From One-Time to Two-Round Reusable Multi-Signatures without Nested Forking

Lior Rotem* Gil Segev† Eylon Yogev‡

Abstract

Multi-signature schemes are gaining significant interest due to their blockchain applications. Of particular interest are two-round schemes in the plain public-key model that offer key aggregation, and whose security is based on the hardness of the DLOG problem. Unfortunately, despite substantial recent progress, the security proofs of the proposed schemes provide rather insufficient concrete guarantees (especially for 256-bit groups). This frustrating situation has so far been approached either by relying on the security of seemingly-stronger assumptions or by considering restricted classes of attackers (e.g., algebraic attackers, which are assumed to provide an algebraic justification of each group element that they produce).

We present a complementing approach by constructing multi-signature schemes that satisfy two relaxed notions of security, whose applicability nevertheless ranges from serving as drop-in replacements to enabling expressive smart contract validation procedures. Our first notion, *one-time unforgeability*, extends the analogous single-signer notion by considering attackers that obtain a single signature for some message and set of signers of their choice. We construct a *non-interactive* one-time scheme based on any ring-homomorphic one-way function, admitting efficient instantiations based on the DLOG and RSA assumptions. Aggregated verification keys and signatures consist of two group elements and a single group element, respectively, and our security proof consists of a *single* application of the forking lemma (thus avoiding the substantial security loss exhibited by the proposed two-round schemes). Additionally, we demonstrate that our scheme naturally extends to a t-time scheme, where aggregated verification keys consist of $t+1$ group elements, while aggregated signatures still consist of a single group element.

Our second notion, *single-set unforgeability*, considers attackers that obtain *any polynomial number* of signatures but are restricted to a single set of signers of their choice. We transform any non-interactive one-time scheme into a two-round single-set scheme via a novel forking-free construction that extends the seminal Naor-Yung tree-based approach to the multi-signer setting. Aggregated verification keys are essentially identical to those of the underlying one-time scheme, and the length of aggregated signatures is determined by that of the underlying scheme while scaling linearly with the length of messages (noting that long messages can always be hashed using a collision-resistant function). Instantiated with our one-time scheme, we obtain

*Computer Science Department, Stanford University, 353 Jane Stanford Way, Stanford, CA 94305, USA. Email: lrotem@cs.stanford.edu. Supported by a research grant from Protocol Labs.

†School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: segev@cs.huji.ac.il. Supported by the Israel Science Foundation (Grant No. 1336/22) and by the European Union (ERC, FTRC, 101043243). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

‡Department of Computer Science, Bar-Ilan University, Israel. Email: eylon.yogev@biu.ac.il. Supported by the Israel Science Foundation (Grant No. 2302/22), European Research Union (ERC, CRYPTOPROOF, 101164375), and by an Alon Young Faculty Fellowship. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

aggregated verification keys and signatures whose lengths are completely independent of the number of signers.

Contents

1 Introduction

A multi-signature scheme [IN83, BN06] enables any set of signers, within a large and potentially permissionless system, to jointly produce a compact signature on a given message. Research on the design and analysis of multi-signature schemes has recently gained significant renewed interest, as such schemes were found particularly suitable for blockchain applications. These range from drop-in replacements for standard signatures (e.g., [BDN18, MPS+19]), to smart contracts with expressive multi-owner validation procedures (e.g., [BWG+21, Arg22, Sta23]).

Two breakthroughs: Plain PK model and key aggregation. The high-level of suitability exhibited by multi-signatures to blockchain applications follows mostly due to two major breakthroughs. First, Bellare and Neven [BN06] showed that multi-signature schemes can provide security in the plain public-key model, capturing a realistic and permissionless environment, while not compromising on practicality. Specifically, in this model, each signer locally produces their signing and verification keys, without engaging in an interactive key-generation process with other signers or with a registration authority, and without augmenting verification keys with proofs of knowledge that need to be individually verified by all other signers. Second, Boneh, Drijvers and Neven [BDN18] and Maxwell, Poelstra, Seurin and Wuille [MPS+19] showed that multi-signature schemes can support non-interactive aggregation of verification keys (in the plain public-key model). Therefore, once an aggregated verification key has been verified to correspond to a particular set of signers, any dependence on the number of signers during all future signature verifications may be completely eliminated. This essentially turns the verification of multi-signatures as practical as (and even fully compatible with) that of standard signatures.

Following up on earlier constructions (e.g., [OO91, LHL94, MOR01, Bol03, LOS+06, BGO+07, RY07] and the references therein), this significant progress has led to a host of recent multi-signature schemes [DEF+19, NRS+20, AB21, BD21, NRS21, BTT22, DOT+22, FSZ22, LK23, PW23, TZ23]. Of particular interest in the blockchain setting are two-round schemes in the plain public-key model whose security is based on the hardness of the discrete logarithm (DLOG) problem in prime-order groups, as such schemes may admit practical implementations over standard elliptic curves, such as Secp256k1 or Curve25519.

Concrete security guarantees in 256-bit groups? As observed by Bellare and Dai [BD21] (and most recently also by Pan and Wagner [PW23]), despite the substantial recent efforts, the existing proofs that base the security of the proposed two-round schemes in the plain public-key model on the hardness of the DLOG problem provide rather insufficient concrete guarantees in 256-bit groups. As common for DLOG-based signatures, these proofs of security are based on the classic forking lemma [PS00, BN06], but instead of relying on a single application of the lemma, they rely on two nested applications – leading to a substantial loss in the provable concrete security.

At a high level, under the widely-accepted assumption that the success probability of any t-time algorithm in solving the DLOG problem in a group of order p is at most t^2/p [Sho97], proofs of security that rely on two nested applications of the forking lemma seem limited to bounding the success probability of t-time attackers with roughly $\left(t^2/p\right)^{1/4}$. For a 256-bit prime p, such a bound falls short of providing sufficient concrete security guarantees, especially when compared to the $\left(t^2/p\right)^{1/2}$ bound resulting from a single application of the forking lemma (as is the case, for example, with Schnorr signatures, and more generally with signature schemes obtained from identification protocols via the Fiat-Shamir transform [FS86, Sch91, AAB+02, KMP16]).[1]

[1] For simplicity, in the above discussion we did not include additional factors that depend on the number of random-

This frustrating situation has so far been approached for DLOG-based schemes in the plain public-key model via two relaxations: Either relying on the security of a recently-introduced stronger variant of the DLOG assumption (the interactive XIDL assumption introduced by Bellare and Dai [BD21]), or by proving security with respect to restricted classes of attackers [AB21, BD21, NRS21, LK23] (most notably, algebraic attackers, which are assumed to provide an algebraic justification of each group element that they produce [FKL18, AHK20, BFL20, FPS20, MTT19, RS20]). On the one hand, these relaxations have indeed led to tighter concrete security bounds. On the other hand, however, the extent to which the concrete security bounds resulting from these relaxations capture the security of the relevant schemes relative to hardness of the DLOG problem, is naturally rather limited.

1.1 Our Contributions

We present a complementing approach for designing multi-signature schemes in the plain public-key model and obtaining a better understanding of their security: Instead of relying on recently-introduced assumptions or considering restricted classes of attackers via the algebraic group model, we construct multi-signature schemes that satisfy relaxed notions of security. Our approach relaxes the full-fledged notion of security for multi-signature schemes in the plain public-key model by restricting attackers' abilities to obtain signatures of their choice. This leads us to formalizing two notions of security, *one-time unforgeability* and *single-set unforgeability*, and to construct schemes that satisfy them. Although our notions are not as strong as the full-fledged one, their applicability nevertheless ranges from serving as drop-in replacements to enabling expressive smart contract validation procedures, as we discuss in Section 1.2.

One-time multi-signatures. Our notion of one-time unforgeability naturally extends the analogous single-signer notion to the multi-signer setting by considering attackers that obtain a single signature for some message and set of signers of their choice. We construct a multi-signature scheme satisfying this notion of security in the random-oracle model based on any ring-homomorphic one-way function, admitting instantiations based on the DLOG and RSA assumptions [CD98, CFG+15].[2]

Our scheme's aggregated verification keys consist of two group elements, and when compared to the known two-round multi-signature schemes that satisfy the full-fledged notion of security for such schemes based on the hardness of the DLOG problem without relying on the algebraic group model [BD21, NRS21, TZ23], our scheme offers:[3] (1) non-interactive signing, (2) aggregate signatures that consist of a single group element, and (3) security proof that consists of a single application of the forking lemma and thus avoids the substantial security loss resulting from two nested applications (without restricting adversaries to algebraic ones). In particular, when relying on the hardness of the DLOG problem, we recover the above-discussed $\left(t^2/p\right)^{1/2}$ bound, similarly to single-signer Schnorr signatures. Although here we focus mainly on the dependence on the order p of the group, we note that our concrete bound includes additional terms that depend on the number of random-oracle queries issued by attackers. In addition, we demonstrate that our scheme naturally extends to a t-time scheme, where aggregated verification keys consist of $t+1$ group elements, while aggregated signatures still consist of a single group element.

oracle queries and signing queries issued by attackers, but rather focused mainly on the dependence on the order p of the group.

[2] We note that a notion of one-time unforgeability in the context of *aggregate* signatures was introduced by Boneh and Kim [BK20], as we discuss in Section 1.2.

[3] It is not clear how to compare the efficiency and concrete security guarantees of our one-time scheme to those of schemes that satisfy the full-fledged notion of security for multi-signature schemes. See Section 1.3 for more details.

Single-set multi-signatures. Our notion of single-set unforgeability considers attackers that obtain *any polynomial number* of signatures for messages of their choice, but are restricted to requesting all of these signatures with respect to a *single set* of signers. In this context, a single set of signers corresponds to a single vector of verification keys, which may be adversarially chosen based on the public parameters of the scheme and on the honestly-generated verification key that is attacked. As we discuss in Section 1.2, this notion already suffices for various applications of multi-signatures, such as validating blockchain transactions in a wide range of settings.

We show that any one-time multi-signature scheme with non-interactive signing can be transformed into a scheme satisfying our notion of single-set unforgeability, where the signing process of the resulting scheme consists of two rounds. Our transformation is obtained via a tree-based construction that relies on standard cryptographic tools, most notably on a non-interactive zero-knowledge proof system (we rely on these standard tools for overcoming the challenges that arise when extending the seminal Naor-Yung tree-based signature scheme [NY89] to the multi-signer setting).[4]

The length of the resulting scheme's verification keys and aggregated verification keys is independent of the number of signers, and the keys themselves are essentially identical to those of the underlying one-time scheme. The length of the resulting scheme's signatures and aggregated signatures is also independent of the number of signers, and determined by that of the underlying scheme while scaling linearly with the length of messages ($\ell\lambda$-bit signatures for ℓ-bit messages, where λ is the security parameter, as in the Naor-Yung transformation[5]). Instantiated with our one-time scheme, we obtain verification keys and signatures whose lengths are completely independent of the number of signers.

Our transformation demonstrates that at least for short messages there is no inherent and significant security loss when transforming one-time multi-signatures into single-set multi-signatures, and that our security loss essentially matches that of transforming single-user one-time signatures into re-usable ones.[6] Finally, We note that in the single-signer setting, tree-based signatures that utilize one-time signatures were initially mostly of foundational interest, whereas additional substantial efforts have demonstrated their practical applicability (see, for example, [BHH+15, AE18, BHK+19, HK22, KHR+22] and the references therein).

One-time multi-signatures: Structure vs. hardness. Revisiting our one-time multi-signature scheme, it is quite noticeable that whereas one-time single-signer signatures can be constructed based on any one-way function [Lam79], our multi-signer scheme is based on the more structured notion of a ring-homomorphic one-way function. Although such functions admit realizations based on standard number-theoretic assumptions [CD98, CFG+15], this raises the question of whether one-time multi-signatures require more structured forms of cryptographic hardness when compared to one-time single-signer signatures.

Addressing this fundamental question, we prove that one-time multi-signature schemes satisfying a natural property (which is satisfied by our one-time scheme) cannot be constructed in a fully black-box manner based on one-way functions. An interesting question for further research is whether this

[4] We emphasize that we rely on standard non-interactive zero-knowledge proofs, which can be realized based on well-studied falsifiable assumptions, and that we do not rely on succinct non-interactive arguments of knowledge (SNARKs) [Mic00, Gro10, GW11, BCI+13, BCS16]. In particular, we do not have any requirements regarding the length of the resulting proofs (they are not included in our signatures, and only play an intermediate role) and do not assume any form of proofs of knowledge.

[5] Without loss of generality, $\ell \leq \lambda$ as otherwise longer messages can first be hashed using a collision-resistant function.

[6] The work of Blazy, Kakvi, Kiltz, and Pan [BKK+15] presented a construction of re-usable signatures with a tight security reduction. However, their starting point was not a one-time signature scheme, but rather a Chameleon hash function, which is a significant more structured object.

can be circumvented via a seemingly less-natural construction.

1.2 Overview of Our Approach

In this section we first briefly discuss the applicability of schemes that satisfy our notions of security. Then, we present a high-level overview of our constructions.

The applicability of one-time and single-set multi-signatures. Our relaxed notions naturally serve as intermediate notions for obtaining a better understanding of the concrete security of full-fledged multi-signatures. At the same time, a direct application of one-time multi-signatures is for validating one-time transactions, such Bitcoin UTXOs [Nak09] (as described by Boneh and Kim [BK20] in the somewhat incomparable context of one-time aggregated signatures, which we discuss below). Specifically, a UTXO is spent after validating one or more signatures with respect to the verification keys committed in the UTXO. Once spent, it cannot be spent again, and the funds are transferred to a different UTXO. Thus, using a one-time multi-signature scheme for UTXOs can eliminate any dependence on the number of signers during verification, and to reduce both communication and storage cost. This comes at the cost of not using the same signing key for more than one UTXO, and this can be managed, for example, by deriving any number of one-time signing keys via a single master key for a pseudorandom function (thus, the one-time signing keys need not be stored, but can instead be reproduced upon demand).

A somewhat less direct application of one-time multi-signatures is for validating standard (i.e., reusable) transactions via account abstraction (e.g., [BWG+21, Arg22, Sta23]). At a high level, account abstraction (among its various features) enables smart contracts to offer arbitrary validation logic. Already in the single-signer setting, consider a smart contract whose storage includes a one-time verification key, and whenever the user provides a transaction they provide a signature with respect to the currently-stored verification key both on the provided input to the smart contract and on a newly-generated one-time verification key that the contract will store instead of its current one. This is motivated by the textbook path-based construction of signatures from one-time signatures [KL21], which is typically presented as a warm-up for the classic Naor-Yung tree-based construction [NY89]. Unlike the textbook construction, here the verification time and signature length do not scale with the number of previously-generated signatures, since each newly-generated verification key replaces its predecessor in the contract's storage.[7] Such a mechanism extends to the multi-signer setting in various ways using a one-time multi-signature scheme, where each transaction additionally updates the stored aggregated verification key. Here, the advantages of using a one-time multi-signature scheme with non-interactive signing and concrete security guarantees based on the hardness of the DLOG problem may be significant.

Finally, for our notion of single-set multi-signatures, where an adversary may observe any polynomial number of signatures for a single set of signers, account abstraction is again useful. Specifically, for any smart contract whose transactions require validating multiple signers, as long as each signer allocates a contract-specific signing key that is not used for any other purpose, then single-set security suffices (and there is no need for any key updates as with one-time multi-signatures). As discussed above, such contract-specific keys can be derived via a single master key, and thus do not have to be explicitly stored.

[7] It should be noted that, over time, such a mechanism may run into various synchronization issues since each signing key can be used only once. Such issues can be dealt with either by using a t-time multi-signature scheme, or by using a recovery mode (again, enabled by account abstraction) that allows key updates via a standard multi-signature scheme. Assuming that such issues are hopefully not-too-frequent, the overall efficiency of the validation procedure would be determined by that of the one-time scheme.

Our one-time multi-signature scheme. The starting point of our one-time scheme is the Schnorr-based one-time signature scheme designed by Bellare and Shoup [BS08], which was extended by Boneh and Kim [BK20] to DLOG-based and lattice-based one-time *aggregate* signature schemes. Recall that aggregate signature schemes enable to aggregate signatures on *any* set of messages, whereas multi-signature schemes enable to aggregate signatures on the same message. As a result, the schemes of Boneh and Kim do not support aggregation of verification keys, and their verification time scales linearly with the number of signers. We use a different aggregation method that is tailored to aggregating signatures on the same message. Once such an aggregated key has been verified to correspond to a particular set of signers (e.g., in a preliminary phase as discussed above for blockchain transactions), this enables us to guarantee that the signing and verification operations are independent of the number of signers.

Our scheme is based on the abstract notion of a ring-homomorphic one-way function, introduced by Catalano, Fiore, Gennaro and Vamvourellis [CFG+15]. At a high level, we consider an efficiently-computable homomorphism $f : \mathcal{X} \to \mathcal{Y}$ for cyclic groups $\mathcal{X}$ and $\mathcal{Y}$ that allows computing linear operations "in the exponent" over a ring $\mathbb{K} = \mathbb{Z}_q$ for some prime q (in the DLOG-based instantiation, the prime q corresponds to the order of the cyclic groups[8], but in the RSA-based instantiation this is not the case – see Section 2.1 for a formal definition).

Each signer in our scheme samples $x, r \leftarrow \mathcal{X}$, and sets $\mathsf{sk} = (x, r)$ and $\mathsf{vk} = (X, R) = (f(x), f(r))$ as their signing key and verification key, respectively. Then, a vector $\vec{\mathsf{vk}} = ((X_1, R_1), \ldots, (X_n, R_n))$ of verification keys, corresponding to n signers, is aggregated by computing $\mathsf{aggvk} = (\mathsf{agg}X, \mathsf{agg}R) = (\prod_{i=1}^{n} X_i^{a_i}, \prod_{i=1}^{n} R_i^{a_i})$, where $(a_1, \ldots, a_n) = \mathsf{H}_1\left(\vec{\mathsf{vk}}\right)$ for a hash function H_1. For any message m and vector $\vec{\mathsf{vk}}$ of verification keys, each signer computes a signature $\sigma_i = r_i + \mathsf{H}_0(m, \mathsf{aggvk}) \cdot x_i$ for a hash function H_0, and signatures are similarly aggregated by computing $\mathsf{agg}\sigma = \sum_{i=1}^{n} a_i \cdot \sigma_i$. In turn, this enables to verify an aggregate signature $\mathsf{agg}\sigma$ with respect to an aggregated verification key $\mathsf{aggvk} = (\mathsf{agg}X, \mathsf{agg}R)$ by checking whether $f(\mathsf{agg}\sigma) = (\mathsf{agg}X)^{\mathsf{H}_0(m, \mathsf{aggvk})} \cdot \mathsf{agg}R$.

Our security proof models the hash function H_0 and H_1 as random oracles, and reduces the task of breaking the one-time unforgeability of the scheme to that of breaking the one-wayness property of the ring-homomorphic function (see Definition 2.1). Specifically, given any attacker for our scheme, we construct an inverter that is given $X = f(x)$ for a randomly chosen x, and outputs a pair (x', d) such that $f(x') = X^d$ and $d \neq 0$ (note that, in the DLOG-setting d can always be efficiently inverted, and thus without loss of generality $d = 1$, but in the RSA-setting this is not the case). At a very high level, our proof programs the random oracle H_0 for embedding the value X in the honestly-generated verification key given as input to the attacker, while generating R in a way that would be consistent with the response to the attacker's single signing query. Then, the proof relies on a single application of the forking lemma using the random oracle H_1 for resampling the value of a_i that corresponds to position of the honestly-generated verification key in the aggregated verification key with respect to which the attacker produces a forgery. Given two such forgeries, we are then able to efficiently produce x' and d as required. The proof naturally contains a variety of challenges for implementing this high-level idea, and we refer the reader to Section 4 for a complete and formal description.

From one-time to single-set multi-signatures. As mentioned above, our approach is inspired by the Naor-Yung transformation of a one-time signature scheme into a reusable one [NY89]. Recall that, in the Naor-Yung transformation, the signer implicitly holds a binary tree of exponential size, where each node of the tree is associated with a pair of one-time signing and verification keys. Specifically, for signing ℓ-bit messages, the tree has 2^ℓ leaves, where each leaf and each internal node

[8] In the DLOG-based instantiation, the homomorphism is simply the group exponentiation operation relative to a given generator.

$\alpha \in \{0,1\}^{\leq \ell} \cup \{\varepsilon\}$ is associated with a pair $(\mathsf{sk}_\alpha, \mathsf{vk}_\alpha)$ of one-time keys.[9] The keys sk_ε and vk_ε corresponding to the root ε serve as the signing key and verification key, respectively, and all other keys (and randomness that may be needed for using them) do not have to be explicitly generated or stored, but rather can be produced whenever needed using a pseudorandom function whose key is additionally included in the signing key.

For signing a message $m \in \{0,1\}^\ell$, the signer first uses the signing key sk_m associated with the leaf correspond to the binary string m for signing the message m itself. Then, it uses the signing keys positioned on the path connecting the root ε to the leaf m for certifying the path: For every $j \in \{0,\ldots,\ell-1\}$, the signer uses the key $\mathsf{sk}_{m|_j}$ associated with the internal node corresponding to the binary string $m|_j$ to sign the concatenation of the two verification keys $\mathsf{vk}_{m|_j 0}$ and $\mathsf{vk}_{m|_j 1}$ corresponding to its two children in the tree.[10] This makes sure that each signing key is used at most once, thus enabling to rely on the one-time security of the underlying scheme.

Equipped with the Naor-Yung transformation, let us attempt extending it to the multi-signer setting. As in the single-signer setting, suppose that each signer implicitly holds a tree, where each node $\alpha \in \{0,1\}^{\leq \ell} \cup \{\varepsilon\}$ is associated with a pair $(\mathsf{sk}_\alpha, \mathsf{vk}_\alpha)$ of keys for a one-time *multi-signature* scheme. The keys sk_ε and vk_ε corresponding to the root ε again serve as the signing key and verification key, respectively, and all other keys are similarly generated using a pseudorandom function upon demand. Note that this structure enables to aggregate the root verification keys $\mathsf{vk}_\varepsilon^{(1)},\ldots,\mathsf{vk}_\varepsilon^{(n)}$ of any n signers by using the key-aggregation algorithm of the underlying one-time scheme. More generally, it enables to aggregate not only the root keys, but to implicitly define an *aggregated tree*. In the aggregated tree, each node $\alpha \in \{0,1\}^{\leq \ell} \cup \{\varepsilon\}$ is associated with an aggregated verification key aggvk_α that is obtained by aggregating the one-time verification keys $\mathsf{vk}_\alpha^{(1)},\ldots,\mathsf{vk}_\alpha^{(n)}$ associated with the node α in the n individual trees.

This observation leads to the following elegant, yet insecure, two-round signing protocol. For signing a message m with respect to signers with root verification keys $\mathsf{vk}_\varepsilon^{(1)},\ldots,\mathsf{vk}_\varepsilon^{(n)}$, each signer first sends all other signers the 2ℓ verification keys on the path leading from the signer's root to the leaf m. At this point, all signers know the verification keys $\mathsf{vk}_{m|_j b}^{(i)}$ for all $i \in [n]$, $j \in \{0,\ldots,\ell-1\}$ and $b \in \{0,1\}$. This enables each signer to compute the aggregated verification keys $\mathsf{aggvk}_{m|_j b}$ along the path from the root to the leaf m in the aggregated tree. Now, for each level $j \in \{0,\ldots,\ell-1\}$, each signer uses their one-time signing key $\mathsf{sk}_{m|_j}^{(i)}$ to non-interactively compute a signature $\sigma_{m|_j}^{(i)}$ on the concatenation of the two aggregated verification keys $\mathsf{aggvk}_{m|_j 0}$ and $\mathsf{aggvk}_{m|_j 1}$ with respect to the signer set $\mathsf{vk}_{m|_j}^{(1)},\ldots,\mathsf{vk}_{m|_j}^{(n)}$. Finally, each signer uses their one-time signing key $\mathsf{sk}_m^{(i)}$ to compute a signature $\sigma_m^{(i)}$ on the message m with respect to the signer set $\mathsf{vk}_m^{(1)},\ldots,\mathsf{vk}_m^{(n)}$. These signatures are then aggregated for each level $j \in \{1,\ldots,\ell\}$ to produce a signature that consists of ℓ aggregated one-time signatures. Note that both the length of the resulting signature and the time required to verify it scale linearly with the length of the message (as in the Naor-Yung scheme), but are completely independent of the number n of signers.

At this point, we would like to argue that since each one-time signing key is used at most once, then we can rely on the security of the underlying one-time multi-signature scheme to claim that our tree-based scheme is secure against attackers issuing any polynomial number of signing queries. This argument fails, however, if the attacker can request signatures with respect to more than one set of signers. The reason is that different sets of signers induce different aggregated trees. Consider an adversary that requests a signature from some signer i with respect to a set $\mathcal{S}$, and then requests

[9] We denote by $\{0,1\}^{\leq \ell}$ the set of all binary strings of length at most ℓ, and by ε the empty string.

[10] We denote by $m|_j$ the leftmost j bits of a binary string m (where $m|_0 = \varepsilon$), and we denote by $m|_j b$ the binary string obtained by concatenating the strings $m|_j$ and b.

a signature from the same signer i with respect to a different set $\mathcal{S}'$. The first signature includes a signature with respect to the one-time signing key $\mathsf{sk}_\varepsilon^{(i)}$ on information derived from the aggregated tree corresponding to $\mathcal{S}$, and the second signature includes a signature with respect to the same one-time signing key $\mathsf{sk}_\varepsilon^{(i)}$ on information derived from the aggregated tree corresponding to $\mathcal{S}'$. Since the underlying scheme is only guaranteed to be one-time secure, the security of the tree-based construction breaks down.

Still, one might hope that the construction is secure as long as the attacker issues signature queries with respect to a single set of signers, eliminating the issue we just described. This coincides with our single-set notion of security. However, this is not the case. This added restriction is insufficient because a root verification key may be used by malicious signers for different trees. That is, a malicious signer can still send different verification keys in the first round of two invocations of the signing protocol. This again results in two distinct aggregated trees (even though the two invocations of the signing protocol share the same set of signers).

We resolve this additional challenge by identifying signers not only with their root verification key, but also with a commitment containing a key for a pseudorandom function from which their entire tree is derived. Now, in the first round of the signing protocol, each signer sends all one-time verification keys on the path from their root to the respective leaf, while augmenting each such key with a non-interactive zero-knowledge proof asserting that it has been generated correctly. From a foundational standpoint, such proofs can be based on the existence of trapdoor functions[11] [FLS90]. For practical instantiations, these NIZK proofs can be based on one of the many recent efficient protocols[12]. Crucially, these proofs are not included as part of the resulting signature, but rather only serve as "proofs of semi-honest behavior" that enable each signer to continue to the second round of the signing protocol. This describes the high-level intuitive structure of our scheme, and we refer the reader to Section 5 for a complete and formal description.

1.3 Related Work

DDH-based multi-signatures. Whereas most of the work on multi-signatures in prime-order groups focused on DLOG-based schemes, several schemes were suggested also based on the DDH assumption (e.g. [LYG19, FH21, TSS+23, PW23]). As in the single-signer setting, DDH-based signatures may lead to tighter reductions. Although, when implemented in concrete groups, such schemes typically do not offer the same efficiency guarantees as DLOG-based ones.

DLOG-based two-round multi-signatures. Existing DLOG-based two-round multi-signature schemes can be roughly divided into three categories: (1) schemes whose security is established in the algebraic group model [AB21, BD21, NRS21, LK23], (2) schemes whose security is established based on interactive variants of the DLOG problem (without relying on the algebraic group model) [BD21, NRS21], and (3) schemes whose security is established based on the standard DLOG problem (again, without relying on the algebraic group model) [BD21, NRS21, TZ23]. As discussed in Section 1.1, it is challenging to compare the efficiency and concrete security guarantees of our one-time scheme to those of the existing DLOG-based schemes, as these schemes satisfy the full-fledged notion of security for multi-signature schemes. If any comparison can be made, it would seem essential for focus on category 3, since the schemes in categories 1 and 2 seem to inherently avoid nested applications of the forking lemma.

[11] Technically speaking, these have to be certifiably injective and doubly-enhanced [BY96, Gol11, GR13, CL18].

[12] See [GS08, Gro16, BBB+18, BSBH+18, GWC19, CHM+20, XZS22, GLS+23] and the many references therein for a highly non-exhaustive list of examples.

Focusing on category 3, the schemes of Bellare and Dai [BD21], Nick, Ruffing and Seurin [NRS21], and Tessaro and Zhu [TZ23] rely on two nested applications of the forking lemma (whereas we rely on a single application), their verification keys consist of a single group element (whereas our verification keys consist of two group elements), and their signatures consist of two, two and three group elements, respectively (whereas our signatures consist of a single group element).

Synchronized multi-signatures. A substantially different tree-based approach, both in terms of its goals and in terms of its structure, was recently presented by Fleischhacker, Simkin and Zhang [FSZ22]. They constructed a lattice-based multi-signature scheme in the *synchronized* model, where it is assumed that signers share a global notion of time, and the signing algorithm takes the current time step as an additional input. Most notably, it is additionally assumed that no signer signs more than one message per time step, and the goal is to aggregate signatures for the same message and *same time step*, *without knowing the set of signers in advance*. Thus, a signature may be aggregated together with those of any subset of other signers. As noted by Fleischhacker et al. such flexibility seems particularly useful in the blockchain setting for the task of confirming newly-generated blocks: Block validators may be synchronized by the number of the block that they sign, each validator does not sign more than one block in each time period, and validators do not know which of the other potential validators will actually participate.

In contrast to the synchronized model, we design our scheme in the plain public key model [BN06], where there is no global notion of time (or any other form of synchronization), and we aim at aggregating signatures not for a given time step but rather for a given set of signers that is specified during the signing process. In particular, our scheme guarantees that a signature may be aggregated only with a specific set of signers that is provided to the signing algorithm as input, while keeping the verification time independent of the number of signers.

From the technical perspective, as discussed above, our approach relies on trees of exponential size, which are never explicitly constructed in their entirety. Each of the exponential number of leaves corresponds to a message-dedicated verification key for a one-time multi-signature scheme, and the path leading to each leaf is constructed upon demand using a pseudorandom function. In contrast, for supporting T time periods, Fleischhacker et al. explicitly sample T key pairs, and then compute a homomorphic Merkle tree with the corresponding T verification keys as its leaves. As a result, their scheme seems limited to supporting only a polynomial number T of time periods, and the efficiency of their key-generation algorithm scales linearly with T.

Finally, we note that since messages in their scheme are not signed with respect to a given set of signers (but rather a signature can be aggregated together with those of any subset of other signers), the notion of security required from their one-time primitive does not explicitly consider multiple signers. Specifically, Fleischhacker et al. introduce a notion of single-signer one-time key-homomorphic signatures, whereas we explicitly introduce a notion of one-time multi-signatures satisfying security guarantees tailored to the multi-signer setting, and our tree-based construction can rely on any multi-signature scheme that satisfies it.

2 Preliminaries

In this section we present the basic notions and the cryptographic primitives and tools that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$.

2.1 Ring-Homomorphic One-Way Functions

Our construction of a one-time multi-signature scheme relies on the notion of ring-homomorphic one-way functions, introduced by Catalano, Fiore, Gennaro and Vamvourellis [CFG+15]. This notion was presented by Catalano et al. as part of their framework for algebraic one-way functions, which is closely-related to the notion of group-homomorphic one-way functions introduced by Cramer and Damgård [CD98]. The notion considers function families $\mathsf{F} = (\mathsf{Setup}, \mathsf{Eval})$ where for any $\lambda \in \mathbb{N}$ and for any function index Ind produced by $\mathsf{F.Setup}(1^\lambda)$ it holds that $\mathsf{F.Eval}(\mathsf{Ind}, \cdot) : \mathcal{X}_{\mathsf{Ind}} \to \mathcal{Y}_{\mathsf{Ind}}$ is an efficiently-computable homomorphism for cyclic groups $\mathcal{X}_{\mathsf{Ind}}$ and $\mathcal{Y}_{\mathsf{Ind}}$ that allows computing linear operations "in the exponent" over a ring $\mathbb{K}_{\mathsf{Ind}}$. For our purposes, we consider the specific case where $\mathbb{K}_{\mathsf{Ind}} = \mathbb{Z}_q$ for some prime $q = q(\mathsf{Ind})$, and note that already this case captures the known constructions based on the hardness of the discrete-logarithm problem (i.e., the exponentiation function in a cyclic group) and RSA problem (i.e., the RSA function) as we discuss below [CD98, CFG+15].

As formalize by Catalano et al. [CFG+15], such a function family F is *ring homomorphic* if there exists an efficient algorithm Eval' such that for any $\lambda \in \mathbb{N}$, Ind produced by $\mathsf{Setup}(1^\lambda)$, generators $h_1, \ldots, h_m \in \mathcal{X}_{\mathsf{Ind}}$, vector of elements $\boldsymbol{W} = (W_1, \ldots, W_\ell) \in \mathcal{X}_{\mathsf{Ind}}^\ell$ where $W_i = h_1^{\omega_i^{(1)}} \cdots h_m^{\omega_i^{(m)}} \cdot R_i$, for some $R_i \in \mathcal{X}_{\mathsf{Ind}}$ and some integers $\omega_i^{(j)} \in \mathbb{Z}$ (note that this decomposition may not be unique), and vector of integers $\boldsymbol{\alpha} \in \mathbb{Z}^\ell$, it holds that

$$\mathsf{Eval}'(\mathsf{Ind}, \boldsymbol{A}, \boldsymbol{W}, \boldsymbol{\Omega}, \boldsymbol{\alpha}) = h_1^{\left\langle \boldsymbol{\omega}^{(1)}, \boldsymbol{\alpha} \right\rangle} \cdots h_m^{\left\langle \boldsymbol{\omega}^{(m)}, \boldsymbol{\alpha} \right\rangle} \prod_{i=1}^{\ell} R_i^{\alpha_i}$$

where $\boldsymbol{A} = (A_1, \ldots, A_m) \in \mathcal{Y}_{\mathsf{Ind}}^m$ is such that $A_i = F_{\mathsf{Ind}}(h_i)$, $\boldsymbol{\Omega} = \left(\omega_i^{(j)}\right)_{i,j} \in \mathbb{Z}^{\ell \times m}$, and each product $\left\langle \boldsymbol{\omega}^{(j)}, \boldsymbol{\alpha} \right\rangle$ in the exponent is computed over the ring $\mathbb{K}_{\mathsf{Ind}}$.

In terms of one-wayness, Catalano et al. formalized the following notion of *flexible one-wayness*, asking that given $X = \mathsf{F}_{\mathsf{Ind}}(x)$ for a uniformly distributed $x \in \mathcal{X}_{\mathsf{Ind}}$, it should be infeasible to output $x' \in \mathcal{X}_{\mathsf{Ind}}$ and $d \in \mathbb{K}_{\mathsf{Ind}}$ such that $\mathsf{F}_{\mathsf{Ind}}(x') = X^d$ and $d \neq 0_{\mathbb{K}_{\mathsf{Ind}}}$.

Definition 2.1. A ring-homomorphic function family $\mathsf{F} = (\mathsf{Setup}, \mathsf{Eval})$ is *flexible one-way* if for every probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{hom\text{-}ow}}_{\mathsf{F},A}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\begin{array}{c} A(\mathsf{Ind}, X) = (x', d) \text{ s.t.} \\ \mathsf{F}_{\mathsf{Ind}}(x') = X^d \text{ and } d \neq 0_{\mathbb{K}_{\mathsf{Ind}}} \end{array}\right] \leq \nu(\lambda),$$

where $\mathsf{Ind} \leftarrow \mathsf{Setup}(1^\lambda)$, $x \leftarrow \mathcal{X}_{\mathsf{Ind}}$ and $X = \mathsf{F}_{\mathsf{Ind}}(x)$.

Note that if $\mathbb{K}_{\mathsf{Ind}} = \mathbb{Z}_q$, where q is the order of $\mathcal{X}_\lambda$, then the above definition is equivalent to the standard notion of one-wayness. In particular, the hardness of computing discrete logarithms in cyclic groups is equivalent to the flexible one-wayness of the group exponentiation function in such groups. In addition, Catalano et al. [CFG+15] showed that the RSA function $x \to x^e \bmod N$ in the subgroup $\mathbb{QR}_N \subset \mathbb{Z}_N^*$ of quadratic residues (where N is the product of two "safe primes" and thus $\mathbb{QR}_N$ is cyclic) is flexible one-way based on the RSA assumption. In this case, $\mathbb{K}_{\mathsf{Ind}} = \mathbb{Z}_e$ for any prime $e \geq 3$.

2.2 The Forking Lemma

The proof of security for our one-time multi-signature scheme relies on the "forking lemma" of Bellare and Neven [BN06] (following Pointcheval and Stern [PS00]). Let $q \geq 1$ be an integer, and let $\mathcal{H}$, $\mathcal{X}$

and $\mathcal{Y}$ be a sets such that $|\mathcal{H}| \geq 2$. Let A be a randomized algorithm that on input $(x, \vec{h}) \in \mathcal{X} \times \mathcal{H}^q$ returns either a pair $(i, y) \in [q] \times \mathcal{Y}$ or the dedicated symbol $\perp$. Let F_A be an algorithm that takes inputs in $\mathcal{X}$ and returns either an output $(y, y') \in \mathcal{Y}^2$ or the dedicated symbol $\perp$, and is defined as follows:

1. Sample random coins $\rho \leftarrow \{0,1\}^*$ for A.
2. Sample $h_1, \ldots, h_q \leftarrow \mathcal{H}$ and compute $\mathsf{out}_1 = A(x, h_1, \ldots, h_q; \rho)$.
3. If $\mathsf{out}_1 = \perp$ then return $\perp$, and otherwise let $\mathsf{out}_1 = (i, y)$.
4. Sample $h'_i, \ldots, h'_q \leftarrow \mathcal{H}$ and compute $\mathsf{out}_2 = A(x, h_1, \ldots, h_{i-1}, h'_i, \ldots, h_q; \rho)$.
5. If $\mathsf{out}_2 = \perp$ then return $\perp$, and otherwise let $\mathsf{out}_2 = (i', y')$.
6. If $i' = i$ and $h_i \neq h'_i$ then return (i, y, y'), and otherwise return $\perp$.

The following lemma, due to Bellare and Neven [BN06], relates the probability that F_A successfully provides an output (other than $\perp$) to the corresponding probability of A.

Lemma 2.2 ([BN06])**.** *For any algorithm A and for any distribution D over $\mathcal{X}$ it holds that*

$$\Pr_{x \leftarrow D}\left[F_A(x) \neq \perp\right] \geq \epsilon \cdot \left(\frac{\epsilon}{q} - \frac{1}{|\mathcal{H}|}\right),$$

where

$$\epsilon = \Pr_{\substack{x \leftarrow D \\ \vec{h} \leftarrow \mathcal{H}^q}}\left[A\left(x, \vec{h}\right) \neq \perp\right].$$

2.3 Additional Cryptographic Primitives

In this section we formally define additional basic cryptographic primitives that are used as building blocks in our construction of a single-set multi-signature scheme: Pseudorandom functions, collision-resistant hash functions, non-interactive statistically-binding commitment schemes, and non-interactive zero-knowledge proofs.

Pseudorandom functions. For integers ℓ_{in} and ℓ_{out} we denote by $\mathsf{Func}_{\ell_{\mathsf{in}},\ell_{\mathsf{out}}}$ the set of all functions mapping ℓ_{in}-bit inputs to ℓ_{out}-bit outputs. We rely on the following standard notion of pseudorandom functions (e.g., [Gol01]):

Definition 2.3. Let $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(\lambda)$ and $\ell_{\mathsf{out}} = \ell_{\mathsf{out}}(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A *pseudorandom function* is a pair of polynomial-time algorithms $\Pi = (\mathsf{KG}, \mathsf{Eval})$ with the following two properties:

1. Eval is a deterministic algorithm such that for any $\lambda \in \mathbb{N}$ and for any k produced by $\mathsf{KG}(1^\lambda)$ it holds that $\mathsf{Eval}(\mathsf{k}, \cdot) : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$.
2. For every probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that
$$\mathsf{Adv}^{\mathsf{prf}}_{\Pi,A}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[A^{\mathsf{Eval}(\mathsf{k},\cdot)}(1^\lambda) = 1\right] - \Pr\left[A^{f(\cdot)}(1^\lambda) = 1\right]\right| \leq \nu(\lambda),$$
where the probability is taken over the choices of $\mathsf{k} \leftarrow \mathsf{KG}(1^\lambda)$, and $f \leftarrow \mathsf{Func}_{\ell_{\mathsf{in}}(\lambda),\ell_{\mathsf{out}}(\lambda)}$, and over the internal randomness of A.

Collision-resistant hash functions. We rely on the standard notion of collision-resistant hash functions (e.g., [Gol01]):

Definition 2.4. Let $\ell_{\mathsf{in}} = \ell_{\mathsf{in}}(\lambda)$ and $\ell_{\mathsf{out}} = \ell_{\mathsf{out}}(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A *collision-resistant hash family* is a pair of polynomial-time algorithms $\Pi = (\mathsf{KG}, \mathsf{Eval})$ with the following two properties:

1. Eval is a deterministic algorithm such that for any $\lambda \in \mathbb{N}$ and for any k produced by $\mathsf{KG}(1^\lambda)$ it holds that $\mathsf{Eval}(\mathsf{k}, \cdot) : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$.

2. For every probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that
$$\mathsf{Adv}^{\mathsf{crh}}_{\Pi,A}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[A\left(1^\lambda, \mathsf{k}\right) = (x, x') \text{ s.t. } \begin{array}{c} x \neq x' \in \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \text{ and} \\ \mathsf{Eval}(\mathsf{k}, x) = \mathsf{Eval}(\mathsf{k}, x') \end{array}\right],$$
where the probability is taken over the choice of $\mathsf{k} \leftarrow \mathsf{KG}(1^\lambda)$ and over the internal randomness of A.

Non-interactive statistically-binding commitment schemes. We rely on the following standard notion of a non-interactive statistically-binding commitment scheme [Nao91, Gol01]:

Definition 2.5. Let $\epsilon_{\mathsf{binding}} = \epsilon_{\mathsf{binding}}(\lambda)$ be a function of the security parameter $\lambda \in \mathbb{N}$. A *non-interactive $\epsilon_{\mathsf{binding}}$-statistically-binding commitment scheme* for a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is a triplet of probabilistic polynomial-time algorithms $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ with the following properties:

1. **Perfect completeness:** For every $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}_\lambda$ it holds that
$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, \mathsf{decom}, m) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{crs}, m) \end{array}\right] = 1$$
where the probability is taken over the internal randomness of Setup, Commit and Verify.

2. **Statistical binding:** For every $\lambda \in \mathbb{N}$ it holds that
$$\Pr\left[\begin{array}{c} \exists \mathsf{com}, m \neq m', \mathsf{decom}, \mathsf{decom}' \text{ s.t.} \\ \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, \mathsf{decom}, m) = \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, \mathsf{decom}', m') = 1 \end{array}\right] \leq \epsilon_{\mathsf{binding}}(\lambda)$$
where the probability is taken over the choice of $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.

3. **Computational hiding:** For every probabilistic polynomial-time algorithm $A = (A_1, A_2)$ there exists a negligible function $\nu(\cdot)$ such that
$$\mathsf{Adv}^{\mathsf{hiding}}_{\Pi,A}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{hiding}}_{\Pi,A}(0,\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{hiding}}_{\Pi,A}(1,\lambda) = 1\right]\right| \leq \nu(\lambda)$$
for all sufficiently large n, where for each $b \in \{0,1\}$ the experiment $\mathsf{Exp}^{\mathsf{hiding}}_{\Pi,A}(b,\lambda)$ is defined as:
   (a) $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.
   (b) $(m_0, m_1, \mathsf{st}) \leftarrow A_1(1^\lambda, \mathsf{crs})$.
   (c) $(\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{crs}, m_b)$.
   (d) $b' \leftarrow A_2(\mathsf{crs}, \mathsf{com}, \mathsf{st})$.
   (e) Output b'.

Non-interactive zero-knowledge proof systems. We rely on the following standard notion of a non-interactive simulation-sound zero-knowledge proof system [BFM88, BSM+91, FLS90, Sah99, SCO+01, Gro06, GOS06, Lin06]:

Definition 2.6. A *non-interactive simulation-sound zero-knowledge proof system* for a language $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda\in\mathbb{N}}$ with a witness relation $R(\mathcal{L}) = \{R_\lambda\}_{\lambda\in\mathbb{N}}$ is a 5-tuple of probabilistic polynomial-time algorithms $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ with the following properties:

1. **Perfect completeness:** For every $\lambda \in \mathbb{N}$ and $(x, w) \in R_\lambda$ it holds that

$$\Pr\left[\mathsf{V}(1^\lambda, \mathsf{crs}, x, \pi) = 1 \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{P}(1^\lambda, \mathsf{crs}, x, w) \end{array}\right] = 1$$

   where the probability is taken over the internal randomness of Setup, P and V.

2. **Adaptive zero knowledge:** For every probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{zk}}_{\Pi,A}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{zk}}_{\Pi,A}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{zk}}_{\Pi,A,\mathsf{Sim}_1,\mathsf{Sim}_2}(\lambda) = 1\right]\right| \leq \nu(\lambda)$$

   for all sufficiently large n, where the experiment $\mathsf{Exp}^{\mathsf{zk}}_{\Pi,A}(\lambda)$ is defined as:

   (a) $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.
   (b) $b \leftarrow A^{\mathsf{P}(1^\lambda,\mathsf{crs},\cdot,\cdot)}(1^\lambda, \mathsf{crs})$.
   (c) Output b.

   and the experiment $\mathsf{Exp}^{\mathsf{zk}}_{\Pi,A,\mathsf{Sim}_1,\mathsf{Sim}_2}(\lambda)$ is defined as:

   (a) $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda)$.
   (b) $b \leftarrow A^{\mathsf{Sim}'_2(1^\lambda,\tau,\cdot,\cdot)}(1^\lambda, \mathsf{crs})$, where $\mathsf{Sim}'_2(1^\lambda, \tau, x, w) = \mathsf{Sim}_2(1^\lambda, \tau, x)$ if $(x, w) \in R_\lambda$ and $\mathsf{Sim}'_2(1^\lambda, \tau, x, w) = \perp$ otherwise.
   (c) output b.

3. **Simulation soundness:** For every probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{ss}}_{\Pi,A}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{Exp}^{\mathsf{ss}}_{\Pi,A}(\lambda) = 1\right] \leq \nu(\lambda)$$

   for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{Exp}^{\mathsf{ss}}_{\Pi,A}(\lambda)$ is defined as:

   (a) $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda)$.
   (b) $(x, \pi) \leftarrow A^{\mathsf{Sim}_2(1^\lambda,\tau,\cdot)}(1^\lambda, \mathsf{crs})$.
   (c) Denote by Q the set of Sim_2's query-response pairs.
   (d) Output 1 if and only if $x \notin \mathcal{L}_\lambda$, $(x, \pi) \notin Q$, and $\mathsf{V}(1^\lambda, \mathsf{crs}, x, \pi) = 1$.

3 One-Time and Single-Set Security for Multi-Signature Schemes

In this section we formalize our two notions of security for multi-signature schemes. These notions are obtained by relaxing the notion of security for multi-signature schemes, formalized by Bellare and Neven [BN06] and recently refined by Bellare and Dai [BD21], by restricting adversaries' abilities to obtain signatures of their choice. In what follows we briefly recall the syntax and correctness requirement of multi-signature schemes, and then formally present our notions of security in Sections 3.1 and 3.2.

A multi-signature scheme is a six-tuple $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KAgg}, \mathsf{Sign}, \mathsf{SAgg}, \mathsf{Verify})$ of polynomial-time algorithms. The setup algorithm Setup receives as input the unary representation of the security parameter $\lambda \in \mathbb{N}$ and outputs public parameters pp. The key-generation algorithm KG receives as input the public parameters pp, and outputs a signing key sk and a verification key vk. The key-aggregation algorithm KAgg is a deterministic algorithm that takes as input the public parameters pp and a vector of verification keys $\vec{\mathsf{vk}}$, and outputs an aggregated verification key aggvk.[13] For schemes with non-interactive signing, the signing algorithm Sign receives as input the public parameters pp, a signing key sk, a vector $\vec{\mathsf{vk}}$ of verification keys, and a message m that is taken from a message space $\mathcal{M}$, and outputs a signature σ. For schemes with interactive signing, the signing algorithm defines an interactive protocol by additionally receiving as input at each round the relevant party's internal state the communication produced by all other parties. The signature-aggregation algorithm SAgg is a deterministic algorithm that takes as input the public parameters pp, a vector of verification keys $\vec{\mathsf{vk}}$, and a vector of signatures $\vec{\sigma}$, and outputs an aggregated signature σ. Finally, the verification algorithm Verify receives as input the public parameters pp, an aggregated verification key aggvk, a message m and an aggregated signature σ, and outputs either 0 or 1.

In terms of correctness, we consider the following standard requirement, which we formalize for simplicity for schemes with non-interactive signing and then discuss its extension to schemes with interactive signing:

Definition 3.1. A multi-signature scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KAgg}, \mathsf{Sign}, \mathsf{SAgg}, \mathsf{Verify})$ with non-interactive signing over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is *perfectly correct* if for any polynomial $n = n(\cdot)$, security parameter $\lambda \in \mathbb{N}$, and message $m \in \mathcal{M}_\lambda$ it holds that

$$\Pr\left[\mathsf{Verify}\left(\mathsf{pp}, \mathsf{KAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}\right), m, \mathsf{SAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}, \vec{\sigma}\right)\right) = 1\right] = 1$$

for $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, where the probability is taken over the choice of $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, and over the choices of $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KG}(\mathsf{pp})$ and $\sigma_i \leftarrow \mathsf{Sign}\left(\mathsf{pp}, \mathsf{sk}_i, \vec{\mathsf{vk}}, m\right)$ for every $i \in [n]$.

The above definition extends to schemes with interactive signing by letting $(\sigma_1, \ldots, \sigma_n)$ denote the local output of each party in the interactive signing protocol, where each party $i \in [n]$ is provided with $\left(\mathsf{pp}, \mathsf{sk}_i, \vec{\mathsf{vk}}, m\right)$ as input. We refer the reader to the work of Bellare and Dai [BD21] for a formal treatment of the correctness requirement for schemes with interactive signing.

A standard relaxation of the above definition allows for a negligible error probability. Concretely, our single-set multi-signature scheme presented in Section 5 provides perfect correctness whenever the vector $\vec{\mathsf{vk}}$ consists of distinct verification keys. Since essentially any notion of security for signature schemes guarantees that collisions among honestly-generated verification keys may occur only with a negligible probability, this yields at most a negligible error probability.

[13] For our framework we view collections of verification keys as vectors and not sets. Note that any set can be uniquely transformed into a vector by determining an order among its elements (e.g., lexicographic order).

3.1 One-Time Unforgeability

For presenting the notion of one-time unforgeability for multi-signature schemes, we focus on multi-signature schemes with non-interactive signing, and note that the following treatment naturally extends to schemes with interactive signing (our one-time multi-signature scheme in Section 4 has non-interactive signing).

This notion of security captures attacks in which an adversary may obtain a single signature of their choice. Specifically, we consider a security experiment in which the adversary first receives the public parameters pp of the scheme and an honestly-generated verification key vk. Then, the adversary may issue a single signing query for some message m with respect to some set of signers indicated via a vector $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ of verification keys. Both the message m and the vector $\vec{\mathsf{vk}}$ of verification keys may be arbitrarily chosen (in polynomial time) by the adversary based on the public parameters and the honest verification key. Next, the adversary obtains a signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, m)$ produced using the signing key sk corresponding to the honest verification key vk (note that if $\mathsf{vk} \notin \vec{\mathsf{vk}}$ then the signing algorithm may be defined to output $\perp$), and the adversary's goal is to output a non-trivial forgery $(\vec{\mathsf{vk}}^*, m^*, \mathsf{agg}\sigma^*)$. The non-triviality of the forgery is reflected in the fact that $\mathsf{vk} \in \vec{\mathsf{vk}}^*$ and $(\vec{\mathsf{vk}}^*, m^*) \neq (\vec{\mathsf{vk}}, m)$ (i.e., it could not have been directly obtained as the result of the signing query), and that the aggregated signature $\mathsf{agg}\sigma^*$ verifies correctly for the message m^* with respect to the aggregate verification key corresponding to $\vec{\mathsf{vk}}^*$.

Definition 3.2. Let $t = t(\lambda)$ and $\epsilon = \epsilon(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A non-interactive multi-signature scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KAgg}, \mathsf{Sign}, \mathsf{SAgg}, \mathsf{Verify})$ is *one-time* (t, ϵ)-*unforgeable* if for any algorithm $A = (A_1, A_2)$ that runs in time at most t it holds that

$$\mathsf{Adv}^{\mathsf{1TimeMS}}_{\Pi,A}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{Exp}^{\mathsf{1TimeMS}}_{\Pi,A}(\lambda) = 1\right] \leq \epsilon(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\Pi,A}(\lambda)$ is defined as follows:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KG}(\mathsf{pp})$.
3. $\left(\vec{\mathsf{vk}}, m, \mathsf{st}\right) \leftarrow A_1(1^\lambda, \mathsf{pp}, \mathsf{vk})$.
4. $\sigma \leftarrow \mathsf{Sign}\left(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, m\right)$.
5. $\left(\vec{\mathsf{vk}}^*, m^*, \mathsf{agg}\sigma^*\right) \leftarrow A_2(\mathsf{st}, \sigma)$.
6. If the following conditions are satisfied then output 1 and otherwise output 0:
   (a) $\mathsf{vk} \in \vec{\mathsf{vk}}^*$.
   (b) $\left(\vec{\mathsf{vk}}^*, m^*\right) \neq \left(\vec{\mathsf{vk}}, m\right)$
   (c) $\mathsf{Verify}\left(\mathsf{pp}, \mathsf{KAgg}\left(\vec{\mathsf{vk}}^*\right), m^*, \mathsf{agg}\sigma^*\right) = 1$.

In some cases we omit the parameters t and ϵ corresponding to the running time and success probability of adversaries, respectively, and consider polynomial-time adversaries with negligible success probabilities. In addition, when considering schemes whose security is analyzed in the random-oracle model [BR93], we augment all algorithms (including the adversary) with access to the random oracle, introduce an additional parameter q_{H} that upper bounds the number of direct random-oracle queries issued by the adversary, and consider all probabilities also over the randomness of the oracle.

3.2 Single-Set Unforgeability

Our notion of single-set unforgeability for multi-signature schemes considers adversaries that obtain any polynomial number of signatures of their choice but are restricted to requesting all of these signatures with respect to a single set of signers. In this context, a single set of signers corresponds to a single vector of verification keys, which may be adversarially chosen based on the public parameters of the scheme and on the honestly-generated verification key that the adversary is attacking.

Looking ahead, our construction of a multi-signature scheme that provides single-set security consists of a two-round signing protocol: Each party sends one message to all other parties, receives one message from all other parties, and then produces their output. Following Bellare and Dai [BD21], for formalizing the security of schemes with an interactive signing protocol, adversaries are provided access to a stateful signing oracle that enables to initiate sessions and to continue previously-initiated ones. Specifically, given a multi-signature scheme Π with a two-round signing protocol $\mathsf{Sign} = (\mathsf{Sign}_1, \mathsf{Sign}_2)$, we denote by $\mathcal{O}_{\mathsf{Sign}}(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, \cdot)$ the corresponding stateful signing oracle that is provided to the adversary, where pp denotes the public parameters, sk denotes the signing key corresponding to the honestly-generated verification key vk given as input to the adversary, and $\vec{\mathsf{vk}}$ denotes the vector of verification keys corresponding to the single set of signers chosen by the adversary. For this oracle, an adversary may issue two types of queries:

- Session-initiation queries: On query a message m, the oracle first assigns a unique session identifier sid (e.g., in an incremental manner) and computes $(\mathsf{msg}_1, \mathsf{st}) \leftarrow \mathsf{Sign}_1\left(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, m\right)$. Then, it locally stores the pair $(\mathsf{sid}, \mathsf{st})$ and outputs $(\mathsf{sid}, \mathsf{msg}_1)$.
- Communication queries: On query a pair $(\mathsf{sid}, \mathsf{msg}_2)$, the oracle first retrieves the stored pair $(\mathsf{sid}, \mathsf{st})$, and then computes and outputs $\mathsf{out} \leftarrow \mathsf{Sign}_2(\mathsf{st}, \mathsf{msg}_2)$. If no stored pair exists for the session identifier sid then the oracle outputs $\perp$.

Definition 3.3. Let $t = t(\lambda), q_{\mathsf{sign}} = q_{\mathsf{sign}}(\lambda)$, and $\epsilon = \epsilon(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A non-interactive multi-signature scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KAgg}, \mathsf{Sign}, \mathsf{SAgg}, \mathsf{Verify})$ is *single-set* $(t, q_{\mathsf{sign}}, \epsilon)$*-unforgeable* if for any algorithm $A = (A_1, A_2)$ that runs in time at most t and issues at most q_{sign} signing queries, it holds that

$$\mathsf{Adv}^{\mathsf{SSetMS}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}(\lambda) = 1\right] \leq \epsilon(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}(\lambda)$ is defined as follows:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KG}(\mathsf{pp})$.
3. $\left(\vec{\mathsf{vk}}, \mathsf{st}\right) \leftarrow A_1(1^\lambda, \mathsf{pp}, \mathsf{vk})$.
4. $\left(\vec{\mathsf{vk}}^*, m^*, \mathsf{agg}\sigma^*\right) \leftarrow A_2^{\mathcal{O}_{\mathsf{Sign}}(\mathsf{pp},\mathsf{sk},\vec{\mathsf{vk}},\cdot)}(\mathsf{st})$.
5. If the following conditions are satisfied then output 1 and otherwise output 0:
   (a) $\mathsf{vk} \in \vec{\mathsf{vk}}^*$.
   (b) $\vec{\mathsf{vk}}^* \neq \vec{\mathsf{vk}}$ or A_2 did not query $\mathcal{O}_{\mathsf{Sign}}(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, \cdot)$ with m^*.
   (c) $\mathsf{Verify}\left(\mathsf{pp}, \mathsf{KAgg}\left(\vec{\mathsf{vk}}^*\right), m^*, \mathsf{agg}\sigma^*\right) = 1$.

As noted in Section 3.1, when considering schemes whose security is analyzed in the random-oracle model [BR93], we augment all algorithms (including the adversary) with access to the random oracle, introduce an additional parameter q_{H} that upper bounds the number of direct random-oracle queries issued by the adversary, and consider all probabilities also over the randomness of the oracle.

4 One-Time Multi-Signatures via Ring-Homomorphic One-Way Functions

In this section we describe our construction of a one-time multi-signature scheme and prove its security. Our construction is parameterized by the security parameter $\lambda \in \mathbb{N}$ and by an integer $n = n(\lambda)$ determining an upper bound on the size of supported signer sets. The construction relies on the following building blocks:

- A ring-homomorphic one-way function $\mathsf{F} = (\mathsf{F.Setup}, \mathsf{F.Eval})$. As formalized in Section 2.1 following Catalano et al. [CFG+15], recall that for any $\lambda \in \mathbb{N}$ and for any function index Ind produced by $\mathsf{F.Setup}(1^\lambda)$ it holds that $\mathsf{F.Eval}(\mathsf{Ind}, \cdot) : \mathcal{X}_{\mathsf{Ind}} \to \mathcal{Y}_{\mathsf{Ind}}$ is a homomorphism for cyclic groups $\mathcal{X}_{\mathsf{Ind}}$ and $\mathcal{Y}_{\mathsf{Ind}}$ that allows computing linear operations "in the exponent" over a ring $\mathbb{K}_{\mathsf{Ind}} = \mathbb{Z}_q$ for some prime $q = q(\mathsf{Ind}) \geq K(\lambda)$, where q is at most the order of the cyclic groups. For simplifying our notation, for any function index Ind we let $\mathsf{F}_{\mathsf{Ind}}(\cdot) = \mathsf{F.Eval}(\mathsf{Ind}, \cdot)$.
- Hash functions H_0 and H_1 that will be modeled, for the security analysis, as random oracles. For any function index Ind of the ring-homomorphic function F we assume that $\mathsf{H}_0 : \mathcal{M}_\lambda \times \mathcal{Y}_{\mathsf{Ind}} \times \mathcal{Y}_{\mathsf{Ind}} \to \mathbb{K}_{\mathsf{Ind}}$ and $\mathsf{H}_1 : (\mathcal{Y}_{\mathsf{Ind}} \times \mathcal{Y}_{\mathsf{Ind}})^n \to \mathbb{K}_{\mathsf{Ind}}^n$, where $\mathcal{M} = \mathcal{M}_\lambda$ is the supported message space of the constructed multi-signature scheme (we can choose, for example, $\mathcal{M}_\lambda = \{0,1\}^\lambda$). In terms of input lengths, this means that we assume sufficiently long input lengths for H_0 and H_1. In terms of output lengths, recall that $\mathbb{K}_{\mathsf{Ind}} = \mathbb{Z}_q$ for some prime q, where q is at most the order of the cyclic groups. In practice, this can be realized in a standard manner by producing sufficiently-long outputs using a cryptographic hash function, and then reducing the results modulo q to obtain a statistically-small error.

In what follows we first describe our one-time scheme, denoted $\mathsf{1T}$, and then prove its correctness and security. For simplicity and for avoiding the introduction of additional notation, when describing the scheme and proving its security we assume that all signer sets are of size $n = n(\lambda)$, but we note that an upper bound on the size of supported signer sets would suffice. In addition, we note that, in practice our scheme does not essentially require any setup. Specifically, when realizing the ring-homomorphic one-way function via group exponentiation in a prime-order group [CD98, CFG+15], the function index Ind consists of the description of the group, which is typically fixed (e.g., standard 256-bit curves such as Secp256k1 or Curve25519).

The scheme $\mathsf{1T} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KAgg}, \mathsf{Sign}, \mathsf{SAgg}, \mathsf{Verify})$

$\mathsf{Setup}(1^\lambda)$. On input 1^λ the setup algorithm samples $\mathsf{Ind} \leftarrow \mathsf{F.Setup}(1^\lambda)$ and outputs $\mathsf{pp} = \mathsf{Ind}$.

$\mathsf{KG}(\mathsf{pp})$. On input pp as above, the key-generation algorithm samples $x, r \leftarrow \mathcal{X}_{\mathsf{Ind}}$, computes $X = \mathsf{F}_{\mathsf{Ind}}(x)$ and $R = \mathsf{F}_{\mathsf{Ind}}(r)$, and then outputs $(\mathsf{sk}, \mathsf{vk}) = ((x, r), (X, R))$.

$\mathsf{KAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}\right)$. On input pp as above and $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, where $\mathsf{vk}_i = (X_i, R_i)$ for every $i \in [n]$, the key-aggregation algorithm computes

$$(a_1, \ldots, a_n) = \mathsf{H}_1\left(\vec{\mathsf{vk}}\right) \in \mathbb{K}_{\mathsf{Ind}}^n$$

$$\mathsf{agg}X = \prod_{i=1}^{n} X_i^{a_i} \in \mathcal{Y}_{\mathsf{Ind}}$$

$$\mathsf{agg}R = \prod_{i=1}^{n} R_i^{a_i} \in \mathcal{Y}_{\mathsf{Ind}},$$

and outputs $\mathsf{aggvk} = (\mathsf{agg}X, \mathsf{agg}R)$.

Sign $\left(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, m\right)$**.** On input pp as above, $\mathsf{sk} = (x, r)$, $\vec{\mathsf{vk}}$ and $m \in \mathcal{M}_\lambda$, the signing algorithm is defined as follows:

1. If $(\mathsf{F}_{\mathsf{Ind}}(x), \mathsf{F}_{\mathsf{Ind}}(r)) \notin \vec{\mathsf{vk}}$ then output $\perp$.
2. Otherwise, compute $\mathsf{aggvk} = \mathsf{KAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}\right)$, and output

$$\sigma = r + \mathsf{H}_0\left(m, \mathsf{aggvk}\right) \cdot x \in \mathcal{X}_{\mathsf{Ind}}.$$

SAgg $\left(\mathsf{pp}, \vec{\mathsf{vk}}, \vec{\sigma}\right)$**.** On input pp as above, $\vec{\mathsf{vk}}$ and $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the signature-aggregation algorithm computes $(a_1, \ldots, a_n) = \mathsf{H}_1\left(\vec{\mathsf{vk}}\right) \in \mathbb{K}_{\mathsf{Ind}}$ and outputs $\mathsf{agg}\sigma = \sum_{i=1}^{n} a_i \cdot \sigma_i \in \mathcal{X}_{\mathsf{Ind}}$.

Verify $(\mathsf{pp}, \mathsf{aggvk}, m, \mathsf{agg}\sigma)$**.** On input pp as above, $\mathsf{aggvk} = (\mathsf{agg}X, \mathsf{agg}R)$, m and $\mathsf{agg}\sigma$, if

$$\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma) = (\mathsf{agg}X)^{\mathsf{H}_0(m, \mathsf{aggvk})} \cdot \mathsf{agg}R$$

then the verification algorithm outputs 1 and otherwise it outputs 0.

Correctness. Fix any $\lambda \in \mathbb{N}$, any public parameters $\mathsf{pp} = \mathsf{Ind}$, and any vector of verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$. For every $i \in [n]$ let $\mathsf{vk}_i = (X_i, R_i)$, where $X_i = \mathsf{F}_{\mathsf{Ind}}(x_i)$ and $R_i = \mathsf{F}_{\mathsf{Ind}}(r_i)$. Then, for any message m it holds that

$$\begin{aligned}
\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma) &= \mathsf{F}_{\mathsf{Ind}}\left(\sum_{i=1}^{n} a_i \cdot r_i + \mathsf{H}_0\left(m, \mathsf{aggvk}\right) \cdot \sum_{i=1}^{n} a_i \cdot x_i\right) \\
&= \left(\prod_{i=1}^{n} \mathsf{F}_{\mathsf{Ind}}(x_i)^{a_i}\right)^{\mathsf{H}_0(m, \mathsf{aggvk})} \cdot \left(\prod_{i=1}^{n} \mathsf{F}_{\mathsf{Ind}}(r_i)^{a_i}\right) \\
&= (\mathsf{agg}X)^{\mathsf{H}_0(m, \mathsf{aggvk})} \cdot \mathsf{agg}R \ ,
\end{aligned}$$

where $(a_1, \ldots, a_n) = \mathsf{H}_1\left(\vec{\mathsf{vk}}\right)$, and thus the scheme provides perfect correctness.

Security. The following theorem establishes the security of our scheme based on that of the underling ring-homomorphic one-way function (recall Definition 2.1):

Theorem 4.1. *Let A be a probabilistic polynomial-time algorithm that issues $q_{\mathsf{H}_0} = q_{\mathsf{H}_0}(\lambda)$ and $q_{\mathsf{H}_1} = q_{\mathsf{H}_1}(\lambda)$ queries to the oracles H_0 and H_1, respectively. Then, there exists a probabilistic polynomial-time algorithm I such that for every $\lambda \in \mathbb{N}$ it holds that*

$$\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda) \leq (q_{\mathsf{H}_0})^2 \cdot q_{\mathsf{H}_1} \cdot n \cdot \left(\sqrt{\mathsf{Adv}^{\mathsf{hom\text{-}ow}}_{\mathsf{F},I}(\lambda) + \frac{(q_{\mathsf{H}_0})^2 + (q_{\mathsf{H}_1})^2}{K(\lambda)}} + \frac{1}{K(\lambda)}\right)$$

Proof of Theorem 4.1. Let A be a probabilistic polynomial-time algorithm that issues a single signing query, and issues q_{H_0} and q_{H_1} queries to the oracles H_0 and H_1, respectively. Without loss of generality, we assume that A does no query either H_0 and H_1 more than once with the same input. We denote by $\left(\vec{\mathsf{vk}}, m\right)$ the signing query issued by A, and by $\left(\vec{\mathsf{vk}}^*, m^*, \mathsf{agg}\sigma^*\right)$ the output provided by A. By increasing the number of A's queries to H_0 by at most 2, we assume that A always queries H_0 with (m, aggvk) and (m^*, aggvk^*), where $\mathsf{aggvk} = \mathsf{KAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}\right)$ and $\mathsf{aggvk}^* = \mathsf{KAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}^*\right)$. Similarly, by increasing the number of A's queries to H_1 by at most 2, we assume that A always queries H_1 with $\vec{\mathsf{vk}}$ and $\vec{\mathsf{vk}}^*$.

We construct a probabilistic polynomial-time inversion algorithm I using Lemma 2.2. Concretely, we define an algorithm B which simulates the experiment $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}$ to A and enforces some additional checks. Then, on input (Ind, X), the algorithm I invokes the forking algorithm F_B for B, guaranteed by Lemma 2.2, to obtain two correlated forgeries which can be used to compute $x' \in \mathcal{X}_{\mathsf{Ind}}$ and $0 \neq d \in \mathbb{K}_{\mathsf{Ind}}$ such that $\mathsf{F}_{\mathsf{Ind}}(x') = X^d$.

The algorithm $B\left(\mathsf{pp}, X, \vec{h}\right)$

On input $\mathsf{pp} = \mathsf{Ind}$, $X = \mathsf{F}_{\mathsf{Ind}}(x)$, and $\vec{h} = \left(h^{(1)}, \ldots, h^{(q_{\mathsf{H}_1})}\right)$, the algorithm B is defined as follows:

1. Sample internal randomness

$$\rho = \left(i_0^*, i_1^*, k^*, \sigma, \left\{c^{(i)}\right\}_{i\in[q_{\mathsf{H}_0}]}, \left\{\left(a_1^{(i)}, \ldots, a_n^{(i)}\right)\right\}_{i\in[q_{\mathsf{H}_1}]}\right),$$

   where:

   - $i_0^*, i_1^* \leftarrow [q_{\mathsf{H}_0}]$. Looking ahead, i_0^* and i_1^* are B's guesses for the indices of A's H_0-queries (m, aggvk) and (m^*, aggvk^*), respectively.
   - $k^* \leftarrow [n]$. Looking ahead, k^* is B's guess for the coordinate in which A will position vk within $\vec{\mathsf{vk}}^*$.
   - $\sigma \leftarrow \mathcal{X}_{\mathsf{pp}}$. Looking ahead, σ is B's response to A's signing query.
   - $c^{(i)} \leftarrow \mathbb{K}_{\mathsf{Ind}}$ for every $i \in [q_{\mathsf{H}_0}]$. Looking ahead, these values are B's responses to A's H_0-queries.
   - $\left(a_1^{(j)}, \ldots, a_n^{(j)}\right) \leftarrow \mathbb{K}_{\mathsf{Ind}}^n$ for every $j \in [q_{\mathsf{H}_1}]$. Looking ahead, these are B's responses to A's H_1-queries (except for the k^*-th coordinate $a_{k^*}^{(j)}$ that will be replaced).

2. Set $R = \mathsf{F}_{\mathsf{Ind}}(\sigma) \cdot X^{-c^{(i_0^*)}}$ (assuming that i_0^* is guessed correctly, this defines $c^{(i_0^*)} = \mathsf{H}_0\left(m, \mathsf{aggvk}\right)$), and for every $j \in [q_{\mathsf{H}_1}]$ set $a_{k^*}^{(j)} = h^{(j)}$.
3. Invoke A on input $(\mathsf{pp}, \mathsf{vk})$, where $\mathsf{vk} = (X, R)$, and respond to A's oracle queries as follows:
   - When A issues a signing query return σ.
   - For every $i \in [q_{\mathsf{H}_0}]$, when A issues their ith H_0-query return $c^{(i)}$.
   - For every $j \in [q_{\mathsf{H}_1}]$, when A issues their jth H_1-query return $\left(a_1^{(j)}, \ldots, a_n^{(j)}\right)$.
4. Given A's output $\left(\vec{\mathsf{vk}}^*, m^*, \mathsf{agg}\sigma^*\right)$, if all of the following conditions are satisfied then output $(j^*, \mathsf{agg}\sigma^*)$, where $j^* \in [q_{\mathsf{H}_1}]$ is the index of the H_1-query $\vec{\mathsf{vk}}^*$, and otherwise output $\perp$:
   (a) $\mathsf{Verify}\left(\mathsf{pp}, \vec{\mathsf{vk}}^*, m^*, \mathsf{agg}\sigma^*\right) = 1$, $\mathsf{vk} \in \vec{\mathsf{vk}}^*$, and $\left(\vec{\mathsf{vk}}^*, m^*\right) \neq \left(\vec{\mathsf{vk}}, m\right)$ (i.e., A won the simulated experiment).
   (b) A's i_0^*-th H_0-query was (m, aggvk) (i.e., B guessed i_0^* correctly).
   (c) A's i_1^*-th H_0-query was (m^*, aggvk^*) (i.e., B guessed i_1^* correctly).
   (d) $\vec{\mathsf{vk}}^* = (\mathsf{vk}_1^*, \ldots, \mathsf{vk}_n^*)$ where $\mathsf{vk}_{k^*}^* = \mathsf{vk}$ (i.e., B guessed k^* correctly).

Claim 4.2. *For every $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[B\left(\mathsf{pp}, \mathsf{F}_{\mathsf{Ind}}(x), \vec{h}\right) \neq \perp\right] \geq \frac{1}{(q_{\mathsf{H}_0})^2 \cdot n} \cdot \mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda),$$

where $\mathsf{pp} = \mathsf{Ind} \leftarrow \mathsf{F.Setup}(1^\lambda)$, $x \leftarrow \mathcal{X}_{\mathsf{Ind}}$ *and* $\vec{h} \leftarrow \mathbb{K}_{\mathsf{Ind}}^{q_{\mathsf{H}_1}}$.

Proof of Claim 4.2. Let win denote the event in which A wins the simulated experiment, and let hit denote the event in which the index i_0^* was guessed correctly by B. The choices of i_1^* and k^* are completely independent of A's execution, and therefore can be sampled by B at the end. Hence,

$$\begin{aligned}\Pr\left[B\left(\mathsf{pp},\mathsf{F}_{\mathsf{Ind}}(x),\vec{h}\right)\neq\perp\right] &= \frac{1}{q_{\mathsf{H}_0}\cdot n}\cdot\Pr\left[\mathsf{win}|\mathsf{hit}\right]\cdot\Pr\left[\mathsf{hit}\right] \qquad (4.1)\\ &= \frac{1}{(q_{\mathsf{H}_0})^2\cdot n}\cdot\Pr\left[\mathsf{win}|\mathsf{hit}\right]\\ &= \frac{1}{(q_{\mathsf{H}_0})^2\cdot n}\cdot\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda),\end{aligned}$$

where Eq. (4.1) also follows from the fact that B outputs $\perp$ with probability 1 conditioned on $\overline{\mathsf{hit}}$. Since conditioned on hit, B perfectly simulates the one-time unforgeability experiment to A, the claim follows. ■

Claim 4.2 directly enables us to lower bound the probability that $F_B(\mathsf{Ind},X)\neq\perp$ via the forking lemma (see Lemma 2.2):

Corollary 4.3. *For every $\lambda\in\mathbb{N}$ it holds that*

$$\Pr\left[F_B\left(\mathsf{Ind},\mathsf{F}_{\mathsf{Ind}}(x)\right)\neq\perp\right]\geq\frac{\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda)}{(q_{\mathsf{H}_0})^2\cdot n}\cdot\left(\frac{\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda)}{(q_{\mathsf{H}_0})^2\cdot q_{\mathsf{H}_1}\cdot n}-\frac{1}{K(\lambda)}\right),$$

where $\mathsf{Ind}\leftarrow\mathsf{F.Setup}(1^\lambda)$ *and* $x\leftarrow\mathcal{X}_{\mathsf{Ind}}$.

Having described the algorithms B and F_B, and analyzed their success probability, we can now describe the inversion algorithm I. On input (Ind,X), the algorithm I invokes the algorithm F_B provided by the forking lemma (see Section 2.2) on the input (pp,X), where $\mathsf{pp}=\mathsf{Ind}$. If the algorithm F_B returns $\perp$ then I fails (e.g., I outputs some fixed values $x'\in\mathcal{X}_\lambda$ and $d\in\mathbb{K}_{\mathsf{Ind}}$). Otherwise, if the algorithm F_B returns a triplet $(j^*,\mathsf{agg}\sigma^*,\mathsf{agg}\sigma^{**})$, then denote by ρ the random string sampled by F_B for running B, where

$$\rho=\left(i_0^*,i_1^*,k^*,\sigma,\left\{c^{(i)}\right\}_{i\in[q_{\mathsf{H}_0}]},\left\{\left(a_1^{(j)},\ldots,a_n^{(j)}\right)\right\}_{j\in[q_{\mathsf{H}_1}]}\right),$$

and denote by $h^{(1)},\ldots,h^{(q_{\mathsf{H}_1})}$ the sequence of oracle responses sampled by F_B for B's first execution, and by $\hat{h}^{(j^*)},\ldots,\hat{h}^{(q_{\mathsf{H}_1})}$ the additional oracle responses sampled by F_B for B's second execution. That is, using this notation, the algorithm F_B first invoked B on the input $\left(\mathsf{pp},X,h^{(1)},\ldots,h^{(q_{\mathsf{H}_1})}\right)$ to obtain an output $\left(\vec{\mathsf{vk}}^*,m^*,\mathsf{agg}\sigma^*\right)$ from A and an output $(j^*,\mathsf{agg}\sigma^*)$ from B, and then invoked B on the input $\left(\mathsf{pp},X,h^{(1)},\ldots,h^{(j^*-1)},\hat{h}^{(j^*)},\ldots,\hat{h}^{(q_{\mathsf{H}_1})}\right)$ to obtain an output $\left(\vec{\mathsf{vk}}^{**},m^{**},\mathsf{agg}\sigma^{**}\right)$ from A and an output $(j^*,\mathsf{agg}\sigma^{**})$ from B. The inversion algorithm I then computes and outputs

$$\begin{aligned}x' &= \mathsf{agg}\sigma^*-\mathsf{agg}\sigma^{**}-\left(h^{(j^*)}-\hat{h}^{(j^*)}\right)\cdot\sigma\in\mathcal{X}_{\mathsf{Ind}}\\ d &= \left(c^{(i_1^*)}-c^{(i_0^*)}\right)\cdot\left(h^{(j^*)}-\hat{h}^{(j^*)}\right)\in\mathbb{K}_{\mathsf{Ind}}.\end{aligned}$$

We now show that as long as the algorithm F_B does not return $\perp$, then $\mathsf{F}_{\mathsf{Ind}}(x')=X^d$ and thus the inversion algorithm I is successful. Note that both invocations of the algorithm B by the algorithm F_B use the random string ρ, and therefore B uses the same values i_0^*, i_1^*, k^*, σ, $\left\{c^{(i)}\right\}_{i\in[q_{\mathsf{H}_0}]}$ and $\left\{\left(a_1^{(j)},\ldots,a_n^{(j)}\right)\right\}_{j\in[q_{\mathsf{H}_1}]}$ for these two executions. In particular it holds that:

1. In both executions of B the vector of verification keys $\vec{\mathsf{vk}}^*$ relative to which A produces the forgery was provided as input to A's j^*-th H_1-query (this is the query to which A is rewinded by B). Therefore, since the executions are identical up to the response to the j^*-th H_1-query, and since in both executions the same index k^* is used for guessing the position of vk^* and vk^{**} in $\vec{\mathsf{vk}}^*$, it holds that $\mathsf{vk}^* = \mathsf{vk}^{**}$. We let
$$\mathsf{vk}^* = \mathsf{vk}^{**} = \left((X_1, R_1), \ldots, (X_n, R_n)\right).$$
2. In both executions of B the verification key $\mathsf{vk} = (X, R)$ was positioned by A in coordinate $k^* \in [n]$ of vk^* and vk^{**}. Therefore, $(X_{k^*}, R_{k^*}) = (X, R)$.
3. In both executions of B, the value returned as the response to A's i_1^*-th query to H_0 was $c^{(i_1^*)}$. Therefore, in the first execution $c^{(i_1^*)} = \mathsf{H}_0\left(m^*, \mathsf{aggvk}^*\right)$ and in the second execution $c^{(i_1^*)} = \mathsf{H}_0\left(m^{**}, \mathsf{aggvk}^{**}\right)$.

Thus, in B's first execution, we have that
$$\begin{aligned}\mathsf{H}_0\left(m^*, \mathsf{aggvk}^*\right) &= c^{(i_1^*)} \\ \mathsf{H}_1\left(\vec{\mathsf{vk}}^*\right) &= \left(a_1^{(j^*)}, \ldots, a_{k^*-1}^{(j^*)}, h^{(j^*)}, a_{k^*+1}^{(j^*)}, a_n^{(j^*)}\right),\end{aligned}$$
and given that A produced a valid forgery it holds that
$$\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma^*) = \left(X_{k^*}^{h^{(j^*)}} \cdot \prod_{i \in [n]\setminus\{k^*\}} X_i^{a_i^{(j^*)}}\right)^{c^{(i_1^*)}} \cdot \left(R_{k^*}^{h^{(j^*)}} \cdot \prod_{i \in [n]\setminus\{k^*\}} R_i^{a_i^{(j^*)}}\right). \tag{4.2}$$
Similarly, in B's second execution, we have that
$$\begin{aligned}\mathsf{H}_0\left(m^{**}, \mathsf{aggvk}^{**}\right) &= c^{(i_1^*)} \\ \mathsf{H}_1\left(\vec{\mathsf{vk}}^{**}\right) &= \left(a_1^{(j^*)}, \ldots, a_{k^*-1}^{(j^*)}, \hat{h}^{(j^*)}, a_{k^*+1}^{(j^*)}, a_n^{(j^*)}\right),\end{aligned}$$
and given that A produced a valid forgery it holds that
$$\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma^{**}) = \left(X_{k^*}^{\hat{h}^{(j^*)}} \cdot \prod_{i \in [n]\setminus\{k^*\}} X_i^{a_i^{(j^*)}}\right)^{c^{(i_1^*)}} \cdot \left(R_{k^*}^{\hat{h}^{(j^*)}} \cdot \prod_{i \in [n]\setminus\{k^*\}} R_i^{a_i^{(j^*)}}\right). \tag{4.3}$$
Therefore, using the facts that $X_{k^*} = X$ and $R_{k^*} = R = \mathsf{F}_{\mathsf{Ind}}(\sigma) \cdot X^{-c_0^{(i_0^*)}}$, Equations (4.2) and (4.3) imply
$$\begin{aligned}\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma^* - \mathsf{agg}\sigma^{**}) &= X_{k^*}^{c^{(i_1^*)} \cdot \left(h^{(j^*)} - \hat{h}^{(j^*)}\right)} \cdot R_{k^*}^{h^{(j^*)} - \hat{h}^{(j^*)}} \\ &= X^{c^{(i_1^*)} \cdot \left(h^{(j^*)} - \hat{h}^{(j^*)}\right)} \cdot \left(\mathsf{F}_{\mathsf{Ind}}(\sigma) \cdot X^{-c^{(i_0^*)}}\right)^{h^{(j^*)} - \hat{h}^{(j^*)}} \\ &= X^{\left(c^{(i_1^*)} - c^{(i_0^*)}\right) \cdot \left(h^{(j^*)} - \hat{h}^{(j^*)}\right)} \cdot \mathsf{F}_{\mathsf{Ind}}\left(\left(h^{(j^*)} - \hat{h}^{(j^*)}\right) \cdot \sigma\right)\end{aligned}$$
which, for the above setting of x' and d, implies that
$$\begin{aligned}\mathsf{F}_{\mathsf{Ind}}(x') &= \mathsf{F}_{\mathsf{Ind}}\left(\mathsf{agg}\sigma^* - \mathsf{agg}\sigma^{**} - \left(h^{(j^*)} - \hat{h}^{(j^*)}\right) \cdot \sigma\right) \\ &= X^{\left(c^{(i_1^*)} - c^{(i_0^*)}\right) \cdot \left(h^{(j^*)} - \hat{h}^{(j^*)}\right)} \\ &= X^d.\end{aligned}$$

The following claim concludes the proof by bounding the probability that $d = 0_{\mathbb{K}_{\mathsf{Ind}}}$. This shows, in particular, that the scheme's key-aggregation procedure is collision resistant as a function of the vector of verification keys that it aggregates.

Claim 4.4. *For every $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[\begin{matrix} I(\mathsf{Ind}, X) = (x', d) \\ s.t.\ d = 0_{\mathbb{K}_{\mathsf{Ind}}} \end{matrix} \wedge F_B(\mathsf{pp}, X) \neq \perp\right] \leq \frac{(q_{\mathsf{H}_0})^2 + (q_{\mathsf{H}_1})^2}{K(\lambda)},$$

where $\mathsf{pp} = \mathsf{Ind} \leftarrow \mathsf{F.Setup}(1^\lambda)$, $x \leftarrow \mathcal{X}_{\mathsf{Ind}}$, $X = \mathsf{F}_{\mathsf{Ind}}(x)$ and the probability is additionally taken over the internal randomness of the algorithms I and B.

Proof of Claim 4.4. Conditioned on the event $F_B(\mathsf{pp}, X) \neq \perp$, the algorithm I outputs $d = \left(c^{(i_1^*)} - c^{(i_0^*)}\right) \cdot \left(h^{(j^*)} - \hat{h}^{(j^*)}\right) \in \mathbb{K}_{\mathsf{Ind}}$, where $c^{(i_1^*)}, c^{(i_0^*)}, h^{(j^*)}, \hat{h}^{(j^*)} \in \mathbb{K}_{\mathsf{Ind}}$ and $\mathbb{K}_{\mathsf{Ind}} = \mathbb{Z}_q$ for some prime $q = q(\mathsf{Ind}) \geq K(\lambda)$. Recall, in addition, that according to the forking lemma, if $h^{(j^*)} = \hat{h}^{(j^*)}$ then F_B outputs $\perp$. Therefore, conditioned on the event $F_B(\mathsf{pp}, X) \neq \perp$, it holds that $h^{(j^*)} \neq \hat{h}^{(j^*)}$, and we are left to consider the event $c^{(i_1^*)} = c^{(i_0^*)}$. Note that these two values are defined already for the first invocation of B by F_B, and conditioned on $B(\mathsf{pp}, X, \vec{h}) \neq \perp$ (which follows from $F_B(\mathsf{pp}, X) \neq \perp$) it holds that

$$\begin{aligned} c^{(i_0^*)} &= \mathsf{H}_0\left(m, \mathsf{aggvk}\right) \\ c^{(i_1^*)} &= \mathsf{H}_0\left(m^*, \mathsf{aggvk}^*\right). \end{aligned}$$

Now, there are two disjoint events to consider, denoted $\mathcal{E}$ and $\bar{\mathcal{E}}$:

Event $\mathcal{E}$: $m \neq m^*$ or $\mathsf{aggvk} \neq \mathsf{aggvk}^*$. Here $(m, \mathsf{aggvk}) \neq (m^*, \mathsf{aggvk}^*)$, and therefore $c^{(i_0^*)} = c^{(i_1^*)}$ implies that the algorithm A found a non-trivial collision for the random oracle H_0 during B's first invocation. Such a collision may be found with probability at most $(q_{\mathsf{H}_0})^2/|\mathbb{K}_{\mathsf{Ind}}| \leq (q_{\mathsf{H}_0})^2/K(\lambda)$, and therefore

$$\Pr\left[\begin{matrix} I(\mathsf{Ind}, X) = (x', d) \\ \text{s.t. } d = 0_{\mathbb{K}_{\mathsf{Ind}}} \end{matrix} \wedge F_B(\mathsf{pp}, X) \neq \perp \wedge \mathcal{E}\right] \leq \frac{(q_{\mathsf{H}_0})^2}{K(\lambda)}. \tag{4.4}$$

Event $\bar{\mathcal{E}}$: $m = m^*$ and $\mathsf{aggvk} = \mathsf{aggvk}^*$. Note that by the definition of B, if $F_B(\mathsf{pp}, X) \neq \perp$ then $\left(m, \vec{\mathsf{vk}}\right) \neq \left(m^*, \vec{\mathsf{vk}^*}\right)$ and therefore the event $\bar{\mathcal{E}}$ implies that the algorithm A produced $\vec{\mathsf{vk}}$ and $\vec{\mathsf{vk}^*}$ such that $\vec{\mathsf{vk}} \neq \vec{\mathsf{vk}^*}$ and $\mathsf{aggvk} = \mathsf{aggvk}^*$. Among the q_{H_1} queries issued by A to the random oracle H_1, there are at most $(q_{\mathsf{H}_0})^2$ potential choices for such $\vec{\mathsf{vk}} \neq \vec{\mathsf{vk}^*}$, and we now show that for each such choice it holds that $\mathsf{aggvk} = \mathsf{aggvk}^*$ with probability at most $1/K(\lambda)$.

Specifically, let

$$\vec{\mathsf{vk}} = ((X_1, R_1), \ldots, (X_n, R_n))$$

and

$$\vec{\mathsf{vk}^*} = ((X_1^*, R_1^*), \ldots, (X_n^*, R_n^*)),$$

and assume without loss of generality that A issues the H_1-query $\vec{\mathsf{vk}}$ before the H_1-query $\vec{\mathsf{vk}^*}$. Then, if $(X_1, \ldots, X_n) \neq (X_1^*, \ldots, X_n^*)$, then by the time A issues that H_1 query $\vec{\mathsf{vk}^*}$ the value $\mathsf{agg}X = \prod_{i=1}^n X_i^{a_i}$ is already fixed, where $(a_1, \ldots, a_n) = \mathsf{H}_1\left(\vec{\mathsf{vk}}\right)$. Therefore, the probability over the choice of $(a_1^*, \ldots, a_n^*) = \mathsf{H}_1\left(\vec{\mathsf{vk}^*}\right)$ that $\mathsf{agg}X^* = \prod_{i=1}^n (X_i^*)^{a_i^*} = \mathsf{agg}X$ is at most

$1/|\mathbb{K}_{\mathsf{Ind}}| \leq 1/K(\lambda)$. An identical argument holds if $(R_1, \ldots, R_n) \neq (R_1^*, \ldots, R_n^*)$, and overall we obtain

$$\Pr\left[\begin{matrix} I(\mathsf{Ind}, X) = (x', d) \\ \text{s.t. } d = 0_{\mathbb{K}_{\mathsf{Ind}}} \end{matrix} \wedge F_B(\mathsf{pp}, X) \neq \bot \wedge \bar{\mathcal{E}}\right] \leq \frac{(q_{\mathsf{H}_1})^2}{K(\lambda)}. \tag{4.5}$$

The claim now follows by combining Equations (4.4) and (4.5). ∎

Equipped with Corollary 4.3 and Claim 4.4, we have

$$\begin{aligned}
&\Pr\left[\begin{matrix} I(\mathsf{Ind}, X) = (x', d) \text{ s.t.} \\ \mathsf{F}_{\mathsf{Ind}}(x') \neq X^d \text{ or } d = 0_{\mathbb{K}_{\mathsf{Ind}}} \end{matrix}\right] \\
&\qquad \leq \Pr\left[F_B(\mathsf{pp}, X) = \bot\right] \\
&\qquad\quad + \Pr\left[\begin{matrix} I(\mathsf{Ind}, X) = (x', d) \\ \text{s.t. } d = 0_{\mathbb{K}_{\mathsf{Ind}}} \end{matrix} \wedge F_B(\mathsf{pp}, X) \neq \bot\right] \\
&\qquad \leq 1 - \frac{\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda)}{(q_{\mathsf{H}_0})^2 \cdot n} \cdot \left(\frac{\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda)}{(q_{\mathsf{H}_0})^2 \cdot q_{\mathsf{H}_1} \cdot n} - \frac{1}{K(\lambda)}\right) \\
&\qquad\quad + \frac{(q_{\mathsf{H}_0})^2 + (q_{\mathsf{H}_1})^2}{K(\lambda)}.
\end{aligned}$$

Thus, for any $\lambda \in \mathbb{N}$ it holds that

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{hom\text{-}ow}}_{\mathsf{F},I}(\lambda) &= \Pr\left[\begin{matrix} I(\mathsf{Ind}, X) = (x', d) \text{ s.t.} \\ \mathsf{F}_{\mathsf{Ind}}(x') = X^d \text{ and } d \neq 0_{\mathbb{K}_{\mathsf{Ind}}} \end{matrix}\right] \\
&\geq \left(\frac{\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},A}(\lambda)}{(q_{\mathsf{H}_0})^2 \cdot q_{\mathsf{H}_1} \cdot n} - \frac{1}{|K(\lambda)|}\right)^2 - \frac{(q_{\mathsf{H}_0})^2 + (q_{\mathsf{H}_1})^2}{K(\lambda)},
\end{aligned}$$

and therefore

$$\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T,A}}(\lambda) \leq (q_{\mathsf{H}_0})^2 \cdot q_{\mathsf{H}_1} \cdot n \cdot \left(\sqrt{\mathsf{Adv}^{\mathsf{hom\text{-}ow}}_{\mathsf{F},I}(\lambda) + \frac{(q_{\mathsf{H}_0})^2 + (q_{\mathsf{H}_1})^2}{K(\lambda)}} + \frac{1}{K(\lambda)}\right).$$

This settles the proof of Theorem 4.1. ∎

5 From One-Time to Single-Set Multi-Signatures

In this section we show that any one-time multi-signature scheme with non-interactive signing can be transformed into a multi-signature scheme that satisfies our notion of single-set unforgeability with a two-round signing protocol. Our construction relies on the following building blocks:

- A multi-signature scheme $\mathsf{1T} = (\mathsf{1T.Setup}, \mathsf{1T.KG}, \mathsf{1T.KAgg}, \mathsf{1T.Sign}, \mathsf{1T.SAgg}, \mathsf{1T.Verify})$ that is one-time unforgeable with non-interactive signing over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$. For simplicity we assume that $\mathcal{M}_\lambda = \{0,1\}^\ell$ for some polynomial $\ell = \ell(\lambda)$, and that ℓ is sufficiently large for enabling the scheme to sign, for example, the concatenation of two verification keys produced by its key-generation algorithm (otherwise, we can additionally rely on a collision-resistant hash function in the standard "hash-then-sign" manner). In addition, since the scheme has non-interactive signing, we assume without loss of generality that its signing algorithm is deterministic.

- A pseudorandom function $\mathsf{PRF} = (\mathsf{PRF.KG}, \mathsf{PRF.Eval})$, where for each $\lambda \in \mathbb{N}$ and for each key k produced by $\mathsf{PRF.KG}(1^\lambda)$ it holds that $\mathsf{PRF.Eval}(\mathsf{k}, \cdot) : \{0,1\}^{\leq \ell(\lambda)} \to \{0,1\}^{\ell'(\lambda)}$, and $\ell'(\lambda)$ is the length of the internal random string sampled by the key-generation algorithm $\mathsf{1T.KG}$.

- A collision-resistant hash function $\mathsf{CRH} = (\mathsf{CRH.KG}, \mathsf{CRH.Eval})$. As noted below, our usage of a collision-resistant hash function is in fact not required, and is done for providing a more direct proof of security.

- A non-interactive statistically-binding commitment scheme $\mathsf{COM} = (\mathsf{COM.Setup}, \mathsf{COM.Commit}, \mathsf{COM.Verify})$.

- A non-interactive simulation-sound zero-knowledge proof system $\mathsf{ZK} = (\mathsf{ZK.Setup}, \mathsf{ZK.P}, \mathsf{ZK.V}, \mathsf{ZK.Sim}_1, \mathsf{ZK.Sim}_2)$ for the language $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda \in \mathbb{N}}$, where

$$\mathcal{L}_\lambda = \left\{ \begin{pmatrix} \mathsf{pp_{1T}}, \mathsf{crs_{com}}, \\ \mathsf{com}, x, \mathsf{vk} \end{pmatrix} : \begin{array}{c} \exists\, (\mathsf{decom}, \mathsf{k_{PRF}}, \mathsf{sk}) \text{ s.t.} \\ \mathsf{COM.Verify}\left(\mathsf{crs_{com}}, \mathsf{com}, \mathsf{decom}, \mathsf{k_{PRF}}\right) = 1 \text{ and} \\ (\mathsf{sk}, \mathsf{vk}) = \mathsf{1T.KG}(\mathsf{pp_{1T}}; \mathsf{PRF.Eval}(\mathsf{k_{PRF}}, x)) \end{array} \right\}.$$

  Note that even if the security of the scheme $\mathsf{1T}$ is proved in the random-oracle model, then as long as the key-generation algorithm $\mathsf{1T.KG}$ does not access the random oracle (as with our scheme in Section 4), then $\mathcal{L}$ is an NP-language (assuming, of course, that COM is a standard-model commitment scheme, and that PRF is a standard-model pseudorandom function [Nao91, Gol01]). In this case, the single-set security of our construction is proved in the random-oracle model.

For presenting our scheme, we denote by ε the empty string, we denote by $m|_i$ the leftmost i bits of a binary string m (where $m|_0 = \varepsilon$), and we denote by $m|_i b$ the binary string obtained by concatenating the strings $m|_i$ and b.

The scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{KAgg}, \mathsf{Sign}, \mathsf{SAgg}, \mathsf{Verify})$

$\mathsf{Setup}(1^\lambda)$. On input 1^λ the setup algorithm computes

$$\begin{aligned} \mathsf{pp_{1T}} &\leftarrow \mathsf{1T.Setup}(1^\lambda) \\ \mathsf{crs_{ZK}} &\leftarrow \mathsf{ZK.Setup}(1^\lambda) \\ \mathsf{crs_{COM}} &\leftarrow \mathsf{COM.Setup}(1^\lambda) \\ \mathsf{k_{CRH}} &\leftarrow \mathsf{CRH.KG}(1^\lambda), \end{aligned}$$

and returns $\mathsf{pp} = (\mathsf{pp_{1T}}, \mathsf{crs_{ZK}}, \mathsf{crs_{COM}}, \mathsf{k_{CRH}})$.

$\mathsf{KG}(\mathsf{pp})$. On input pp as above, the key-generation algorithm computes

$$\begin{aligned} (\mathsf{sk}_\varepsilon, \mathsf{vk}_\varepsilon) &\leftarrow \mathsf{1T.KG}(\mathsf{pp_{1T}}) \\ \mathsf{k_{PRF}} &\leftarrow \mathsf{PRF.KG}(1^\lambda) \\ (\mathsf{com}, \mathsf{decom}) &\leftarrow \mathsf{COM.Commit}\left(\mathsf{crs_{COM}}, \mathsf{k_{PRF}}\right), \end{aligned}$$

and returns $\mathsf{sk} = (\mathsf{sk}_\varepsilon, \mathsf{vk}_\varepsilon, \mathsf{k_{PRF}}, \mathsf{com}, \mathsf{decom})$ and $\mathsf{vk} = (\mathsf{vk}_\varepsilon, \mathsf{com})$.

$\mathsf{KAgg}\left(\mathsf{pp}, \vec{\mathsf{vk}}\right)$. On input pp as above and $\vec{\mathsf{vk}} = \left(\left(\mathsf{vk}_\varepsilon^{(1)}, \mathsf{com}^{(1)}\right), \ldots, \left(\mathsf{vk}_\varepsilon^{(n)}, \mathsf{com}^{(n)}\right)\right)$ the key-aggregation

algorithm computes

$$\begin{aligned}\mathsf{aggvk}_{\varepsilon} &= \mathsf{1T.KAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \left(\mathsf{vk}_{\varepsilon}^{(1)}, \ldots, \mathsf{vk}_{\varepsilon}^{(n)}\right)\right)\\ \mathsf{tag}_{\vec{\mathsf{vk}}} &= \mathsf{CRH.Eval}\left(\mathsf{k}_{\mathsf{CRH}}, \vec{\mathsf{vk}}\right),\end{aligned}$$

and returns $\mathsf{aggvk} = \left(\mathsf{aggvk}_{\varepsilon}, \mathsf{tag}_{\vec{\mathsf{vk}}}\right)$.

Sign $\left(\mathsf{pp}, \mathsf{sk}, \vec{\mathsf{vk}}, m\right)$**.** On input pp as above, $\mathsf{sk} = (\mathsf{sk}_{\varepsilon}, \mathsf{vk}_{\varepsilon}, \mathsf{k}, \mathsf{com}, \mathsf{decom})$, $\vec{\mathsf{vk}}$ and $m \in \{0,1\}^{\ell}$, the signing algorithm proceeds as follows:

1. Let $\vec{\mathsf{vk}} = \left(\left(\mathsf{vk}_{\varepsilon}^{(1)}, \mathsf{com}^{(1)}\right), \ldots, \left(\mathsf{vk}_{\varepsilon}^{(n)}, \mathsf{com}^{(n)}\right)\right)$. If there is no index $i \in [n]$ for which $(\mathsf{vk}_{\varepsilon}, \mathsf{com}) = \left(\mathsf{vk}_{\varepsilon}^{(i)}, \mathsf{com}^{(i)}\right)$ or there is more than one such index, then abort. Otherwise, denote by $k^* \in [n]$ the unique such index.
2. For any $j \in \{0, \ldots, \ell-1\}$ and $b \in \{0,1\}$ compute

$$\begin{aligned}\left(\mathsf{sk}_{m|_j b}^{(k^*)}, \mathsf{vk}_{m|_j b}^{(k^*)}\right) &= \mathsf{1T.KG}(\mathsf{pp}_{\mathsf{1T}}; \mathsf{PRF.Eval}(\mathsf{k}, m|_j b))\\ \pi_{m|_j b}^{(k^*)} &= \mathsf{ZK.P}\left(\mathsf{crs}_{\mathsf{ZK}}, \left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}, m|_j b, \mathsf{vk}_{m|_j b}^{(k^*)}\right), \left(\mathsf{decom}, \mathsf{k}, \mathsf{sk}_{m|_j b}^{(k^*)}\right)\right)\end{aligned}$$

   and send $\left\{\left(\mathsf{vk}_{m|_j b}^{(k^*)}, \pi_{m|_j b}^{(k^*)}\right)\right\}_{j \in \{0,\ldots,\ell-1\}, b \in \{0,1\}}$ to all other parties.
3. Upon receiving $\left\{\left(\mathsf{vk}_{m|_j b}^{(i)}, \pi_{m|_j b}^{(i)}\right)\right\}_{i \in [n] \setminus \{k^*\}, j \in \{0,\ldots,\ell-1\}, b \in \{0,1\}}$ from all other parties, if there exists $(i, j, b) \in ([n] \setminus \{k^*\}) \times \{0, \ldots, \ell-1\} \times \{0,1\}$ for which $\mathsf{vk}_{m|_j b}^{(i)} = \mathsf{vk}_{m|_j b}^{(k^*)}$ or

$$\mathsf{ZK.V}\left(\mathsf{crs}_{\mathsf{ZK}}, \left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, m|_j b, \mathsf{vk}_{m|_j b}^{(i)}\right), \pi_{m|_j b}^{(i)}\right) = 0$$

   then abort. Otherwise, compute

$$\begin{aligned}\vec{\mathsf{vk}}_{m|_j b} &= \left(\mathsf{vk}_{m|_j b}^{(1)}, \ldots, \mathsf{vk}_{m|_j b}^{(n)}\right) \text{ for all } j \in \{0, \ldots, \ell-1\} \text{ and } b \in \{0,1\}\\ \mathsf{aggvk}_{m|_j b} &= \mathsf{1T.KAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \vec{\mathsf{vk}}_{m|_j b}\right) \text{ for all } j \in \{0, \ldots, \ell-1\} \text{ and } b \in \{0,1\}\\ \sigma_{m|_j}^{(k^*)} &= \mathsf{1T.Sign}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_{m|_j}^{(k^*)}, \vec{\mathsf{vk}}_{m|_j}, \left(\mathsf{aggvk}_{m|_j 0}, \mathsf{aggvk}_{m|_j 1}\right)\right) \text{ for all } j \in \{0, \ldots, \ell-1\}\\ \mathsf{tag}_{\vec{\mathsf{vk}}} &= \mathsf{CRH.Eval}\left(\mathsf{k}_{\mathsf{CRH}}, \vec{\mathsf{vk}}\right)\\ \sigma_m^{(k^*)} &= \mathsf{1T.Sign}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_m^{(k^*)}, \vec{\mathsf{vk}}_m, (m, \mathsf{tag}_{\vec{\mathsf{vk}}})\right)\end{aligned}$$

   and output

$$\sigma^{(k^*)} = \left(\left\{\left(\sigma_{m|_j}^{(k^*)}, \mathsf{vk}_{m|_j 0}^{(k^*)}, \mathsf{vk}_{m|_j 1}^{(k^*)}\right)\right\}_{j \in \{0,\ldots,\ell-1\}}, \sigma_m^{(k^*)}\right).$$

SAgg $\left(\mathsf{pp}, \vec{\mathsf{vk}}, \vec{\sigma}\right)$**.** On input pp as above, $\vec{\mathsf{vk}}$ and $\vec{\sigma} = \left(\sigma^{(1)}, \ldots, \sigma^{(n)}\right)$ where

$$\sigma^{(i)} = \left(\left\{\left(\sigma_{m|_j}^{(i)}, \mathsf{vk}_{m|_j 0}^{(i)}, \mathsf{vk}_{m|_j 1}^{(i)}\right)\right\}_{j \in \{0,\ldots,\ell-1\}}, \sigma_m^{(i)}\right)$$

for every $i \in [n]$, the signature-aggregation algorithm computes

$$\begin{aligned}\mathsf{aggvk}_{m|_j b} &= \mathsf{1T.KAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \left(\mathsf{vk}^{(1)}_{m|_j b}, \ldots, \mathsf{vk}^{(n)}_{m|_j b}\right)\right) \text{ for all } j \in \{0,\ldots,\ell-1\} \text{ and } b \in \{0,1\}\\ \mathsf{agg}\sigma_{m|j} &= \mathsf{1T.SAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \vec{\mathsf{vk}}_{m|j}, \left(\sigma^{(1)}_{m|j}, \ldots, \sigma^{(n)}_{m|j}\right)\right) \text{ for all } j \in \{0,\ldots,\ell\},\end{aligned}$$

and outputs

$$\mathsf{agg}\sigma = \left(\left\{\left(\mathsf{agg}\sigma_{m|_j}, \mathsf{aggvk}_{m|_j 0}, \mathsf{aggvk}_{m|_j 1}\right)\right\}_{j\in\{0,\ldots,\ell-1\}}, \mathsf{agg}\sigma_m\right).$$

Verify$(\mathsf{pp}, \mathsf{aggvk}, m, \mathsf{agg}\sigma)$. On input pp as above, $\mathsf{aggvk} = \left(\mathsf{aggvk}_\varepsilon, \mathsf{tag}_{\vec{\mathsf{vk}}}\right)$, $m \in \{0,1\}^\ell$ and $\mathsf{agg}\sigma$, where

$$\mathsf{agg}\sigma = \left(\left\{\left(\mathsf{agg}\sigma_{m|_j}, \mathsf{aggvk}_{m|_j 0}, \mathsf{aggvk}_{m|_j 1}\right)\right\}_{j\in\{0,\ldots,\ell-1\}}, \mathsf{agg}\sigma_m\right),$$

the verification algorithm outputs 1 if and only if the following two requirements are satisfied:

1. $\mathsf{1T.Verify}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{aggvk}_{m|_j}, \left(\mathsf{aggvk}_{m|_j 0}, \mathsf{aggvk}_{m|_j 1}\right), \mathsf{agg}\sigma_{m|j}\right) = 1$ for every $j \in \{0,\ldots,\ell-1\}$.
2. $\mathsf{1T.Verify}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{aggvk}_m, (m, \mathsf{tag}_{\vec{\mathsf{vk}}}), \mathsf{agg}\sigma_m\right) = 1$.

The following theorem captures the security of the scheme Π based on that on its underlying building blocks. Here, we note that our usage of a collision-resistant hash function for computing the tags $\mathsf{tag}_{\vec{\mathsf{vk}}} = \mathsf{CRH.Eval}\left(\mathsf{k}_{\mathsf{CRH}}, \vec{\mathsf{vk}}\right)$ is in fact not required, and is done for providing a more direct proof of security. Specifically, we could have instead used $\mathsf{tag}_{\vec{\mathsf{vk}}} = \mathsf{aggvk}_\varepsilon$, and rely on the fact that the key aggregation of any one-time multi-signature scheme is collision resistant (as effectively shown by the proof of Claim 4.4 for our one-time scheme).

Theorem 5.1. *Let $\epsilon_{\mathsf{binding}} = \epsilon_{\mathsf{binding}}(\lambda)$, $q_{\mathsf{Sign}} = q_{\mathsf{Sign}}(\lambda)$ and $\ell = \ell(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$, and let A be a probabilistic polynomial-time algorithm that issues q_{Sign} signing queries for ℓ-bit messages. Then, assuming that COM is $\epsilon_{\mathsf{binding}}$-statistically binding, there exist probabilistic polynomial-time algorithms $B_1, \ldots, B_6$ such that for every $\lambda \in \mathbb{N}$ it holds that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SSetMS}}_{\Pi,A}(\lambda) \leq\ & \epsilon_{\mathsf{binding}}(\lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{ZK},B_1}(\lambda) + \mathsf{Adv}^{\mathsf{hiding}}_{\mathsf{COM},B_2}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},B_3}(\lambda)\\ &+ \mathsf{Adv}^{\mathsf{ss}}_{\mathsf{ZK},B_4}(\lambda) + \mathsf{Adv}^{\mathsf{crh}}_{\mathsf{CRH},B_5}(\lambda) + (2\cdot\ell\cdot q_{\mathsf{Sign}} + 1)\cdot \mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda).\end{aligned}$$

Proof of Theorem 5.1. Let A be a probabilistic polynomial-time adversary that participates in the experiment $\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}(\lambda)$ by issuing $q_{\mathsf{Sign}} = q_{\mathsf{Sign}}(\lambda)$ signing queries. Note that the signing oracle, responding to these queries, may invoke the key-generation algorithm $\mathsf{1T.KAgg}$ at most $2\cdot\ell\cdot q_{\mathsf{Sign}}$ times during the experiment $\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}$ (recall that ℓ denotes bit-length of messages). In addition, at the beginning of the experiment $\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}$, the key-generation algorithm $\mathsf{1T.KAgg}$ is invoked once by Π's key-generation algorithm KG. Thus, throughout the experiment $\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}$, the key-generation algorithm $\mathsf{1T.KAgg}$ is invoked at most $L = 2\cdot\ell\cdot q_{\mathsf{Sign}} + 1$ times.

In what follows we describe a sequence of hybrid experiments $\mathsf{Exp}_0, \ldots, \mathsf{Exp}_5$, starting with the experiment $\mathsf{Exp}_0 = \mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}$. Based on the security of the building blocks used in the construction of the scheme Π, we bound the differences between the adversary's success probabilities in each consecutive pair of hybrid experiments, and then show that any adversary participating in experiment Exp_5 may be used to attack the underlying scheme $\mathsf{1T}$.

Experiment $\mathbf{Exp_0}$. This is the original experiment $\mathsf{Exp}^{\mathsf{SSetMS}}_{\Pi,A}$. Recall that at the beginning of the experiment, the setup algorithm Setup and the key-generation algorithm KG of the scheme Π

are invoked to produce $\mathsf{pp} = (\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{ZK}}, \mathsf{crs}_{\mathsf{COM}}, \mathsf{k}_{\mathsf{CRH}})$ and $\mathsf{vk} = (\mathsf{vk}_\varepsilon, \mathsf{com})$ which are provided A, whereas the signing key $\mathsf{sk} = (\mathsf{sk}_\varepsilon, \mathsf{vk}_\varepsilon, \mathsf{k}_{\mathsf{PRF}}, \mathsf{com}, \mathsf{decom})$ is not provided to A.

Experiment $\mathbf{Exp_1}$. This experiment is obtained from Exp_0 via the following modifications. First, instead of producing the common-reference string $\mathsf{crs}_{\mathsf{ZK}}$ that is included in the public parameters using the setup algorithm $\mathsf{crs}_{\mathsf{ZK}} \leftarrow \mathsf{ZK.Setup}(1^\lambda)$, it is produced by the simulated setup algorithm $(\mathsf{crs}_{\mathsf{ZK}}, \tau_{\mathsf{ZK}}) \leftarrow \mathsf{Sim}_1(1^\lambda)$. Second, when the experiment produces one-time signing and verification keys as

$$\left(\mathsf{sk}_{m|_j b}^{(k^*)}, \mathsf{vk}_{m|_j b}^{(k^*)}\right) = \mathsf{1T.KG}(\mathsf{pp}_{\mathsf{1T}}; \mathsf{PRF.Eval}(\mathsf{k}_{\mathsf{PRF}}, m|_j b))$$

for some $k^* \in [n]$ and for all $j \in \{0, \ldots, j-1\}$ and $b \in \{0,1\}$, then instead of computing

$$\pi_{m|_j b}^{(k^*)} = \mathsf{ZK.P}\left(\mathsf{crs}_{\mathsf{ZK}}, \left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}, m|_j b, \mathsf{vk}_{m|_j b}^{(k^*)}\right), \left(\mathsf{decom}, \mathsf{k}_{\mathsf{PRF}}, \mathsf{sk}_{m|_j b}^{(k^*)}\right)\right)$$

it uses the simulated prover for computing

$$\pi_{m|_j b}^{(k^*)} = \mathsf{ZK.Sim}_2\left(\tau_{\mathsf{ZK}}, \left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}, m|_j b, \mathsf{vk}_{m|_j b}^{(k^*)}\right)\right).$$

That is, in this experiment it still holds that $\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}, m|_j b, \mathsf{vk}_{m|_j b}^{(k^*)}\right) \in \mathcal{L}$, but the proofs are generated using the trapdoor τ_{ZK} instead of the witness $\left(\mathsf{decom}, \mathsf{k}_{\mathsf{PRF}}, \mathsf{sk}_{m|_j b}^{(k^*)}\right)$.

A direct reduction to the adaptive zero-knowledge property of the proof system ZK (see Definition 2.6) yields the following claim:

Claim 5.2. *There exists a probabilistic polynomial-time algorithm B_1 such that for every $\lambda \in \mathbb{N}$ it holds that*

$$|\Pr[\mathsf{Exp}_0(\lambda) = 1] - \Pr[\mathsf{Exp}_1(\lambda) = 1]| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{ZK},B_1}(\lambda).$$

Experiment $\mathbf{Exp_2}$. This experiment is obtained from Exp_1 by modifying the distribution of the commitment com that is included in the verification key $\mathsf{vk} = (\mathsf{vk}_\varepsilon, \mathsf{com})$: Instead of computing $(\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{COM.Commit}(\mathsf{crs}_{\mathsf{COM}}, \mathsf{k}_{\mathsf{PRF}})$, the experiment computes $(\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{COM.Commit}\left(\mathsf{crs}_{\mathsf{COM}}, 0^{|\mathsf{k}_{\mathsf{PRF}}|}\right)$. Note that the key $\mathsf{k}_{\mathsf{PRF}}$ is still sampled and used for deriving randomness for generating the one-time signing and verification keys.

A direct reduction to the computational hiding property of the commitment scheme COM (see Definition 2.5) yields the following claim:

Claim 5.3. *There exists a probabilistic polynomial-time algorithm B_2 such that for every $\lambda \in \mathbb{N}$ it holds that*

$$|\Pr[\mathsf{Exp}_1(\lambda) = 1] - \Pr[\mathsf{Exp}_2(\lambda) = 1]| \leq \mathsf{Adv}^{\mathsf{hiding}}_{\mathsf{COM},B_2}(\lambda).$$

Experiment $\mathbf{Exp_3}$. This experiment is obtained from Exp_2 by producing the one-time signing and verification keys using true randomness instead of using the pseudorandom function. That is, instead of computing

$$\left(\mathsf{sk}_{m|_j b}^{(k^*)}, \mathsf{vk}_{m|_j b}^{(k^*)}\right) = \mathsf{1T.KG}(\mathsf{pp}_{\mathsf{1T}}; \mathsf{PRF.Eval}(\mathsf{k}_{\mathsf{PRF}}, m|_j b))$$

for some $k^* \in [n]$ and for all $j \in \{0,\ldots,\ell-1\}$ and $b \in \{0,1\}$, the experiment samples $r^{(k^*)}_{m|_j b} \leftarrow \{0,1\}^*$ of the appropriate length, and computes

$$\left(\mathsf{sk}^{(k^*)}_{m|_j b}, \mathsf{vk}^{(k^*)}_{m|_j b}\right) = \mathsf{1T.KG}\left(\mathsf{pp}_{\mathsf{1T}}; r^{(k^*)}_{m|_j b}\right).$$

The experiment stores all of the one-time signing and verification keys that it produces, and whenever such keys are used more than once (i.e., on signing queries corresponding to intersecting paths from the root ε to different leaves), the corresponding one-time signing and verification keys are reused (or, equivalently, produced again using the same random strings).

A direct reduction to the security of the pseudorandom function PRF (see Definition 2.3) yields the following claim:

Claim 5.4. *There exists a probabilistic polynomial-time algorithm B_3 such that for every $\lambda \in \mathbb{N}$ it holds that*

$$|\Pr[\mathsf{Exp}_2(\lambda)=1] - \Pr[\mathsf{Exp}_3(\lambda)=1]| \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},B_3}(\lambda).$$

Experiment Exp_4. This experiment is obtained from Exp_3 by modifying its output in some cases, as follows. If there exists a one-time signing key produced by the experiment Exp_3 that is used for signing more than one message then Exp_4 outputs 0, and otherwise it outputs the output of Exp_3.

The following claim upper bounds the difference between the adversary's success probabilities in the experiments Exp_3 and Exp_4 based on the simulation soundness property of the proof system ZK (see Definition 2.6) and on the statistical binding property of the commitment scheme COM (see Definition 2.5):

Claim 5.5. *Assuming that COM is $\epsilon_{\mathsf{binding}}$-statistically binding, there exists a probabilistic polynomial-time algorithm B_4 such that for every $\lambda \in \mathbb{N}$ it holds that*

$$|\Pr[\mathsf{Exp}_3(\lambda)=1] - \Pr[\mathsf{Exp}_4(\lambda)=1]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{ss}}_{\mathsf{ZK},B_4}(\lambda) + \epsilon_{\mathsf{binding}}(\lambda).$$

Proof of Claim 5.5. Recall that the adversary A is restricted to issuing signing queries with respect to a single vector of verification keys. We denote this vector by

$$\vec{\mathsf{vk}} = \left(\left(\mathsf{vk}^{(1)}_{\varepsilon}, \mathsf{com}^{(1)}\right), \ldots, \left(\mathsf{vk}^{(n)}_{\varepsilon}, \mathsf{com}^{(n)}\right)\right),$$

and we denote by $k^* \in [n]$ the unique integer for which it holds that $(\mathsf{vk}_\varepsilon, \mathsf{com}) = \left(\mathsf{vk}^{(k^*)}_{\varepsilon}, \mathsf{com}^{(k^*)}\right)$, where $\mathsf{vk} = (\mathsf{vk}_\varepsilon, \mathsf{com})$ is the honest verification key provided as input to the adversary (note that without loss of generality such an integer exists and it is unique since otherwise the adversary obtains no responses to signing queries). Additionally, denote by $\left\{\left(\mathsf{sk}^{(k^*)}_{\alpha}, \mathsf{vk}^{(k^*)}_{\alpha}\right)\right\}_{\alpha \in \{\varepsilon\} \cup \{0,1\}^{\leq \ell}}$ the subset of one-time signing and verification keys used by experiment Exp_3 for responding to the A's signing queries (note that not all of these keys have actually being generated, but rather a subset that corresponds to the paths from the root ε to the leaves corresponding to the signed messages). Denote by $\mathsf{ReUsedKey}$ the event in which on these pairs is used for signing more than one message in experiment Exp_3, then by the definition of experiment Exp_4 it holds that

$$|\Pr[\mathsf{Exp}_3(\lambda)=1] - \Pr[\mathsf{Exp}_4(\lambda)=1]| \leq \Pr[\mathsf{ReUsedKey}].$$

Consider now an execution of Exp_3 in which some signing key $\mathsf{sk}_\alpha^{(k^*)}$ was used for signing more than one message. Note that a signing key $\mathsf{sk}_\alpha^{(k^*)}$ for $\alpha \in \{0,1\}^\ell$ is always used for signing exactly one message (which is the message $\left(\alpha, \mathsf{tag}_{\vec{\mathsf{vk}}}\right)$). Therefore, it must be that $\alpha \in \{\varepsilon\} \cup \{0,1\}^{<\ell}$, and thus $\mathsf{sk}_\alpha^{(k^*)}$ was used for signing the concatenation of two aggregated verification keys. Moreover, since all signing queries are issued with respect to the same "root" keys $\mathsf{vk}_\varepsilon^{(1)}, \ldots, \mathsf{vk}_\varepsilon^{(n)}$, then there must exists a signing key $\mathsf{sk}_\alpha^{(k^*)}$ for $\alpha \in \{0,1\}^\ell$ that was used for signing more than one message with respect to the same vector of verification keys $\vec{\mathsf{vk}}_\alpha = \left(\mathsf{vk}_\alpha^{(1)}, \ldots, \mathsf{vk}_\alpha^{(n)}\right)$. We denote the two signing operations as

$$\begin{aligned}
\sigma_\alpha^{(k^*)} &= \mathsf{1T.Sign}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_\alpha^{(k^*)}, \vec{\mathsf{vk}}_\alpha, (\mathsf{aggvk}_{\alpha 0}, \mathsf{aggvk}_{\alpha 1})\right) \\
\hat{\sigma}_\alpha^{(k^*)} &= \mathsf{1T.Sign}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_\alpha^{(k^*)}, \vec{\mathsf{vk}}_\alpha, \left(\hat{\mathsf{aggvk}}_{\alpha 0}, \hat{\mathsf{aggvk}}_{\alpha 1}\right)\right)
\end{aligned}$$

where $(\mathsf{aggvk}_{\alpha 0}, \mathsf{aggvk}_{\alpha 1}) \neq \left(\hat{\mathsf{aggvk}}_{\alpha 0}, \hat{\mathsf{aggvk}}_{\alpha 1}\right)$. This means that for some $b \in \{0,1\}$, the adversary provided two inputs $\left(\mathsf{vk}_{\alpha b}^{(i)}, \pi_{\alpha b}^{(i)}\right)$ and $\left(\hat{\mathsf{vk}}_{\alpha b}^{(i)}, \hat{\pi}_{\alpha b}^{(i)}\right)$ such that $\mathsf{vk}_{\alpha b}^{(i)} \neq \hat{\mathsf{vk}}_{\alpha b}^{(i)}$, and these were used for computing the aggregated verification keys $\mathsf{aggvk}_{\alpha b} \neq \hat{\mathsf{aggvk}}_{\alpha b}$ after verifying the proofs $\pi_{\alpha b}^{(i)}$ and $\hat{\pi}_{\alpha b}^{(i)}$.

However, except with probability $\epsilon_{\mathsf{binding}}$, the common-references string $\mathsf{crs}_{\mathsf{com}}$ is such that $\mathsf{com}^{(i)}$ uniquely determines a key $\mathsf{k}_{\mathsf{PRF}}^{(i)}$ for the pseudorandom function, and therefore at most one of the keys $\mathsf{vk}_{\alpha b}^{(i)}$ and $\hat{\mathsf{vk}}_{\alpha b}^{(i)}$ was actually produced using the randomness $\mathsf{PRF.Eval}\left(\mathsf{k}_{\mathsf{PRF}}^{(i)}, \alpha b\right)$. That is, although both proofs pass verification, it must hold that either

$$\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \mathsf{vk}_{\alpha b}^{(i)}\right) \notin \mathcal{L}$$

or

$$\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \hat{\mathsf{vk}}_{\alpha b}^{(i)}\right) \notin \mathcal{L}.$$

This leads to an algorithm B_4 that participates in the simulation-soundness experiment $\mathsf{Exp}_{\mathsf{ZK},B_4}^{\mathsf{ss}}(\lambda)$ by simulating the experiment Exp_3 to A, where $\mathsf{crs}_{\mathsf{ZK}}$ is generated by Sim_1 and provided as input to B_4, and where B_4 generates proofs using oracle access to Sim_2 (recall Definition 2.6). Whenever B_4 observes $\left(\mathsf{vk}_{\alpha b}^{(i)}, \pi_{\alpha b}^{(i)}\right)$ and $\left(\hat{\mathsf{vk}}_{\alpha b}^{(i)}, \hat{\pi}_{\alpha b}^{(i)}\right)$ such that $\mathsf{vk}_{\alpha b}^{(i)} \neq \hat{\mathsf{vk}}_{\alpha b}^{(i)}$ and both pass verification as above, then B_4 outputs

$$\left(\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \mathsf{vk}_{\alpha b}^{(i)}\right), \pi_{\alpha b}^{(i)}\right)$$

with probability $1/2$ and

$$\left(\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \hat{\mathsf{vk}}_{\alpha b}^{(i)}\right), \hat{\pi}_{\alpha b}^{(i)}\right)$$

with probability $1/2$. We observe that the proofs $\pi_{\alpha b}^{(i)}$ and $\hat{\pi}_{\alpha b}^{(i)}$ could not have been previously provided by the oracle Sim_2 to the algorithm B_4 as responses to the queries $\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \mathsf{vk}_{\alpha b}^{(i)}\right)$ or $\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \hat{\mathsf{vk}}_{\alpha b}^{(i)}\right)$, respectively (if such queries were issued). Specifically, for the internal nodes $\alpha 0$ and $\alpha 1$, the algorithm B_4 queries the oracle Sim_2 only with the two instances $\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(k^*)}, \alpha b, \mathsf{vk}_{\alpha b}^{(k^*)}\right)$ for $b = 0$ and $b = 1$. However, as instructed in Steps 1 and 3 of the algorithm Sign, the algorithm B_4 aborts the signing protocol if either $\mathsf{vk}_{\alpha b}^{(k^*)} = \mathsf{vk}_{\alpha b}^{(i)}$ or $\mathsf{vk}_{\alpha b}^{(k^*)} = \hat{\mathsf{vk}}_{\alpha b}^{(i)}$. Therefore, the algorithm B_4 does not query the oracle Sim_2 with either $\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \mathsf{vk}_{\alpha b}^{(i)}\right)$

or $\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, \alpha b, \hat{\mathsf{vk}}^{(i)}_{\alpha b}\right)$. This means that for the algorithm B_4 it holds that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{ZK},B_4}(\lambda) &\geq \frac{1}{2}\cdot\left(\Pr\left[\mathsf{ReUsedKey}\right] - \epsilon_{\mathsf{binding}}(\lambda)\right)\\ &\geq \frac{1}{2}\cdot\left(\left|\Pr\left[\mathsf{Exp}_3(\lambda)=1\right] - \Pr\left[\mathsf{Exp}_4(\lambda)=1\right]\right| - \epsilon_{\mathsf{binding}}(\lambda)\right)\end{aligned}$$

as required. ■

Experiment Exp_5. This experiment is obtained from Exp_4 by modifying its output in some cases, as follows. Denote by $\vec{\mathsf{vk}}$ the vector of one-time verification keys with respect to which A issues signing queries, and denote by $\hat{\vec{\mathsf{vk}}}$ the vector of verification keys included in A's output. If $\vec{\mathsf{vk}} \neq \hat{\vec{\mathsf{vk}}}$ and $\mathsf{CRH.Eval}\left(\mathsf{k}_{\mathsf{CRH}}, \vec{\mathsf{vk}}\right) = \mathsf{CRH.Eval}\left(\mathsf{k}_{\mathsf{CRH}}, \hat{\vec{\mathsf{vk}}}\right)$ then Exp_5 outputs 0, and otherwise it outputs the output of Exp_4.

A direct reduction to the collision resistance property of the function family CRH (see Definition 2.4) yields the following claim:

Claim 5.6. *There exists a probabilistic polynomial-time algorithm B_5 such that for every $\lambda \in \mathbb{N}$ it holds that*

$$\left|\Pr\left[\mathsf{Exp}_4(\lambda)=1\right] - \Pr\left[\mathsf{Exp}_5(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{crh}}_{\mathsf{CRH},B_5}(\lambda).$$

The following claim upper bounds the success probability of the adversary A in the experiment Exp_5 based on the security of the one-time multi-signature scheme $\mathsf{1T}$:

Claim 5.7. *There exists a probabilistic polynomial-time algorithm B_6 such that for every $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[\mathsf{Exp}_5(\lambda)=1\right] \leq L \cdot \mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda).$$

Proof of Claim 5.7. Let B_6 be the algorithm that on input $\mathsf{crs}_{\mathsf{1T}}$ and $\mathsf{vk}_{\mathsf{1T}}$ participates in the experiment $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}$ by emulating the experiment Exp_5 to the algorithm A as follows. First, the algorithm B_6 computes $(\mathsf{crs}_{\mathsf{ZK}}, \tau_{\mathsf{ZK}}) \leftarrow \mathsf{Sim}_1(1^\lambda)$, $\mathsf{crs}_{\mathsf{COM}} \leftarrow \mathsf{COM.Setup}(1^\lambda)$, $\mathsf{k}_{\mathsf{CRH}} \leftarrow \mathsf{CRH.KG}(1^\lambda)$, and $(\mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{COM.Commit}\left(\mathsf{crs}_{\mathsf{COM}}, 0^{|\mathsf{k}|}\right)$. Then, recall that in the experiment Exp_5, the algorithm B_6 generates at most $L = 2 \cdot \ell \cdot q_{\mathsf{Sign}} + 1$ one-time signing and verification keys. The algorithm B_6 samples $v \leftarrow [L]$, determining the index of the one-time verification key for which $\mathsf{vk}_{\mathsf{1T}}$ will be used instead of sampling a fresh key pair. In particular, if $v = 1$ then B_6 sets $\mathsf{vk}_\varepsilon = \mathsf{vk}_{\mathsf{1T}}$, and otherwise B_6 samples $(\mathsf{sk}_\varepsilon, \mathsf{vk}_\varepsilon) \leftarrow \mathsf{1T.KG}(\mathsf{pp}_{\mathsf{1T}})$. Next, B_6 invokes the algorithm A on the public parameters $\mathsf{pp} = (\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{ZK}}, \mathsf{crs}_{\mathsf{COM}}, \mathsf{k}_{\mathsf{CRH}})$ and the verification key $\mathsf{vk} = (\mathsf{vk}_\varepsilon, \mathsf{com})$, and responds to A's signing queries as follows.

When A issues a signing query of the form $\left(\vec{\mathsf{vk}}, m\right)$, where

$$\vec{\mathsf{vk}} = \left(\left(\mathsf{vk}^{(1)}_\varepsilon, \mathsf{com}^{(1)}\right), \ldots, \left(\mathsf{vk}^{(n)}_\varepsilon, \mathsf{com}^{(n)}\right)\right),$$

the algorithm B_6 first verifies that there is a unique $k^* \in [n]$ for which $(\mathsf{vk}_\varepsilon, \mathsf{com}) = \left(\mathsf{vk}^{(i)}_\varepsilon, \mathsf{com}^{(i)}\right)$, and otherwise responds with $\perp$. Then, for any $j \in \{0, \ldots, \ell-1\}$ and $b \in \{0,1\}$, the algorithm B_6 samples $r^{(k^*)}_{m|_j b} \leftarrow \{0,1\}^*$ of the appropriate length, and computes

$$\begin{aligned}\left(\mathsf{sk}^{(k^*)}_{m|_j b}, \mathsf{vk}^{(k^*)}_{m|_j b}\right) &= \mathsf{1T.KG}\left(\mathsf{pp}_{\mathsf{1T}}; r^{(k^*)}_{m|_j b}\right)\\ \pi^{(k^*)}_{m|_j b} &= \mathsf{ZK.Sim}_2\left(\tau_{\mathsf{ZK}}, \left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}, m|_j b, \mathsf{vk}^{(k^*)}_{m|_j b}\right)\right),\end{aligned}$$

with the exception of setting $\mathsf{vk}_{m|_j b}^{(k^*)} = \mathsf{vk}_{\mathsf{1T}}$ for the v-th key $\mathsf{vk}_{m|_j b}^{(k^*)}$ that is produced as part of the emulation whenever $v > 1$ (recall that the case $v = 1$ was already handled differently by setting $\mathsf{vk}_{\varepsilon} = \mathsf{vk}_{\mathsf{1T}}$). The algorithm B_6 responds to the signing query with $\left\{\left(\mathsf{vk}_{m|_j b}^{(k^*)}, \pi_{m|_j b}^{(k^*)}\right)\right\}_{j\in\{0,\ldots,\ell-1\},b\in\{0,1\}}$. In addition, when A issues a signing query of the form

$$\left\{\left(\mathsf{vk}_{m|_j b}^{(i)}, \pi_{m|_j b}^{(i)}\right)\right\}_{i\in[n]\setminus\{k^*\},j\in\{0,\ldots,\ell-1\},b\in\{0,1\}},$$

corresponding to an earlier signing query of the form $\left(\vec{\mathsf{vk}}, m\right)$, the algorithm B_6 first verifies that for all $(i, j, b) \in ([n] \setminus \{k^*\}) \times \{0, \ldots, \ell - 1\} \times b \in \{0, 1\}$ it holds that

$$\mathsf{ZK.V}\left(\mathsf{crs}_{\mathsf{ZK}}, \left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{crs}_{\mathsf{com}}, \mathsf{com}^{(i)}, m|_j b, \mathsf{vk}_{m|_j b}^{(i)}\right), \pi_{m|_j b}^{(i)}\right) = 1$$

and $\mathsf{vk}_{m|_j b}^{(i)} \neq \mathsf{vk}_{m|_j b}^{(k^*)}$, and otherwise responds with $\perp$. Then, the algorithm B_6 computes

$$\begin{aligned}
\vec{\mathsf{vk}}_{m|_j b} &= \left(\mathsf{vk}_{m|_j b}^{(1)}, \ldots, \mathsf{vk}_{m|_j b}^{(n)}\right) \text{ for all } j \in \{0, \ldots, \ell - 1\} \text{ and } b \in \{0, 1\} \\
\mathsf{aggvk}_{m|_j b} &= \mathsf{1T.KAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \vec{\mathsf{vk}}_{m|_j b}\right) \text{ for all } j \in \{0, \ldots, \ell - 1\} \text{ and } b \in \{0, 1\} \\
\sigma_{m|_j}^{(k^*)} &= \mathsf{1T.Sign}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_{m|_j}^{(k^*)}, \vec{\mathsf{vk}}_{m|_j}, \left(\mathsf{aggvk}_{m|_j 0}, \mathsf{aggvk}_{m|_j 1}\right)\right) \text{ for all } j \in \{0, \ldots, \ell - 1\} \\
\sigma_{m}^{(k^*)} &= \mathsf{1T.Sign}\left(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_{m}^{(k^*)}, \vec{\mathsf{vk}}_{m}, m\right),
\end{aligned}$$

with the exception of accessing the signing oracle $\mathsf{1T.Sign}(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_{\mathsf{1T}}, \cdot, \cdot)$ for obtaining the signature $\sigma_{m|_j}^{(k^*)}$ or $\sigma_{m}^{(k^*)}$ corresponding to the v-th one-time verification key that is produced as part of the emulation (recall that the v-th one-time verification key was set to $\mathsf{vk}_{\mathsf{1T}}$ and thus its corresponding signing key $\mathsf{sk}_{\mathsf{1T}}$ is not known to B_6). The algorithm B_6 locally stores all signatures produced throughout the emulation, so that no signature has to be recomputed, and no query to the oracle $\mathsf{1T.Sign}(\mathsf{pp}_{\mathsf{1T}}, \mathsf{sk}_{\mathsf{1T}}, \cdot, \cdot)$ has to be repeated (as specified above, we assume without loss of generality that the signing algorithm $\mathsf{1T.Sign}$ is deterministic). The algorithm B_6 responds to the signing query with

$$\sigma^{(k^*)} = \left(\left\{\left(\sigma_{m|_j}^{(k^*)}, \mathsf{vk}_{m|_j 0}^{(k^*)}, \mathsf{vk}_{m|_j 1}^{(k^*)}\right)\right\}_{j\in\{0,\ldots,\ell-1\}}, \sigma_{m}^{(k^*)}\right).$$

At some point, the algorithm A produces an output $\left(\hat{\vec{\mathsf{vk}}}, \hat{m}, \hat{\mathsf{agg}}\sigma\right)$ and halts. We let

$$\begin{aligned}
\hat{\vec{\mathsf{vk}}} &= \left(\left(\hat{\mathsf{vk}}_{\varepsilon}^{(1)}, \hat{\mathsf{com}}^{(1)}\right), \ldots, \left(\hat{\mathsf{vk}}_{\varepsilon}^{(n)}, \hat{\mathsf{com}}^{(n)}\right)\right) \\
\mathsf{tag}_{\hat{\vec{\mathsf{vk}}}} &= \mathsf{CRH.Eval}\left(\mathsf{k}_{\mathsf{CRH}}, \hat{\vec{\mathsf{vk}}}\right) \\
\hat{\mathsf{aggvk}}_{\varepsilon} &= \mathsf{1T.KAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \left(\hat{\mathsf{vk}}_{\varepsilon}^{(1)}, \ldots, \hat{\mathsf{vk}}_{\varepsilon}^{(n)}\right)\right) \\
\hat{\mathsf{agg}}\sigma &= \left(\left\{\left(\hat{\mathsf{agg}}\sigma_{\hat{m}|_j}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right)\right\}_{j\in\{0,\ldots,\ell-1\}}, \hat{\mathsf{agg}}\sigma_{\hat{m}}\right).
\end{aligned}$$

In addition, as above, we denote by $\vec{\mathsf{vk}} = \left(\left(\mathsf{vk}_{\varepsilon}^{(1)}, \mathsf{com}^{(1)}\right), \ldots, \left(\mathsf{vk}_{\varepsilon}^{(n)}, \mathsf{com}^{(n)}\right)\right)$ the vector of verification keys relative to which A issued signing queries (recall that there is only one such vector). For determining its output, the algorithm B_6 first checks whether or not the event $\{\mathsf{Exp}_5(\lambda) = 1\}$

occurs, when the event is defined within the probability space induced by the emulation. If the event $\{\mathsf{Exp}_5(\lambda) = 1\}$ does not occur, then B_6 outputs $\perp$. Otherwise, note that if the event $\{\mathsf{Exp}_5(\lambda) = 1\}$ does occur, then since A issued all signing queries with respect to the same vector of verification keys $\vec{\mathsf{vk}}$, the fact that no signing key was used to sign more than one message implies that A's signing queries and their corresponding responses uniquely define the values

$$\vec{\mathsf{vk}}_{m|_j b} = \left(\mathsf{vk}^{(1)}_{m|_j b}, \ldots, \mathsf{vk}^{(n)}_{m|_j b}\right) \text{ for all } j \in \{0, \ldots, \ell - 1\} \text{ and } b \in \{0, 1\}$$

$$\mathsf{aggvk}_{m|_j b} = \mathsf{1T.KAgg}\left(\mathsf{pp}_{\mathsf{1T}}, \vec{\mathsf{vk}}_{m|_j b}\right) \text{ for all } j \in \{0, \ldots, \ell - 1\} \text{ and } b \in \{0, 1\}$$

for each message m for which a signing query was issued. Equipped with this observation, if the event $\{\mathsf{Exp}_5(\lambda) = 1\}$ occurs then the algorithm B_6 determines its output by distinguishing between the following three disjoint cases:

Case I: If $\left(\mathsf{vk}^{(1)}_{\varepsilon}, \ldots, \mathsf{vk}^{(n)}_{\varepsilon}\right) \neq \left(\hat{\mathsf{vk}}^{(1)}_{\varepsilon}, \ldots, \hat{\mathsf{vk}}^{(n)}_{\varepsilon}\right)$, then B_6 outputs

$$\begin{aligned} \vec{\mathsf{vk}}^* &= \left(\hat{\mathsf{vk}}^{(1)}_{\varepsilon}, \ldots, \hat{\mathsf{vk}}^{(n)}_{\varepsilon}\right) \\ m^* &= \left(\hat{\mathsf{aggvk}}_0, \hat{\mathsf{aggvk}}_1\right) \\ \mathsf{agg}\sigma^* &= \hat{\mathsf{agg}}\sigma_{\varepsilon} \end{aligned}$$

Case II: Else, if there exists $j \in \{0, \ldots, \ell - 1\}$ for which $\mathsf{aggvk}_{\hat{m}|_j} = \hat{\mathsf{aggvk}}_{\hat{m}|_j}$ but it holds that $\left(\mathsf{aggvk}_{\hat{m}|_j 0}, \mathsf{aggvk}_{\hat{m}|_j 1}\right) \neq \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right)$ (including the case where $\mathsf{aggvk}_{\hat{m}|_j 0}$ or $\mathsf{aggvk}_{\hat{m}|_j 1}$ were not at all generated during B_6's responses to A's signing queries), then B_6 outputs

$$\begin{aligned} \vec{\mathsf{vk}}^* &= \left(\mathsf{vk}^{(1)}_{\hat{m}|_j}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}|_j}\right) \\ m^* &= \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right) \\ \mathsf{agg}\sigma^* &= \hat{\mathsf{agg}}\sigma_{\hat{m}|_j} \end{aligned}$$

for the minimal such $j \in \{0, \ldots, \ell - 1\}$.

Case III: Else, B_6 outputs

$$\begin{aligned} \vec{\mathsf{vk}}^* &= \left(\mathsf{vk}^{(1)}_{\hat{m}}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}}\right) \\ m^* &= \left(\hat{m}, \mathsf{tag}_{\vec{\mathsf{vk}}}\right) \\ \mathsf{agg}\sigma^* &= \hat{\mathsf{agg}}\sigma_{\hat{m}}. \end{aligned}$$

This concludes the description of the algorithm B_6, and now we turn to analyzing its success probability in the experiment $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}$. The algorithm B_6 perfectly emulates the experiment Exp_5 to A, and therefore

$$\begin{aligned} \mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},\mathsf{B}_6}(\lambda) &= \Pr\left[\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda) = 1\right] \\ &= \Pr\left[\mathsf{Exp}_5(\lambda) = 1\right] \cdot \Pr\left[\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda) = 1 \,\middle|\, \mathsf{Exp}_5(\lambda) = 1\right], \end{aligned} \tag{5.1}$$

where, as noted above, the event $\{\mathsf{Exp}_5(\lambda) = 1\}$ is defined within the probability space induced by the emulation. Note that, conditioned on the event $\{\mathsf{Exp}_5(\lambda) = 1\}$, no one-time signing key used throughout the emulation was used to sign more than one message, and in addition:

1. $\mathsf{vk} = (\mathsf{vk}_\varepsilon, \mathsf{com}) \in \hat{\vec{\mathsf{vk}}}$ and the algorithm A did not issue a signing query $\left(\hat{\vec{\mathsf{vk}}}, \hat{m}\right)$.
2. $\mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_{m|_j}, \left(\hat{\mathsf{aggvk}}_{m|_j 0}, \hat{\mathsf{aggvk}}_{m|_j 1}\right), \hat{\mathsf{aggsig}}_{m|_j}\right) = 1$ for every $j \in \{0, \ldots, \ell - 1\}$.
3. $\mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_{\hat{m}}, \left(\hat{m}, \mathsf{tag}_{\vec{\mathsf{vk}}}\right), \hat{\mathsf{aggsig}}_{\hat{m}}\right) = 1$.

We now show that, conditioned on the event $\{\mathsf{Exp}_5(\lambda) = 1\}$, then in each of Case I, Case II and Case II it holds that $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda) = 1$ with probability at least $1/L$ over the choice of $v \leftarrow [L]$ (which is independent of A's view):

Case I: In this case, if $v = 1$ (an event that occurs with probability $1/L$), then $\mathsf{vk}_\varepsilon = \mathsf{vk_{1T}}$ (recall that $\mathsf{vk_{1T}}$ is the one-time verification key provided to B_6 at the beginning of the experiment $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}$). We now verify that the three conditions required for $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda) = 1$ are satisfied:

- Since $(\mathsf{vk}_\varepsilon, \mathsf{com}) \in \hat{\vec{\mathsf{vk}}} = \left(\left(\hat{\mathsf{vk}}^{(1)}_\varepsilon, \hat{\mathsf{com}}^{(1)}\right), \ldots, \left(\hat{\mathsf{vk}}^{(n)}_\varepsilon, \hat{\mathsf{com}}^{(n)}\right)\right)$ and $\mathsf{vk}_\varepsilon = \mathsf{vk_{1T}}$, then $\mathsf{vk_{1T}} \in \left(\hat{\mathsf{vk}}^{(1)}_\varepsilon, \ldots, \hat{\mathsf{vk}}^{(n)}_\varepsilon\right) = \vec{\mathsf{vk}}^*$.
- Since $\mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_\varepsilon, \left(\hat{\mathsf{aggvk}}_0, \hat{\mathsf{aggvk}}_1\right), \hat{\mathsf{aggsig}}_\varepsilon\right) = 1$, it holds that

$$\begin{aligned}\mathsf{1T.Verify}&\left(\mathsf{pp_{1T}}, \mathsf{KAgg}\left(\mathsf{pp_{1T}}, \vec{\mathsf{vk}}^*\right), m^*, \mathsf{aggsig}^*\right)\\ &= \mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_\varepsilon, \left(\hat{\mathsf{aggvk}}_0, \hat{\mathsf{aggvk}}_1\right), \hat{\mathsf{aggsig}}_\varepsilon\right)\\ &= 1\end{aligned}$$

- Since B_6 queried the signing oracle $\mathsf{1T.Sign}(\mathsf{pp_{1T}}, \mathsf{sk_{1T}}, \cdot, \cdot)$ exactly once, and the input to this query was $\left(\left(\mathsf{vk}^{(1)}_\varepsilon, \ldots, \mathsf{vk}^{(n)}_\varepsilon\right), (\mathsf{aggvk}_0, \mathsf{aggvk}_1)\right)$, and since in Case I we have

$$\left(\mathsf{vk}^{(1)}_\varepsilon, \ldots, \mathsf{vk}^{(n)}_\varepsilon\right) \neq \left(\hat{\mathsf{vk}}^{(1)}_\varepsilon, \ldots, \hat{\mathsf{vk}}^{(n)}_\varepsilon\right),$$

  this means that B_6 did not query the signing oracle with

$$\left(\vec{\mathsf{vk}}^*, m^*\right) = \left(\left(\hat{\mathsf{vk}}^{(1)}_\varepsilon, \ldots, \hat{\mathsf{vk}}^{(n)}_\varepsilon\right), \left(\hat{\mathsf{aggvk}}_0, \hat{\mathsf{aggvk}}_1\right)\right).$$

Case II: Let $j \in \{0, \ldots, \ell - 1\}$ be the minimal integer for which $\mathsf{aggvk}_{\hat{m}|_j} = \hat{\mathsf{aggvk}}_{\hat{m}|_j}$ but it holds that $\left(\mathsf{aggvk}_{\hat{m}|_j 0}, \mathsf{aggvk}_{\hat{m}|_j 1}\right) \neq \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right)$ (or that $\mathsf{aggvk}_{\hat{m}|_j 0}$ or $\mathsf{aggvk}_{\hat{m}|_j 1}$ were not at all generated during B_6's responses to A's signing queries), and let $\vec{\mathsf{vk}}_{\hat{m}|_j} = \left(\mathsf{vk}^{(1)}_{\hat{m}|_j}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}|_j}\right)$. In this case, if $v \in [L]$ equals the index of the verification key $\mathsf{vk}^{(k^*)}_{\hat{m}|_j b}$ (an event that occurs with probability $1/L$), then $\mathsf{vk}^{(k^*)}_{\hat{m}|_j b} = \mathsf{vk_{1T}}$. We now verify that the three conditions required for $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda) = 1$ are satisfied:

- Since $\mathsf{vk}^{(k^*)}_{\hat{m}|_j b} \in \left(\mathsf{vk}^{(1)}_{\hat{m}|_j}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}|_j}\right) = \vec{\mathsf{vk}}^*$ and $\mathsf{vk}^{(k^*)}_{\hat{m}|_j b} = \mathsf{vk_{1T}}$, then $\mathsf{vk_{1T}} \in \vec{\mathsf{vk}}^*$.

- Since $\mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_{\hat{m}|_j}, \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right), \hat{\mathsf{agg}}\sigma_{\hat{m}|_j}\right) = 1$ and $\mathsf{aggvk}_{\hat{m}|_j} = \hat{\mathsf{aggvk}}_{\hat{m}|_j}$, it holds that

$$\begin{aligned}\mathsf{1T.Verify}&\left(\mathsf{pp_{1T}}, \mathsf{KAgg}\left(\mathsf{pp_{1T}}, \vec{\mathsf{vk}}^*\right), m^*, \mathsf{agg}\sigma^*\right)\\ &= \mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_{\hat{m}|_j}, \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right), \hat{\mathsf{agg}}\sigma_{\hat{m}|_j}\right)\\ &= 1\end{aligned}$$

- Since B_6 queried the signing oracle $\mathsf{1T.Sign}(\mathsf{pp_{1T}}, \mathsf{sk_{1T}}, \cdot, \cdot)$ at most once, and the input to this query (if existed) was

$$\left(\left(\mathsf{vk}^{(1)}_{\hat{m}|_j}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}|_j}\right), \left(\mathsf{aggvk}_{\hat{m}|_j 0}, \mathsf{aggvk}_{\hat{m}|_j 1}\right)\right),$$

and since in Case II we have

$$\left(\mathsf{aggvk}_{\hat{m}|_j 0}, \mathsf{aggvk}_{\hat{m}|_j 1}\right) \neq \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right),$$

this means that B_6 did not query the signing oracle with

$$\left(\vec{\mathsf{vk}}^*, m^*\right) = \left(\left(\mathsf{vk}^{(1)}_{\hat{m}|_j}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}|_j}\right), \left(\hat{\mathsf{aggvk}}_{\hat{m}|_j 0}, \hat{\mathsf{aggvk}}_{\hat{m}|_j 1}\right)\right).$$

Case III: In this case, if $v \in [L]$ equals the index of the verification key $\mathsf{vk}^{(k^*)}_{\hat{m}}$ (an event that occurs with probability $1/L$), then $\mathsf{vk}^{(k^*)}_{\hat{m}} = \mathsf{vk_{1T}}$. We now verify that the three conditions required for $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda) = 1$ are satisfied:

- Since $\mathsf{vk}^{(k^*)}_{\hat{m}} \in \left(\mathsf{vk}^{(1)}_{\hat{m}}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}}\right) = \vec{\mathsf{vk}}^*$ and $\mathsf{vk}^{(k^*)}_{\hat{m}} = \mathsf{vk_{1T}}$, then $\mathsf{vk_{1T}} \in \vec{\mathsf{vk}}^*$.
- Since $\mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_{\hat{m}}, \left(\hat{m}, \mathsf{tag}_{\vec{\hat{\mathsf{vk}}}}\right), \hat{\mathsf{agg}}\sigma_{\hat{m}}\right) = 1$, and since in Case III we have $\mathsf{KAgg}\left(\mathsf{pp_{1T}}, \vec{\mathsf{vk}}^*\right) = \mathsf{aggvk}_{\hat{m}} = \hat{\mathsf{aggvk}}_{\hat{m}}$, it holds that

$$\begin{aligned}\mathsf{1T.Verify}&\left(\mathsf{pp_{1T}}, \mathsf{KAgg}\left(\mathsf{pp_{1T}}, \vec{\mathsf{vk}}^*\right), m^*, \mathsf{agg}\sigma^*\right)\\ &= \mathsf{1T.Verify}\left(\mathsf{pp_{1T}}, \hat{\mathsf{aggvk}}_{\hat{m}}, \left(\hat{m}, \mathsf{tag}_{\vec{\hat{\mathsf{vk}}}}\right), \hat{\mathsf{agg}}\sigma_{\hat{m}}\right)\\ &= 1\end{aligned}$$

- The algorithm A did not issue the signing query $\left(\vec{\hat{\mathsf{vk}}}, \hat{m}\right)$, and therefore if $\vec{\hat{\mathsf{vk}}} = \vec{\mathsf{vk}}$ this means that A did not issue any signing query with the message $\hat{m}$. Therefore, if $\vec{\hat{\mathsf{vk}}} = \vec{\mathsf{vk}}$, then B_6 did not query the oracle $\mathsf{1T.Sign}(\mathsf{pp_{1T}}, \mathsf{sk_{1T}}, \cdot, \cdot)$ even once.

  In addition, if $\vec{\hat{\mathsf{vk}}} \neq \vec{\mathsf{vk}}$, it may be that A issued the signing query $(\vec{\mathsf{vk}}, \hat{m})$, and then B_6 queried the oracle $\mathsf{1T.Sign}(\mathsf{pp_{1T}}, \mathsf{sk_{1T}}, \cdot, \cdot)$ exactly once. However, the input to this query was $\left(\left(\mathsf{vk}^{(1)}_{\hat{m}}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}}\right), \left(\hat{m}, \mathsf{tag}_{\vec{\mathsf{vk}}}\right)\right)$, and since $\mathsf{tag}_{\vec{\mathsf{vk}}} \neq \mathsf{tag}_{\vec{\hat{\mathsf{vk}}}}$ this means that B_6 did not query the signing oracle with

$$\left(\vec{\mathsf{vk}}^*, m^*\right) = \left(\left(\mathsf{vk}^{(1)}_{\hat{m}}, \ldots, \mathsf{vk}^{(n)}_{\hat{m}}\right), \left(\hat{m}, \mathsf{tag}_{\vec{\hat{\mathsf{vk}}}}\right)\right).$$

Using the fact that Case I, Case II and Case III correspond to disjoint events, we obtain

$$\Pr\left[\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda)=1\middle|\,\mathsf{Exp}_5(\lambda)=1\right]\geq\frac{1}{L},$$

and then Equation (5.1) yields

$$\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda)\geq\Pr\left[\mathsf{Exp}_5(\lambda)=1\right]\cdot\frac{1}{L}.$$

■

Putting together Claims 5.2–5.7, there exist probabilistic polynomial-time algorithms $B_1,\ldots,B_6$ such that for every $\lambda\in\mathbb{N}$ it holds that

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{SSetMS}}_{\Pi,A}(\lambda) &= \Pr\left[\mathsf{Exp}_0(\lambda)=1\right]\\
&\leq \left|\Pr\left[\mathsf{Exp}_0(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_1(\lambda)=1\right]\right|\\
&\quad+\left|\Pr\left[\mathsf{Exp}_1(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_2(\lambda)=1\right]\right|\\
&\quad+\left|\Pr\left[\mathsf{Exp}_2(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_3(\lambda)=1\right]\right|\\
&\quad+\left|\Pr\left[\mathsf{Exp}_3(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_4\lambda)=1\right]\right|\\
&\quad+\left|\Pr\left[\mathsf{Exp}_4(\lambda)=1\right]-\Pr\left[\mathsf{Exp}_5\lambda)=1\right]\right|\\
&\quad+\Pr\left[\mathsf{Exp}_5\lambda)=1\right]\\
&\leq \epsilon_{\mathsf{binding}}(\lambda)+\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{ZK},B_1}(\lambda)+\mathsf{Adv}^{\mathsf{hiding}}_{\mathsf{COM},B_2}(\lambda)+\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},B_3}(\lambda)\\
&\quad+\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{ZK},B_4}(\lambda)+\mathsf{Adv}^{\mathsf{crh}}_{\mathsf{CRH},B_5}(\lambda)+L\cdot\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},B_6}(\lambda),
\end{aligned}$$

where $L=2\cdot\ell\cdot q_{\mathsf{Sign}}+1$. This settles the proof of Theorem 5.1.

■

References

[AAB+02] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *Advances in Cryptology – EUROCRYPT '02*, pages 418–433, 2002.

[AB21] H. K. Alper and J. Burdges. Two-round trip Schnorr multi-signatures via delinearized witnesses. In *Advances in Cryptology – CRYPTO '21*, pages 157–188, 2021.

[AE18] J.-P. Aumasson and G. Endignoux. Improving stateless hash-based signatures. In *Cryptographers' Track at the RSA Conference*, pages 219–242. Springer, 2018.

[AHK20] T. Agrikola, D. Hofheinz, and J. Kastner. On instantiating the algebraic group model from falsifiable assumptions. In *Advances in Cryptology – EUROCRYPT '20*, pages 96–126, 2020.

[Arg22] Argent. Part I: WTF is account abstraction. Available at https://www.argent.xyz/blog/wtf-is-account-abstraction/, 2022.

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy*, pages 315–334, 2018.

[BCI+13] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Proceedings of the 10th Theory of Cryptography Conference*, pages 315–333, 2013.

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, pages 31–60, 2016.

[BD21] M. Bellare and W. Dai. Chain reductions for multi-signatures and the HBMS scheme. In *Advances in Cryptology – ASIACRYPT '21*, pages 650–678, 2021.

[BDN18] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *Advances in Cryptology – ASIACRYPT '18*, pages 435–464, 2018.

[BFL20] B. Bauer, G. Fuchsbauer, and J. Loss. A classification of computational assumptions in the algebraic group model. In *Advances in Cryptology – CRYPTO '20*, pages 121–151, 2020.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112, 1988.

[BGO+07] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 276–285, 2007.

[BHH+15] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In *Advances in Cryptology – EUROCRYPT '15*, pages 368–397, 2015.

[BHK+19] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe. The SPHINCS$^+$ signature framework. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 2129–2146. ACM, 2019.

[BK20] D. Boneh and S. Kim. One-time and interactive aggregate signatures from lattices. Available at https://crypto.stanford.edu/~skim13/agg_ots.pdf, 2020.

[BKK+15] O. Blazy, S. A. Kakvi, E. Kiltz, , and J. Pan. Tightly-secure signatures from chameleon hash functions. In *Public Key Cryptography – PKC '15*, pages 256–279, 2015.

[BN06] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 390–399, 2006.

[Bol03] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In *Public Key Cryptography – PKC '03*, pages 31–46, 2003.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS08] M. Bellare and S. Shoup. Two-tier signatures from the Fiat-Shamir transform, with applications to strongly unforgeable and one-time signatures. *IET Information Security*, 2(2):47–63, 2008.

[BSBH+18] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018.

[BSM+91] M. Blum, A. D. Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BTT22] C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In *Advances in Cryptology – CRYPTO '22*, pages 276–305, 2022.

[BWG+21] V. Buterin, Y. Weiss, K. Gazso, N. Patel, D. Tirosh, S. Nacson, and T. Hess. EIP-4337: Account abstraction using Alt mempool [DRAFT]. Ethereum Improvement Proposals, no. 4337 (available at https://eips.ethereum.org/EIPS/eip-4337), 2021.

[BY96] M. Bellare and M. Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996.

[CD98] R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *Advances in Cryptology – CRYPTO '98*, pages 424–441, 1998.

[CFG+15] D. Catalano, D. Fiore, R. Gennaro, and K. Vamvourellis. Algebraic (trapdoor) one-way functions: Constructions and applications. *Theoretical Computer Science*, 592:143–165, 2015.

[CHM+20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology – EUROCRYPT '20*, pages 738–768, 2020.

[CL18] R. Canetti and A. Lichtenberg. Certifying trapdoor permutations, revisited. In *Proceedings of the 16th Theory of Cryptography Conference*, pages 476–506, 2018.

[DEF+19] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. On the security of two-round multi-signatures. In *IEEE Symposium on Security and Privacy*, pages 1084–1101, 2019.

[DOT+22] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, 2022.

[FH21] M. Fukumitsu and S. Hasegawa. A tightly secure DDH-based multisignature with public-key aggregation. *International Journal of Networking and Computing*, 11(2):319–337, 2021.

[FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Advances in Cryptology – CRYPTO '18*, pages 33–62, 2018.

[FLS90] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 308–317, 1990.

[FPS20] G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In *Advances in Cryptology – EUROCRYPT '20*, pages 63–95, 2020.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86*, pages 186–194, 1986.

[FSZ22] N. Fleischhacker, M. Simkin, and Z. Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1109–1123, 2022.

[GLS+04] R. Gennaro, D. Leigh, R. Sundaram, and W. S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *Advances in Cryptology – ASIACRYPT '04*, pages 276–292, 2004.

[GLS+23] A. Golovnev, J. Lee, S. Setty, J. Thaler, and R. S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *Advances in Cryptology – CRYPTO '23*, pages 193–226, 2023.

[Gol01] O. Goldreich. Foundations of Cryptography – Volume 1: Basic Techniques. Cambridge University Press, 2001.

[Gol11] O. Goldreich. In a world of P= BPP. In *Studies in Complexity and Cryptography*, pages 191–232. Springer, 2011.

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In *Advances in Cryptology – CRYPTO '06*, pages 97–111, 2006.

[GR13] O. Goldreich and R. D. Rothblum. Enhancements of trapdoor permutations. *Journal of Cryptology*, 26(3):484–512, 2013.

[Gro06] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology – ASIACRYPT '06*, pages 444–459, 2006.

[Gro10] J. Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology – ASIACRYPT '10*, pages 341–358, 2010.

[Gro16] J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology – EUROCRYPT '16*, pages 305–326, 2016.

[GS08] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EUROCRYPT '08*, pages 415–432, 2008.

[GW11] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, 2011.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019.

[HK22] A. Hülsing and M. A. Kudinov. Recovering the tight security proof of SPHINCS$^+$. Cryptology ePrint Archive, Paper 2022/346, 2022.

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1–8, 1983.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.

[KHR+22] M. A. Kudinov, A. Hülsing, E. Ronen, and E. Yogev. SPHINCS+C: Compressing SPHINCS+ with (almost) no cost. Cryptology ePrint Archive, Paper 2022/778, 2022.

[KL21] J. Katz and Y. Lindell. Introduction to Modern Cryptography (3rd Edition). CRC Press, 2021.

[KMP16] E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In *Advances in Cryptology – CRYPTO '16*, pages 33–61, 2016.

[Lam79] L. Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[LHL94] C. Li, T. Hwang, and N. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In *Advances in Cryptology – EUROCRYPT '94*, pages 194–204, 1994.

[Lin06] Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *Journal of Cryptology*, 19(3):359–377, 2006.

[LK23] K. Lee and H. Kim. Two-round multi-signatures from Okamoto signatures. *Mathematics*, 11(14), 2023.

[LOS+06] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology – EUROCRYPT '06*, pages 465–485, 2006.

[LYG19] D. Le, G. Yang, and A. A. Ghorbani. A new multisignature scheme with public key aggregation for blockchain. In *Proceedings of the 17th International Conference on Privacy, Security and Trust*, pages 1–7, 2019.

[Mic00] S. Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MOR01] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254, 2001.

[MPS+19] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[MTT19] T. Mizuide, A. Takayasu, and T. Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In *Topics in Cryptology – CT-RSA '19*, pages 169–188, 2019.

[Nak09] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf, 2009.

[Nao91] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NRS+20] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1717–1731, 2020.

[NRS21] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In *Advances in Cryptology – CRYPTO '21*, pages 189–221, 2021.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[OO91] K. Ohta and T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In *Advances in Cryptology – ASIACRYPT '91*, pages 139–148, 1991.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.

[PW23] J. Pan and B. Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In *Advances in Cryptology – EUROCRYPT '23*, pages 597–627, 2023.

[RS20] L. Rotem and G. Segev. Algebraic distinguishers: From discrete logarithms to decisional Uber assumptions. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 366–389, 2020.

[RY07] T. Ristenpart and S. Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In M. Naor, editor, *Advances in Cryptology – EUROCRYPT '07*, pages 228–245, 2007.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 543–553, 1999.

[Sch91] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[SCO+01] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology – CRYPTO '01*, pages 566–598, 2001.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT '97*, pages 256–266, 1997.

[Sim98] D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT '98*, pages 334–345, 1998.

[Sta23] StarkWare. Account abstraction: Improving security and user experience for mainstream crypto adoption. Available at https://medium.com/starkware/account-abstraction-improving-security-and-user-experience-for-mainstream-crypto-adoption-eb57cb09023, 2023.

[TSS+23] K. Takemure, Y. Sakai, B. Santoso, G. Hanaoka, and K. Ohta. More efficient two-round multi-signature scheme with provably secure parameters. Cryptology ePrint Archive, Paper 2023/155, 2023.

[TZ23] S. Tessaro and C. Zhu. Threshold and multi-signature schemes from linear hash functions. In *Advances in Cryptology – EUROCRYPT '23*, pages 628–658, 2023.

[XZS22] T. Xie, Y. Zhang, and D. Song. Orion: Zero knowledge proof with linear prover time. In *Advances in Cryptology – CRYPTO '22*, pages 299–328, 2022.

A Extension: A t-Time Multi-Signature Scheme

In this section we show that our one-time multi-signature scheme, described in Section 4, naturally extends to a t-time multi-signature scheme without introducing any additional assumptions.

The t-time scheme's verification and aggregated verification keys consist of $t+1$ group elements, whereas signatures still consist of a single group element. At a high level, the extension relies on ideas developed originally for batch identification schemes, such as the batch Schnorr identification scheme constructed by Gennaro, Leigh, Sundaram and Yerazunis [GLS+04].

For extending our one-time scheme, the key-generation algorithm invoked by each signer samples $x^{(1)},\ldots,x^{(t)},r \leftarrow \mathcal{X}_{\mathsf{Ind}}$, computes $X^{(1)} = \mathsf{F}_{\mathsf{Ind}}\left(x^{(1)}\right),\ldots,X^{(t)} = \mathsf{F}_{\mathsf{Ind}}\left(x^{(t)}\right), R = \mathsf{F}_{\mathsf{Ind}}(r)$, and then outputs

$$(\mathsf{sk},\mathsf{vk}) = \left(\left(x^{(1)},\ldots,x^{(t)},r\right),\left(X^{(1)},\ldots,X^{(t)},R\right)\right).$$

For aggregating n verification keys, a vector $\vec{\mathsf{vk}} = (\mathsf{vk}_1,\ldots,\mathsf{vk}_n)$ of verification keys, where $\mathsf{vk}_i = \left(X_i^{(1)},\ldots,X_i^{(t)},R_i\right)$ for every $i \in [n]$, is aggregated by computing

$$\begin{aligned}(a_1,\ldots,a_n) &= \mathsf{H}_1\left(\vec{\mathsf{vk}}\right) \in \mathbb{K}_{\mathsf{Ind}}^n\\ \mathsf{agg}X^{(j)} &= \prod_{i=1}^{n}\left(X_i^{(j)}\right)^{a_i} \in \mathcal{Y}_{\mathsf{Ind}} \text{ for every } j \in [t]\\ \mathsf{agg}R &= \prod_{i=1}^{n} R_i^{a_i} \in \mathcal{Y}_{\mathsf{Ind}},\end{aligned}$$

and outputting the aggregated verification key

$$\mathsf{aggvk} = \left(\mathsf{agg}X^{(1)},\ldots,\mathsf{agg}X^{(t)},\mathsf{agg}R\right).$$

For signing a message m with respect to an aggregated verification key aggvk, each signer first computes $\beta = \mathsf{H}_0\left(m,\mathsf{aggvk}\right)$ and then uses their signing key $\mathsf{sk}_i = \left(x_i^{(1)},\ldots,x_i^{(t)},r_i\right)$ for computing

a signature $\sigma_i = r_i + \sum_{j=1}^{t} \beta^j \cdot x_i^{(j)}$. The signatures $\sigma_1, \ldots, \sigma_n$ are then aggregated exactly as in the one-time scheme $\mathsf{1T}$ by computing $\mathsf{agg}\sigma = \sum_{i=1}^{n} a_i \cdot \sigma_i$, and the verification algorithm verifies that

$$\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma) = \mathsf{agg}R \cdot \prod_{j=1}^{t} \left(\mathsf{agg}X^{(j)}\right)^{\beta^j} .$$

In terms of correctness, letting $\beta = \mathsf{H}_0\left(m, \mathsf{aggvk}\right)$, it holds that

$$\begin{aligned}
\mathsf{F}_{\mathsf{Ind}}(\mathsf{agg}\sigma) &= \mathsf{F}_{\mathsf{Ind}}\left(\sum_{i=1}^{n} a_i \cdot \sigma_i\right) \\
&= \mathsf{F}_{\mathsf{Ind}}\left(\sum_{i=1}^{n} a_i \cdot r_i + \sum_{j=1}^{t} \beta^j \cdot \sum_{i=1}^{n} a_i \cdot x_i^{(j)}\right) \\
&= \left(\prod_{i=1}^{n} \mathsf{F}_{\mathsf{Ind}}(r_i)^{a_i}\right) \cdot \prod_{j=1}^{t} \left(\prod_{i=1}^{n} \mathsf{F}_{\mathsf{Ind}}(x_i)^{a_i}\right)^{\beta^j} \\
&= \mathsf{agg}R \cdot \prod_{j=1}^{t} \left(\mathsf{agg}X^{(j)}\right)^{\beta^j} ,
\end{aligned}$$

and thus the scheme provides perfect correctness.

In terms of security, the extended analysis is obtained from the proof of Theorem 4.1 by following the exact same structure, while enabling the algorithm B to respond to A's t signing queries. This is done via the following two main modifications. First, the algorithm B guesses the indices of A's H_0-queries $\{(m_i, \mathsf{aggvk}_i)\}_{i\in[t]}$ and (m^*, aggvk^*), where $m_1, \ldots, m_t$ are the messages for which A issues signing queries, and $\mathsf{aggvk}_1, \ldots, \mathsf{aggvk}_t$ are the corresponding aggregated verification keys. This leads to a security loss of $q_{\mathsf{H}_0}^{t+1}$, instead of $q_{\mathsf{H}_0}^2$ as in the one-time scheme (similar to the loss in the batch protocol of Gennaro et al. [GLS+04]), which is still polynomial for any constant $t \geq 1$. Second, given $X = \mathsf{F}_{\mathsf{Ind}}(x)$, the algorithm B needs to define the honest verification key $\mathsf{vk} = \left(X^{(1)}, \ldots, X^{(t)}, R\right)$ that will be provided to the algorithm A. This is done by sampling responses $\beta_1, \ldots, \beta_t \leftarrow \mathbb{K}_{\mathsf{Ind}}$ that will be provided by B to A's H_0-queries that correspond to the t signing queries, as well as signatures $\sigma_1, \ldots, \sigma_t \leftarrow \mathcal{X}_{\mathsf{pp}}$ that will be provided by B as responses to A's signing queries. Letting $X^{(1)} = \mathsf{F}_{\mathsf{Ind}}\left(x^{(1)}\right), \ldots, X^{(t)} = \mathsf{F}_{\mathsf{Ind}}\left(x^{(t)}\right)$, and $R = \mathsf{F}_{\mathsf{Ind}}(r)$ for unknowns $x^{(1)}, \ldots, x^{(t)}$ and r, the following linear equations must be satisfied:

$$\begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_t \end{pmatrix} = \begin{pmatrix} 1 & \beta_1 & \ldots & \beta_1^t \\ \vdots & & \ddots & \\ 1 & \beta_t & \ldots & \beta_t^t \end{pmatrix} \cdot \begin{pmatrix} r \\ x^{(1)} \\ \vdots \\ x^{(t)} \end{pmatrix}$$

Note that whenever the values $\beta_1, \ldots, \beta_t$ are distinct, then the above matrix has full rank. Therefore, for any fixing of one of the $x^{(i)}$'s to unknown value x, there is a unique (and linear) solution for the remaining $x^{(j)}$'s and r. Relying on this fact, the algorithm B uniformly samples an index $i \leftarrow [t]$, sets $X^{(i)} = X$, and can then compute the values of the remaining $X^{(j)}$'s and R in a linearly homomorphic manner given X, $\sigma_1, \ldots, \sigma_t$ and $\beta_1, \ldots, \beta_t$ (note that the case $t = 1$ corresponds to the proof of Theorem 4.1).

B From One-Time Multi-Signatures to Collision-Resistant Hashing

In this section we show that a collision-resistant hash function can be constructed in a black-box manner from any one-time multi-signature scheme satisfying a natural property (which is satisfied by our one-time multi-signature scheme in Section 4). The property asks that the scheme's key-generation algorithm defines an injective mapping of its randomness to its verification keys. We refer to this property as "randomness-injective key generation", and an interesting open problem is to study the extent to which this property is essential in this context. Given that one-way functions cannot be used in a black-box manner to construct a collision-resistant hash function [Sim98], this rules our various natural construction of one-time multi-signature schemes based on one-way function.

Note that, in addition to randomness-injective key generation, one must also always assume that the scheme's key-aggregation algorithm produces rather short keys. Otherwise, any one-time single-signer scheme (which can be constructed based on any one-way function [Lam79]) can be used to construct a one-time multi-signer scheme in a black-box manner by concatenating individual verification keys for producing an aggregated verification key.

Our result shows that, unlike the case of standard (i.e., single-user) one-time signatures, a one-time multi-signature scheme with short keys (and randomness-injective key generation) cannot be constructed based on any one-way function in a black-box manner [IR89]. This justifies, in some sense, the fact that our one-time multi-signature scheme is based on an assumption that seems stronger than the assumption that one-way functions exist. Specifically, our one-time multi-signature scheme is based on any ring-homomorphic one-way function, which is a significantly more structured primitive when compared to plain one-way functions.

In what follows we describe the construction and prove its collision resistance based on the security of the underlying one-time multi-signature scheme. Let $\mathsf{1T} = (\mathsf{1T.Setup}, \mathsf{1T.KG}, \mathsf{1T.KAgg}, \mathsf{1T.Sign}, \mathsf{1T.SAgg}, \mathsf{1T.Verify})$ be a one-time multi-signature scheme over the message space $\mathcal{M} = \{0,1\}$. We denote by $\ell = \ell(\lambda)$ the bit-length of the randomness provided to the key-generation algorithm $\mathsf{1T.KG}$.

The function family $\mathsf{CRH} = (\mathsf{KG}, \mathsf{Eval})$

$\mathsf{KG}(1^\lambda)$. On input 1^λ the key-generation algorithm computes

$$\begin{aligned} \mathsf{pp} &\leftarrow \mathsf{1T.Setup}(1^\lambda) \\ (\mathsf{sk}_0, \mathsf{vk}_0) &\leftarrow \mathsf{1T.KG}(1^\lambda) \end{aligned}$$

and outputs $\mathsf{k} = (\mathsf{pp}, \mathsf{vk}_0)$.

$\mathsf{Eval}(\mathsf{k}, x_1, \ldots, x_n)$. On input $\mathsf{k} = (\mathsf{pp}, \mathsf{vk}_0)$ and $x = (x_1, \ldots, x_n) \in \left(\{0,1\}^\ell\right)^n$ the evaluation algorithm computes

$$\begin{aligned} (\mathsf{sk}_i, \mathsf{vk}_i) &\leftarrow \mathsf{1T.KG}(\mathsf{pp}; x_i) \text{ for all } i \in [n] \\ y &= \mathsf{1T.KAgg}(\mathsf{pp}, \mathsf{vk}_0, \mathsf{vk}_1, \ldots, \mathsf{vk}_n) \end{aligned}$$

and outputs y.

In terms of compression, note that if there exists any $n = n(\lambda)$ for which the scheme's key-aggregation algorithm aggregates $n+1$ keys into an aggregated key of length less than $n \cdot \ell$ bits, then the function family CRH is compressing. This holds, in particular if the length of aggregated keys is independent of the number of signers, as in our one-time multi-signature scheme from Section 4.

The following theorem captures the collision resistance of the function family CRH based on the security of the one-time multi-signature scheme $\mathsf{1T}$. As discussed above, the theorem assumes that,

for the key-generation algorithm $\mathsf{1T.KG}$, different random strings lead to different verification keys. Formally, we assume that for any $\lambda \in \mathbb{N}$, for any pp generated by $\mathsf{1T.Setup}(1^\lambda)$ and for any $x \neq x' \in \{0,1\}^{\ell(\lambda)}$ it holds that $\mathsf{vk} \neq \mathsf{vk}'$, where $(\mathsf{sk},\mathsf{vk}) = \mathsf{1T.KG}(\mathsf{pp};x)$ and $(\mathsf{sk}',\mathsf{vk}') = \mathsf{1T.KG}(\mathsf{pp};x')$. We refer to this property as "randomness-injective key generation", and note once again that it is satisfied in particular by our one-time multi-signature scheme from Section 4.

Theorem B.1. *Assuming that the scheme* $\mathsf{1T}$ *has randomness-injective key generation, then for any probabilistic polynomial-time algorithm* A *there exists a probabilistic polynomial-time algorithm* B *such that for every* $\lambda \in \mathbb{N}$ *it holds that*

$$\mathsf{Adv}^{\mathsf{crh}}_{\mathsf{CRH},A}(\lambda) \leq \mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},B}(\lambda).$$

Proof of Theorem B.1. Let A be a probabilistic polynomial-time algorithm that participates in the collision-resistance experiment $\mathsf{Exp}^{\mathsf{crh}}_{\mathsf{CRH},A}$, and consider the algorithm B that participates in the experiment $\mathsf{Exp}^{\mathsf{1TimeMS}}_{\mathsf{1T},B}$ as follows. On input $(\mathsf{pp},\mathsf{vk}_0)$, the algorithm B invokes the algorithm A on input $\mathsf{k} = (\mathsf{pp},\mathsf{vk}_0)$ to obtain $x = (x_1,\ldots,x_n)$ and $x' = (x'_1,\ldots,x'_n)$. If $x = x'$ or $\mathsf{CRH.Eval}(\mathsf{k},x) = \mathsf{CRH.Eval}(\mathsf{k},x')$ then B outputs $\perp$ and halts. Otherwise, B computes

$$\begin{aligned}
(\mathsf{sk}_i,\mathsf{vk}_i) &\leftarrow \mathsf{1T.KG}(\mathsf{pp};x_i) \text{ for all } i \in [n] \\
\vec{\mathsf{vk}} &= (\mathsf{vk}_0,\mathsf{vk}_1,\ldots,\mathsf{vk}_n) \\
\sigma_i &= \mathsf{1T.Sign}\left(\mathsf{pp},\mathsf{sk}_i,\vec{\mathsf{vk}},0\right) \text{ for all } i \in [n] \\
(\mathsf{sk}'_i,\mathsf{vk}'_i) &\leftarrow \mathsf{1T.KG}(\mathsf{pp};x'_i) \text{ for all } i \in [n] \\
\vec{\mathsf{vk}}' &= \left(\mathsf{vk}_0,\mathsf{vk}'_1,\ldots,\mathsf{vk}'_n\right),
\end{aligned}$$

and queries the signing oracle with $\left(\vec{\mathsf{vk}},0\right)$ to obtain a signature σ_0. Then, the algorithm B outputs

$$\begin{aligned}
\vec{\mathsf{vk}}^* &= \vec{\mathsf{vk}}' \\
m^* &= 0 \\
\mathsf{agg}\sigma^* &= \mathsf{1T.SAgg}\left(\mathsf{pp},(\sigma_0,\sigma_1,\ldots,\sigma_n)\right).
\end{aligned}$$

For analyzing the success probability of the algorithm B, note that $\mathsf{CRH.Eval}(\mathsf{k},x) = \mathsf{CRH.Eval}(\mathsf{k},x')$ implies $\mathsf{1T.KAgg}\left(\mathsf{pp},\vec{\mathsf{vk}}\right) = \mathsf{1T.KAgg}\left(\mathsf{pp},\vec{\mathsf{vk}}'\right)$, and therefore the correctness of the scheme $\mathsf{1T}$ implies that

$$\begin{aligned}
&\mathsf{1T.Verify}\left(\mathsf{pp},\mathsf{1T.KAgg}\left(\mathsf{pp},\vec{\mathsf{vk}}^*\right),m^*,\mathsf{agg}\sigma^*\right) \\
&\quad= \mathsf{1T.Verify}\left(\mathsf{pp},\mathsf{1T.KAgg}\left(\mathsf{pp},\vec{\mathsf{vk}}'\right),0,\mathsf{1T.SAgg}\left(\mathsf{pp},(\sigma_0,\sigma_1,\ldots,\sigma_n)\right)\right) \\
&\quad= \mathsf{1T.Verify}\left(\mathsf{pp},\mathsf{1T.KAgg}\left(\mathsf{pp},\vec{\mathsf{vk}}\right),0,\mathsf{1T.SAgg}\left(\mathsf{pp},(\sigma_0,\sigma_1,\ldots,\sigma_n)\right)\right) \\
&\quad= 1.
\end{aligned}$$

In addition, the description of B guarantees that $\mathsf{vk}_0 \in \vec{\mathsf{vk}}^*$, and the assumption that $\mathsf{1T}$ has randomness-injective key generation together with the fact that $x \neq x'$ guarantee that $\vec{\mathsf{vk}} \neq \vec{\mathsf{vk}}'$, and therefore B did not query the signing oracle with $\left(\vec{\mathsf{vk}}^*,m^*\right)$. Therefore, it holds that

$$\mathsf{Adv}^{\mathsf{1TimeMS}}_{\mathsf{1T},B}(\lambda) \geq \mathsf{Adv}^{\mathsf{crh}}_{\mathsf{CRH},A}(\lambda).$$

■