $k \in [n^*-1]$ before it queries $(\mathbf{s}^*|_1^{n^*}, \mathbf{m}^*|_1^{n^*})$. In the verification algorithm, for $k = n^*, n^*-1, \ldots, 1$, it computes $\sigma^*_{k-1} = Evaluate(s_k, \sigma_k) \odot H(\mathbf{s}^*|_1^k, \mathbf{m}^*|_1^k)^{-1}$, and it must hold that $\sigma^*_0 = 1$ since the verification algorithm accepts the signature σ. Considering the inverse map, we observe that it holds that $\sigma^*_k = Invert(s^*_k, t^*_k, \sigma^*_{k-1} \odot H(\mathbf{s}^*|_1^k, \mathbf{m}^*|_1^k))$ for all $k \in [n^*]$. Thus, $\sigma^*_{n^*-1}$ that appears in successful verification is uniquely determined by $(s^*_1, s^*_2, ..., s^*_{n^*-1}, m^*_1, m^*_2, ..., m^*_{n^*-1})$. Therefore, $\sigma^*_{n^*-1}$ has been uniquely determined when $\mathcal{A}$ queries $(\mathbf{s}^*|_i^{n^*-1}, \mathbf{m}^*|_i^{n^*-1})$ to the hash oracle before $\mathcal{A}$ queries $(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*})$ to the hash oracle. Since $H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*})$ is uniformly random, $\sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*})$ is also uniformly random regardless of $\sigma^*_{n^*-1}$. Since the description $s^*_{n^*}$ of the permutation has been determined before $\sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*})$ is obtained, $Invert(s^*_{n^*}, t^*_{n^*}, \sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*}))$ is uniformly random.

Likewise, $Invert(s^{**}_{n^{**}}, t^{**}_{n^{**}}, \sigma^{**}_{n^{**}-1} \odot H(\mathbf{s}^{**}|_i^{n^{**}}, \mathbf{m}^{**}|_i^{n^{**}}))$ is uniformly random. From the winning condition of $\mathsf{ExpSAS}^{NKS}_{\Sigma_{SAS},\mathcal{A}}(\lambda)$, we have that $(m_i^*, \mathsf{pk}_i^*)_{i\in[n^*]} \neq (m_i^{**}, \mathsf{pk}_i^{**})_{i\in[n^{**}]}$. Thus, the distributions of $H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*})$ and $H(\mathbf{s}^{**}|_i^{n^{**}}, \mathbf{m}^{**}|_i^{n^{**}})$ are independent, and hence those of $Invert(s^*_{n^*}, t^*_{n^*}, \sigma^*_{n^*-1} \odot H(\mathbf{s}^*|_i^{n^*}, \mathbf{m}^*|_i^{n^*}))$ and $Invert(s^{**}_{n^{**}}, t^{**}_{n^{**}}, \sigma^{**}_{n^{**}-1} \odot H(\mathbf{s}^{**}|_i^{n^{**}}, \mathbf{m}^{**}|_i^{n^{**}}))$ are independent. Thus, $\mathsf{ExpSAS}^{NKS}_{\Sigma_{SAS},\mathcal{A}}(\lambda) = 1$ means the collision of the uniformly random values.

Since $\mathcal{A}$ is a PPT algorithm, it can query at most polynomial number of times, and hence the times of trials of collisions is also at most polynomial number.

Thus, if such an $\mathcal{A}$ exists, we can construct an algorithm that finds collisions of random numbers chosen from an exponential-size set in polynomial time. □

5 Key Substitution Attacks on Multi-signatures

We first introduce NKS and wNKS of multi-signatures. Then, we show that the scheme by Bellare and Neven [3] (MS-BN) and MuSig2 [16] satisfy wNKS, but not NKS. We further improve these schemes so that they achieve NKS.

We here define a group generation algorithm $\mathcal{P}$ as a PPT algorithm that takes a security parameter 1^λ as input, and outputs a prime p, a group G of order p, and a generator g of G.

5.1 Non-Key Substitutability

We define NKS and wNKS of multi-signatures as follows.

Definition 10 (Non-Key Substitutability). *A multi-signature* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ *satisfies non-key substitutability if for all PPT algorithm* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{ExpMS}^{NKS}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ *where* $\mathsf{ExpMS}^{NKS}_{\Sigma,\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpMS}^{NKS}_{\Sigma,\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \Sigma.\mathsf{Setup}(1^\lambda)$;
$(m^*, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}, m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}, \sigma) \leftarrow \mathcal{A}(\mathsf{pp})$:
Output 1 if $\Sigma.\mathsf{Vrf}(\mathsf{pp}, \sigma, m^*, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}) = 1$
$\wedge \Sigma.\mathsf{Vrf}(\mathsf{pp}, \sigma, m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}) = 1$
$\wedge (m^* \neq m^{**} \vee \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]} \neq \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}$ (as multiset))
Otherwise output 0

Definition 11 (Weak Non-Key Substitutability). *A multi-signature* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ *satisfies weak non-key substitutability if for all stateful PPT algorithm* $\mathcal{A}$*, it holds that* $\Pr[\mathsf{ExpMS}^{wNKS}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ *where* $\mathsf{ExpMS}^{wNKS}_{\Sigma,\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpMS}^{wNKS}_{\Sigma,\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \Sigma.\mathsf{Setup}(1^\lambda)$;
$n^* \leftarrow \mathcal{A}(\mathsf{pp})$;
$(\mathsf{pk}_i^*, \mathsf{sk}_i^*) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for all $i \in [n^*]$;
$m^* \leftarrow \mathcal{A}(\{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]})$;
$\sigma \leftarrow \langle \{\mathsf{Sign}(\mathsf{sk}_i^*)\}_{i=1}^{n^*} \rangle (\mathsf{pp}, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}, m^*)$;
$(m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}) \leftarrow \mathcal{A}(\mathsf{pp}, m^*, \sigma, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]})$:
Output 1 if $\Sigma.\mathsf{Vrf}(\mathsf{pp}, \sigma, m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}) = 1$
$\wedge (m^* \neq m^{**} \vee \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]} \neq \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}$ (as multiset));
Otherwise output 0

5.2 MS-BN

MS-BN, proposed by Bellare and Neven [3], is based on Schnorr signatures. It consists of the four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ that work as follows:

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Given a security parameter 1^λ as input, it obtains (G, p, g) by running $\mathcal{P}(1^\lambda)$ and outputs a public parameter $\mathsf{pp} \leftarrow (G, p, g)$.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Given a public parameter pp as input, it chooses $x \xleftarrow{U} \mathbb{Z}_p$, computes $X \leftarrow g^x$, sets $\mathsf{sk} \leftarrow x$ and $\mathsf{pk} \leftarrow X$, and outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\langle\{\mathsf{Sign}(\mathsf{sk}_i)\}_{i=1}^n\rangle(\mathsf{pp}, \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m) \to \sigma$: Sign is an interactive protocol among $n = \mathrm{poly}(\lambda)$ signers which consists of the following four rounds. Each round is conducted by each signer locally. Each signer considers their own index as 1 and executes each round accordingly.

- $\mathsf{round1}(\mathsf{sk}, L = \{\{X_i\}\}_{i\in[n]}, m) \to t_1$: Given a secret key sk, a multiset of public keys L and message m, it chooses $r_1 \xleftarrow{U} \mathbb{Z}_p$, computes $R_1 \leftarrow g^{r_1}$ and $t_1 \leftarrow H_0(R_1)$, and sends t_1 to the other signers.
- $\mathsf{round2}(\{\{t_i | 2 \le i \le n\}\}) \to R_1$: Given a multiset of hash values $\{\{t_i | 2 \le i \le n\}\}$ from the other signers, it sends R_1 to the other signers.
- $\mathsf{round3}(\{\{R_i | 2 \le i \le n\}\}) \to s_1$: Given a multiset of random elements $\{\{R_i | 2 \le i \le n\}\}$ from the other signers, it checks if for all $2 \le i \le n$, it holds that $t_i = H_0(R_i)$, otherwise it outputs $\perp$ and terminates. If it continues, then it computes $R \leftarrow \prod_{i=1}^n R_i$, $c_i \leftarrow H_1(X_1||R||L||m)$ and $s_1 \leftarrow x_1 c_1 + r_1 \mod p$, and sends s_1 to the other signers.
- $\mathsf{round4}(\{\{s_i | 2 \le i \le n\}\}) \to \sigma$: Given a multiset of random integers $\{\{s_i | 2 \le i \le n\}\}$ from the other signers, it computes $s \leftarrow \sum_{i=1}^n s_i \mod p$, sets $\sigma \leftarrow (R, s)$, and outputs σ.

$\mathsf{Vrf}(\mathsf{pp}, L = \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m, \sigma) \to \{0, 1\}$: Given a public parameter pp, a multiset of public keys $\{\{\mathsf{pk}_i\}\}_{i\in[n]} = \{\{X_i\}\}_{i\in[n]}$, a message m, and a signature $\sigma = (R, s)$ as input, it computes $c_i \leftarrow H_1(X_i||R||L||m)$ for all $1 \le i \le n$. If it holds that $g^s = R \prod_{i=1}^n X_i^{c_i}$, then it outputs 1, otherwise outputs 0.

Key Substitution Attacks on MS-BN. We show that MS-BN is not secure against key substitution attacks with the trivial key.

Theorem 4. *MS-BN does not satisfy NKS.*

Proof. We provide a PPT adversary $\mathcal{A}$ that breaks NKS of MS-BN as follows. We first describe how $\mathcal{A}$ sets keys, messages, and a signature. Let $\mathcal{M}$ be a message space. The adversary $\mathcal{A}$ sets the first multiset of public keys $X_1^*, X_2^* \leftarrow 1_G$, chooses a message $m^* \xleftarrow{U} \mathcal{M}$ and an integer $s \xleftarrow{U} \mathbb{Z}_p$ randomly, computes $R \leftarrow g^s$, and sets a signature $\sigma \leftarrow (R, s)$. It sets the second multiset of public keys $X_1^{**}, X_2^{**}, X_3^{**} \leftarrow 1_G$ and chooses the second message $m^{**} \xleftarrow{U} \mathcal{M} \setminus \{m^*\}$.

Let $c_i^* = H_1(X_i^*||R||\{\{X_1^*, X_2^*\}\}||m^*)$ and $c_i^{**} = H_1(X_i^{**}||R||\{\{X_1^{**}, X_2^{**}\}\}||m^{**})$. Observe that it holds that $\prod_{i=1}^2 (X_i^*)^{c_i^*} = \prod_{i=1}^2 1_G^{c_i^*} = 1_G$ and $\prod_{i=1}^3 (X_i^{**})^{c_i^{**}} = \prod_{i=1}^3 1_G^{c_i^{**}} = 1_G$. Hence, the signature σ is accepted by the verification algorithm with both $(\{\{X_1^*, X_2^*\}\}, m^*)$ and $(\{\{X_1^{**}, X_2^{**}, X_3^{**}\}\}, m^{**})$. □

Theorem 5. *MS-BN satisfies wNKS in the random oracle model.*

Proof. We show that there are collisions of random numbers if there exists a PPT algorithm that breaks wNKS of the MS-BN. Suppose that there exists a PPT algorithm $\mathcal{A}$ that breaks wNKS of MS-BN with non-negligible probability. Let Σ_{MS} be the MS-BN scheme. In the experiment $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda)$, let $(m^*, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}, (R,s))$ be the final input to $\mathcal{A}$ and $(m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]})$ be the final output from $\mathcal{A}$ that results in $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$. Since it holds that $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$, we have that $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}} = g^s/R$. Without loss of generality, we may assume that $\mathcal{A}$ queries $(X_i^{**}||R||\{\{X_k^{**}\}\}_{k\in[n^{**}]}||m^{**})$ $(1 \le i \le n^{**})$ to the hash oracle before it outputs $(m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]})$. We show that $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*}$ and $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}}$ are uniformly random and independently distributed.

First, we show that it holds that $\exists i \in [n^{**}], x_i^{**} \neq 0$. Suppose for the first case that $\forall i \in [n^{**}], x_i^{**} = 0$. As the verification succeeds, we have that $g^s/R = \prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}} = 1_G$. Also, we have that $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*} = g^{\sum_{i=1}^{n^*} x_i^* c_i^*} = g^s/R = 1_G$. However, x_i^* and c_i^* are chosen uniformly randomly, so the probability that $\sum_{i=1}^{n^*} x_i^* c_i^* \equiv 0 \pmod p$ is negligible. Therefore, the probability that it holds $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$ is negligible unless $\exists i \in [n^{**}], x_i^{**} \neq 0$.

Then, we can assume that $\exists i \in [n^{**}], x_i^{**} \neq 0$. For some $1 \le j \le n^{**}$ such that $x_j^{**} \neq 0$, we denote the multiplicity in the set $\{\{X_i^{**}\}\}_{i\in[n^{**}]}$ by $t_j^{**} = \sharp\{X \in \{\{X_i^{**}\}\}_{i\in[n^{**}]} | X = g^{x_j^{**}}\}$. For all $1 \le i \le n^{**}$, we have that $c_i^{**} \leftarrow H_1(X_i^{**}||R||\{\{X_k^{**}\}\}_{k\in[n^{**}]}||m^{**})$, so $\{\{X_k^{**}\}\}_{k\in[n^{**}]}$ has been uniquely determined before c_i^{**} is determined. For all $1 \le k \le n^{**}$, x_k^{**} is uniquely determined such that $X_k^{**} = g^{x_k^{**}}$ when X_k^{**} is determined. Thus, c_i^{**} is determined after $\{\{x_k^{**}\}\}_{k\in[n^{**}]}$ and t_i^{**} are determined. Since c_j^{**} is chosen uniformly randomly, $t_j^{**}c_j^{**}x_j^{**}$ is also uniformly random. For $I = \{i \in [n^{**}] | x_i \neq x_j\}$, $\sum_{i=1}^{n^{**}} c_i^{**}x_i^{**} = t_j^{**}c_j^{**}x_j^{**} + \sum_{i\in I}^{n^{**}} c_i^{**}x_i^{**}$ is uniformly random. Therefore, $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}} = g^{\sum_{i=1}^{n^{**}} c_i^{**}x_i^{**}} = g^{t_j^{**}c_j^{**}x_j^{**} + \sum_{i\in I}^{n^{**}} c_i^{**}x_i^{**}}$ is uniformly random.

Let $c_i^* \leftarrow H_1(X_i^*||R||\{\{X_k^*\}\}_{k\in[n^*]}||m^*)$, and we have $g^s/R = \prod_{i=1}^{n^*}(X_i^*)^{c_i^*} = g^{\sum_{i=1}^{n^*} x_i^* c_i^*}$. Since x_i^* and c_i^* are chosen uniformly randomly, $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*}$ is also uniformly random. The distributions of c_i^* and c_i^{**} are independent because it holds that $\{\{X_i^*\}\}_{i\in[n^*]} \neq \{\{X_i^{**}\}\}_{i\in[n^{**}]}$. Therefore, $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$ means the collision of $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*}$ and $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}}$ whereas they are uniformly random and independently distributed.

Since $\mathcal{A}$ is a PPT algorithm, it can query at most polynomial number of times. Thus, if such an $\mathcal{A}$ exists, we can construct an algorithm that finds collisions of random numbers chosen from an exponential-size set in polynomial time. □

5.3 Modified MS-BN

We modify KeyGen and Vrf to prohibit the trivial key so MS-BN has NKS.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Given a public parameter pp as input, it chooses $x \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}$, computes $X \leftarrow g^x$, sets $\mathsf{sk} \leftarrow x$ and $\mathsf{pk} \leftarrow X$, and outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Vrf}(\mathsf{pp}, L = \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m, \sigma) \to \{0, 1\}$: It takes a public parameter pp, a multiset of public keys $\{\{\mathsf{pk}_i\}\}_{i\in[n]} = \{\{X_i\}\}_{i\in[n]}$, a message m, and a signature $\sigma = (R, s)$ as input. It outputs 0 if it holds that $\exists i (1 \leq i \leq n), X_i = 1$. Otherwise, it computes $c_i \leftarrow H_1(X_i||R||L||m)$ for all $1 \leq i \leq n$. If it holds that $g^s = R\prod_{i=1}^{n} X_i^{c_i}$, then it outputs 1, otherwise outputs 0.

Security against Key Substitution Attacks. We can prove the modified MS-BN has NKS in almost the same way as wNKS of original MS-BN.

Theorem 6. *Modified MS-BN satisfies NKS in the random oracle model.*

Due to the space limitation, we put the proof in Appendix D.

Unforgeability. The difference in input distribution between the original one and the modified one is negligibly small, so unforgeability of the modified one can be reduced to that of the original one.

Theorem 7. *Modified MS-BN satisfies the UF-MS security if MS-BN satisfies the UF-MS security.*

Due to the space limitation, we put the proof in Appendix E.

5.4 MuSig2

MuSig2 is proposed to be used as a replacement for Schnorr signatures in Bitcoin [16]. It consists of the four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ that work as follows:

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: Given a security parameter 1^λ as input, it obtains (G, p, g) by running $\mathcal{P}(1^\lambda)$, chooses three hash functions $H_{agg}, H_{non}, H_{sig} : \{0,1\}^* \to \mathbb{Z}_p$, and outputs a public parameter $\mathsf{pp} \leftarrow (G, p, g, H_{agg}, H_{non}, H_{sig})$.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Given a public parameter pp as input, it chooses $x \xleftarrow{U} \mathbb{Z}_p$, computes $X \leftarrow g^x$, sets $\mathsf{sk} \leftarrow x$ and $\mathsf{pk} \leftarrow X$, and outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{KeyAgg}(\mathsf{pp}, L = \{\{X_i\}\}_{i\in[n]}) \to \tilde{X}$: Given a public parameter pp and a multiset of public keys $L = \{\{X_i\}\}_{i\in[n]}$ as input, it computes $a_i \leftarrow H_{agg}(L, X_i)$ for $i \in [n]$ and $\tilde{X} \leftarrow \prod_{i=1}^{n} X_i^{a_i}$, and outputs $\tilde{X}$.

$\langle\{\mathsf{Sign}(\mathsf{sk}_i)\}_{i=1}^{n}\rangle(\mathsf{pp}, \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m) \to \sigma$: Sign is an interactive protocol among $n = \mathrm{poly}(\lambda)$ signers which consists of the following five rounds. Each round is conducted by each signer locally. Each signer considers their own index as 1 and executes each round accordingly.

$\mathsf{Round1}(\mathsf{pp}) \to (out_1, state_1)$: Given a public parameter pp, it computes $r_{1,j} \xleftarrow{U} \mathbb{Z}_p, R_{1,j} \leftarrow g^{r_{1,i}}$ for all $j \in [v]$, sets $out_1 \leftarrow (R_{1,1}, \cdots, R_{1,v})$ and $state_1 \leftarrow (r_{1,1}, \cdots, r_{1,v})$, and outputs out_1 and $state_1$.

$\mathsf{Round1Agg}(\mathsf{pp}, \{\{out_i\}\}_{i\in[n]}) \to out$: Given a public parameter pp and a multiset of a sequence of group elements $\{\{out_i\}\}_{i\in[n]}$, it computes $R_j \leftarrow \prod_{i=1}^{n} R_{i,j}$, and outputs $out \leftarrow (R_j)_{j\in[v]}$.

$\mathsf{Round2}(state_1, out, sk_1, m, \{\{pk_i\}\}_{2\le i\le n}) \to (state'_1, out'_1)$: Given integers $state_1$, a sequence of group elements out, a secret key sk_1, a message m, and a multiset of public keys $\{\{pk_i\}\}_{2\le i\le n}$, it sets $(r_{1,j})_{j\in[v]} \leftarrow state_1, x_1 \leftarrow sk_1, X_1 \leftarrow g^{x_1}, \{\{X_i\}\}_{2\le i\le n} \leftarrow \{\{pk_i\}\}_{2\le i\le n}$, and $L \leftarrow \{\{X_i\}\}_{i\in[n]}$, computes $a_1 \leftarrow H_{agg}(L, X_1)$, $\tilde{X} \leftarrow \mathsf{KeyAgg}(L)$, $(R_i)_{i\in[v]} \leftarrow out$, $b \leftarrow H_{non}(\tilde{X}, (R_j)_{j\in[v]}, m)$, $R \leftarrow \prod_{j=1}^{v} R_j^{b^{j-1}}$, $c \leftarrow H_{sig}(\tilde{X}, R, m)$, and $s_1 \leftarrow (ca_1x_1 + \sum_{j=1}^{v} r_{1,j}b^{j-1}) \mod p$, sets $state'_1 \leftarrow R$ and $out'_1 \leftarrow s$, and outputs $state'_1$ and out'_1.

$\mathsf{Round2Agg}(\{\{out'_i\}\}_{i\in[n]}) \to out'$: Given a multiset of integers $\{\{s_i\}\}_{i\in[n]} \leftarrow \{\{out'_i\}\}_{i\in[n]}$ as input, it computes $s \leftarrow \sum_{i=1}^{n} s_i \mod p$, and outputs $out' \leftarrow s$.

$\mathsf{Round3}(state'_1, out') \to \sigma$: Given $state'_1$ and out', it computes $R \leftarrow state'_1$ and $s \leftarrow out'$, and outputs a signature $\sigma \leftarrow (R, s)$.

$\mathsf{Vrf}(\mathsf{pp}, \mathsf{KeyAgg}(\mathsf{pp}, L = \{\{X_i\}\}_{i\in[n]}), m, \sigma) \to \{0, 1\}$: Given a public parameter pp, an aggregated public key $\tilde{X}$, a message m, and a signature $\sigma = (R, s)$ as input, it computes $c \leftarrow H_{sig}(\tilde{X}, R, m)$. If it holds that $g^s = R\tilde{X}^c$ then it outputs 1, otherwise outputs 0.

Key Substitution Attacks on MuSig2. We show MuSig2 is not secure against key substitution attacks with the trivial key in almost the same way as MS-BN.

Theorem 8. *MuSig2 does not satisfy NKS.*

Due to the space limitation, we put the proof in Appendix F.

Theorem 9. *MuSig2 satisfies wNKS in the random oracle model.*

Due to the space limitation, we put the proof in Appendix G.

5.5 Modified MuSig2

We modify KeyGen to prohibit the trivial key so that MuSig2 satisfies NKS.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Given a public parameter pp as input, it chooses $x \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}$, computes $X \leftarrow g^x$, sets $\mathsf{sk} \leftarrow x$ and $\mathsf{pk} \leftarrow X$, and outputs a public/secret key pair $(\mathsf{pk}, \mathsf{sk})$.

Security against Key Substitution Attacks.

Theorem 10. *Modified MuSig2 satisfies NKS in the random oracle model.*

Due to the space limitation, we put the proof in Appendix H.

Unforgeability. We can prove the unforgeability of the modified MuSig2 in the exact same way as MS-BN.

Theorem 11. *Modified MuSig2 satisfies the UF-MS security if MuSig2 satisfies the UF-MS security.*

6 Key Substitution Attacks on BGOY OMS

We first define NKS and wNKS of ordered multi-signatures. Then, we introduce the construction of a multi-signature scheme by Boldyreva et al. [6] (BGOY OMS), and demonstrate that it does not satisfy wNKS. Finally, we modify the scheme and upgrade the security to wNKS.

Definition 12 (Non-Key Substitutability). *An ordred multi-signature* $\Sigma_{OMS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{OSign}, \mathsf{OVrf})$ *satisfies the non-key substitutability if it holds that* $Pr[\mathsf{ExpOMS}^{NKS}_{\Sigma_{OMS},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ *for all PPT algorithms* $\mathcal{A}$ *where* $\mathsf{ExpOMS}^{NKS}_{\Sigma_{OMS},\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpOMS}^{NKS}_{\Sigma_{OMS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \Sigma_{OMS}.\mathsf{Setup}(1^\lambda)$;
$(m^*, (\mathsf{pk}_i^*)_{i\in[n^*]}, m^{**}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]}, \sigma) \leftarrow \mathcal{A}(\mathsf{pp})$:
Output 1 if $\Sigma_{OMS}.\mathsf{OVrf}(\mathsf{pp}, m^*, \sigma, (\mathsf{pk}_i^*)_{i\in[n^*]}) = 1$
$\wedge \Sigma_{OMS}.\mathsf{OVrf}(\mathsf{pp}, m^{**}, \sigma, (\mathsf{pk}_i^{**})_{i\in[n^{**}]}) = 1$
$\wedge (m^*, (\mathsf{pk}_i^*)_{i\in[n^*]}) \neq (m^{**}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]})$ (as ordered set);
Otherwise output 0

Definition 13 (Weak Non-Key Substitutability).
An ordered multi-signature $\Sigma_{OMS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{OSign}, \mathsf{OVrf})$ *satisfies the weak non-key substitutability if it holds that* $Pr[\mathsf{ExpOMS}^{wNKS}_{\Sigma_{OMS},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ *for any stateful PPT algorithms* $\mathcal{A}$ *where* $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{OMS},\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpOMS}^{wNKS}_{\Sigma_{OMS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \Sigma_{OMS}.\mathsf{Setup}(1^\lambda)$;
$n^* \leftarrow \mathcal{A}(\mathsf{pp})$;
$(\mathsf{pk}_i^*, \mathsf{sk}_i^*) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for all $i \in [n^*]$;
$m^* \leftarrow \mathcal{A}((\mathsf{pk}_i^*)_{i\in[n^*]})$;
$\sigma_i \leftarrow \Sigma_{OMS}.\mathsf{OSign}(\mathsf{pp}, \mathsf{sk}_i^*, m^*, \sigma_{i-1}, (\mathsf{pk}_k^*)_{k\in[i]})$ for all $i \in [n^*]$;
$(m^{**}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]}) \leftarrow \mathcal{A}(\mathsf{pp}, m^*, \sigma_{n^*}, (\mathsf{pk}_i^*)_{i\in[n^*]})$:
Output 1 if $\Sigma_{OMS}.\mathsf{OVrf}(\mathsf{pp}, m^{**}, \sigma_{n^*}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]}) = 1$
$\wedge (m^*, (\mathsf{pk}_i^*)_{i\in[n^*]}) \neq (m^{**}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]})$ (as ordered set);
Otherwise output 0

6.1 BGOY

We discuss the scheme BGOY OMS proposed by Boldyreva et al. [6] as a concrete instantiation of the ordered multi-signature. As for a bilinear map, we here

consider type-I or type-II setting. Recall that we write G_1 and G_2 simply as G in type-I or type-II setting for simplicity.

Definition 14 (BGOY OMS). *BGOY OMS consists of the four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{OSign}, \mathsf{OVrf})$ *that work as follows.*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: Given a security parameter* 1^λ *as input, it obtains* (p, G, G_T, e) *by running* $\mathcal{G}(1^\lambda)$*, choose a random generator* $g \in G^*$ *and a hash function* $H : \{0,1\}^* \to G$*, and outputs a public parameter* $\mathsf{pp} = (p, G, G_T, e, g, H)$*.*
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$*: Given a public parameter* pp *as input, it chooses three integers* $s, t, u \xleftarrow{U} \mathbb{Z}_p$*, computes* $S \leftarrow g^s, T \leftarrow g^t$ *and* $U \leftarrow g^u$*, sets* $\mathsf{pk} \leftarrow (S, T, U)$ *and* $\mathsf{sk} \leftarrow (s, t, u)$*, and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{OSign}(\mathsf{pp}, \mathsf{sk}_i, m, \sigma_{i-1}, (\mathsf{pk}_j)_{j\in[i-1]}) \to \sigma_i$*: It takes a public parameter* pp*, a secret key* sk_i*, a message* m*, a signature-so-far* $\sigma_{i-1} = (Q_{i-1}, R_{i-1})$ *(set* $\sigma_0 = (1, 1)$ *for the first signing), and a sequence of public keys* $(\mathsf{pk}_j)_{j\in[i-1]}$ *as input. It outputs* $\perp$ *if* $\mathsf{OVrf}(\mathsf{pp}, m, \sigma_{i-1}, (\mathsf{pk}_j)_{j\in[i-1]}) = 0$*. Otherwise, it chooses an integer* $r \xleftarrow{U} \mathbb{Z}_p$*, computes* $R_i \leftarrow R_{i-1} \cdot g^r, X_i \leftarrow R_i^{t_i + iu_i}, Y_i \leftarrow (\prod_{j=1}^{i-1} T_j (U_j)^j)^r$*, and* $Q_i \leftarrow H(m)^{s_i} \cdot Q_{i-1} \cdot X_i \cdot Y_i$*, and outputs a signature* $\sigma_i \leftarrow (Q_i, R_i)$*.*
- $\mathsf{OVrf}(\mathsf{pp}, m, \sigma_i, (\mathsf{pk}_j)_{j\in[i]}) \to \{0,1\}$*: Given a public parameter* pp*, a signature* $\sigma_i = (Q, R)$*, a message* m*, and a sequence of public keys* $(\mathsf{pk}_j)_{j\in[i]}$ *as input, it outputs* 1 *if* $e(Q, g) = e(H(m), \prod_{j=1}^{i} S_j) \cdot e(\prod_{j=1}^{i} T_j (U_j)^j, R)$*, otherwise outputs* 0*.*

Theorem 12. *BGOY does not satisfy wNKS.*

Due to the space limitation, we put the proof in Appendix J.

6.2 Modified BGOY OMS

We add the following three modifications on the BGOY OMS so that it satisfies wNKS (it does not satisfy NKS though). First, we prohibit $s_i = 0$ for a secret key in the key generation algorithm. Second, we change the usage of hash from $H(m)^{s_i}$ to $H(m||\,\mathsf{pk}_i)^{is_i}$ in the signing algorithm. Third, we modify the verification algorithm according to the signing algorithm. We do not change the setup algorithm. The modified BGOY OMS works as follows:

- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$: Given a public parameter pp as input, it chooses random integers $t, u \leftarrow \mathbb{Z}_p$ and $s \leftarrow \mathbb{Z}_p \backslash \{0\}$, computes $S \leftarrow g^s, T \leftarrow g^t$, and $U \leftarrow g^u$, and outputs a public key $\mathsf{pk} \leftarrow (S, T, U)$ and a secret key $\mathsf{sk} \leftarrow (s, t, u)$.
- $\mathsf{OSign}(\mathsf{pp}, \mathsf{sk}_i, m, \sigma_{i-1}, (\mathsf{pk}_j)_{j\in[i-1]}) \to \sigma_i$: It takes a public parameter pp, a secret key sk_i, a message m, a so-far signature $\sigma_{i-1} = (Q_{i-1}, R_{i-1})$, and a sequence of public keys $(\mathsf{pk}_j)_{j\in[i-1]}$ as input. It outputs $\perp$ if $\mathsf{OVrf}(\mathsf{pp}, m, \sigma_{i-1}, (\mathsf{pk}_j)_{j\in[i-1]}) = 0$. Otherwise, it chooses a random integer $r \leftarrow \mathbb{Z}_p$, computes $R_i \leftarrow R_{i-1} \cdot g^r, X_i \leftarrow R_i^{t_i + iu_i}, Y_i \leftarrow (\prod_{j=1}^{i-1} T_j (U_j)^j)^r$, $Q_i \leftarrow H(m||\,\mathsf{pk}_i)^{is_i} \cdot Q_{i-1} \cdot X_i \cdot Y_i$, and outputs a signature $\sigma_i = (Q_i, R_i)$.

$\mathsf{OVrf}(\mathsf{pp}, m, \sigma_i, (\mathsf{pk}_j)_{j\in[i]}) \to \{0,1\}$: Given a public parameter pp, a signature $\sigma_i = (Q, R)$, a message m, and a sequence of public keys $(\mathsf{pk}_j)_{j\in[i]}$ as input, if $(\forall j \in [i], S_j \neq 1) \wedge e(Q,g) = \{\prod_{j=1}^{i} e(H(m||\,\mathsf{pk}_j), (S_j)^j)\} \cdot e(\prod_{j=1}^{i} T_j (U_j)^j, R_n)$, then it outputs 1, otherwise outputs 0.

Theorem 13. *Modified BGOY OMS satisfies the correctness.*

Due to the space limitation, we put the proof in Appendix K.

Key Substitution Attacks on Modified BGOY OMS.

Theorem 14. *Modified BGOY OMS does not satisfy NKS.*

Due to the space limitation, we put the proof in Appendix L.

Theorem 15. *Modified BGOY OMS satisfies wNKS in the random oracle model under the DBP assumption.*

Due to the space limitation, we put the proof in Appendix M.

Unforgeability. Unforgeability of Modified BGOY OMS can be proven in almost the same way as the original one [6].

Theorem 16. *Modified BGOY OMS satisfies the UF-OMS security in the random oracle model under the CDH assumption.*

Due to the space limitation, we put the proof in Appendix N.

7 Conclusion

In this paper, we explored security against key substitution attacks of a wide range of aggregate signature schemes and multi-signer signature schemes, and if not secure, we proposed new schemes. It remains to consider additional properties such as message-bound and non re-signability of multi-signer signatures.

References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. pp. 209–236. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
2. An, Y., Lee, H.S., Lee, J., Lim, S., D'Antonio, S.: Key substitution attacks on lattice signature schemes based on sis problem. Sec. and Commun. Netw. **2018**, 8525163:1–8525163:13 (jan 2018)
3. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. p. 390–399. CCS '06, Association for Computing Machinery, New York, NY, USA (2006)

4. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (sts) protocol. In: Public Key Cryptography. pp. 154–170. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
5. Bohli, J.M., Röhrich, S., Steinwandt, R.: Key substitution attacks revisited: Taking into account malicious signers. International Journal of Information Security **5**, 30–36 (2005)
6. Boldyreva, A., Gentry, C., O'Neill, A., Yum, D.H.: Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. p. 276–285. CCS '07, Association for Computing Machinery, New York, NY, USA (2007)
7. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. pp. 416–432. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
8. Boneh, D., Gorbunov, S., Wahby, R.S., Wee, H., Wood, C.A., Zhang, Z.: Bls signatures. Internet Draft draft-irtf-cfrg-bls-signature-05 (dec 2022)
9. Brendel, J., Cremers, C., Jackson, D., Zhao, M.: The provable security of ed25519: Theory and practice. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1659–1676 (2021)
10. Cremers, C.J.F., Düzlü, S., Fiedler, R., Fischlin, M., Janson, C.: Buffing signature schemes beyond unforgeability and the case of post-quantum signatures. 2021 IEEE Symposium on Security and Privacy (SP) pp. 1696–1714 (2021)
11. Jackson, D., Cremers, C., Cohn-Gordon, K., Sasse, R.: Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2165–2180. CCS '19, Association for Computing Machinery, New York, NY, USA (2019)
12. Knapp, E.: On Pairing-Based Signature and Aggregate Signature Schemes. Master's thesis, University of Waterloo (2008)
13. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. pp. 74–90. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
14. Menezes, A., Smart, N.: Security of signature schemes in a multi-user setting. Des. Codes Cryptography **33**, 261–274 (11 2004)
15. Nick, J., Ruffing, T., Jin, E.: Musig2 for bip340-compatible multi-signatures. BIP327 (mar 2022)
16. Nick, J., Ruffing, T., Seurin, Y.: Musig2: Simple two-round schnorr multi-signatures. In: Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I. p. 189–221. Springer-Verlag, Berlin, Heidelberg (2021)
17. Park, S., Sealfon, A.: It wasn't me! repudiability and unclaimability of ring signatures. In: Annual International Cryptology Conference. pp. 159–190. Springer (2019)
18. Pornin, T., Stern, J.P.: Digital signatures do not guarantee exclusive ownership. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 138–150. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
19. Richard, B., Jacob, H.A., James, K.: Automatic certificate management environment (ACME) (2015)

20. Sakai, Y., Schuldt, J.C.N., Emura, K., Hanaoka, G., Ohta, K.: On the security of dynamic group signatures: Preventing signature hijacking. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography – PKC 2012. pp. 715–732. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
21. Yamashita, K., Hara, K., Watanabe, Y., Yanai, N., Shikata, J.: Designated verifier signature with claimability. In: Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop. p. 21–32. APKC '23 (2023)

A Assumptions

We use the following computational hardness assumption in this paper.

Definition 15 (The DDH assumption). *The decisional Diffie-Hellman (DDH) assumption in G_1 holds with respect to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$, it holds that*

$$|\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); a, b \xleftarrow{U} \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, g_1^a, g_1^b, g_1^{ab}) = 1] - \Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); a, b, c, \xleftarrow{U} \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, g_1^a, g_1^b, g_1^c) = 1]| \leq \mathrm{negl}(\lambda).$$

Definition 16 (The CDH assumption). *The computational Diffie-Hellman (CDH) assumption in G_1 holds with respect to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$, it holds that*

$$Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); g_1 \xleftarrow{U} G^*; a, b \xleftarrow{U} \mathbb{Z}_p : g_1^{ab} = \mathcal{A}(\mathsf{gk}, g_1, g_1^a, g_1^b)] \leq negl(\lambda)$$

Definition 17 (The co-CDH assumption). *The co-computational Diffie-Hellman (co-CDH) assumption holds with respect to $\mathcal{G}$ if for all PPT algorithms $\mathcal{A}$ it holds that*

$$\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); h \xleftarrow{U} G_1; a \xleftarrow{U} \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, h, g_2^a) = h^a] \leq \mathrm{negl}(\lambda).$$

B Proof of Lemma 2: Unforgeability of Modified BGLS

B.1 Requirements of Aggregate Signature Schemes

An aggregate signature scheme $\Sigma_{\mathrm{AS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Agg}, \mathsf{AggVrf})$ satisfies correctness if the following conditions are satisfied.

- For any $\lambda \in \mathbb{N}$ and any m, it holds that $\mathsf{Vrf}(\mathsf{pp}, \mathsf{pk}, m, \sigma) = 1$ where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m)$.
- For any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any $m_1, \ldots, m_n$, it holds that $\mathsf{AggVrf}(\mathsf{pp}, ((\mathsf{pk}_i, m_i))_{i\in[n]}, \tau) = 1$ where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\forall i \in [n]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_i, m_i)$, and $\tau \leftarrow \mathsf{Agg}(\mathsf{pp}, ((\mathsf{pk}_i, m_i, \sigma_i))_{i\in[n]})$.

Unforgeability of an aggregate signature scheme is defined as follows.

Definition 18 (EUF-CMA security). *An aggregate signature scheme $\Sigma_{\mathrm{AS}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf}, \mathsf{Agg}, \mathsf{AggVrf})$ satisfies existentially unforgeability under chosen-message attacks (EUF-CMA) if for any $n = \mathrm{poly}(\lambda)$, and any PPT adversary $\mathcal{A}$, it holds that $\Pr[\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ where*

$\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Sigma_{\mathrm{AS}},\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Setup}(1^\lambda)$;
$(\mathsf{pk},\mathsf{sk}) \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{KeyGen}(\mathsf{pp}); Q := \emptyset$;
$(n,(m_i)_{i\in[n]},\tau,(\mathsf{pk}_i)_{i\in[n]}) \leftarrow \mathcal{A}^{\Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{pp},\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{pk})$:
Output 1 if $\exists i \in [n]$: *(*$\mathsf{pk}_i = \mathsf{pk}$ *and* $m_i \notin Q$*)*
$\wedge\, \Sigma_{\mathrm{AS}}.\mathsf{AggVrf}(\mathsf{pp},((\mathsf{pk}_i,m_i))_{i\in[n]},\tau) = 1$;
Otherwise output 0

where when $\mathcal{A}$ *makes a query* m *to the oracle* $\Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{pp},\mathsf{sk}_1,\cdot)$*, it computes* $\sigma \leftarrow \Sigma_{\mathrm{AS}}.\mathsf{Sign}(\mathsf{pp},\mathsf{sk}_1,m)$*, returns* σ *to* $\mathcal{A}$*, and sets* $Q \leftarrow Q \cup \{m\}$*.*

B.2 The co-CDH′ Assumption

Definition 19. *The co-CDH′ assumption holds with respect to* $\mathcal{G}$ *if for all PPT algorithms* $\mathcal{A}$ *it holds that*

$$\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); a,b \xleftarrow{U} \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}, g_1^a, g_2^a, g_2^b) = g_1^{ab}] \leq \mathrm{negl}(\lambda).$$

It is known that the co-CDH′ assumption is equivalent to the CDH assumption in type-I setting, and to the co-CDH assumption in type-II setting [12].

B.3 Proof of Lemma 2

First, we provide an overview of the proof. We prove Lemma 3 and Lemma 4, that are necessary as we prohibit $\mathsf{pk} = 1$. Then, by Lemma 5, we construct a PPT adversary $\mathcal{A}'$ against the co-CDH′ assumption and show that it indeed breaks the co-CDH′ assumption with non-negligible probability if there exists a PPT adversary $\mathcal{A}$ that violates unforgeability of Π_{BP} with non-negligible probability.

Assume for contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the EUF-CMA security of Π_{BP} with non-negligible probability ϵ. We construct a PPT adversary $\mathcal{A}'$ that breaks the co-CDH′ assumption with non-negligible probability, if such $\mathcal{A}$ exists. Let $n = \mathrm{poly}(\lambda)$ be the maximum size of a multiset in a forgery by $\mathcal{A}$. We assume that $\mathcal{A}$ makes at most $q_S = \mathrm{poly}(\lambda)$ queries to the signing oracle.

We construct an algorithm $\mathcal{A}'$ that solves the following modified version of the co-CDH′ problem:

$$\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); a \leftarrow \mathbb{Z}_p; b \leftarrow \mathbb{Z}_p \setminus \{0\} : \mathcal{A}(\mathsf{gk}, g_1^a, g_1^b, g_2^b) = g_1^{ab}].$$

We claim that if we assume the co-CDH′ assumption, then the above probability is negligible for any adversary $\mathcal{A}$.

Lemma 3. *Assume the co-CDH′ assumption. For any PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr[\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda); a \leftarrow \mathbb{Z}_p; b \leftarrow \mathbb{Z}_p \setminus \{0\} : \mathcal{A}(\mathsf{gk}, g_1^a, g_1^b, g_2^b) = g_1^{ab}] \leq \mathrm{negl}(\lambda).$$

Proof: Given an adversary $\mathcal{A}$, we consider the following distinguisher that distinguishes the uniform distributions $b \xleftarrow{U} \mathbb{Z}_p$ and $b \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}$: Given a sample b, the distinguisher simulates the co-CDH$'$ experiment except for using the given b for the exponent b in the experiment; if $\mathcal{A}$ terminates, the distinguisher checks whether the winning condition is satisfied; if the winning condition is satisfied, the distinguisher outputs 1 and outputs 0 otherwise. The distinguisher's advantage is equal to the absolute difference between the probabilities that $\mathcal{A}$ wins in the original or modified co-CDH$'$ experiments. Furthermore, the advantage is smaller than or equal to the statistical distance between two uniform distributions, which is equal to $1/p$. Thus, the winning probabilities of the two experiments are negligibly close, and one of them is negligible by the assumption. Then the other is also negligible. □

To construct an algorithm $\mathcal{A}'$ that solves the above variant of the co-CDH$'$ problem, we consider an experiment which is a modification of $\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda)$ where the random oracle H_P returns a uniformly random element in $G \setminus \{1\}$. We denote this modified experiment by $\mathsf{ExpAS}^{EUF\text{-}CMA'}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda)$. Since the statistical distance between the uniformly random variables over G and $G \setminus \{1\}$ is $1/p$, by a hybrid argument, the statistical distance between two experiments is bounded by q_{H_P}/p.

Lemma 4. *For any PPT $\mathcal{A}$, it holds that*

$$|\Pr[\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpAS}^{EUF\text{-}CMA'}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda) = 1]|$$

is negligible.

Proof: Since the statistical distance between the uniform distributions $h \xleftarrow{U} G_1$ and $h \xleftarrow{U} G_1 \setminus \{1\}$ is $1/p$, the statistical distance between the uniform distributions $\vec{h} \xleftarrow{U} (G_1)^{q_{H_P}}$ and $\vec{h} \xleftarrow{U} (G_1 \setminus \{1\})^{q_{H_P}}$ is smaller than or equal to q_{H_P}/p, by a hybrid argument. Then, the difference $|\Pr[\mathsf{ExpAS}^{EUF\text{-}CMA}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpAS}^{EUF\text{-}CMA'}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda) = 1]|$ is smaller than or equal to q_{H_P}/p, if we let q_{H_P} be the maximum number of $\mathcal{A}$'s hash queries to H_P. Since q_{H_P}/p is negligible, the lemma holds. □

Then we describe an algorithm $\mathcal{A}'$ that interacts with $\mathcal{A}$ by simulating the modified experiment $\mathsf{ExpAS}^{EUF\text{-}CMA'}_{\Pi_{\mathsf{BP}},\mathcal{A}}(\lambda)$. We assume that before issuing a signing query m, $\mathcal{A}$ issues an H query $m \,\|\, \mathsf{pk}$ and that before outputting a forgery $(\{(m_i, \hat{\mathsf{pk}}_i)\}_{i\in[n]}, \sigma)$, $\mathcal{A}$ issues an H_P queries pk_i where $(\mathsf{pk}_i, \sigma_{\mathsf{pop},i}) \leftarrow \hat{\mathsf{pk}}_i$.

- **Setup.** The algorithm $\mathcal{A}'$ is given as input $(\mathsf{gk}, g_1^a, g_1^b, g_2^b)$. The algorithm $\mathcal{A}'$ maintains two hash lists for H and H_P, which are initialized as empty. Then $\mathcal{A}'$ sets $\mathsf{pk}^* \leftarrow g_2^b$, chooses $z \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}$, records (pk^*, z) to the hash list for H_P, and computes $\sigma^*_{\mathsf{pop}} \leftarrow (g_1^b)^z$. Finally, $\mathcal{A}'$ sets $\mathsf{pp} \leftarrow \mathsf{gk}$ and $\hat{\mathsf{pk}}^* \leftarrow (\mathsf{pk}^*, \sigma^*_{\mathsf{pop}})$ and runs $\mathcal{A}(\mathsf{pp}, \hat{\mathsf{pk}}^*)$.
- H **query.** Given an H query $m \,\|\, \mathsf{pk}$, if $\mathsf{pk} \neq \mathsf{pk}^*$, then $\mathcal{A}'$ searches for an entry $(m \,\|\, \mathsf{pk}, w, \perp)$ for some w in the hash list for H. If not found, $\mathcal{A}'$ chooses

$w \xleftarrow{U} \mathbb{Z}_p$ and records $(m \,\|\, \mathsf{pk}, w, \perp)$ to the hash list for H. Then $\mathcal{A}'$ returns g_1^w. If $\mathsf{pk} = \mathsf{pk}^*$, then $\mathcal{A}'$ searches for an entry $(m \,\|\, \mathsf{pk}^*, w, c)$ for some w and c in the hash list for H. If not found, $\mathcal{A}'$ chooses $w \xleftarrow{U} \mathbb{Z}_p$ and $c \in \{0, 1\}$ which is 0 with probability $1/(q_S + n)$ and 1 otherwise. Then $\mathcal{A}'$ records $(m \,\|\, \mathsf{pk}^*, w, c)$ to the hash list for H. If $b = 0$, $\mathcal{A}'$ returns $g_1^a g_1^w$. If $b = 1$, $\mathcal{A}'$ returns g_1^w.

- H_P **query.** Given an H_P query pk, if there is an entry (pk, z) for some z in the hash list for H_P, then $\mathcal{A}'$ returns g_1^z. If there is no entry of this form, $\mathcal{A}'$ chooses $z \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}$, records (pk, z) to the hash list for H_P. Then $\mathcal{A}'$ returns g_1^z.
- **Signing query.** Given a signing query m, $\mathcal{A}'$ retrieves an entry $(m \,\|\, \mathsf{pk}^*, w, b)$ and checks if $b = 1$. If not, $\mathcal{A}'$ terminates the simulation and output $\perp$. If $b = 1$, $\mathcal{A}'$ returns $(g_1^b)^w$.
- **Forgery.** When $\mathcal{A}$ outputs a forgery $(\{(m_i, \hat{\mathsf{pk}}_i)\}_{i \in [n]}, \sigma)$, $\mathcal{A}'$ verifies the winning condition. If the winning condition is not satisfied, $\mathcal{A}'$ terminates and outputs $\perp$. Then $\mathcal{A}'$ finds some $i^* \in [n]$ satisfying that m_{i^*} is not issued as a signing query, $\mathcal{A}'$ parses $\hat{\mathsf{pk}}_i$ as $(\mathsf{pk}_i, \sigma_{\mathsf{pop},i})$ and computes the following three sets

$$I = \{i \in [n] \mid m_i = m_{i^*} \wedge \mathsf{pk}_i = \mathsf{pk}^*\},$$
$$J = \{i \in [n] \mid m_i \neq m_{i^*} \wedge \mathsf{pk}_i = \mathsf{pk}^*\},$$
$$K = \{i \in [n] \mid \mathsf{pk}_i \neq \mathsf{pk}^*\}.$$

Then $\mathcal{A}'$ lets l be the modular inverse of $|I|$ modulo p. If l is a multiple of p, then $\mathcal{A}'$ terminates and outputs $\perp$. For all $i \in J$, $\mathcal{A}'$ confirms that $c = 1$ where $(m_i \,\|\, \mathsf{pk}^*, w, c)$ is in the hash list for H. If some $i \in J$ does not satisfy this, then $\mathcal{A}'$ terminates and outputs $\perp$. If all $i \in J$ satisfy this, then $\mathcal{A}'$ computes

$$\sigma_i \leftarrow (g_1^b)^w.$$

For all $i \in K$, $\mathcal{A}'$ computes

$$\sigma_i \leftarrow (\sigma_{\mathsf{pop},i})^{w/z}$$

where $(m_i \,\|\, \mathsf{pk}_i, w, \perp)$ is in the hash list for H and (pk_i, z) is in the hash list for H_P. Finally, $\mathcal{A}'$ confirms that $c = 0$ where $(m_{i^*} \,\|\, \mathsf{pk}^*, w, c)$ is in the hash list for H. If not, $\mathcal{A}'$ terminates and outputs $\perp$. If it is, $\mathcal{A}'$ computes

$$h \leftarrow \Big(\sigma \prod_{i \in J \cup K} \sigma_i^{-1}\Big)^l (g_1^b)^{-w}$$

and outputs h.

We claim that if $\mathcal{A}'$ does not output $\perp$, then the output h is equal to g_1^{ab}.

Firstly we claim that for all $i \in J \cup K$, it holds that $e(\sigma_i, g_2) = e(H(m_i \,\|\, \mathsf{pk}_i), \mathsf{pk}_i)$. For $i \in J$, if $(m_i \,\|\, \mathsf{pk}^*, w, c)$ is in the hash list for H, it holds that

$$e(\sigma_i, g_2) = e((g_1^b)^w, g_2) = e(g_1^w, g_2^b) = e(H(m_i \,\|\, \mathsf{pk}_i), \mathsf{pk}^*).$$

For $i \in K$, if $(m_i \,\|\, \mathsf{pk}_i, w, \perp)$ is in the hash list for H and (pk_i, z) is in the hash list for H_P, then it holds that

$$\begin{aligned} e(\sigma_i, g_2) &= e((\sigma_{\mathsf{pop},i})^{w/z}, g_2) = e(\sigma_{\mathsf{pop},i}, g_2)^{w/z} \\ &= e(H_P(\mathsf{pk}_i), \mathsf{pk}_i)^{w/z} = e(H(m_i \,\|\, \mathsf{pk}_i), \mathsf{pk}_i). \end{aligned}$$

Then we claim that

$$e\Big(\Big(\sigma \prod_{i \in J \cup K} \sigma_i^{-1}\Big)^l, g_2\Big) = e(H(m_{i^*} \,\|\, \mathsf{pk}^*), \mathsf{pk}^*).$$

We have that

$$\begin{aligned} e(\sigma, g_2) &= \prod_{i \in [n]} e(H(m_i \,\|\, \mathsf{pk}_i), \mathsf{pk}_i) \\ &= \prod_{i \in I} e(H(m_{i^*} \,\|\, \mathsf{pk}^*), \mathsf{pk}^*) \prod_{i \in J \cup K} e(H(m_i \,\|\, \mathsf{pk}_i), \mathsf{pk}_i). \end{aligned}$$

As for all $i \in J \cup K$, it holds that $e(\sigma_i, g_2) = e(H(m_i \,\|\, \mathsf{pk}_i), \mathsf{pk}_i)$, the above equation is equivalent to

$$e(\sigma, g_2) = e(H(m_{i^*} \,\|\, \mathsf{pk}^*), \mathsf{pk}^*)^{|I|} \prod_{i \in J \cup K} e(\sigma_i, g_2).$$

Since $l \cdot |I| \equiv 1 \bmod p$, we have that

$$e\Big(\Big(\sigma \prod_{i \in J \cup K} \sigma_i^{-1}\Big)^l, g_2\Big) = e(H(m_{i^*} \,\|\, \mathsf{pk}^*), \mathsf{pk}^*).$$

We claim that the output h of $\mathcal{A}'$ is equal to g_1^{ab}. We have that

$$e\Big(\Big(\sigma \prod_{i \in J \cup K} \sigma_i^{-1}\Big)^l, g_2\Big) \quad = \quad e(H(m_{i^*} \,\|\, \mathsf{pk}^*), \mathsf{pk}^*) \quad = \quad e(g_1^a g_1^w, g_2^b)$$

where $(m_{i^*} \,\|\, \mathsf{pk}^*, w, c)$ is in the hash list for H. Thus we have

$$e\Big(\Big(\sigma \prod_{i \in J \cup K} \sigma_i^{-1}\Big)^l (g_2^b)^{-w}, g_2\Big) \quad = \quad e(g_1^a, g_2^b) \quad = \quad e(g_1^{ab}, g_2).$$

Due to the non-degenerateness of e, we have that $h = g_1^{ab}$.

Finally, we prove an upper bound on the probability that $\mathcal{A}$ wins by the probability that $\mathcal{A}'$ outputs $h \neq \perp$. Let ϵ' be the probability that $\mathcal{A}'$ outputs $h \neq \perp$.

To this end, we first argue that $|I|$ is not a multiple of p for sufficiently large security parameters. This is because $|I|$ is polynomially large and p is exponentially large. From now on, we assume that $|I|$ is not a multiple of p.

Then we define the following events.

- E_1. The algorithm $\mathcal{A}'$ does not terminate due to $\mathcal{A}$'s signing queries.
- E_2. The adversary $\mathcal{A}$ satisfies the winning condition.
- E_3. The event E_2 occurs. In addition, $H(m_{i^*} \,\|\, \mathsf{pk}^*)$ is responded with $c = 0$, and $H(m_i \,\|\, \mathsf{pk}^*)$ for $i \in J$ is responded with $c = 1$.

The algorithm $\mathcal{A}'$ successfully outputs $h \neq \perp$ if all the events happen. The probability $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1 \wedge E_3]$ is equal to

$$\Pr[E_1 \wedge E_3] = \Pr[E_1] \Pr[E_2 \mid E_1] \Pr[E_3 \mid E_1 \wedge E_2].$$

We give a lower bounds for each term.

Lemma 5. *The probability that $\mathcal{A}'$ does not terminate by $\mathcal{A}$'s signing queries is at least $(1 - 1/(q_S + n))^{q_S}$.*

Proof: Without loss of generality, we can assume that $\mathcal{A}$ does not repeat the same signing query twice. We prove by induction that after k signing queries, the probability that $\mathcal{A}'$ does not terminate by the signing query is at least $(1 - 1/(q_S + n))^k$. For $k = 0$, the statement is clearly true. For the k-th query m, the probability that $\mathcal{A}'$ does not terminates before this query is $(1 - 1/(q_S + n))^{k-1}$ by the induction hypothesis. At the k-th query, $\mathcal{A}'$ terminates if and only if the bit c for $m \,\|\, \mathsf{pk}^*$ is 0. This happens with probability $1/(q_S + n)$. The distribution of c is independent of $\mathcal{A}$'s view because $H(m \,\|\, \mathsf{pk}^*)$ is distributed independently of c. Then, after k-th query, $\mathcal{A}'$ does not terminates with probability at least $(1 - 1/(q_S + n))^k$. Since $\mathcal{A}$ issues at most q_S signing queries, the probability $\Pr[E_1]$ is at least $(1 - 1/(q_S + n))^{q_S}$. □

Lemma 6. *If $\mathcal{A}'$ does not terminates by $\mathcal{A}$'s signing queries, $\mathcal{A}$'s view is identical to that of the (modified) unforgeability game. Hence, we have that $\Pr[E_2 \mid E_1] = \epsilon$.*

Proof: The public key given to $\mathcal{A}$ is identically distributed as that in the (modified) unforgeability game. Responses to the random oracles and the signing oracle is also identically distributed conditioned on that $\mathcal{A}'$ does not terminates by signing queries. Then, the output of $\mathcal{A}$ interacting with $\mathcal{A}'$ is not equal to $\perp$ with probability ϵ. Hence we have that $\Pr[E_2 \mid E_1] = \epsilon$. □

Lemma 7. *After $\mathcal{A}$ outputs a successful forgery, it causes the event E_3 with probability at least $(1 - 1/(q_S + n))^{n-1} \cdot 1/(q_S + n)$.*

Proof: Let $(m_{i^*} \,\|\, \mathsf{pk}^*, w, c)$ be in the hash list for H and $(m_i \,\|\, \mathsf{pk}^*, w_i, c_i)$ be in the hash list for H for $i \in J$. For the event E_3 to occur, we need $c = 0$ and $c_i = 1$ for all $i \in J$. Since the bit c is independent of $\mathcal{A}$'s view, $c = 0$ with probability $1/(q_S + n)$. Regarding c_i, it is independent if m_i is not issued as a signing query. In that case, $c_i = 1$ with probability $1 - 1/(q_S + n)$. If m_i is issued as a signing query, then with probability 1, it holds that $c_i = 1$. In any case, with probability at least $1 - 1/(q_S + n)$, it holds that $c_i = 1$. Therefore, it holds that $\Pr[E_3 \mid E_1 \wedge E_2] \geq (1 - 1/(q_S + n))^{n-1} 1/(q_S + n)$. □

Finally, we bound the probability ϵ. From the lemmas, we have that

$$\begin{aligned}
\epsilon' &\geq \left(1-\frac{1}{q_S+n}\right)^{q_S}\cdot\epsilon\cdot\left(1-\frac{1}{q_S+n}\right)^{n-1}\cdot\frac{1}{q_S+n}\\
&= \left(1-\frac{1}{q_S+n}\right)^{q_S+n-1}\cdot\frac{1}{q_S+n}\cdot\epsilon\\
&= \left(\frac{q_S+n-1}{q_S+n}\right)^{q_S+n-1}\cdot\frac{1}{q_S+n}\cdot\epsilon\\
&= \left(\frac{q_S+n}{q_S+n-1}\right)^{-(q_S+n-1)}\cdot\frac{1}{q_S+n}\cdot\epsilon\\
&= \left(1+\frac{1}{q_S+n-1}\right)^{-(q_S+n-1)}\cdot\frac{1}{q_S+n}\cdot\epsilon\\
&\geq \frac{\epsilon}{e(q_S+n)}.
\end{aligned}$$

This inequality holds for all sufficiently large λ. This fact proves the theorem. □

C Requirements of Unforgeability of Multi-Signatures

A multi-signature scheme $\Sigma_{MS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ satisfies correctness if the following conditions are satisfied.

- For any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any m, it holds that $\mathsf{Vrf}(\mathsf{pp}, \{\{\mathsf{pk}_i\}\}_{i\in[n]}, m, \sigma)$ where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\forall i \in [n]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\sigma \leftarrow \langle\{\mathsf{Sign}(\mathsf{sk}_i^*)\}_{i=1}^{n^*}\rangle(\mathsf{pp}, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}, m^*)$.

Definition 20 (UF-MS). *A multi-signature $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vrf})$ satisfies existentially unforgeability under chosen-message attacks (UF-MS) if for any PPT algorithm $\mathcal{A}$, it holds that $\Pr[\mathsf{ExpMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ where $\mathsf{ExpMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda)$ is the following experiment.*

$\mathsf{ExpMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda)$
$Q \leftarrow \emptyset$;
$\mathsf{pp} \leftarrow \Sigma.\mathsf{Setup}(1^\lambda)$;
$(\mathsf{pk}^*, \mathsf{sk}^*) \xleftarrow{R} \mathsf{KeyGen}(\mathsf{pp})$;
$(L = \{\{\mathsf{pk}_i\}\}_{i=1}^n, m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{pp},\mathsf{sk}^*,\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk}^*)$;
Output 1 if $\Sigma.\mathsf{Vrf}(\mathsf{pp}, \sigma, m, L) = 1 \wedge pk^* \in L \wedge (L, m) \notin Q$;
Otherwise output 0

An adversary $\mathcal{A}$ can interact with the signing oracle $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}^*, \cdot, \cdot)$ to obtain a signature σ by making a query (L', m') where L' is the set of public keys that includes pk^*. It can run an arbitrary number of sessions to the signing oracle concurrently. When $\mathcal{A}$ makes a query (L', m') to the signing oracle, it sets $Q \leftarrow Q \cup \{(L', m')\}$.

D Proof of Theorem 6 : NKS of Modified MS-BN

We show that there are collisions of random numbers if there exists a PPT algorithm $\mathcal{A}$ that breaks NKS of the modified MS-BN. Suppose that there exists a PPT algorithm $\mathcal{A}$ that breaks NKS of modified MS-BN. Let Σ_{MS} be the modified MS-BN scheme. In the experiment $\mathsf{ExpMS}^{NKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda)$, let $(m^*, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}, m^{**}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}, \sigma)$ be the output of $\mathcal{A}$ where it results in $\mathsf{ExpMS}^{NKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$. Without loss of generality, we may assume that $\mathcal{A}$ queries $(X_i^*||R||\{\{X_i^*\}\}_{i\in[n^*]}||m^*)\,(1 \le i \le n^*)$ to the hash oracle before outputting a signature. We show that $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*}$ and $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}}$ are uniformly random and independently distributed.

Let j be an arbitrary number in $[n^*]$. we denote the multiplicity in the set $\{\{X_i^*\}\}_{i\in[n^*]}$ by $t_j^* = \sharp\{X \in \{\{X_i^*\}\}_{i\in[n^*]} | X = g^{x_j^*}\}$. For all $1 \le i \le n^*$, we have that $c_i^* \leftarrow H_1(X_i^*||R||\{\{X_k^*\}\}_{k\in[n^*]}||m^*)$, so $\{\{X_k^*\}\}_{k\in[n^*]}$ has been uniquely determined before c_i^* is determined. For all $1 \le k \le n^*$, x_k^* is uniquely determined such that $X_k^* = g^{x_k^*}$ when X_k^* is determined. Thus, c_i^* is determined after $\{\{x_k^*\}\}_{k\in[n^*]}$ and t_i^* are determined. Since c_j^* is chosen uniformly randomly, $t_j^* c_j^* x_j^*$ is uniformly random. If we let I denote $\{i \in [n^*] | x_i \neq x_j\}$, $\sum_{i=1}^{n^*} c_i^* x_i^* = t_j^* c_j^* x_j^* + \sum_{i\in I}^{n^*} c_i^* x_i^*$ is uniformly random. Therefore, $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*} = g^{\sum_{i=1}^{n^*} c_i^* x_i^*} = g^{t_j^* c_j^* x_j^* + \sum_{i\in I}^{n^*} c_i^* x_i^*}$ is uniformly random. We can show that $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}}$ is also uniformly random in the same manner.

Since we have $\{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]} \neq \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}$, the distributions of $(c_i^*)_{i\in n^*}$ and $(c_i^{**})_{i\in n^{**}}$ are independent, and hence the distributions of $t_j^* x_j^* c_j^*$ and $t_j^{**} x_j^{**} c_j^{**}$ are also independent. Therefore, $\mathsf{ExpMS}^{NKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$ means the collision of $\prod_{i=1}^{n^*}(X_i^*)^{c_i^*}$ and $\prod_{i=1}^{n^{**}}(X_i^{**})^{c_i^{**}}$ whereas they are uniformly random and independently distributed.

Since $\mathcal{A}$ is a PPT algorithm, it can query at most polynomial number of times, and hence the times of trials of collisions is also at most polynomial number. Thus, if such an $\mathcal{A}$ exists, we can construct an algorithm that finds collisions of random numbers chosen from an exponential-size set in polynomial time. □

E Proof of Theorem 7 : Unforgeability of Modified MS-BN

Here let Σ be the MS-BN scheme and Π be the modified MS-BN scheme. Let $\mathsf{AdvMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda)$ denote an adversary $\mathcal{A}$'s advantage in the $\mathsf{ExpMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda)$ experiment against the MS-BN scheme. Then we introduce a modified $\mathsf{ExpMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda)$ experiment, denoted by the $\mathsf{ExpMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)$ experiment, in which the way to choose a challenge key is changed from $\mathsf{sk}^* \xleftarrow{U} \mathbb{Z}_p, \mathsf{pk}^* \leftarrow g^x$ to $\mathsf{sk}^* \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}, \mathsf{pk}^* \leftarrow g^x$. Let $\mathsf{AdvMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)$ denote $\mathcal{A}$'s advantage in the $\mathsf{ExpMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)$ experiment against the MS-BN scheme.

We show that the modified MS-BN scheme is unforgeable if the MS-BN scheme is unforgeable by proving that for any PPT algorithm $\mathcal{A}$, the following two inequations hold:

$$|\,\mathsf{AdvMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda) - \mathsf{AdvMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)| \leq \mathrm{negl}(\lambda),$$
$$\mathsf{AdvMS}^{UF-MS}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{AdvMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda).$$

The only difference between the $\mathsf{ExpMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda)$ experiment and the $\mathsf{ExpMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)$ experiment is the distribution of secret keys. Let p be the size of a secret key space. Since the distribution of output does not change with regard to the same input, the following inequation holds:

$$\begin{aligned}
&|\,\mathsf{AdvMS}^{UF-MS}_{\Sigma,\mathcal{A}}(\lambda) - \mathsf{AdvMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)| \\
&\leq \tfrac{1}{2}\textstyle\sum_{i=0}^{p-1} |\Pr[x = i : x \xleftarrow{U} \mathbb{Z}_p] - \Pr[x = i : x \xleftarrow{U} \mathbb{Z}_p \setminus \{0\}]| \\
&= \tfrac{1}{2}(|\tfrac{1}{p} - 0| + \textstyle\sum_{i=1}^{p-1} |\tfrac{1}{p} - \tfrac{1}{p-1}|) = \tfrac{1}{2}(\tfrac{1}{p} + (p-1)\tfrac{1}{p(p-1)}) = \tfrac{1}{2}(\tfrac{1}{p} + \tfrac{1}{p}) = \tfrac{1}{p}.
\end{aligned}$$

Next, we compare the $\mathsf{ExpMS}^{UF-MS}_{\Pi,\mathcal{A}}(\lambda)$ experiment on the modified MS-BN with the $\mathsf{ExpMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)$ experiment on MS-BN. In the $\mathsf{ExpMS}^{UF-MS}_{\Pi,\mathcal{A}}(\lambda)$ experiment on the modified MS-BN, an adversary cannot include $pk = 1_G$ in a sequence of public keys with which a forged signature σ is accepted by the verification algorithm. The two experiments are otherwise same. Thus, for all PPT algorithm $\mathcal{A}$, it holds that $\mathsf{AdvMS}^{UF-MS}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{AdvMS}^{UF-MS'}_{\Sigma,\mathcal{A}}(\lambda)$.

F Proof of Theorem 8 : Key Substitution Attacks on MuSig2

Proof. We provide a PPT adversary $\mathcal{A}$ that breaks NKS of MuSig2 as follows. We first describe how $\mathcal{A}$ sets keys, messages, and a signature. Let $\mathcal{M}$ be a message space. The adversary $\mathcal{A}$ sets the first sequence of public keys $X^*_1, X^*_2 \leftarrow 1_G$, chooses a message $m^* \xleftarrow{U} \mathcal{M}$ and an integer $s \xleftarrow{U} \mathbb{Z}_p$ randomly, computes $R \leftarrow g^s$, and sets a signature $\sigma \leftarrow (R, s)$. It sets the second sequence of public keys $X^{**}_1, X^{**}_2, X^{**}_3 \leftarrow 1_G$ and chooses the second message $m^{**} \xleftarrow{U} \mathcal{M}/\{m^*\}$ randomly.

Let a^*_i be $H_{agg}(L^*, X^*_i)$ and a^{**}_i be $H_{agg}(L^{**}, X^{**}_i)$, and then we observe that two aggregated public keys are as below:

$$\begin{aligned}
\tilde{X}^* &= (X^*_1)^{a^*_1}(X^*_2)^{a^*_2} = 1_G \\
\tilde{X}^{**} &= (X^{**}_1)^{a^{**}_1}(X^{**}_2)^{a^{**}_2}(X^{**}_3)^{a^{**}_3} = 1_G.
\end{aligned}$$

Observe that it holds that $g^s = R \cdot 1_G = R(\tilde{X}^*)^{H_{sig}(\tilde{X}^*, R, m^*)}$ and $g^s = R \cdot 1_G = R(\tilde{X}^{**})^{H_{sig}(\tilde{X}^{**}, R, m^{**})}$. Hence, the signature σ is accepted by the verification algorithm with both $(\{\{X^*_1, X^*_2\}\}, m^*)$ and $(\{\{X^{**}_1, X^{**}_2, X^{**}_3\}\}, m^{**})$. □

G Proof of Theorem 9 : wNKS of MuSig2

Proof. We show that there are collisions of random numbers if there exists a PPT algorithm $\mathcal{A}$ that breaks wNKS of MuSig2. Suppose that there exists $\mathcal{A}$ that breaks wNKS of MuSig2. Let Σ_{MS} be MuSig2. In the experiment $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda)$, let $m^*, L^* = \{\{X_i^*\}\}_{i\in[n^*]} = \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}$, and $\sigma = (R, s)$ be the final input to $\mathcal{A}$, and m^{**} and $L^{**} = \{\{X_i^{**}\}\}_{i\in[n^{**}]} = \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^{**}]}$ be the final output from $\mathcal{A}$ where it results in $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$. Let $a_i^* \leftarrow H_{agg}(L^*, X_i^*), a_i^{**} \leftarrow H_{agg}(L^{**}, X_i^{**}), c^* \leftarrow H_{sig}(\tilde{X}^*, R, m^*)$ and $c^{**} \leftarrow H_{sig}(\tilde{X}^{**}, R, m^{**})$. For aggregated keys, we have that $\tilde{X}^* \leftarrow \mathsf{KeyAgg}(\mathsf{pp}, \{\{\mathsf{pk}_i^*\}\}_{i\in[n^*]}) = \prod_{i=1}^{n^*}(X_i^*)^{a_i^*} = \prod_{i=1}^{n^*} g^{x_i^* a_i^*} = g^{\sum_{i=1}^{n^*} x_i^* a_i^*}$ and $\tilde{X}^{**} \leftarrow \mathsf{KeyAgg}(\mathsf{pp}, \{\{\mathsf{pk}_i^{**}\}\}_{i\in[n^*]}) = g^{\sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}}$. Also, let $\{\{x_i^*\}\}_{i\in[n^*]}, \{\{x_i^{**}\}\}_{i\in[n^{**}]}$ and r be the elements in $\mathbb{Z}_p$ such that $g^{x_i^*} = X_i^*, g^{x_i^{**}} = X_i^{**}$ and $R = g^r$. We show that $c^* \sum_{i=1}^{n^*} x_i^* a_i^*$ and $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}$ are uniformly random and independently distributed.

Since the signature σ is accepted in the verification with (m^*, L^*), we have $g^s = R(\tilde{X}^*)^{a_i^*} = g^r g^{c^* \sum_{i=1}^{n^*} x_i^* a_i^*}$, and thus it holds that $s - r \equiv c^* \sum_{i=1}^{n^*} x_i^{**} a_i^* \pmod p$. We can show that $s - r \equiv c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**} \pmod p$ in the same way. Therefore, it holds that $c^* \sum_{i=1}^{n^*} x_i^* a_i^* \equiv c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**} \pmod p$.

First, we show that it holds that $\exists i \in [n^{**}], x_i^{**} \neq 0$. Suppose the case where it holds that $\forall i \in [n^{**}], x_i^{**} = 0$. We have $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**} = 0$. As the verification succeeds, we have that $c^* \sum_{i=1}^{n^*} x_i^* a_i^* \equiv c^* \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**} \pmod p$. However, x_i^*, a_i^* and c^* are chosen uniformly randomly, so the probability that it holds that $c^* \sum_{i=1}^{n^*} x_i^* a_i^* \mod p = 0$ is negligible. Therefore, the probability that $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$ is negligible unless $\exists i \in [n^{**}], x_i^{**} \neq 0$.

Then, we can assume that $\exists i \in [n^{**}], x_i^{**} \neq 0$. For some $1 \leq j \leq n^{**}$ such that $x_j^{**} \neq 0$, we denote the multiplicity in the set $\{\{X_i^{**}\}\}_{i\in[n^{**}]}$ by $t_j^{**} = \sharp\{X \in \{\{X_i^{**}\}\}_{i\in[n^{**}]} | X = g^{x_j^{**}}\}$. For all $1 \leq i \leq n^{**}$, we have that $a_i^{**} \leftarrow H_{agg}(L^{**}, X_i^{**})$, so $\{\{X_k^{**}\}\}_{k\in[n^{**}]}$ has been uniquely determined before c_i^{**} is determined. For all $1 \leq k \leq n^{**}$, x_k^{**} is uniquely determined such that $X_k^{**} = g^{x_k^{**}}$ when X_k^{**} is determined. Thus, a_i^{**} is determined after $\{\{x_k^{**}\}\}_{k\in[n^{**}]}$ and t_i^{**} are determined. Since a_j^{**} is chosen uniformly randomly, $t_j^{**} a_j^{**} x_j^{**}$ is uniformly random. If we let I denote $\{i \in [n^{**}] | x_i \neq x_j\}$, $\sum_{i=1}^{n^{**}} a_i^{**} x_i^{**} = t_j^{**} a_j^{**} x_j^{**} + \sum_{i\in I}^{n^{**}} a_i^{**} x_i^{**}$ is uniformly random. Therefore, $\tilde{X}^* = \prod_{i=1}^{n^{**}}(X_i^{**})^{a_i^{**}} = g^{\sum_{i=1}^{n^{**}} a_i^{**} x_i^{**}} = g^{t_j^{**} a_j^{**} x_j^{**} + \sum_{i\in I}^{n^{**}} a_i^{**} x_i^{**}}$ is uniformly random. Then, $c_i^{**} = H_{sig}(\tilde{X}^*, R, m^*)$ is also uniformly random, and hence $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}$ is uniformly random.

Since we have $\{\{X_i^*\}\}_{i\in[n^*]} \neq \{\{X_i^{**}\}\}_{i\in[n^{**}]}$, the distributions of a_i^* and a_i^{**} are independent. If $\tilde{X}^{**} \neq \tilde{X}^*$, the distributions of c^* and c^{**} are also independent. Therefore, $\mathsf{ExpMS}^{wNKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$ means the collision of $c^* \sum_{i=1}^{n^*} x_i^* a_i^*$ and $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}$ whereas they are uniformly random and independently distributed.

Since $\mathcal{A}$ is a PPT algorithm, it can query at most polynomial number of times, and hence the times of trials of collisions is also at most polynomial number. Thus, if such an $\mathcal{A}$ exists, we can construct an algorithm that finds collisions of random numbers chosen from an exponential-size set in polynomial time. □

H Proof of Theorem 10 : NKS of Modified MuSig2

Proof. We show that there are collisions of random numbers if there exists a PPT algorithm that breaks NKS of modified MuSig2. Suppose that there exists a PPT algorithm $\mathcal{A}$ that breaks NKS of modified MuSig2. Let Σ_{MS} be the modified MuSig2. Let $m^*, L^* = \{\{X_i^*\}\} = \{\{\mathsf{pk}_i^*\}\}, m^{**}, L^{**} = \{\{X_i^{**}\}\} = \{\{\mathsf{pk}_i^{**}\}\}$ and $\sigma = (R, s)$ be the output of $\mathcal{A}$ where it holds that $\mathsf{ExpMS}^{NKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}} = 1$. Let $a_i^* \leftarrow H_{agg}(L^*, X_i^*), a_i^{**} \leftarrow H_{agg}(L^{**}, X_i^{**}), c^* \leftarrow H_{sig}(\tilde{X}^*, R, m^*)$ and $c^{**} \leftarrow H_{sig}(\tilde{X}^{**}, R, m^{**})$. Let $\{\{x_i^*\}\}_{i\in[n^*]}, \{\{x_i^{**}\}\}_{i\in[n^{**}]}$ and r be the elements in $\mathbb{Z}_p$ such that $g^{x_i^*} = X_i^*, g^{x_i^{**}} = X_i^{**}$ and $R = g^r$. We show that $c^* \sum_{i=1}^{n^*} x_i^* a_i^*$ and $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}$ are uniformly random and independently distributed.

Since the signature σ is accepted in the verification with (m^*, L^*), we have that $g^s = R(\tilde{X}^*)^{c^*} = g^r g^{c^* \sum_{i=1}^{n^*} x_i^* a_i^*}$, and hence we have $s - r \equiv c^* \sum_{i=1}^{n^*} x_i^{**} a_i^* \pmod p$. We can also show that $s - r \equiv c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**} \pmod p$ in the same way. Therefore, it holds that $c^* \sum_{i=1}^{n^*} x_i^* a_i^* \equiv c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**} \pmod p$.

Let j be an arbitrary number in $[n^*]$. we denote the multiplicity in the set $\{\{X_i^*\}\}_{i\in[n^*]}$ by $t_j^* = \sharp\{X \in \{\{X_i^*\}\}_{i\in[n^*]} | X = g^{x_j^*}\}$. For all $1 \le i \le n^*$, we have that $a_i^* = H_{agg}(L^*, X_i^*)$, so $\{\{X_k^*\}\}_{k\in[n^*]}$ has been uniquely determined before a_i^* is determined. For all $1 \le k \le n^*$, x_k^* is uniquely determined such that $X_k^* = g^{x_k^*}$ when X_k^* is determined. Thus, a_i^* is determined after $\{\{x_k^*\}\}_{k\in[n^*]}$ and t_i^* are determined. Since a_j^* is chosen uniformly randomly, $t_j^* a_j^* x_j^*$ is uniformly random. If we let I denote $\{i \in [n^*] | x_i \ne x_j\}$, $\sum_{i=1}^{n^*} a_i^* x_i^* = t_j^* a_j^* x_j^* + \sum_{i\in I}^{n^*} a_i^* x_i^*$ is uniformly random. Therefore, $\tilde{X}^* = \prod_{i=1}^{n^*} (X_i^*)^{a_i^*} = g^{\sum_{i=1}^{n^*} a_i^* x_i^*} = g^{t_j^* a_j^* x_j^* + \sum_{i\in I}^{n^*} a_i^* x_i^*}$ is uniformly random. Then, $c_i^{**} = H_{sig}(\tilde{X}^*, R, m^*)$ is also uniformly random. If $\tilde{X}^{**} \ne \tilde{X}^*$, $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}$ is uniformly random. We can show that $\tilde{X}^{**} = \prod_{i=1}^{n^{**}} (X_i^{**})^{a_i^{**}}$ is also uniformly random in the same manner.

Since we have $\{\{X_i^*\}\} \ne \{\{X_i^{**}\}\}$, the distributions of a_i^* and a_i^{**} are independent, and hence those of c^* and c^{**} are independent. Therefore, $\mathsf{ExpMS}^{NKS}_{\Sigma_{\mathrm{MS}},\mathcal{A}}(\lambda) = 1$ means the collision of $c^* \sum_{i=1}^{n^*} x_i^* a_i^*$ and $c^{**} \sum_{i=1}^{n^{**}} x_i^{**} a_i^{**}$ whereas they are uniformly random and independently distributed.

Since $\mathcal{A}$ is a PPT algorithm, it can query at most polynomial number of times, and hence the times of trials of collisions is also at most polynomial number. Thus, if such an $\mathcal{A}$ exists, we can construct an algorithm that finds collisions of random numbers chosen from an exponential-size set in polynomial time. □

I Requirements of Unforgeability of Ordered Multi-Signatures

An ordered multi-signature scheme $\Sigma_{OMS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{OSign}, \mathsf{OVrf})$ satisfies correctness if the following conditions are satisfied.

– For any $\lambda \in \mathbb{N}$, any $n = \mathrm{poly}(\lambda)$, and any m, it holds that $\mathsf{OVrf}(\mathsf{pp}, m, \sigma_n, (\mathsf{pk}_i)_{i\in[n]})$ where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\forall i \in [n]$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\sigma_i \leftarrow \mathsf{OSign}(\mathsf{pp}, \mathsf{sk}_i, m, \sigma_{i-1}, (\mathsf{pk}_k)_{k\in[i]})$.

We next introduce unforgeability of ordered multi-signatures. It means not only the usual unforgeability but also the impossibility of reordering the signing order.

Definition 21 (UF-OMS). *An ordered multi-signature* $\Sigma_{OMS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{OSign}, \mathsf{OVrf})$ *satisfies existentially unforgeability under chosen-message attacks (UF-OMS) if it holds that* $\Pr[\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{OMS},\mathcal{A}}(\lambda) = 1] \leq \mathrm{negl}(\lambda)$ *for all PPT algorithms* $\mathcal{A}$ *where* $\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{OMS},\mathcal{A}}(\lambda)$ *is the following experiment:*

$\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{OMS},\mathcal{A}}(\lambda)$

$K \leftarrow \emptyset; Q \leftarrow \emptyset;$
$\mathsf{pp} \leftarrow \Sigma_{OMS}.\mathsf{Setup}(1^\lambda); (\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \Sigma_{OMS}.\mathsf{KeyGen}(\mathsf{pp});$
$((\mathsf{pk}_i^{**})_{i\in[n^{**}]}, m^{**}, \sigma^{**}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{OSign}}(\mathsf{pp},\mathsf{sk}^*,\cdot,\cdot,\cdot),\mathcal{O}_{Reg}(\cdot,\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk}^*)$:
Output 1 if $\mathsf{OVrf}(\mathsf{pp}, m^{**}, \sigma^{**}, (\mathsf{pk}_i^{**})_{i\in[n]}) = 1$
$\wedge \exists i^{**}, (\mathsf{pk}_{i^{**}}^{**} = \mathsf{pk}^* \wedge (\forall (m, \sigma, L) \in Q, |L| \neq i^{**} - 1 \vee m \neq m^{**})$
$\wedge (\forall j \in [n^{**}], j = i^{**} \vee (\exists (\mathsf{pk}, \mathsf{sk}, c) \in K, \mathsf{pk}_j = \mathsf{pk})));$
Otherwise output 0

An adversary $\mathcal{A}$ *can query* $(m, \sigma, (\mathsf{pk}_i)_{i\in[n]})$ *to the signing oracle* $\mathcal{O}_{\mathsf{OSign}}(\mathsf{pp}, \mathsf{sk}^*, \cdot, \cdot, \cdot)$. *Given a query, the oracle responds* $\sigma' \leftarrow \Sigma_{OMS}.\mathsf{OSign}(\mathsf{pp}, \mathsf{sk}^*, m, \sigma, (\mathsf{pk}_i)_{i\in[n]})$ *to* $\mathcal{A}$, *and sets* $L \leftarrow (\mathsf{pk}_i)_{i\in[n]}$ *and* $Q \leftarrow Q \cup \{(m, \sigma, L)\}$. *An adversary* $\mathcal{A}$ *can also query to the key registration oracle* $\mathcal{O}_{Reg}(\cdot, \cdot, \cdot)$ *a key pair* $(\mathsf{pk}, \mathsf{sk})$ *and the random tape* c *to generate the key pair with. The oracle sets* $K \leftarrow K \cup \{(\mathsf{pk}, \mathsf{sk}, c)\}$ *if it obtains* $(\mathsf{pk}, \mathsf{sk})$ *by running* $\mathsf{KeyGen}(1^\lambda)$ *with the random tape* c. *It returns nothing to the adversary.*

J Proof of Theorem 12 : Key Substitution Attacks on BGOY OMS

We first observe the form of a signature before the proof. Let h^* be the discrete logarithm of $H(m^*)$ to the base g and $(r_i)_{i\in[n^*]}$ be random numbers. When n^* signers sign on a message m^* with corresponding secret keys $(\mathsf{sk}_i^*)_{i\in[n^*]} = (s_i^*, t_i^*, u_i^*)_{i\in[n^*]}$, the form of the signature is as below:

$$\begin{aligned}\sigma = (Q, R) &= (H(m^*)^{\sum_{i=1}^{n^*} s_i^*} g^{(\sum_{i=1}^{n^*} r_i)\cdot(\sum_{i=1}^{n^*} t_i^* + i u_i^*)}, g^{\sum_{i=1}^{n^*} r_i}) \\ &= (g^{h^*\cdot(\sum_{i=1}^{n^*} s_i^*) + (\sum_{i=1}^{n^*} r_i)\cdot(\sum_{i=1}^{n^*} t_i^* + i u_i^*)}, g^{\sum_{i=1}^{n^*} r_i}).\end{aligned}$$

If the signature is accepted in the verification with a message m^{**} and a sequence of public keys $(\mathsf{pk}_i^{**})_{i\in[n^{**}]} = (S_i^{**}, T_i^{**}, U_i^{**})_{i\in[n^{**}]} = (g^{s_i^{**}}, g^{t_i^{**}}, g^{u_i^{**}})_{i\in[n^{**}]}$, the equation below holds:

$$e(Q,g) = e(H(m^{**}), \prod_{i=1}^{n^{**}} S_i^{**}) \cdot e(\prod_{i=1}^{n^{**}} T_i^{**}(U_i^{**})^i, g^{\sum_{i=1}^{n^*} r_i}).$$

Therefore, for the discrete logarithm h^{**} of $H(m^{**})$ to the base g, we observe that

$$h^*(\sum_{i=1}^{n^*} s_i^*) + (\sum_{i=1}^{n^*} r_i)(\sum_{i=1}^{n^*} t_i^* + iu_i^*) \equiv h^{**}(\sum_{i=1}^{n^{**}} s_i^{**}) + (\sum_{i}^{n^*} r_i)(\sum_{i=1}^{n^{**}} t_i^{**} + iu_i^{**}) \pmod p.$$

The verification succeeds if these three equations below hold:

$$h^* \equiv h^{**} \pmod p$$
$$\sum_{i=1}^{n^*} s_i^* \equiv \sum_{i=1}^{n^{**}} s_i^{**} \pmod p$$
$$\sum_{i=1}^{n^*} (t_i^* + iu_i^*) \equiv \sum_{i=1}^{n^{**}} (t_i^{**} + iu_i^{**}) \pmod p.$$

The three equations above hold if the following three equations hold:

$$m^* = m^{**}$$
$$\prod_{i=1}^{n^*} S_i^* \equiv \prod_{i=1}^{n^{**}} S_i^{**} \pmod p$$
$$\prod_{i=1}^{n^*} (T_i^*(U_i^*)^i) \equiv \prod_{i=1}^{n^{**}} (T_i^{**}(U_i^{**})^i) \pmod p.$$

Therefore, we can break the weak non-key substitutability if we set m^{**} and $(S_i^{**}, T_i^{**}, U_i^{**})_{i\in[n^{**}]}$ so that the equations above hold. We give a concrete example.

Proof. Let Σ_{OMS} be the BGOY OMS. We can construct a PPT algorithm $\mathcal{A}$ that breaks wNKS of Σ_{OMS} as follows: Given a signature σ, a message m^* and a sequence of public keys $(\mathsf{pk}_i^*)_{i\in[n^*]} = (S_i^*, T_i^*, U_i^*)_{i\in[n^*]}$ as input, $\mathcal{A}$ sets $m^{**} \leftarrow m^*$ and $S_1^{**} \xleftarrow{U} G\backslash\{S_1^*\}$, computes $S_2^{**} \leftarrow \{\prod_{i=1}^{n^*} S_i^*\}/S_1^{**} \bmod p$, $T_1^{**}, U_1^{**}, U_2^{**} \xleftarrow{U} G$, and $T_2^{**} \leftarrow \{\prod_{i=1}^{n^*}(T_i^*(U_i^*)^i)\}/(T_1^{**}U_1^{**}(U_2^{**})^2)$, and outputs a message m^{**} and a sequence of public keys $(\mathsf{pk}_1^{**} = (S_1^{**}, T_1^{**}, U_1^{**}), \mathsf{pk}_2^{**} = (S_2^{**}, T_2^{**}, U_2^{**}))$. Then, we have that $m^* = m^{**}$, $\prod_{i=1}^{n^*} S_i^* \equiv S_1^{**}S_2^{**}$, and $\prod_{i=1}^{n^*}(T_i^*(U_i^*)^i) \equiv T_1^{**}U_1^{**}T_2^{**}(U_2^{**})^2 \pmod p$. It holds that $(\mathsf{pk}_i^*)_{i\in[n^*]} \neq (\mathsf{pk}_1^{**}, \mathsf{pk}_2^{**})$ since $S_1^{**} \neq S_1^*$, so it results in $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}(\lambda) = 1$. □

K Proof of Theorem 13 : Correctness of Modified BGOY OMS

The form of a signature is written as below:

$$\sigma = (\{\prod_{i=1}^{n}(H(m||\,\mathsf{pk}_i)^{is_i})\}(g^{\sum_{i=1}^{n} t_i+iu_i})^{\sum_{i=1}^{n} r_i}, g^{\sum_{i=1}^{n} r_i}).$$

Let $(h_i)_{i\in[n]}$ be an element in $\mathbb{Z}_p$ where it holds that $g^{h_i} = H(m||\,\mathsf{pk}_i)$. Then, we have that

$$\begin{aligned}
&e(Q,g)\\
&= e(\{\textstyle\prod_{i=1}^{n} H(m||\,\mathsf{pk}_i)^{is_i}\}(g^{\sum_{i=1}^{n} t_i+iu_i})^{\sum_{i=1}^{n} r_i}, g)\\
&= e(\{\textstyle\prod_{i=1}^{n} g^{ih_is_i}\}(g^{\sum_{i=1}^{n} t_i+iu_i})^{\sum_{i=1}^{n} r_i}, g)\\
&= e(g^{\sum_{i=1}^{n} ih_is_i} \cdot g^{(\sum_{i=1}^{n} r_i)\sum_{i=1}^{n} t_i+iu_i}, g)\\
&= e(g^{\sum_{i=1}^{n} ih_is_i+(\sum_{i=1}^{n} t_i+iu_i)\sum_{i=1}^{n} r_i}, g)\\
&= e(g,g)^{\sum_{i=1}^{n} ih_is_i+(\sum_{i=1}^{n} t_i+iu_i)\sum_{i=1}^{n} r_i}\\
&= e(g,g)^{\sum_{i=1}^{n} ih_is_i} \cdot e(g,g)^{(\sum_{i=1}^{n} t_i+iu_i)\sum_{i=1}^{n} r_i}\\
&= \textstyle\prod_{i=1}^{n}\{e(g,g)^{ih_is_i}\} \cdot e(g^{\sum_{i=1}^{n} t_i+iu_i}, g^{\sum_{i=1}^{n} r_i})\\
&= \textstyle\prod_{i=1}^{n}\{e(g^{h_i}, g^{is_i})\} \cdot e(\prod_{i=1}^{n} g^{t_i+iu_i}, g^{\sum_{i=1}^{n} r_i})\\
&= \textstyle\prod_{i=1}^{n}\{e(H(m||\,\mathsf{pk}_i), (S_i)^i)\} \cdot e(\prod_{i=1}^{n} T_i(U_i)^i, R_n),
\end{aligned}$$

so the signature σ is accepted in the verification with m and $(\mathsf{pk}_j)_{j\in[i]}$.

L Proof of Theorem 14: Key Substitution Attacks on Modified BGOY OMS

Although R_n is usually determined randomly, by setting R_n to a specific value, we can create a signature that is accepted in the verification with two sequences of public keys corresponding to two arbitrary sequences of secret keys. We provide observations before the formal proof.

We set the values that our adversary uses during the attack. Let n^* and n^{**} be the numbers of keys. Let two sequences of secret keys be $(\mathsf{sk}_i^*)_{i\in[n^*]} = (s_i^*, t_i^*, u_i^*)_{i\in[n^*]}$ and $(\mathsf{sk}_i^{**})_{i\in[n^{**}]} = (s_i^{**}, t_i^{**}, u_i^{**})_{i\in[n^{**}]}$. We denote corresponding sequences of public keys by $(\mathsf{pk}_i^*)_{i\in[n^*]} = (S_i^*, T_i^*, U_i^*) = (g^{s_i^*}, g^{t_i^*}, g^{u_i^*})_{i\in[n^*]}$, and $(\mathsf{pk}_i^{**})_{i\in[n^{**}]} = (S_i^{**}, T_i^{**}, U_i^{**}) = (g^{s_i^{**}}, g^{t_i^{**}}, g^{u_i^{**}})_{i\in[n^{**}]}$. Let m^* and m^{**} be two arbitrary messages. If we sign a message m^* using a sequence of secret keys $(\mathsf{sk}_i^*)_{i\in[n^*]}$, we have that $\sigma = (Q,R) = (\{\prod_{i=1}^{n^*} H(m^*||\,\mathsf{pk}_i^*)^{is_i^*}\}g^{\sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^*}(t_i^*+iu_i^*)}, g^{\sum_{i=1}^{n^*} r_i})$. Let $(h_i^*)_{i\in[n^*]}, (h_i^{**})_{i\in[n^{**}]}$ denote sequences of discrete logarithms where it holds that $g^{h_i^*} = H(m^*||\,\mathsf{pk}_i^*), g^{h_i^{**}} = H(m^{**}||\,\mathsf{pk}_i^{**})$.

If the signature σ is accepted in the verification with $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ and m^{**}, we have that

$$e(Q,g) = \prod_{i=1}^{n^{**}}\{e(H(m^{**}||\,\mathsf{pk}_i^{**}), (S_i^{**})^i)\} \cdot e(\prod_{i=1}^{n^{**}} T_i^{**}(U_i^{**})^i, R).$$

We can convert the left side on the equation as follows: $e(Q,g) = e(g^{\sum_{i=1}^{n^*} ih_i^* s_i^* + \sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^*}(t_i^*+iu_i^*)}, g) = e(g,g)^{\sum_{i=1}^{n^*} ih_i^* s_i^* + \sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^*}(t_i^*+iu_i^*)}$. Also, we can convert the right side on the equation as follows: $\{\prod_{i=1}^{n^{**}} e(H(m^{**}||\,\mathsf{pk}_i^{**}), (S_i^{**})^i)\} \cdot e(\prod_{i=1}^{n^{**}} T_i^{**}(U_i^{**})^i, R)$
$= \{\prod_{i=1}^{n^{**}} e(g^{h_i^{**}}, g^{is_i^{**}})\} \cdot e(\prod_{i=1}^{n^{**}} g^{t_i^{**}+iu_i^{**}}, g^{\sum_{i=1}^{n^*} r_i})$
$= \{\prod_{i=1}^{n^{**}} e(g,g)^{ih_i^{**}s_i^{**}}\} \cdot e(g^{\sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**})}, g^{\sum_{i=1}^{n^*} r_i})$
$= e(g,g)^{\sum_{i=1}^{n^{**}} ih_i^{**}s_i^{**}} \cdot e(g,g)^{\sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**})}$
$= e(g,g)^{\sum_{i=1}^{n^{**}} ih_i^{**}s_i^{**} + \sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**})}$.

Due to the non-degenerateness of the bilinear map e, we have that $e(g,g) \neq 1_{G_T}$. Thus, we observe that $\sum_{i=1}^{n^*} ih_i^* s_i^* + \sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^*}(t_i^* + iu_i^*) \equiv \sum_{i=1}^{n^{**}} ih_i^{**}s_i^{**} + \sum_{i=1}^{n^*} r_i \sum_{i=1}^{n^{**}}(t_i^{**} + iu_i^{**}) \pmod p$. We further convert the congruence equation as follows: $\sum_{i=1}^{n^*} r_i \equiv (\sum_{i=1}^{n^{**}} ih_i^{**}s_i^{**} - \sum_{i=1}^{n^*} ih_i^* s_i^*)/(\sum_{i=1}^{n^*}(t_i^* + iu_i^*) - \sum_{i=1}^{n^{**}}(t_i^{**} + iu_i^{**})) \pmod p$. Since $g \in G$ is a generator, σ is accepted in the verification with $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ and m^{**} if it holds that

$$
\begin{aligned}
&g^{(\sum_{i=1}^{n^*} r_i)} = g^{(\sum_{i=1}^{n^{**}} ih_i^{**}s_i^{**} - \sum_{i=1}^{n^*} ih_i^* s_i^*)/(\sum_{i=1}^{n^*}(t_i^*+iu_i^*) - \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**}))} \\
&R = (g^{\sum_{i=1}^{n^{**}} ih_i^{**}s_i^{**}} / g^{\sum_{i=1}^{n^*} ih_i^* s_i^*})^{1/(\sum_{i=1}^{n^*}(t_i^*+iu_i^*) - \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**}))} \\
&= \{\prod_{i=1}^{n^{**}} H(m^{**}||\,\mathsf{pk}_i^{**})^{is_i^{**}} / \\
&\prod_{i=1}^{n^*} H(m^*||\,\mathsf{pk}_i^*)^{is_i^*}\}^{1/(\sum_{i=1}^{n^*}(t_i^*+iu_i^*) - \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**}))}.
\end{aligned}
$$

Proof. Let Σ_{OMS} be the modified BGOY OMS scheme. An adversary chooses $n^* + n^{**}$ secret keys arbitrarily, and let $(\mathsf{sk}_i^*)_{i\in[n^*]} = (s_i^*, t_i^*, u_i^*)_{i\in[n^*]}$ and $(\mathsf{sk}_i^{**})_{i\in[n^{**}]} = (s_i^{**}, t_i^{**}, u_i^{**})_{i\in[n^{**}]}$ be two sequences of secret keys. It computes two sequences of public keys $(\mathsf{pk}_i^*)_{i\in[n^*]}$ and $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ that correspond to the two sequences of secret keys. Let m^* and m^{**} be two messages chosen arbitrarily from a message space. Then, it obtains hash values $H_i^* \leftarrow H(m^*||\,\mathsf{pk}_i^*)$ and $H_i^{**} \leftarrow H(m^{**}||\,\mathsf{pk}_i^{**})$, computes $R \leftarrow (\prod_{i=1}^{n^{**}}(H_i^{**})^{is_i^{**}} / \prod_{i=1}^{n^*}(H_i^*)^{is_i^*})^{1/(\sum_{i=1}^{n^*}(t_i^*+iu_i^*) - \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**}))}$ and $Q \leftarrow H(m^*||\,\mathsf{pk}_i^*)^{is_i^*} R^{\sum_{i=1}^{n^*}(t_i^*+iu_i^*)}$, and sets a signature $\sigma \leftarrow (Q,R)$. The signature σ is accepted with $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ and m^{**}. Due to the correctness of the scheme, the signature σ is also accepted with $(\mathsf{pk}_i^*)_{i\in[n^*]}$ and m^*. Therefore, if an adversary outputs $(m^*, (\mathsf{pk}_i^*)_{i\in[n^*]}, m^{**}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]})$, it holds that $\mathsf{ExpOMS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}^{NKS}(\lambda) = 1$.

M Proof of Theorem 15 : wNKS of Modified BGOY OMS

Proof. Suppose that there exists a PPT adversary $\mathcal{B}$ that breaks wNKS of the modified BGOY OMS. Then we can construct a PPT algorithm $\mathcal{A}$ that makes the probability in Eq. (1) non-negligible using $\mathcal{B}$, which means we can solve the intermediate problem (but here the index of random input starts from 0). Let Σ_{OMS} be the modified BGOY OMS scheme. Given input of the intermediate problem pp and $h_0, ..., h_n$, an algorithm $\mathcal{A}$ simulates $\mathsf{ExpOMS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}^{wNKS}(\lambda)$, takes

a response from $\mathcal{B}$, and can output the answer to the intermediate problem. An adversary $\mathcal{B}$ can use the hash oracle whenever it wants to, and $\mathcal{A}$ responds to these queries by simulating the random oracle. Let $q_H \leq \mathrm{poly}(\lambda)$ be the maximum number of hash queries from $\mathcal{B}$. First, $\mathcal{A}$ obtains n^* by giving pp to $\mathcal{B}$. Without loss of generality, we can set $n = n^* + q_H$ since both $\mathcal{A}$ and $\mathcal{B}$ are PPT algorithms. $\mathcal{A}$ uses the input of the intermediate problem $h_0, ..., h_{n^*+q_H}$ to create a challenge signature σ^* and to respond to $\mathcal{B}$'s hash queries. Since $h_0, ..., h_{n^*+q_H}$ are uniformly random, $\mathcal{A}$ can simulate the random oracle and the challenger in $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}(\lambda)$ against $\mathcal{B}$.

Setup Phase

First, $\mathcal{A}$ obtains n^* from $\mathcal{B}$ by giving pp to $\mathcal{B}$. An adversary $\mathcal{B}$ can make hash queries whenever it wants to. We will afterward describe how $\mathcal{A}$ responds to these queries.

Next, $\mathcal{A}$ creates a sequence of secret keys $(\mathsf{sk}^*_i)_{i\in[n^*]} = (s^*_i, t^*_i, u^*_i)_{i\in[n^*]}$ and a sequence of public keys $(\mathsf{pk}^*_i)_{i\in[n^*]} = (g^{s^*_i}, g^{t^*_i}, g^{u^*_i})_{i\in[n^*]}$ by running $\mathsf{KeyGen}(1^\lambda)$. It obtains a message m^* from $\mathcal{B}$ by giving the sequence of public keys $(\mathsf{pk}^*_i)_{i\in[n^*]}$ to $\mathcal{B}$.

Afterward, $\mathcal{A}$ embeds h_0 into a signature as a random number by setting $R^* \leftarrow h_0$. Also, $\mathcal{A}$ embeds the input of the intermediate problem into $H(m^*||\,\mathsf{pk}^*_i)$ as we will discuss later, and it computes $Q^* \leftarrow H(m^*||\,\mathsf{pk}^*_i)^{\sum_{i=1}^{n^*} is^*_i}(R^*)^{\sum_{i=1}^{n^*}(t^*_i+iu^*_i)}$. It sets a signature $\sigma^* \leftarrow (Q^*, R^*)$, and runs $\mathcal{B}$ by giving this signature σ^* and a sequence of public keys $(\mathsf{pk}^*_i)_{i\in[n^*]}$. Since R^* is uniformly random, the distribution of R^* is the same as that of R_n in the signature created by running OSign repeatedly, so $\mathcal{B}$ cannot distinguish between the signature σ^* and one given in the experiment $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}(\lambda)$.

Response to Hash Queries from $\mathcal{B}$

The algorithm $\mathcal{A}$ responds to hash queries using H-List. Using the index j starting from 1, H-List stores values of the form $((m_j, \mathsf{pk}_j), h_j)$. When $\mathcal{A}$ receives a query $(m||\,\mathsf{pk})$, it searches H-List for the same (m_j, pk_j) and proceeds as follows: If it finds the same (m_j, pk_j), then it returns h_j. Otherwise, let j be the minimum index not used in H-List, and it stores $((m, \mathsf{pk}), h_j)$ in H-List and returns h_j.

When creating a signature to give $\mathcal{B}$ using a message m^* and a sequence of public keys $(\mathsf{pk}^*_i)_{i\in[n^*]}$, $\mathcal{A}$ determines $H(m^*||\,\mathsf{pk}^*_i)$ in the same way as it responds to hash queries: It searches H-List for (m_j, pk_j) and proceeds as follows. If it finds the same (m_j, pk_j), then it sets $H(m^*, \mathsf{pk}^*_i) \leftarrow h_j$. Otherwise, let j be the minimum index not used in H-List, and it stores $((m, \mathsf{pk}), h_j)$ in H-List and sets $H(m^*, \mathsf{pk}^*_i) \leftarrow h_j$.

Since $(h_j)_{j\in[n^*+q_H]}$ are chosen uniformly randomly in the intermediate problem, when $\mathcal{A}$ determines hash values as described above, $\mathcal{B}$ cannot distinguish between $\mathcal{A}$'s replies and the random oracle's replies.

$\mathcal{A}$'s response to the intermediate problem

Suppose that $\mathcal{B}$ outputs a message m^{**} and a sequence of public keys $(\mathsf{pk}^{**}_i)_{i\in[n^{**}]}$ where it results in $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}(\lambda) = 1$. Without loss of generality, we may assume that $\mathcal{B}$ queries $(m^{**}||\,\mathsf{pk}^{**}_i)$ before $\mathcal{B}$ outputs m^{**} and $(\mathsf{pk}^{**}_i)_{i\in[n^{**}]}$. Considering the verification equation, $\mathcal{A}$ can solve the intermedi-

ate problem by reordering two sequences of public keys $((\mathsf{pk}_i^*)_{i\in[n^*]}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]})$ from the order in sequences to the order in hash queries.

We first observe what equation holds with regard to m^{**}, $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ and σ^*. The sequence of secret keys $(\mathsf{sk}_i^{**})_{i\in[n^{**}]} = (s_i^{**}, t_i^{**}, u_i^{**})_{i\in[n^{**}]}$ is uniquely determined by the sequence of public keys $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ even if $\mathcal{B}$ itself does not know them. Let $(d_i^*)_{i\in[n^*]}$ and $(d_i^{**})_{i\in[n^{**}]}$ denote discrete logarithms of hash values to the base g as follows: it holds that $g^{d_i^*} = H(m^* || \mathsf{pk}_i^*), g^{d_i^{**}} = H(m^{**} || \mathsf{pk}_i^{**})$. As the verification succeeds, we have the following equation.

$$
\begin{aligned}
&e(Q^*, g) = \{\textstyle\prod_{i=1}^{n^{**}} e(H(m^{**} || \mathsf{pk}_i^{**}), (S_i^{**})^i)\} e(\prod_{i=1}^{n^{**}} T_i^{**} (U_i^{**})^i, R^*) \\
&e(g,g)^{\sum_{i=1}^{n^*} id_i^* s_i^* + r^* \sum_{i=1}^{n^*}(t_i^* + iu_i^*)} = e(g,g)^{\sum_{i=1}^{n^{**}} id_i^{**} s_i^{**} + r^* \sum_{i=1}^{n^{**}}(t_i^{**} + iu_i^{**})} \\
&e(g,g)^{r^*\{\sum_{i=1}^{n^*}((-t_i^*)+i(-u_i^*)) + \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**})\}} \\
&\cdot e(g,g)^{\sum_{i=1}^{n^*} id_i^*(-s_i^*) + \sum_{i=1}^{n^{**}} id_i^{**} s_i^{**}} = 1 \\
&e(R^*, g^{\sum_{i=1}^{n^*}((-t_i^*)+i(-u_i^*)) + \sum_{i=1}^{n^{**}}(t_i^{**}+iu_i^{**})}) \\
&\cdot \{\textstyle\prod_{i=1}^{n^*} e(g,g)^{id_i^*(-s_i^*)}\} \cdot \{\prod_{i=1}^{n^{**}} e(g,g)^{id_i^{**} s_i^{**}}\} = 1 \\
&e(h_0, \{\textstyle\prod_{i=1}^{n^{**}} T_i^{**}(U_i^{**})^i\} / \{\prod_{i=1}^{n^*} T_i^*(U_i^*)^i\}) \\
&\cdot \textstyle\prod_{i=1}^{n^*} e(H(m^* || \mathsf{pk}_i^*), g)^{-is_i^*} \cdot \prod_{i=1}^{n^{**}} e(H(m^{**} || \mathsf{pk}_i^{**}), g)^{is_i^{**}} = 1
\end{aligned}
$$

We convert the product of the sequence in the equation above by considering the order in H-List rather than the order in the sequence of public keys. With regard to two messages m^* and m^{**} and two sequences of public keys $(\mathsf{pk}_i^*)_{i\in[n^*]}, (\mathsf{pk}_i^{**})_{i\in[n^{**}]}$, $\mathcal{A}$ searches H-List for (m_j, pk_j), and it sets $(c_j^*)_{j\in[n^*+q_H]}, (c_j^{**})_{j\in[n^*+q_H]}$ so that it holds that $h_j = H(m^* || \mathsf{pk}_{c_j^*}^*)$ and $h_j = H(m^{**} || \mathsf{pk}_{c_j^{**}}^{**})$. If it cannot find matching value (m_j, pk_j), then it sets $c_j^* = 0$ and $c_j^{**} = 0$. We define $\mathsf{pk}_0^* = (S_0^*, T_0^*, U_0^*) = (1,1,1)$ and $\mathsf{pk}_0^{**} = (S_0^{**}, T_0^{**}, U_0^{**}) = (1,1,1)$. Then, we have that

$$
\begin{aligned}
&\textstyle\prod_{i=1}^{n^*} e(H(m^* || \mathsf{pk}_i^*), g)^{-is_i^*} \cdot \prod_{i=1}^{n^{**}} e(H(m^{**} || \mathsf{pk}_i^{**}), g)^{is_i^{**}} \\
&= \textstyle\prod_{j=1}^{n^*+q_H} e(h_j, g)^{-c_j^* s_{c_j^*}^* } \cdot e(h_j, g)^{c_j^{**} s_{c_j^{**}}^{**}} \\
&= \textstyle\prod_{j=1}^{n^*+q_H} e(h_j, g)^{-c_j^* s_{c_j^*}^* + c_j^{**} s_{c_j^{**}}^{**}} = \prod_{j=1}^{n^*+q_H} e(h_j, g^{-c_j^* s_{c_j^*}^* + c_j^{**} s_{c_j^{**}}^{**}}) \\
&= \textstyle\prod_{j=1}^{n^*+q_H} e(h_j, g^{c_j^{**} s_{c_j^{**}}^{**}} / g^{c_j^* s_{c_j^*}^*}) = \prod_{j=1}^{n^*+q_H} e(h_j, S_{c_j^{**}}^{**\, c_j^{**}} / S_{c_j^*}^{*\, c_j^*}).
\end{aligned}
$$

We next demonstrate how to determine $(r_i)_{i\in[n^*+q_H]}$ where it holds that $\prod_{i=0}^{n^*+q_H} e(h_i, r_i) = 1$. As discussed above, we have that $e(h_0, \{\prod_{i=1}^{n^{**}} T_i^{**}(U_i^{**})^i\} / \{\prod_{i=1}^{n^*} T_i^*(U_i^*)^i\}) \cdot \prod_{j=1}^{n^*+q_H} e(h_j, S_{c_j^{**}}^{**\, c_j^{**}} / S_{c_j^*}^{*\, c_j^*}) = 1$. Therefore, by setting $r_0 \leftarrow \{\prod_{i=1}^{n^{**}} T_i^{**}(U_i^{**})^i\} / \{\prod_{i=1}^{n^*} T_i^*(U_i^*)^i\}$ and $r_j \leftarrow S_{c_j^{**}}^{**\, c_j^{**}} / S_{c_j^*}^{*\, c_j^*}$ for $1 \le j \le n^* + q_H$, we have that $\prod_{i=0}^{n^*+q_H} e(h_i, r_i) = 1$. We next show that it holds that $(r_0, ..., r_{n^*+q_H}) \neq (1, ..., 1)$ for $(r_i)_{i\in[n^*+q_H]}$.

First, we consider the case where the two sequences of public keys $(\mathsf{pk}_i^*)_{i\in[n^*]}$ and $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ are different as unordered set. There exists a public key included in only either $(\mathsf{pk}_i^*)_{i\in[n^*]}$ or $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$. Assume there exists such a public key $pk_{c_j^*}^*$ in $(\mathsf{pk}_i^*)_{i\in[n^*]}$. Since $c_j^{**} = 0$, we have that $S_{c_j^{**}}^{**} = 1$, and it holds that $r_j = S_{c_j^{**}}^{**} / S_{c_j^*}^* = 1 / S_{c_j^*}^* \neq 1$. If there exists a public key $pk_{c_j^{**}}^{**}$ included only

in $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$, we have that $S^*_{c_j^*} = 1$ because of $c_j^* = 0$, and it holds that $r_j = S^{**}_{c_j^{**}}/S^*_{c_j^*} = S^{**}_{c_j^{**}} \neq 1$.

Next, we consider the case where $(\mathsf{pk}_i^{**})_{i\in[n^{**}]}$ can be obtained by reordering $(\mathsf{pk}_i^{*})_{i\in[n^{*}]}$. Considering the public key reordered, it holds that $pk^*_{c_j^*} = pk^{**}_{c_j^{**}}$ and $c_j^* \neq c_j^{**}$. We thus have that $S^*_{c_j^*} = S^{**}_{c_j^{**}}$, and it holds that $S^{**}_{c_j^{**}} \neq 1$ due to the prohibition of the trivial key. Therefore, we have that $r_j = {S^{**}_{c_j^{**}}}^{c_j^{**}}/{S^*_{c_j^*}}^{c_j^*} = {S^{**}_{c_j^{**}}}^{c_j^{**}-c_j^*} \neq 1$. As discussed above, $(r_i)_{i\in[n^*+q_H]}$ is the solution to the intermediate problem.

The algorithm $\mathcal{A}$ can always perform the operations described above if it obtains $\mathcal{B}$'s output such that $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}(\lambda) = 1$. Therefore, the probability that $\mathcal{A}$ solves the intermediate problem is the same as the probability that $\mathcal{B}$ satisfies the winning condition of the $\mathsf{ExpOMS}^{wNKS}_{\Sigma_{\mathrm{OMS}},\mathcal{A}}(\lambda)$ experiment.

N Proof of Theorem 16 : Unforgeability of Modified BGOY OMS

Proof. Suppose that there exists a PPT adversary $\mathcal{B}$ that breaks the UF-OMS security of the modified BGOY OMS. We show that using $\mathcal{B}$, we can construct a PPT algorithm $\mathcal{A}$ that breaks the CDH assumption. Let Σ_{OMS} be the modified BGOY OMS scheme. Given input of the CDH problem from a challenger, $\mathcal{A}$ simulates the experiment $\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$, takes a response from $\mathcal{B}$, and can output the solution to the CDH problem in non-negligible probability. According to the definition of the experiment $\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$, $\mathcal{B}$ can make queries to the key registration oracle and the signing oracle. Simulating these oracles, $\mathcal{A}$ replies to these queries. Since we adopt the random oracle model, $\mathcal{B}$ can also make hash queries, and $\mathcal{A}$ replies to these queries by simulating the random oracle. Without loss of generality, we may assume that $\mathcal{B}$ does not repeat the same query.

Setup Phase

We describe how $\mathcal{A}$ behaves before running $\mathcal{B}$. First, $\mathcal{A}$ takes input of the CDH problem $(p, G, G_T, e, g, g^a, g^b)$. It initializes arrays K, D, H and E, and chooses $k^* \xleftarrow{U} [n_{max}]$ and $t_0, u_0 \xleftarrow{U} \mathbb{Z}_p$. Here n_{max} denotes the maximum number of signers. The adversary $\mathcal{A}$ computes $S^* \leftarrow g^a, T^* \leftarrow (g^a)^{-u_0 k^*} g^{t_0}$ and $U^* \leftarrow (g^a)^{u_0}$ and sets a challenge key $\mathsf{pk}^* \leftarrow (S^*, T^*, U^*)$. Here let s^*, t^* and u^* be the elements in $\mathbb{Z}_p$ where it holds that $S^* = g^{s^*}, T^* = g^{t^*}$ and $U^* = g^{u^*}$. Finally, it runs $\mathcal{B}$ by giving pk^* and (p, G, G_T, e, g).

$\mathcal{A}$'s behavior to key registration queries

We describe $\mathcal{A}$'s behavior when $\mathcal{B}$ queries to the key registration oracle a public key pk, a secret key sk and the random tape c used to generate the key pair. First, $\mathcal{A}$ confirms that it can obtain the key pair $(\mathsf{pk}, \mathsf{sk})$ with random tape c. If so, it sets $K[\mathsf{pk}] \leftarrow \mathsf{sk}$ so that it can refer to the secret key sk using pk as an index afterward. If not, it does nothing and continues.

$\mathcal{A}$'s replies to hash queries

We describe how $\mathcal{A}$ responds when $\mathcal{B}$ queries a message m and a public key pk_i to the hash oracle. It chooses $E[m||\,\mathsf{pk}_i] \xleftarrow{U} \mathbb{Z}_p$. Then, it sets $D[m||\,\mathsf{pk}_i] \leftarrow 1$ with probability δ, and $D[m||\,\mathsf{pk}_i] \leftarrow 0$ with probability $1-\delta$. If $D[m||\,\mathsf{pk}_i] = 1$, then it computes $H[m||\,\mathsf{pk}_i] \leftarrow g^{E[m||\,\mathsf{pk}_i]}$, otherwise computes $H[m||\,\mathsf{pk}_i] \leftarrow g^b g^{E[m||\,\mathsf{pk}_i]}$. Finally, it replies $H[m||\,\mathsf{pk}_i]$ to $\mathcal{B}$. Since $g^{E[m||\,\mathsf{pk}_i]}$ is uniformly random, $\mathcal{B}$ cannot distinguish if the reply is from $\mathcal{A}$ or the random oracle.

$\mathcal{A}$'s replies to signing queries

We show how $\mathcal{A}$ responds when $\mathcal{B}$ queries to the signing oracle a message m, a signature $\sigma = (Q, R)$ and a sequence of public keys $L = (\mathsf{pk}_1, ..., \mathsf{pk}_{i-1})$. If there exists $z \in [i-1]$ such that $K[\mathsf{pk}_z]$ is not registered, or it holds that $\Sigma_{OMS}.\mathsf{OVrf}(\mathsf{pp}, m, \sigma, L) = 0$, then $\mathcal{A}$ outputs $\perp$. If it holds that $\sigma[m||\,\mathsf{pk}^*] = 0$ and $i = k^*$, $\mathcal{A}$ halts. Without loss of generality, we may assume that $\mathcal{B}$ queries $m||\,\mathsf{pk}_1, ..., m||\,\mathsf{pk}_{i-1}, m||\,\mathsf{pk}^*$ to the hash oracle before $\mathcal{B}$ makes the signing query.

We describe how $\mathcal{A}$ computes a response. It chooses $r \xleftarrow{U} \mathbb{Z}_p$. It sets $K[\mathsf{pk}_z] = (s_z, t_z, u_z)$ for all $z \in [i-1]$. If we have that $D[m||\,\mathsf{pk}^*] = 1$, then it computes $Q' \leftarrow (g^a)^{iE[m||\,\mathsf{pk}^*]}((g^a)^{-u_0 k^*} g^{t_0}(g^a)^{iu_0})^r$, $R' \leftarrow g^r$ and $Q'' \leftarrow Q' \prod_{j=1}^{i-1}(g^{E[m||\,\mathsf{pk}_j]})^{js_j}(R')^{(t_j+ju_j)}$, and returns (Q'', R'). If we have that $D[m||\,\mathsf{pk}^*] = 0$, then it computes $Q' \leftarrow (g^a)^{iE[m||\,\mathsf{pk}^*]}(g^b)^{-it_0/(u_0(i-k^*))}((g^a)^{-u_0 k^*} g^{t_0}(g^a)^{iu_0})^r$, $R' \leftarrow g^r (g^b)^{i/(u_0(k^*-i))}$ and $Q'' \leftarrow Q' \prod_{k=1}^{i-1}(g^b g^{E[m||\,\mathsf{pk}_k]})^{ks_k}(R')^{(t_k+ku_k)}$, and returns (Q'', R'). The distribution of this reply is the same as that of the signing oracle's reply from $\mathcal{B}$'s view. Since $r \in \mathbb{Z}_p$ is chosen uniformly randomly, g^r and $g^r(g^b)^{i/(u_0(k^*-i))}$ are also uniformly random.

The Answer to the CDH Problem

We next describe how $\mathcal{A}$ answers to the CDH problem after $\mathcal{B}$ outputs $(L^{**} = (\mathsf{pk}_1^{**}, ..., \mathsf{pk}_{n^{**}}^{**}), m^{**}, \sigma^{**} = (Q^{**}, R^{**}))$ and halts. Without loss of generality, we may assume that $\mathcal{B}$ queries $m||\,\mathsf{pk}_1, ..., m||\,\mathsf{pk}_{i-1}, m||\,\mathsf{pk}^*$ to the hash oracle before it outputs $(L^{**}, m^{**}, \sigma^{**})$. First, $\mathcal{A}$ checks if it holds that $\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda) = 1$, otherwise it halts. Here let $\mathsf{pk}_{i^{**}}^{**}$ be pk^*. If we have that $D[m^{**}||\,\mathsf{pk}^*] = 1$ or $i^{**} \neq k^*$, it halts. Otherwise, it sets $K[\mathsf{pk}_z] = (s_z, t_z, u_z)$ for all $z \in [i-1]$, computes $Q \leftarrow Q^{**}/(\prod_{j \neq k^*}(H[m||\,\mathsf{pk}_j]^{js_j}(R^{**})^{t_j+ju_j})$, and outputs $Z \leftarrow (Q/((R^{**})^{t_0}(g^a)^{k^* E[m^{**}||\,\mathsf{pk}^*]}))^{1/k^*}$ as the answer to the CDH problem. Unless $\mathcal{A}$ halts before outputting the answer, Z is the correct answer to the CDH problem; Since σ^* is accepted, we have that

$Q^{**} = \prod_{j \neq k^*}^{n^{**}} H[m^{**}||\,\mathsf{pk}_j^{**}]^{js_j}(R^{**})^{t_j+ju_j}) \cdot H[m^{**}||\,\mathsf{pk}^*]^{k^* s^*}(R^{**})^{t^*+k^* u^*}$,

$Q = H[m^{**}||\,\mathsf{pk}^*]^{k^* s^*}(R^{**})^{t^*+k^* u^*} = (g^b g^{E[m^{**}||\,\mathsf{pk}^*]})^{k^* a}(R^{**})^{-au_0 k^* + t_0 + au_0 k^*} = g^{k^* ab + k^* aE[m^{**}||\,\mathsf{pk}^*]}(R^{**})^{t_0}$, and $Z = (g^{k^* ab})^{1/k^*} = g^{ab}$.

The Probability That $\mathcal{A}$ Can Answer to the CDH Problem Correctly

The advantage of $\mathcal{A}$ to the CDH problem $\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is non-negligible. Let q_S be the maximum number of signing queries, and $\mathsf{AdvOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$ be the probability that $\mathcal{B}$ satisfies the winning condition of the $\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$ experiment. Then we show the following equation holds:

$$\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda) > \frac{1}{e} \cdot \frac{1}{q_S + 1} \cdot \frac{1}{n_{max}} \cdot \mathsf{AdvOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$$

Since $\mathcal{A}$ is a PPT algorithm, q_S and n_{max} are polynomials of λ, so $\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is non-negligible if $\mathsf{AdvOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$ is non-negligible.

Splitting the event that $\mathcal{A}$ solves the CDH problem to several events, we consider the probability and independence of these events. We define E_1 as the event that $\mathcal{A}$ does not halt by $\mathcal{B}$'s queries, E_2 as the event that $\mathcal{B}$ breaks the UF-OMS security of the scheme, and E_3 as the event that $\mathcal{A}$ does not halts before it outputs the answer to the CDH problem. Then we can write the event that $\mathcal{A}$ answers to the CDH problem correctly as $E_1 \wedge E_2 \wedge E_3$. we have the equation below:

$$\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda) = Pr[E_1 \wedge E_2 \wedge E_3] = Pr[E_3|E_1 \wedge E_2] \cdot Pr[E_2|E_1] \cdot Pr[E_1].$$

The event E_1 occurs when $\mathcal{B}$ has never queried m and $(\mathsf{pk}_1, ..., \mathsf{pk}_{i-1})$ such that $D[m||\,\mathsf{pk}^*] = 0$ and $i = k^*$. Even though we cannot know the number of $\mathcal{B}$'s key registration queries, at least $\mathcal{A}$ does not halt if $D[m||\,\mathsf{pk}^*] = 1$ holds with regard to any queried message m. Since the maximum number of queries is q_S, $\mathcal{A}$ checks if $D[m||\,\mathsf{pk}^*] = 1$ holds at most q_S times even if every queried message is different. Therefore, we have that $Pr[E_1] \geq \delta^{q_S}$.

The event E_2 occurs independently of E_1. The replies of $\mathcal{A}$ are uniform randomly distributed regardless of $D[m||\,\mathsf{pk}^*]$, so $\mathcal{B}$ cannot distinguish the replies are from $\mathcal{A}$ or the true oracles. Therefore, we have that $Pr[E_2|E_1] = Pr[E_2] = \mathsf{AdvOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$.

The event E_3 occurs when we have that $i^{**} = k^*$ and $D[m^{**}||\,\mathsf{pk}^*] = 0$. Since $k^* \in [n_{max}]$ and the responses to signing queries are uniformly random, the responses are independent of $\mathcal{B}$'s output. Also, according to the definition of the experiment $\mathsf{ExpOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$, $(m^{**}, (\mathsf{pk}_1, ..., \mathsf{pk}_{k^*-1}))$ is not queried to the signing oracle when $\mathcal{B}$ breaks the UF-OMS security of the scheme. Therefore, regardless of whether $\mathcal{B}$ makes queries including m^{**} or not, E_1 occurs independently of whether $D[m^{**}||\,\mathsf{pk}^*] = 1$ holds. Thus, we have that $Pr[E_3|E_1 \wedge E_2] = \frac{1}{n_{max}} \cdot (1 - \delta)$.

Finally we decide the value δ to maximize $\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda)$. As discussed above, we have that $\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda) \geq \delta^{q_S} \cdot (1-\delta) \cdot \mathsf{AdvOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda) \cdot \frac{1}{n_{max}}$. At $\delta = \frac{q_s}{q_S+1}$, $\delta^{q_S} \cdot (1 - \delta)$ reaches its maximum value $\frac{1}{q_S+1} \cdot (\frac{q_S}{q_S+1})^{q_S} > \frac{1}{q_S+1} \cdot \frac{1}{e}$. Therefore, we have that $\mathsf{Adv}^{CDH}_{\mathcal{G},\mathcal{A}}(\lambda) > \frac{1}{e} \cdot \frac{1}{q_S+1} \cdot \frac{1}{n_{max}} \cdot \mathsf{AdvOMS}^{UF-OMS}_{\Sigma_{\mathrm{OMS}},\mathcal{B}}(\lambda)$.