Hint-Free Multi-Signatures

Dennis Hofheinz [1] Michael Reichle*[2] Benedikt Wagner† [2]

[1] ETH Zürich
{hofheinz}@inf.ethz.ch
[2] INSAIT, Sofia University "St. Kliment Ohridski"
michael.reichle@insait.ai
[3] Ethereum Foundation
benedikt.wagner@ethereum.org

Abstract

In a (non-interactive) multi-signature scheme, parties independently generate keys and produce signatures on a common message, which can be aggregated into a single signature verifiable with respect to the set of public keys. Existing multi-signature constructions, however, suffer from at least one of two limitations: either (1) aggregate signatures cannot be further aggregated, or (2) verification requires auxiliary information beyond the set of public keys, such as the aggregation topology.

We argue that these limitations significantly restrict the applicability of multi-signatures in large-scale distributed systems, such as proof-of-stake blockchains. To address this gap, we initiate the formal study of *hint-free multi-signatures*, which support multi-hop aggregation while allowing verification using only the set of public keys.

To the best of our knowledge, the only previously known (folklore) construction that is fully hint-free relies on recursive SNARKs, which introduces unclear heuristics for security (e.g., proving statements about random oracle relations) or inherently limits the depth of the aggregation topology.

We show that hint-free multi-signatures can be realized *without proof recursion*, in the standard model. At a high level, we show how to publicly normalize BLS multi-signatures so that verification depends only on the set of public keys rather than a multi-set. While our scheme is not practical (it uses indistinguishability obfuscation), it establishes the feasibility of the primitive and provides a foundation for future work on practical constructions.

*This work originated when the second author was affiliated with ETH Zürich.
†This work originated when the third author was visiting ETH Zürich.

©IACR 2026. This is the full version of an article that will be published in the proceedings of TCC 2026.

Contents

1 Introduction **3**
- 1.1 Our Contribution . . . 4
- 1.2 Application in Voting and Proof-of-Stake . . . 4
- 1.3 Related Work . . . 5
- 1.4 Outline . . . 6

2 Technical Overview **6**

3 Preliminaries **8**
- 3.1 Indistinguishability Obfuscation . . . 8
- 3.2 Groups . . . 9
- 3.3 Signatures and Standard Model BLS . . . 9
- 3.4 Weak Tag-Based Puncturable Encryption . . . 10
- 3.5 Dual-Mode Non-Interactive Zero-Knowledge . . . 11

4 Hint-Free Multi-Signatures **12**

5 Hint-Free Multi-Signatures without Proof Recursion **13**
- 5.1 Construction . . . 13
- 5.2 Security . . . 14

1 Introduction

In a *(non-interactive) multi-signature scheme* [Bol03, BDN18], multiple parties with independently generated keys can sign a message, and their individual signatures can be publicly *aggregated* into a single compact signature. This aggregate certifies that each party has individually signed the message. The aggregation feature makes multi-signatures a powerful primitive for settings where many parties must collectively attest to a common statement, and it enables a variety of applications [BDN18, MPSW19, Edg23].

A canonical example is voting in Proof-of-Stake (PoS) blockchains [Edg23], such as Ethereum[1]: validators vote for blocks by signing them, and the individual signatures on an accepted block are aggregated into a single compact signature stored on-chain. This drastically reduces the storage overhead compared to storing each validator's signature individually, which would be infeasible given the large number of validators in a modern PoS system.

Multi-Hop Aggregation. A multi-signature scheme supporting *multi-hop aggregation* [DGKV22, DMP26] allows aggregates to themselves be further aggregated: given two aggregates over signer sets S_1 and S_2, one can produce a new aggregate over $S_1 \cup S_2$, and this process can be repeated arbitrarily.

This feature is useful whenever signatures are collected and combined in a distributed or parallelized fashion: in a peer-to-peer network, nodes may independently aggregate signatures as they propagate, and multi-hop aggregation allows these partial aggregates to be further combined without any party having to collect all individual signatures. Coming back to our PoS example, this is precisely the setting of vote aggregation: multiple redundant aggregators each collect a subset of validator signatures, and multi-hop aggregation would allow their outputs to be merged into a single compact aggregate stored in the block.

Reliance on Verification Hints. Unfortunately, all existing solutions supporting multi-hop aggregation come with a major caveat: the verifier requires an additional *hint* beyond the message and the set of public keys of the signers. In more detail:

- *Topology.* Solutions based on batch arguments [DGKV22, DMP26] or mergeable SNARKs [BBPS26][2] require the verifier to know the full aggregation topology, i.e., the complete tree of which aggregates were combined to form which intermediate aggregates and eventually the final one. Moreover, the depth of this topology must be bounded at setup time.
- *Multiplicities.* When aggregating BLS multi-signatures [BLS01, Bol03, RY07, BDN18], the verifier needs to know the *multiplicity* of each public key, i.e., how many times it appeared across the aggregated signatures. For instance, aggregating $\sigma_{1,2} = (\mathsf{sk}_1 + \mathsf{sk}_2)\mathsf{H}(\mathsf{m})$ and $\sigma_{2,3} = (\mathsf{sk}_2 + \mathsf{sk}_3)\mathsf{H}(\mathsf{m})$ yields $\sigma_{1,2} + \sigma_{2,3} = (\mathsf{sk}_1 + 2\mathsf{sk}_2 + \mathsf{sk}_3)\mathsf{H}(\mathsf{m})$, which can only be verified knowing that sk_2 contributes with multiplicity 2.[3] The same issue arises in other pairing-based schemes and we refer to Section 1.3 for an overview.

The overhead from multiplicities is more subtle but still significant in practice: For example, even if multiplicities are guaranteed to lie in $\{0, 1, 2, 3\}$ (say we know that there are just two first-layer aggregators), encoding them requires 2 bits per potential signer, resulting in a hint of 256 KiB for 2^{20} signers[4], compared to 128 KiB when only the *set* of signers needs to be encoded[5]. This overhead grows if there are more first-layer aggregators, and it is one reason why Ethereum does *not* currently employ multi-hop aggregation. It motivates the study of *hint-free* multi-hop aggregation, where verification requires only the *set* of signers (given by the set of their public keys) and no additional hint.

[1] See `https://ethereum.github.io/yellowpaper/paper.pdf`.

[2] In Section 1.3, we explain in more detail why the recent work [BBPS26] requires knowledge of the topology.

[3] We assume the proof-of-possession variant [RY07]. For the random-coefficients variant [BDN18], the full topology is needed instead.

[4] At the time of writing, there are approximately 2^{20} validators registered in Ethereum.

[5] In applications such as PoS blockchains, where the universe of potential signers is fixed and publicly known, the latter suffices: one needs only record which members of the universe signed, which can be encoded as a single bitvector.

A Folklore Solution via Recursive SNARKs. There is a folklore solution *without verification hints* based on *recursively* composing succinct non-interactive arguments of knowledge (SNARKs) [Gro10, BCS16, CGSY24]. Concretely, an aggregate for a set of public keys and a message is a succinct proof of knowledge that for every key in the set, there exists either (a) a valid signature under that key, or (b) a valid aggregate covering that key. This solution has recently been proposed for deployment in Ethereum [KKZM23, DKKW25, KZHB25].

From a theoretical standpoint, however, this approach has significant drawbacks depending on the type of SNARK used. For SNARKs based on knowledge assumptions or with non-straightline extractors, the depth of the aggregation topology is inherently bounded: security can only be argued for constant or logarithmic depth. To support unbounded depth, one instead needs SNARKs with straightline extraction, which are typically constructed in the random oracle model (ROM) [BR93, BCS16, CGSY24]. But recursive composition in this model does not admit a clean security argument: the hash function is modeled simultaneously as a random oracle and as an explicit circuit (since the verification algorithm is being proved inside the SNARK), which is inherently contradictory. Indeed, succinctly proving general statements about random oracle relations has been shown to be impossible [BCG24b]. Alternatively, one could heuristically assume that a ROM-based SNARK is (straightline) knowledge-sound in the standard model, when the random oracle is instantiated with a concrete hash function. Still, the heuristic nature of this approach is worrisome from a theory perspective and the security guarantees that we can infer from it are unclear.

1.1 Our Contribution

Motivated by the above discussion, we study *hint-free multi-signatures (with multi-hop aggregation)* as a new cryptographic primitive.

Definition. Our first contribution is a formal definition and introduction of hint-free multi-signatures. While the need for such a primitive is implicit among practitioners—and, as we have argued, practically motivated by PoS blockchain applications—it has not previously been studied as a standalone cryptographic primitive. With our paper, we bring this problem to the attention of the cryptographic research community, which we view as a contribution in its own right.

Feasibility from Standard Assumptions. Our second contribution is showing feasibility of this primitive. Our construction relies on indistinguishability obfuscation (iO), dual-mode non-interactive zero-knowledge proofs, a weak form of puncturable encryption, and unforgeability of BLS signatures (in the standard model). For instance, these can be realized from iO and mild group based assumptions, such as Matrix Diffie-Hellman and Diffie-Hellman Inversion. While we do not claim our construction to be practically efficient, it establishes feasibility *without recursive SNARKs*. We conjecture that our techniques extend to the setting of aggregate signatures, where the messages of the aggregated signatures may be distinct. We hope that our work inspires further research on this practically relevant primitive, and in particular the development of efficient constructions.

Our Insight. While existing works on multi-hop aggregation argue security by threading a single signature of interest through the aggregation topology, we take a fundamentally different approach. Our main technical insight is to equip the scheme with a public *normalization* procedure that reduces multiplicities in an aggregate, thereby mapping the multi-hop setting back to the single-hop setting.

1.2 Application in Voting and Proof-of-Stake

Hint-free multi-signatures are particularly motivated by their application to voting in Proof-of-Stake (PoS) blockchains. We briefly described this motivation in the introduction; here we go more into detail, with Ethereum as an example in mind.

In Ethereum's PoS protocol (simplified), there are N validators, each registered on-chain with a BLS public key. When a validator observes a valid block, it casts its vote by signing the block with its secret key. These individual signatures must then be aggregated before being attached to the block. This task is performed by designated *aggregators*, which listen on the peer-to-peer network, collect individual

signatures, and compute a BLS aggregate. To enable verification, each aggregate is accompanied by a *bitvector* of N bits indicating which validators' votes are included. To mitigate liveness issues, the protocol designates *multiple* aggregators, each of which independently collects signatures and produces its own aggregate-bitvector pair; all of these are then posted on-chain.

The bitvectors and the aggregate signatures together contribute a significant amount of data to each block. A natural approach to reduce this overhead would be to *aggregate the aggregates*, collapsing the multiple aggregate-bitvector pairs produced by the different aggregators into a single one. Unfortunately, this second-level aggregation is not straightforward with BLS, as the sets of validators covered by the different aggregators will have non-trivial overlaps: the same validator's signature may appear in several of the aggregates simultaneously. In this situation, the verification equation for the aggregate of aggregates requires knowledge of the *multiplicity* of each public key in the aggregate. Thus, the bitvector representation no longer suffices and one must instead store a vector of small integers, one per validator. This counter vector is not significantly smaller than simply concatenating the original bitvector-aggregate pairs. This is why, in the current Ethereum protocol, aggregates are *not* aggregated further.

Hint-free multi-signatures resolve this issue directly. Because verification requires only the *set* of signers rather than their multiplicities, a second-level aggregation becomes possible without any overhead: the aggregates from all aggregators can be combined into a single aggregate signature, and the verifier needs only a single bitvector of N bits indicating the set of validators whose votes are represented.

1.3 Related Work

Let us discuss more related work. While there are interactive approaches to aggregation (e.g., interactive multi-signatures [BN06]), we focus on *non-interactive* aggregation and provide a brief overview below.

Single-hop Aggregation from Pairings. The first multi-signatures [Bol03] and aggregate signatures [BGLS03] are based on BLS signatures [BLS01, BLS04]. Bellare et al. [BNN07] remove the restriction that aggregated signatures must have distinct messages that appears in [BGLS03]. While BLS-based constructions rely on random oracles, there are efficient multi-signatures [LOS+06] and aggregate signatures [HWW25] in the standard model. In recent years, several variants have been developed that offer various tradeoffs [BDN18, BCG+24a, BW24, QX10, QLH12, DGNW20, GV22, BW24]. These constructions build on pairing-based signatures, such as BLS [BLS04] or signatures derived from identity-based encryption (namely [BB04, Wat05]). Aggregation of the latter leverages the algebraic structure of such signature schemes by multiplying individual signatures together, similar to aggregation of BLS as discussed in Section 1. As consequence, none of these schemes do support multi-hop aggregation as the issue with multiplicities arises.

We note that there are pairing-and RSA-based constructions [LMRS04, LOS+06, BNN07, BGOY07, Nev08, BGR12, FLS12, GOR18] that avoid the issue of multiplicities if aggregation is *sequential*. That is, to produce an aggregate for message-key pairs $(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)$, signer i receives an aggregate from signer $i-1$ and produces an aggregate for the first i message-key pairs. Unfortunately, sequential aggregation is ill-fit for highly distributed settings such as proof-of-stake.

Universal Aggregation. In a universal aggregation system, any set of signatures (from potentially distinct signature schemes) can be aggregated into a short signature [HKW15]. Here, arbitrary *unique* signatures can be aggregated under iO, puncturable PRFs, and security of RSA. Roughly, a unique signature is mapped (via an obfuscated program) to an RSA signature. This helps as RSA signatures are itself aggregatable, similar to BLS. However, the issue with multiplicities appears and message-key pairs must be distinct, which cannot be guaranteed in a multi-hop scenario.

The uniqueness requirement for the aggregated signatures can be removed in two ways. Either at the expense of relying on VBB obfuscation (which is a much stronger assumption), or by adapting the construction to pass through the forgery within the aggregate in an encrypted manner. The latter approach further relies on iO and subexponential security of some primitives or universal samplers [HJK+16] (which rely on iO and random oracles). As before, the aggregated message-key pairs must be distinct.

The to-be-aggregated signatures are outputs from a vanilla signature scheme, so multi-hop aggregation is not supported out of the box. At the same time, a potential multi-hop extension of [HKW15] seems plausible: it seems natural to view aggregates as signatures on the set of messages $\{(\mathsf{pk}_1, m_1), \ldots, (\mathsf{pk}_n, m_n)\}$

and recursively aggregate such signatures using [HKW15] itself. After recursion, each m_i will be a system of nested sets (whose structure reflects the aggregation topology) which ultimately contain key-message pairs. Therefore this approach is not hint-free.

Multi-hop Aggregation via SNARKs. As we have discussed in Section 1, one can implement multi-hop aggregation via recursive SNARKs. This has been done in [KKZM23, DKKW25, KZHB25]. However, as discussed in Section 1, no theoretically-satisfying instantiation is known.

Aggregation via BARGs. It is possible to generically build signature aggregation via batch arguments (BARGs) [WW22]. A BARG allows to prove multiple instances of NP statements in a succinct manner. Specifically, a BARG for NP allows a prover to construct a proof π of ℓ NP statements such that the size of π scales sublinearly with ℓ. The aggregation procedure takes as input ℓ signature-key pairs (σ_i, π_i) on message m and an aggregate is a BARG proof π which testifies that each pair (σ_i, π_i) verifies for m. As opposed to the SNARK-based approach, BARGs are known from various standard assumptions (e.g., k-LIN [WW22] or LWE and QR [CJJ21]). If the BARG is rate-1 (i.e., the proof size is $\ell + o(\ell)$), then it is possible to compactly aggregate proofs [DGKV22], which gives rise to multi-hop aggregate signatures. However, the verification algorithm in [DGKV22] requires the aggregation topology as hint for the verifier.

Concurrent Work [BBPS26]. Recently, a preprint [BBPS26] provides two constructions of mergeable succinct non-interactive arguments (SNARGs), i.e., SNARGs with a merging procedure for proofs, in the standard model. That is, given n proofs on statements $x_1, \ldots, x_n$ in some language $\mathcal{L}$, it is possible to merge them into a single proof for some merged statement $x \in \mathcal{L}$ which itself is implied by $x_1, \ldots, x_n$. The first construction is from LWE but the proof size scales with the depth of the merge topology. The second construction does not scale with the depth and relies on subexponentially-secure iO. Their constructions work for so-called trapdoor languages, where the validity of a statement can be decided in polynomial time given a trapdoor. The authors of [BBPS26] show that mergeable SNARGs yield multi-hop aggregate signatures, where verification of aggregates does not require the full aggregation topology.

However, verification is *not entirely hint-free*. Concretely, verification takes as input an aggregate key-message pair $(\widehat{\mathsf{pk}, m})$ that is computed iteratively during the aggregation procedure. To verify an aggregate with respect to a signer set $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ and message m, the verifier must either recompute $(\widehat{\mathsf{pk}, m})$ itself, or obtain it from a third party. [6] In either case, the full aggregation topology is required: $(\widehat{\mathsf{pk}, m})$ depends on the topology, so the verifier or any third party computing it on the verifier's behalf must know the full topology. While this may be acceptable in applications where the aggregation topology is naturally available, we argue that it makes the primitive a poor fit for the proof-of-stake setting, where storing or communicating the topology is precisely the overhead one wishes to avoid.

1.4 Outline

After giving a technical overview and formal preliminaries in Section 2 and Section 3, respectively, we introduce the formal definition of hint-free multi-signatures in Section 4. In Section 5, we show feasibility of this new primitive in the standard model.

2 Technical Overview

In this section, we elaborate on why BLS multi-signatures require verification hints in the context of multi-hop aggregation, and we explain our approach to solving this issue, leading to our feasibility result for hint-free multi-signatures.

[6] We note that [BBPS26] enables authentication of individual message-key pairs within the aggregate $(\widehat{\mathsf{pk}, m})$ via an authentication path. However, individual verification is insufficient in the proof-of-stake scenario.

BLS Multi-Signatures. As a starting point, we recall BLS signatures and how to aggregate them. Consider a group $\mathbb{G}$ of prime order p with generator G, equipped with a pairing, and a hash function H mapping into $\mathbb{G}$. A secret key is a scalar $\mathsf{sk} \in \mathbb{Z}_p$, the corresponding public key is $\mathsf{pk} = \mathsf{sk} \cdot G \in \mathbb{G}$, and a signature on a message m is $\sigma = \mathsf{sk} \cdot \mathsf{H}(\mathsf{m}) \in \mathbb{G}$. We work with the proof-of-possession variant [RY07], in which each public key is accompanied by a proof of knowledge of the corresponding secret key[7].

Given a message m and signers with key pairs $(\mathsf{sk}_i, \mathsf{pk}_i = \mathsf{sk}_i \cdot G)$ for $i \in \mathcal{I}$ (where $\mathcal{I}$ denotes some index set), individual signatures are aggregated by simply adding them:

$$\sigma_{\mathcal{I}} = \sum_{i \in \mathcal{I}} \sigma_i = \left(\sum_{i \in \mathcal{I}} \mathsf{sk}_i \right) \mathsf{H}(\mathsf{m}).$$

Verification proceeds as for a standard BLS signature, using the aggregate public key $\mathsf{pk}_{\mathcal{I}} = \sum_{i \in \mathcal{I}} \mathsf{pk}_i = \left(\sum_{i \in \mathcal{I}} \mathsf{sk}_i \right) G$.

Multi-Hop Aggregation and Verification Hints. Now suppose we want to aggregate two signatures that are themselves already aggregates: given $\sigma_{\mathcal{I}}$ and $\sigma_{\mathcal{I}'}$ for two index sets $\mathcal{I}$ and $\mathcal{I}'$, the goal is to obtain a signature certifying the *union* $\mathcal{I} \cup \mathcal{I}'$ of signers. However, note what adding the two aggregates gives:

$$\begin{aligned} \sigma_{\mathcal{I}} + \sigma_{\mathcal{I}'} &= \left(\sum_{i \in \mathcal{I}} \mathsf{sk}_i \right) \mathsf{H}(\mathsf{m}) + \left(\sum_{i \in \mathcal{I}'} \mathsf{sk}_i \right) \mathsf{H}(\mathsf{m}) \\ &= \left(\sum_{i \in \mathcal{I} \cup \mathcal{I}'} \mathsf{sk}_i \right) \mathsf{H}(\mathsf{m}) + \left(\sum_{i \in \mathcal{I} \cap \mathcal{I}'} \mathsf{sk}_i \right) \mathsf{H}(\mathsf{m}). \end{aligned}$$

That is, the secret keys of signers in the intersection $\mathcal{I} \cap \mathcal{I}'$ contribute with multiplicity 2, and more generally, after several rounds of aggregation (or when aggregating more than two aggregates), each key may contribute with an arbitrary multiplicity. A verifier can only check such an aggregate if it knows these multiplicities, or equivalently, the *multi-set* of public keys rather than just their set. This is precisely the verification hint that we aim to eliminate.

The Normalization Approach. Our key idea to eliminate the need for multiplicities is to equip the scheme with a *normalization* procedure: a public algorithm that, given an aggregate whose keys may appear with multiplicities, produces an equivalent aggregate in which every key appears with multiplicity at most 1. For instance, one would want to publicly compute

$$(\mathsf{sk}_1 + 2\mathsf{sk}_2 + \mathsf{sk}_3)\mathsf{H}(\mathsf{m}) \longmapsto (\mathsf{sk}_1 + \mathsf{sk}_2 + \mathsf{sk}_3)\mathsf{H}(\mathsf{m}),$$

thereby reducing the multiplicity of sk_2 back to 1. If such a procedure exists, then after each aggregation step one can normalize the result, and the verifier only ever needs to know the *set* of signers.

Intuitively, however, it seems hard to implement this public normalization: it seems as if one needs the individual signature $\mathsf{sk}_2 \cdot \mathsf{H}(\mathsf{m})$ for that. Even more: given the input-output relation above, it is indeed possible to compute $\mathsf{sk}_2 \cdot \mathsf{H}(\mathsf{m})$. The main challenge is therefore to implement normalization in a secure way.

Implementation using Obfuscation. To understand how normalization can be implemented, consider first a thought experiment in which a trusted party is available. We include in each public key pk_i a ciphertext ct_i encrypting the secret key sk_i towards this party, together with a proof of well-formedness of this ciphertext. The trusted party can then perform normalization on demand: given $\sigma_{\mathcal{I}} + \sigma_{\mathcal{I}'}$, it decrypts the secret keys of the signers in the intersection $\mathcal{I} \cap \mathcal{I}'$ and subtracts their contributions:

$$\sigma_{\mathcal{I}} + \sigma_{\mathcal{I}'} - \sum_{i \in \mathcal{I} \cap \mathcal{I}'} \mathsf{sk}_i \cdot \mathsf{H}(\mathsf{m}) = \left(\sum_{i \in \mathcal{I} \cup \mathcal{I}'} \mathsf{sk}_i \right) \mathsf{H}(\mathsf{m}).$$

[7]The alternative variant using random coefficients derived from a random oracle [BDN18] appears to be fundamentally incompatible with hint-free multi-hop aggregation: the random coefficients depend on the set of public keys involved in a given aggregation step, and after re-aggregation the verifier would need to know the full aggregation topology in order to recompute them. We therefore do not consider this variant further.

This yields a correctly normalized aggregate over $\mathcal{I} \cup \mathcal{I}'$, as desired.

Of course, a trusted party is not available in practice. Our insight is that indistinguishability obfuscation (iO) allows us to replace it with a public *normalization circuit*, included in the public parameters of the scheme. This circuit takes as input two normalized aggregates, verifies them, adds them, identifies the intersection of public keys, decrypts the corresponding secret keys, and subtracts their contributions to normalize the result. Intuitively, obfuscation should ensure that the decryption key hardcoded in the circuit remains hidden, so that an adversary learns nothing about honest secret keys beyond what is available from the public keys themselves. Making this intuition precise while using iO and not black-box obfuscation, however, requires significant care, as we discuss next.

Security Model and Strategy. Our security model for hint-free multi-signatures follows the standard template for non-interactive multi-signatures. There is a single honest key pair $(\mathsf{sk}, \mathsf{pk})$, where the public key is $\mathsf{sk} \cdot G$ together with a ciphertext ct encrypting sk and a proof of well-formedness π. The adversary has access to a signing oracle and wins by producing a forgery: a fresh message m, a set of public keys containing pk, and a multi-signature σ that verifies with respect to m and the given key set.

A natural strategy to argue security would be to reduce to the single-signer unforgeability of BLS. Note that what makes this non-trivial is that there is additional information about sk contained in ct, which is not present for single-signer BLS. To reduce to BLS unforgeability, we would therefore first need to eliminate this extra information by switching ct to an encryption of a random value. Unfortunately, this cannot be done directly: the decryption key is hardcoded inside the obfuscated normalization circuit and is thus effectively leaked to the adversary.

Arguing Security. Our hope is that we can rely on the security of iO to solve this. Note that if black-box obfuscation were available, one could first turn the obfuscated circuit into an oracle and then use a CCA-style argument to switch ct to random. To achieve a similar effect from iO, we instead rely on puncturable encryption: an encryption scheme admitting a punctured decryption key that decrypts all ciphertexts except one, after which that ciphertext can be switched to random.

The remaining challenge is that we cannot simply replace the decryption key hardcoded in the normalization circuit by a punctured one: the two circuits would not be functionally equivalent. This is because it is unclear what the punctured key would output on ct. To resolve this, we first rewrite the normalization circuit so that, in the case where it would need to decrypt ct, it instead recovers the honest signer's contribution by subtracting the dishonest individual signatures from the input. We show that this rewritten circuit is functionally equivalent to the original, assuming that the proof of well-formedness is perfectly sound. With this equivalence in hand, we can then apply iO security to switch to the punctured decryption key, and subsequently switch ct to random. To finalize the proof, we also need to rely on a dual-mode feature of the proof of well-formedness, but we defer details to the technical body.

3 Preliminaries

We adopt standard cryptographic notation and terminology (probabilistic polynomial-time (PPT), negligible functions, security parameter denoted by λ). Negligible functions are denoted by $\mathsf{negl}(\lambda)$ for short. Every algorithm implicitly receives λ as input. For a deterministic algorithm Alg, we write $y := \mathsf{Alg}(x)$ to denote the output y obtained by executing Alg on input x. If Alg is randomized, the notation $y \leftarrow \mathsf{Alg}(x)$ indicates that y is produced by running Alg using freshly sampled uniform randomness ρ. When it is useful to make the randomness explicit, we write $y := \mathsf{Alg}(x; \rho)$, in which case the algorithm $\mathsf{Alg}(\cdot; \rho)$ behaves deterministically. The expression $y \in \mathsf{Alg}(x)$ means $\exists \rho\colon\ y = \mathsf{Alg}(x; \rho)$. Uniform sampling of an element from a finite set X is denoted by $x \xleftarrow{\$} X$, while sampling from a distribution $\mathcal{D}$ is written as $x \leftarrow \mathcal{D}$. We write $[r] := \{1, \ldots, r\} \subseteq \mathbb{N}$, and $\mathbb{Z}_q$ denotes the set $\{0, \ldots, q-1\}$, which we treat as a ring (or a field when q is prime).

3.1 Indistinguishability Obfuscation

We recall the definition of indistinguishability obfuscation [BGI+01, JLS21].

Definition 1 (Indistinguishability Obfuscation)**.** An indistinguishability obfuscator for a class of circuits $\mathcal{C}$ is a PPT algorithm iO that takes as input a circuit $\mathsf{C} \in \mathcal{C}$ and outputs an obfuscated circuit $\hat{\mathsf{C}}$, so that the following properties are satisfied:

- **Correctness.** For all inputs x, we have

$$\Pr\left[\mathsf{C}(x) = \hat{\mathsf{C}}(x) \mid \hat{\mathsf{C}} \leftarrow \mathsf{iO}(\mathsf{C})\right] = 1.$$

- **Security.** For every valid (see below) PPT algorithm $\mathcal{A}$, the following is negligible in λ:

$$\left|\Pr\left[\mathbf{G}_0^{\mathsf{iO},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{G}_1^{\mathsf{iO},\mathcal{A}}(\lambda) = 1\right]\right|,$$

  where the game $\mathbf{G}_b^{\mathsf{iO},\mathcal{A}}(\lambda)$ for $b \in \{0,1\}$ is defined as follows:

  1. Get $(\mathsf{C}_0, \mathsf{C}_1) \leftarrow \mathcal{A}(1^\lambda)$.
  2. Output $b' \leftarrow \mathcal{A}(\mathsf{iO}(\mathsf{C}_b))$.

  We say that a PPT algorithm $\mathcal{A}$ is valid if it only outputs $\mathsf{C}_0, \mathsf{C}_1 \in \mathcal{C}$ with $\mathsf{C}_0(x) = \mathsf{C}_1(x)$ for all x in the above game.

3.2 Groups

We work over bilinear groups of prime order p, which we write additively. That is, we consider pairings (non-degenerate, bilinear maps) of the form $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where G, G_T are generators for $\mathbb{G}, \mathbb{G}_T$, respectively. Note that one could as well consider asymmetric pairings, but for simplicity we stick to the symmetric setting. Note that all groups and pairings are assumed to be implicitly parameterized by the security parameter λ.

3.3 Signatures and Standard Model BLS

We recall the definition of a signature scheme, its existential unforgeability, and a result about BLS signatures [BLS01] in the standard model.

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

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) \to b \in \{0,1\}$ (deterministic)

We require the scheme to be correct according to the following definition: for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$, all $\mathsf{m} \in \mathcal{M}$, we have

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

Definition 3 (Unforgeability)**.** Let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a signature scheme. We say that SIG is existentially unforgeable under chosen message attacks (for short: unforgeable), if for any PPT algorithm $\mathcal{A}$, the probability that the following game outputs 1 is negligible (in λ):

1. Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$.
2. Run $(\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pk})$, where oracle O is defined as follows:
   - $\mathrm{O}(\mathsf{m})$: Add m to $\mathcal{Q}$ and return $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})$.
3. The game outputs 1 if and only if

$$\mathsf{m}^* \notin \mathcal{Q} \wedge \mathsf{Ver}\left(\mathsf{pk}, \mathsf{m}^*, \sigma^*\right) = 1.$$

Let $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a pairing with generators $G \in \mathbb{G}, G_T \in \mathbb{G}_T$ for cyclic groups of prime order p, and let $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$ be a hash function. Then the BLS signature scheme over e and H is defined as follows:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$:
  1. $x \xleftarrow{\$} \mathbb{Z}_p,\ X := xG$
  2. Return $(\mathsf{pk} := X, \mathsf{sk} := x)$
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$:
  1. Return $\sigma := x\mathsf{H}(\mathsf{m})$
- $\mathsf{Ver}(\mathsf{pk} = X, \mathsf{m}, \sigma) \to b \in \{0,1\}$:
  1. If $e(\sigma, G) = e\left(\mathsf{H}(\mathsf{m}), X\right)$: Return $b = 1$
  2. Else: Return $b = 0$

We recall a result from [HSW14], showing that we can instantiate BLS in the standard model.

Theorem 1 ([HSW14]). *Under the assumption that an indistinguishability obfuscator exists and that the Diffie-Hellman Inversion assumption holds in $\mathbb{G}$, there is a hash function H such that the BLS signature scheme above is unforgeable.*

3.4 Weak Tag-Based Puncturable Encryption

For our purposes, we will use a simplified notion of puncturable public-key encryption (PPE, [GM15]). PPE is a variant of tag-based public-key encryption [Kil06] which allows to *puncture* secret keys at an arbitrary set of tags tag_i, such that the punctured key decrypts all ciphertexts except those with one of the tags tag_i.[8] While [GM15] consider keys punctured at multiple tags (consecutively), our simplification is to consider only a single puncturing operation. In other words, we consider only secret keys that are punctured at a single tag tag. However, we note that for our purposes, it is important to have perfectly correct schemes and punctured keys (unlike with the definition from [GM15]).

Definition 4 (Puncturable Encryption)**.** A puncturable encryption scheme PKE with message space $\mathcal{M}$, tag space $\mathcal{T}$, and ciphertext space $\mathcal{C}$ is a tuple of PPT algorithms $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Puncture}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk})$
- $\mathsf{Puncture}(\mathsf{ek}, \mathsf{dk}, \mathsf{tag}) \to \mathsf{dk}_{\mathsf{tag}}$
- $\mathsf{Enc}(\mathsf{ek}, m, \mathsf{tag}) \to c \in \mathcal{C}$, written as $\mathsf{Enc}(\mathsf{ek}, m, \mathsf{tag}; \rho)$ for explicit random coins ρ.
- $\mathsf{Dec}(\mathsf{dk}, c, \mathsf{tag}) \to m$ (deterministic)

We require the scheme to be correct according to the following definitions, which should hold for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$:

- **Correctness.** For all $m \in \mathcal{M}$, and all $\mathsf{tag} \in \mathcal{T}$, we have
$$\Pr\left[\mathsf{Dec}(\mathsf{dk}, c, \mathsf{tag}) = m \mid c \leftarrow \mathsf{Enc}(\mathsf{ek}, m, \mathsf{tag})\right] = 1.$$
- **Punctured Correctness.** For all $c \in \mathcal{C}$, all $\mathsf{tag}^*, \mathsf{tag} \in \mathcal{T}$ with $\mathsf{tag}^* \neq \mathsf{tag}$, and for all $\mathsf{dk}_{\mathsf{tag}^*} \in \mathsf{Puncture}(\mathsf{ek}, \mathsf{dk}, \mathsf{tag}^*)$, we have
$$\mathsf{Dec}(\mathsf{dk}_{\mathsf{tag}^*}, c, \mathsf{tag}) = \mathsf{Dec}(\mathsf{dk}, c, \mathsf{tag}).$$

[8] See also [CHN+16] for a slightly different and non-tag-based definition of puncturable public-key encryption.

Definition 5 (Punctured Indistinguishability)**.** Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Puncture}, \mathsf{Enc}, \mathsf{Dec})$ be a puncturable encryption scheme with message space $\mathcal{M}$ and tag space $\mathcal{T}$. Let $0 \in \mathcal{M}$ denote some default element in the message space. We say that PKE satisfies punctured indistinguishability, if for any PPT algorithm $\mathcal{A}$, the following is negligible (in λ):

$$\left|\Pr\left[\mathbf{G}_0^{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathbf{G}_1^{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right]\right|,$$

where the game $\mathbf{G}_b^{\mathsf{PKE},\mathcal{A}}(\lambda)$ for $b \in \{0,1\}$ is defined as follows:

1. Run $(St, m, \mathsf{tag}) \leftarrow \mathcal{A}(1^\lambda)$.
2. Generate $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\tilde{\mathsf{dk}} \leftarrow \mathsf{PKE.Puncture}(\mathsf{ek}, \mathsf{dk}, \mathsf{tag})$.
3. Set $c \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, m, \mathsf{tag})$ if $b = 0$ and $c \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, 0, \mathsf{tag})$ if $b = 1$.
4. Output $b' \leftarrow \mathcal{A}(St, \mathsf{ek}, c, \tilde{\mathsf{dk}})$.

We remark that puncturable encryption as defined above (and in fact even with multiply punctured keys as in [GM15]) can be constructed generically from any selectively secure hierarchical identity-based encryption (HIBE) scheme, see [GHJL17].[9] Selectively secure HIBE schemes can be generically constructed from selectively secure IBE schemes [DG17a], and thus from a variety of assumptions (e.g., [CHK03, CHKP10, ABB10, DG17b][10]). Specifically, a construction from the Computational Diffie-Hellman assumption is possible (by combining [DG17b, DG17a]), which is implied by the Diffie-Hellman Inversion assumption used in Theorem 1.

3.5 Dual-Mode Non-Interactive Zero-Knowledge

For the following definition, recall that an **NP** relation is a binary relation containing statement-witness pairs that is efficiently decidable. We define the corresponding language as

$$\mathcal{L}_{\mathcal{R}} = \{\mathsf{stmt} \mid \exists\ \mathsf{witn}\ \text{s.t.}\ (\mathsf{stmt}, \mathsf{witn}) \in \mathcal{R}\}.$$

Definition 6 (Non-Interactive Proof System)**.** Let $\mathcal{R}$ be an **NP** relation. A non-interactive proof system PS for $\mathcal{R}$ is a tuple of PPT algorithms $\mathsf{PS} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$
- $\mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{witn}) \to \pi$
- $\mathsf{Ver}(\mathsf{crs}, \mathsf{stmt}, \pi) \to b \in \{0,1\}$ (deterministic)

We require the scheme to be correct according to the following definition: For every $(\mathsf{stmt}, \mathsf{witn}) \in \mathcal{R}$, and every $\mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, we have

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1 \mid \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{witn})\right] = 1.$$

Definition 7 (Dual-Mode NIZK)**.** Let $\mathcal{R}$ be an **NP** relation and PS be a non-interactive proof system for $\mathcal{R}$. We say that PS is a dual-mode NIZK if there are PPT algorithms $\mathsf{SimSetup}, \mathsf{SimProve}$ such that the following properties hold:

- **Perfect Soundness.** For any (not necessarily efficiently computable) function φ, we have

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1 \wedge \mathsf{stmt} \notin \mathcal{L}_{\mathcal{R}} \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{stmt}, \pi) := \varphi(\mathsf{crs})\right] = 0.$$

- **CRS Indistinguishability.** For any PPT algorithm $\mathcal{A}$, the following is negligible (in λ):

$$\left|\Pr\left[\mathcal{A}(\mathsf{crs}) = 1 \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] - \Pr\left[\mathcal{A}(\mathsf{crs}) = 1 \mid (\mathsf{crs}, \mathsf{tr}) \leftarrow \mathsf{SimSetup}(1^\lambda)\right]\right|.$$

[9] While the PPE construction is not explicit in [GHJL17], its idea is simple and elegant: PPE tags correspond to (multi-level) HIBE identities, and a punctured decryption key consists of all HIBE decryption keys for identities on the co-path to the punctured tag. Conceptually, this follows the construction of forward-secure encryption [CHK03], and more generally broadcast encryption [FN94].

[10] Since we need perfect correctness, parameters in the lattice-based constructions [CHKP10, ABB10] have to be chosen aggressively, such that decryption becomes error-free.

- **Zero-Knowledge.** For every $(\mathsf{stmt}, \mathsf{witn}) \in \mathcal{R}$, and any PPT algorithm $\mathcal{A}$, the following is negligible (in λ):

$$\Big|\Pr\left[\mathcal{A}(\mathsf{crs}, \mathsf{tr}, \mathsf{stmt}, \pi) = 1 \mid (\mathsf{crs}, \mathsf{tr}) \leftarrow \mathsf{SimSetup}(1^\lambda),\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{witn})\right]$$
$$-\Pr\left[\mathcal{A}(\mathsf{crs}, \mathsf{tr}, \mathsf{stmt}, \pi) = 1 \mid (\mathsf{crs}, \mathsf{tr}) \leftarrow \mathsf{SimSetup}(1^\lambda),\ \pi \leftarrow \mathsf{SimProve}(\mathsf{crs}, \mathsf{stmt}, \mathsf{tr})\right]\Big|.$$

Dual-mode NIZKs can be instantiated by Groth-Sahai proofs [GS08], e.g., based on the Matrix Diffie-Hellman assumption [EHK+13].

4 Hint-Free Multi-Signatures

We now give the formal definition of hint-free multi-signatures. Essentially, like a (non-interactive) multi-signature scheme it allows to aggregate signatures. In addition, it allows to aggregate aggregates. That is, the aggregation algorithm takes as input sets $\mathcal{S}_i$ of public keys, in combination with signatures σ_i for those sets, and outputs an aggregate signature for the union of these sets. We call the primitive *hint-free* as the verification algorithm does not take as input any additional information beyond the *set* of public keys. In particular, neither a multi-set (i.e., multplicities) is needed, nor the aggregation topology.

Definition 8 (Hint-Free Multi-Signatures)**.** A hint-free multi-signature scheme with message space $\mathcal{M}$ is a tuple of PPT algorithms $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Aggregate}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$; we assume that all algorithms implicitly get par as input.
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$
- $\mathsf{Aggregate}((\mathcal{S}_1, \sigma_1), \ldots, (\mathcal{S}_\ell, \sigma_\ell), \mathsf{m}) \to \sigma$ (deterministic); notably, σ_i may itself be an output of $\mathsf{Aggregate}$.
- $\mathsf{Ver}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{m}, \sigma) \to b \in \{0, 1\}$ (deterministic). notably, $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ is given as a set.

We require the scheme to be correct according to the following two definitions, which should hold for any $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$ and $\mathsf{m} \in \mathcal{M}$:

- **Signing Correctness.** For all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\mathsf{par})$, we have
$$\Pr\left[\mathsf{Ver}(\{\mathsf{pk}\}, \mathsf{m}, \sigma) \mid \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})\right] = 1.$$
- **Aggregation Correctness.** For all $\ell = \mathsf{poly}(\lambda)$, all sets of public keys $\mathcal{S}_i = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_{n_i}\}, i \in [\ell]$, and all signatures $\sigma_i, i \in [\ell]$, we have
$$\forall i \in [\ell] : \mathsf{Ver}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_{n_i}\}, \mathsf{m}, \sigma_i) = 1$$
$$\implies \mathsf{Ver}\left(\bigcup_{i \in [\ell]} \mathcal{S}_i, \mathsf{m}, \sigma\right) = 1 \text{ for } \sigma := \mathsf{Aggregate}((\mathcal{S}_1, \sigma_1), \ldots, (\mathcal{S}_\ell, \sigma_\ell), \mathsf{m}).$$
Notably, the union here is a union of sets.

Typically, we want the scheme to be non-trivial, in the sense that the aggregate is shorter than the list of individual signatures.

Remark 1. In our construction (cf. Section 5), aggregation is only supported as long as the size of the union of the public key sets does not exceed a certain fixed polynomial bound. While this may be a restriction in some applications, it does not trivialize the problem, and it is without loss of generality for our primary application: in PoS blockchains, the universe of potential signers is fixed and publicly known, so the bound is simply set large enough to accommodate this universe. We also note that this restriction does not limit the *depth* of the aggregation topology—arbitrarily deep re-aggregation of aggregates remains possible, as long as the total number of distinct public keys involved stays within the bound. A similar restriction appears in one of the constructions in [HKW15].

We now define unforgeability, which is defined as for regular (non-interactive) multi-signatures: there is one honest signer and the adversary sees its public key. It also gets a signing oracle. Finally, it has to output an (potentially aggregated) signature for a set involving the honest public key and on a fresh message.

Definition 9 (Unforgeability)**.** Let $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Aggregate}, \mathsf{Ver})$ be a hint-free multi-signature scheme. We say that MS is existentially unforgeable under chosen message attacks (for short: unforgeable), if for any PPT algorithm $\mathcal{A}$, the probability that the following game outputs 1 is negligible (in λ):

1. Generate $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$.
2. Run $(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{par}, \mathsf{pk})$, where oracle O is defined as follows:
   - $\mathrm{O}(\mathsf{m})$: Add m to $\mathcal{Q}$ and return $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})$.
3. The game outputs 1 if and only if

$$\mathsf{m}^* \notin \mathcal{Q} \wedge \mathsf{pk} \in \mathcal{S}^* \wedge \mathsf{Ver}\left(\mathcal{S}^*, \mathsf{m}^*, \sigma^*\right) = 1.$$

5 Hint-Free Multi-Signatures without Proof Recursion

We now show how to construct hint-free multi-signatures in the standard model, without relying on proof recursion.

5.1 Construction

Assume the following are given:

- Message space $\mathcal{M}$.
- Pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with generators $G \in \mathbb{G}, G_T \in \mathbb{G}$ for cyclic groups of prime order p.
- Puncturable encryption $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Puncture}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, with tag space $\mathbb{G}$ and message space $\mathbb{Z}_p$, and ciphertext space $\mathcal{C}$.
- A hash $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$, in the standard model (see Theorem 1).
- Dual-mode NIZK proof $\mathsf{PS} = (\mathsf{PS.Setup}, \mathsf{PS.Prove}, \mathsf{PS.Ver})$ for the relation

$$\mathcal{R} = \{(\mathsf{stmt}, \mathsf{witn}) \mid \mathsf{stmt} = (G, X, \mathsf{ek}, c), \mathsf{witn} = (x, \rho), X = xG, c = \mathsf{PKE.Enc}(\mathsf{ek}, x, X; \rho)\}\,.$$

- Indistinguishability obfuscator iO for a circuit class that involves the circuits in this section. We assume that all circuits are appropriately padded to have equal size / fall into this circuit class.

We construct a hint-free multi-signature scheme MS below. Note that we only define $\mathsf{Aggregate}$ for pairs of (aggregate) signatures, which is sufficient, as one can apply it repeatedly.

- $\mathsf{MS.Setup}(1^\lambda) \to \mathsf{par}$:
  1. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$
  2. $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\lambda)$
  3. $\hat{\mathsf{C}} := \mathsf{iO}(\mathsf{C}^{\mathsf{dk}})$ with appropriate padding, for the circuit C^{dk} defined below
  4. $\mathsf{par} = (e, G, G_T, \mathsf{H}, \mathsf{crs}, \hat{\mathsf{C}}, \mathsf{ek})$
- $\mathsf{MS.Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$:
  1. $x \xleftarrow{\$} \mathbb{Z}_p,\ X := xG$
  2. $c := \mathsf{PKE.Enc}(\mathsf{ek}, x, X; \rho)$ for random coins ρ, where X is the tag
  3. $\mathsf{stmt} := (G, X, \mathsf{ek}, c), \mathsf{witn} := (x, \rho)$

4. $\pi \leftarrow \mathsf{PS.Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{witn})$
5. Return $(\mathsf{pk} := (X, c, \pi), \mathsf{sk} := x)$

- $\mathsf{MS.Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$:
  1. Return $\sigma := x\mathsf{H}(\mathsf{m})$
- $\mathsf{MS.Aggregate}((\mathcal{S}, \sigma), (\mathcal{S}', \sigma'), \mathsf{m}) \to \sigma$:
  1. Return $\hat{\mathsf{C}}((\mathcal{S}, \sigma), (\mathcal{S}', \sigma'), \mathsf{m})$
- $\mathsf{MS.Ver}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{m}, \sigma) \to b \in \{0, 1\}$:
  1. For $i \in [n]$: parse $\mathsf{pk}_i = (X_i, c_i, \pi_i)$, set $\mathsf{stmt}_i := (G, X_i, \mathsf{ek}, c_i)$.
  2. If $\exists i \in [n]$ s.t. $\mathsf{PS.Ver}(\mathsf{crs}, \mathsf{stmt}_i, \pi_i) = 0$: Return $b = 0$
  3. If $e(\sigma, G) = e\left(\mathsf{H}(\mathsf{m}), \sum_{i=1}^{n} X_i\right)$: Return $b = 1$
  4. Else: Return $b = 0$

Remark 2. Note that $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ is treated as a set (i.e., no $i \neq j$ exist with $\mathsf{pk}_i = \mathsf{pk}_j$). However, there could exist $i \neq j$ with $X_i = X_j$, in which case this X_i appears (at least) twice in the sum.

The circuit C^{dk} is defined as follows:

- $\mathsf{C}^{\mathsf{dk}}((\mathcal{S}, \sigma), (\mathcal{S}', \sigma'), \mathsf{m}) \to \sigma_{\mathrm{agg}}$:
  1. If $|\mathcal{S}| \geq p$ or $|\mathcal{S}'| \geq p$, return $\perp$.
  2. If $\mathsf{MS.Ver}(\mathcal{S}, \mathsf{m}, \sigma) = 0$ or $\mathsf{MS.Ver}(\mathcal{S}', \mathsf{m}, \sigma') = 0$, return $\perp$.
  3. $\sigma_{\mathrm{agg}} := \sigma + \sigma' \in \mathbb{G}$
  4. $\mathcal{S}^* := \mathcal{S} \cap \mathcal{S}'$
  5. For all $(X_i, c_i, \pi_i) \in \mathcal{S}^*$:
     (a) $\mathsf{tag}_i := X_i$
     (b) $x_i := \mathsf{PKE.Dec}(\mathsf{dk}, c_i, \mathsf{tag}_i)$
     (c) $\sigma_i := x_i\mathsf{H}(\mathsf{m})$
     (d) $\sigma_{\mathrm{agg}} := \sigma_{\mathrm{agg}} - \sigma_i$
  6. Return σ_{agg}

Note that not only dk, but also par is hardcoded in C^{dk}. Correctness follows by inspection. It relies on correctness of all building blocks. Moreover, aggregation correctness (cf. Definition 8) also relies on the soundness of PS.

Remark 3. Since aggregation is realized via an obfuscated circuit, we require that the size of the public key set is bounded by a fixed polynomial, so that the circuit can be fixed at setup time. As discussed in Remark 1, this is not a restriction in the PoS setting, where the (size of the) signer universe is known in advance. We further note that this limitation is in principle not inherent: by instead obfuscating Turing machines, one could support aggregation over sets of arbitrary and unbounded size, at the cost of relying on obfuscation for Turing machines [KLW15, GS18] rather than circuits. A similar restriction appears in one of the constructions in [HKW15].

5.2 Security

We now show security of our construction.

Theorem 2. *If the assumptions from Theorem 1 hold,* PKE *satisfies punctured indistinguishability (Definition 5),* PS *is a dual-mode NIZK (Definition 7) for $\mathcal{R}$, and* iO *is a secure indistinguishability obfuscator (Definition 1), then* MS *as constructed in Section 5.1 is unforgeable (cf. Definition 9).*

Before we prove Theorem 2, we state and prove a useful lemma.

	crs	Dec Key	Circuit	π	c	On Forgery
$\mathcal{G}_0$	Sound	Real	$\mathsf{iO}(\mathsf{C}^{\mathsf{dk}})$	Hon	$\mathsf{Enc}(\mathsf{ek}, x, X)$	-
$\mathcal{G}_1$	Sound	Real	$\mathsf{iO}(\mathsf{C}^{\mathsf{dk}})$	Hon	$\mathsf{Enc}(\mathsf{ek}, x, X)$	$\mathsf{ExtHonSig}_{X,\mathsf{dk}}$ + abort
$\mathcal{G}_2$	Sound	Real	$\mathsf{iO}(\mathsf{D}^{\mathsf{dk},X})$	Hon	$\mathsf{Enc}(\mathsf{ek}, x, X)$	$\mathsf{ExtHonSig}_{X,\mathsf{dk}}$ + abort
$\mathcal{G}_3$	Sound	Punctured	$\mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X,X})$	Hon	$\mathsf{Enc}(\mathsf{ek}, x, X)$	$\mathsf{ExtHonSig}_{X,\mathsf{dk}_X}$ + abort
$\mathcal{G}_4$	Sim/ZK	Punctured	$\mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X,X})$	Hon	$\mathsf{Enc}(\mathsf{ek}, x, X)$	$\mathsf{ExtHonSig}_{X,\mathsf{dk}_X}$ + abort
$\mathcal{G}_5$	Sim/ZK	Punctured	$\mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X,X})$	Sim	$\mathsf{Enc}(\mathsf{ek}, x, X)$	$\mathsf{ExtHonSig}_{X,\mathsf{dk}_X}$ + abort
$\mathcal{G}_6$	Sim/ZK	Punctured	$\mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X,X})$	Sim	$\mathsf{Enc}(\mathsf{ek}, 0, X)$	$\mathsf{ExtHonSig}_{X,\mathsf{dk}_X}$ + abort

Table 1: Overview of the hybrids in the proof of Theorem 2.

Lemma 1 (Honest Signature Extraction in Sound Mode)**.** *Fix any parameters* $\mathsf{par} = (e, G, G_T, \mathsf{H}, \mathsf{crs}, \hat{\mathsf{C}}, \mathsf{ek})$ *with* $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$ *and* $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\lambda)$*, and an honestly generated* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{MS.Gen}(\mathsf{par})$*, with* $\mathsf{pk} = (X, c, \pi)$ *and* $X = xG$*. Consider the following algorithm* $\mathsf{ExtHonSig}_{X,\mathsf{dk}}(\mathcal{S}, \mathsf{m}, \sigma)$*:*

1. *If* $|\mathcal{S}| \geq p$*, return* $\perp$
2. $\hat{\mathcal{S}} := \{(X_j, c_j, \pi_j) \in \mathcal{S} \mid X_j \neq X\} \subseteq \mathcal{S}$
3. *Return* $\left(\sigma - \sum_{(X_j,c_j,\pi_j)\in\hat{\mathcal{S}}} \mathsf{PKE.Dec}(\mathsf{dk}, c_j, X_j)\mathsf{H}(\mathsf{m})\right) \Big/ |\mathcal{S} \setminus \hat{\mathcal{S}}|$

Note that the division is well-defined in $\mathbb{Z}_p$*. Then, the following implication holds (assuming* $|\mathcal{S}| < p$*):*

$$\left(\mathsf{MS.Ver}(\mathcal{S}, \mathsf{m}, \sigma) = 1 \wedge \hat{\mathcal{S}} \neq \mathcal{S}\right) \implies \mathsf{ExtHonSig}_{X,\mathsf{dk}}(\mathcal{S}, \mathsf{m}, \sigma) = x\mathsf{H}(\mathsf{m}).$$

Proof. By verification and the pairing being non-degenerate, we have $\sigma = \sum_{(X_j,c_j,\pi_j)\in\mathcal{S}} x_j\mathsf{H}(\mathsf{m})$ for $X_j = x_jG$. Splitting the sum, we get

$$\begin{aligned}\sigma &= \sum_{(X_j,c_j,\pi_j)\in\mathcal{S}} x_j\mathsf{H}(\mathsf{m}) \\ &= \sum_{(X_j,c_j,\pi_j)\in\hat{\mathcal{S}}} x_j\mathsf{H}(\mathsf{m}) + \sum_{(X_j,c_j,\pi_j)\in\mathcal{S}\setminus\hat{\mathcal{S}}} x_j\mathsf{H}(\mathsf{m}).\end{aligned}$$

By definition of $\hat{\mathcal{S}}$, the second term can be written as

$$\sum_{(X_j,c_j,\pi_j)\in\mathcal{S}\setminus\hat{\mathcal{S}}} x_j\mathsf{H}(\mathsf{m}) = \sum_{(X_j,c_j,\pi_j)\in\mathcal{S}\setminus\hat{\mathcal{S}}} x\mathsf{H}(\mathsf{m}) = |\mathcal{S} \setminus \hat{\mathcal{S}}| \cdot x\mathsf{H}(\mathsf{m}).$$

In combination, we get

$$\sigma = \sum_{(X_j,c_j,\pi_j)\in\hat{\mathcal{S}}} x_j\mathsf{H}(\mathsf{m}) + |\mathcal{S} \setminus \hat{\mathcal{S}}| \cdot x\mathsf{H}(\mathsf{m}).$$

Rearranging and using $\hat{\mathcal{S}} \neq \mathcal{S}$ yields

$$x\mathsf{H}(\mathsf{m}) = \left(\sigma - \sum_{(X_j,c_j,\pi_j)\in\hat{\mathcal{S}}} x_j\mathsf{H}(\mathsf{m})\right) \Big/ |\mathcal{S} \setminus \hat{\mathcal{S}}|.$$

Now, the claim follows if we can argue that $x_j = \mathsf{PKE.Dec}(\mathsf{dk}, c_j, X_j)$ for all $(X_j, c_j, \pi_j) \in \hat{\mathcal{S}}$. This follows directly from the perfect correctness of PKE, definition of $\mathcal{R}$, perfect soundness of PS, and the fact that π_j is verified in $\mathsf{MS.Ver}(\mathcal{S}, \mathsf{m}, \sigma)$. □

Proof of Theorem 2. Let $\mathcal{A}$ be any PPT adversary against unforgeability of MS. We prove the statements via a sequence of hybrid games $\mathcal{G}_i$, where we denote the probability that the i-th game $\mathcal{G}_i$ outputs 1 by ε_i. Our goal will be to bound ε_0. We refer to Table 1 for an overview of the hybrids.

Game $\mathcal{G}_0$ (Real Game). This is the real unforgeability game, as defined in Definition 9, for multi-signature scheme MS. It can be written as follows:

1. *Key Generation (Part 1).* $x \xleftarrow{\$} \mathbb{Z}_p,\ X := xG$
2. *Setup.*
   (a) $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\lambda)$, $\hat{\mathsf{C}} := \mathsf{iO}(\mathsf{C}^{\mathsf{dk}})$ (with padding)
   (b) $\mathsf{par} := (e, G, G_T, \mathsf{H}, \mathsf{crs}, \hat{\mathsf{C}}, \mathsf{ek})$
3. *Key Generation (Part 2).*
   (a) $c := \mathsf{PKE.Enc}(\mathsf{ek}, x, X; \rho)$ for random coins ρ
   (b) $\mathsf{stmt} := (G, X, \mathsf{ek}, c), \mathsf{witn} := (x, \rho),\ \ \pi \leftarrow \mathsf{PS.Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{witn})$
   (c) $\mathsf{pk} := (X, c, \pi)$
4. *Run Adversary.* $(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{par}, \mathsf{pk})$, where $\mathrm{O}(\mathsf{m})$ adds m to $\mathcal{Q}$ and returns $\sigma := x\mathsf{H}(\mathsf{m})$
5. *Winning Condition.*
   (a) If $\mathsf{m}^* \in \mathcal{Q} \vee \mathsf{pk} \notin \mathcal{S}^* \vee \mathsf{Ver}(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) = 0$, return 0.
   (b) Else, return 1.

Our goal is to bound ε_0, which is the probability that this game outputs 1.

Game $\mathcal{G}_1$ (Extract Individual Signature from Forgery). We now run $\mathsf{ExtHonSig}_{X,\mathsf{dk}}$ on the forgery to extract an individual BLS signature from it. If this is not a valid BLS signature, we let the game return 0. Only the winning condition phase changes:

5. *Winning Condition.*
   (a) If $\mathsf{m}^* \in \mathcal{Q} \vee \mathsf{pk} \notin \mathcal{S}^* \vee \mathsf{Ver}(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) = 0$, return 0.
   (b) $\sigma_{\text{ind}} := \mathsf{ExtHonSig}_{X,\mathsf{dk}}(\mathcal{S}^*, \mathsf{m}^*, \sigma^*)$
   (c) If $e(\sigma_{\text{ind}}, G) \neq e(\mathsf{H}(\mathsf{m}^*), X)$, return 0.
   (d) Else, return 1.

We argue that this does not change the probability of winning. To see this, note that the winning probability can only decrease here, as we add an additional condition to return 0. Thus, $\varepsilon_0 \geq \varepsilon_1$. On the other hand, note that if $\mathcal{G}_0$ outputs 1, then $\mathsf{pk} \in \mathcal{S}^*$ and $\mathsf{Ver}(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) = 1$, which implies that the two preconditions in the implication in Lemma 1 are satisfied[11]. The lemma then implies that $\sigma_{\text{ind}} = x\mathsf{H}(\mathsf{m}^*)$, which means that $\mathcal{G}_1$ outputs 1 as well. Thus $\varepsilon_0 \leq \varepsilon_1$. We get $\varepsilon_0 = \varepsilon_1$.

Game $\mathcal{G}_2$ (Change Circuit). We change the circuit that is obfuscated, so that we no longer use dk for ciphertexts with $\mathsf{tag} = X$. Concretely, we now compute $\hat{\mathsf{C}} := \mathsf{iO}(\mathsf{D}^{\mathsf{dk},X})$ in the setup phase, where

- $\mathsf{D}^{\mathsf{dk},X}((\mathcal{S}, \sigma), (\mathcal{S}', \sigma'), \mathsf{m}) \to \sigma_{\text{agg}}$:
  1. If $|\mathcal{S}| \geq p$ or $|\mathcal{S}'| \geq p$, return $\perp$.
  2. If $\mathsf{MS.Ver}(\mathcal{S}, \mathsf{m}, \sigma) = 0$ or $\mathsf{MS.Ver}(\mathcal{S}', \mathsf{m}, \sigma') = 0$, return $\perp$.
  3. $\sigma_{\text{agg}} := \sigma + \sigma' \in \mathbb{G}$
  4. $\mathcal{S}^* := \mathcal{S} \cap \mathcal{S}'$
  5. For all $(X_i, c_i, \pi_i) \in \mathcal{S}^*$ with $X_i \neq X$:
     (a) $\mathsf{tag}_i := X_i$
     (b) $x_i := \mathsf{PKE.Dec}(\mathsf{dk}, c_i, \mathsf{tag}_i)$
     (c) $\sigma_i := x_i\mathsf{H}(\mathsf{m})$
     (d) $\sigma_{\text{agg}} := \sigma_{\text{agg}} - \sigma_i$
  6. For all $(X_i, c_i, \pi_i) \in \mathcal{S}^*$ with $X_i = X$:
     (a) $\sigma_i := \mathsf{ExtHonSig}_{X,\mathsf{dk}}(\mathcal{S}, \mathsf{m}, \sigma)$

[11] Clearly, $|\mathcal{S}^*| < p$ also holds, as we consider a polynomial time adversary.

(b) $\sigma_{\text{agg}} := \sigma_{\text{agg}} - \sigma_i$

7. Return σ_{agg}

To argue that both hybrids are indistinguishable, we use the security of iO. To do that, all that is left to do is to show that the circuits C^{dk} and $\mathsf{D}^{\mathsf{dk},X}$ are functionally equivalent. This can be seen by considering two cases: if the input is such that no $(X_i, c_i, \pi_i) \in \mathcal{S}^*$ satisfies $X_i = X$, then clearly both circuits compute the same (the second loop in $\mathsf{D}^{\mathsf{dk},X}$ is not run, and the first is exactly as in C^{dk}). In the other case, it follows from Lemma 1 that the σ_i computed in the second loop in $\mathsf{D}^{\mathsf{dk},X}$ is the same as the σ_i that C^{dk} would compute in its loop. We can apply this lemma because:

- both circuits initially check that the signatures are valid,
- we have $\hat{\mathcal{S}} \neq \mathcal{S}$, using the notation from Lemma 1. This is because $(X, c_i, \pi_i) \in \mathcal{S}^* \subseteq \mathcal{S}$ and $(X, c_i, \pi_i) \notin \hat{\mathcal{S}}$ by definition of $\hat{\mathcal{S}}$.

Via security of iO, we get $\varepsilon_1 \leq \varepsilon_2 + \mathsf{negl}(\lambda)$.

Game $\mathcal{G}_3$ (Compute Punctured Key). We now change the decryption key that we use: we use a key that is punctured on the tag X. We do so for both the obfuscated circuit and for running $\mathsf{ExtHonSig}$ in the winning condition:

1. *Key Generation (Part 1).* $x \xleftarrow{\$} \mathbb{Z}_p,\ X := xG$
2. *Setup.*
   (a) $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, $\mathsf{crs} \leftarrow \mathsf{PS.Setup}(1^\lambda)$
   (b) $\mathsf{dk}_X \leftarrow \mathsf{PKE.Puncture}(\mathsf{ek}, \mathsf{dk}, X),\ \hat{\mathsf{C}} := \mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X, X})$ (with padding)
   (c) $\mathsf{par} := (e, G, G_T, \mathsf{H}, \mathsf{crs}, \hat{\mathsf{C}}, \mathsf{ek})$
3. *Key Generation (Part 2).*
   (a) $c := \mathsf{PKE.Enc}(\mathsf{ek}, x, X; \rho)$ for random coins ρ
   (b) $\mathsf{stmt} := (G, X, \mathsf{ek}, c), \mathsf{witn} := (x, \rho),\ \ \pi \leftarrow \mathsf{PS.Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{witn})$
   (c) $\mathsf{pk} := (X, c, \pi)$
4. *Run Adversary.* $(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{par}, \mathsf{pk})$, where $\mathrm{O}(\mathsf{m})$ adds m to $\mathcal{Q}$ and returns $\sigma := x\mathsf{H}(\mathsf{m})$
5. *Winning Condition.*
   (a) If $\mathsf{m}^* \in \mathcal{Q} \vee \mathsf{pk} \notin \mathcal{S}^* \vee \mathsf{Ver}(\mathcal{S}^*, \mathsf{m}^*, \sigma^*) = 0$, return 0.
   (b) $\sigma_{\text{ind}} := \mathsf{ExtHonSig}_{X, \mathsf{dk}_X}(\mathcal{S}^*, \mathsf{m}^*, \sigma^*)$
   (c) If $(\sigma_{\text{ind}}, G) \neq e(\mathsf{H}(\mathsf{m}^*), X)$, return 0.
   (d) Else, return 1.

Note that $\mathsf{ExtHonSig}_{X,\cdot}(\cdot)$ never runs Dec on tag X. Therefore, $\mathsf{ExtHonSig}_{X,\mathsf{dk}}(\cdot)$ and $\mathsf{ExtHonSig}_{X,\mathsf{dk}_X}(\cdot)$ are functionally equivalent by punctured correctness of PKE. Consequently, circuits $\mathsf{D}^{\mathsf{dk},X}$ and $\mathsf{D}^{\mathsf{dk}_X,X}$ are also functionally equivalent. Thus, the indistinguishability of the two hybrids follows from security of iO. We get $\varepsilon_2 \leq \varepsilon_3 + \mathsf{negl}(\lambda)$.

Game $\mathcal{G}_4$ (Switch to ZK Mode). We change how crs is generated, now using the $\mathsf{SimSetup}$ algorithm (cf. Definition 7):

2. *Setup.*
   (a) $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, $(\mathsf{crs}, \mathsf{tr}) \leftarrow \mathsf{PS.SimSetup}(1^\lambda)$
   (b) $\mathsf{dk}_X \leftarrow \mathsf{PKE.Puncture}(\mathsf{ek}, \mathsf{dk}, X),\ \hat{\mathsf{C}} := \mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X, X})$ (with padding)
   (c) $\mathsf{par} := (e, G, G_T, \mathsf{H}, \mathsf{crs}, \hat{\mathsf{C}}, \mathsf{ek})$

Indistinguishability follows via a simple reduction to CRS indistinguishability of PS. We get $\varepsilon_3 \leq \varepsilon_4 + \mathsf{negl}(\lambda)$.

Game $\mathcal{G}_5$ (Simulate Proof). We change how the proof π is generated, so that the witness is no longer needed. This eliminates information about x and ρ from π:

3. *Key Generation (Part 2).*
   (a) $c := \mathsf{PKE.Enc}(\mathsf{ek}, x, X; \rho)$ for random coins ρ
   (b) $\mathsf{stmt} := (G, X, \mathsf{ek}, c)$, $\pi \leftarrow \mathsf{SimProve}(\mathsf{crs}, \mathsf{stmt}, \mathsf{tr})$
   (c) $\mathsf{pk} := (X, c, \pi)$

Indistinguishability follows via a simple reduction to Zero-Knowledge of PS. We get $\varepsilon_4 \leq \varepsilon_5 + \mathsf{negl}(\lambda)$.

Game $\mathcal{G}_6$ (Fake Ciphertext). We now eliminate information about x from the ciphertext c:

3. *Key Generation (Part 2).*
   (a) $c \leftarrow \mathsf{PKE.Enc}(\mathsf{ek}, 0, X)$
   (b) $\mathsf{stmt} := (G, X, \mathsf{ek}, c)$, $\pi \leftarrow \mathsf{SimProve}(\mathsf{crs}, \mathsf{stmt}, \mathsf{tr})$
   (c) $\mathsf{pk} := (X, c, \pi)$

Intuitively, we have removed information about the random coins ρ before, and we no longer use the full decryption key dk, but only the punctured key dk_X. Thus, indistinguishability follows from punctured indistinguishability of PKE (cf. Definition 5). More formally, we build a reduction as follows: The reduction runs the first part of key generation and outputs its state and the message x and a tag X to the punctured indistinguishability game. It gets the encryption key ek, the punctured key dk_X and a ciphertext c that either encrypts x or 0 under ek and tag X. It then runs the rest of the setup to generate crs, tr and the obfuscated circuit $\hat{\mathsf{C}} := \mathsf{iO}(\mathsf{D}^{\mathsf{dk}_X, X})$. It runs the adversary, providing oracle O honestly. It checks the winning condition using the adversary's outputs and dk_X, and outputs as its decision bit the result of the game. Clearly, the game is perfectly simulated, the reduction is PPT, and the distinguishing advantage of the reduction is that of the adversary. By punctured indistinguishability, we get $\varepsilon_5 \leq \varepsilon_6 + \mathsf{negl}(\lambda)$.

Reduction to BLS. Intuitively, we have now eliminated all information about x except $X = xG$ and the signing oracle O. We have also ensured that from the adversary's forgery, we can extract a fresh individual BLS signature under X. Therefore, we can easily build a reduction to the unforgeability of BLS (cf. Theorem 1). The reduction gets X as input and access to a signing oracle. It simulates the game for $\mathcal{A}$. This can be done without knowing x, by sampling the encryption key honestly and puncturing the decryption key on X, and by forwarding all signing oracle queries of $\mathcal{A}$ to the reduction's oracle. Finally, it outputs σ_{ind} (cf. $\mathcal{G}_1$). Clearly, the game is perfectly simulated and the reduction is PPT. Also, if $\mathcal{G}_6$ outputs 1, then by the change introduced in $\mathcal{G}_1$, σ_{ind} is a valid BLS signature on public key X and message m^*, which has not been signed. Thus the reduction breaks unforgeability of BLS. We get $\varepsilon_6 \leq \mathsf{negl}(\lambda)$. Combining everything, we get $\varepsilon_0 \leq \mathsf{negl}(\lambda)$. □

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010. (Cited on Page 11.)

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Berlin, Heidelberg, May 2004. (Cited on Page 5.)

[BBPS26] Zvika Brakerski, Maya Farber Brodsky, Omer Paneth, and Tomer Solomon. Mergeable SNARGs for trapdoor languages and their applications. Cryptology ePrint Archive, Paper 2026/678, 2026. (Cited on Page 3, 6.)

[BCG+24a] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, François Garillot, Jonas Lindstrøm, Ben Riva, Arnab Roy, Mahdi Sedaghat, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. Subset-optimized BLS multi-signature with key aggregation. In Jeremy Clark and Elaine Shi, editors, *FC 2024, Part II*, volume 14745 of *LNCS*, pages 188–205. Springer, Cham, March 2024. (Cited on Page 5.)

[BCG24b] Annalisa Barbara, Alessandro Chiesa, and Ziyi Guan. Relativized succinct arguments in the ROM do not exist. Cryptology ePrint Archive, Report 2024/728, 2024. (Cited on Page 4.)

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016. (Cited on Page 4.)

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Cham, December 2018. (Cited on Page 3, 5, 7.)

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001. (Cited on Page 8.)

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

[BGOY07] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 276–285. ACM Press, October 2007. (Cited on Page 5.)

[BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 644–662. Springer, Berlin, Heidelberg, December 2012. (Cited on Page 5.)

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001. (Cited on Page 3, 5, 9.)

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004. (Cited on Page 5.)

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, October / November 2006. (Cited on Page 5.)

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

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

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. (Cited on Page 4.)

[BW24] Renas Bacho and Benedikt Wagner. Tightly secure non-interactive BLS multi-signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 397–422. Springer, Singapore, December 2024. (Cited on Page 5.)

[CGSY24] Alessandro Chiesa, Ziyi Guan, Shahar Samocha, and Eylon Yogev. Security bounds for proof-carrying data from straightline extractors. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024, Part II*, volume 15365 of *LNCS*, pages 464–496. Springer, Cham, December 2024. (Cited on Page 4.)

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, Berlin, Heidelberg, May 2003. (Cited on Page 11.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Berlin, Heidelberg, May / June 2010. (Cited on Page 11.)

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1115–1127. ACM Press, June 2016. (Cited on Page 10.)

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 394–423, Virtual Event, August 2021. Springer, Cham. (Cited on Page 6.)

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408. Springer, Cham, November 2017. (Cited on Page 11.)

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Cham, August 2017. (Cited on Page 11.)

[DGKV22] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-NP and applications. In *63rd FOCS*, pages 1057–1068. IEEE Computer Society Press, October / November 2022. (Cited on Page 3, 6.)

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

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum ethereum. *CiC*, 2(1):13, 2025. (Cited on Page 4, 6.)

[DMP26] Nico Döttling, Giulio Malavolta, and Omer Paneth. Batch arguments with optimal communication. Cryptology ePrint Archive, Paper 2026/003, 2026. (Cited on Page 3.)

[Edg23] Ben Edgington. 2.9.1 BLS signatures, 2023. Accessed: 2026-04-07. (Cited on Page 3.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013. (Cited on Page 12.)

[FLS12] Marc Fischlin, Anja Lehmann, and Dominique Schröder. History-free sequential aggregate signatures. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 113–130. Springer, Berlin, Heidelberg, September 2012. (Cited on Page 5.)

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Berlin, Heidelberg, August 1994. (Cited on Page 11.)

[GHJL17] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 519–548. Springer, Cham, April / May 2017. (Cited on Page 11.)

[GM15] Matthew D. Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320. IEEE Computer Society Press, May 2015. (Cited on Page 10, 11.)

[GOR18] Craig Gentry, Adam O'Neill, and Leonid Reyzin. A unified framework for trapdoor-permutation-based sequential aggregate signatures. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 34–57. Springer, Cham, March 2018. (Cited on Page 5.)

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Berlin, Heidelberg, December 2010. (Cited on Page 4.)

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008. (Cited on Page 12.)

[GS18] Sanjam Garg and Akshayaram Srinivasan. A simple construction of iO for Turing machines. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 425–454. Springer, Cham, November 2018. (Cited on Page 14.)

[GV22] Rishab Goyal and Vinod Vaikuntanathan. Locally verifiable signature and key aggregation. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 761–791. Springer, Cham, August 2022. (Cited on Page 5.)

[HJK+16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 715–744. Springer, Berlin, Heidelberg, December 2016. (Cited on Page 5.)

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 3–34. Springer, Berlin, Heidelberg, April 2015. (Cited on Page 5, 6, 12, 14.)

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Berlin, Heidelberg, May 2014. (Cited on Page 10.)

[HWW25] Susan Hohenberger, Brent Waters, and David J Wu. Pairing-based aggregate signatures without random oracles. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 271–303. Springer, 2025. (Cited on Page 5.)

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021. (Cited on Page 8.)

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, Berlin, Heidelberg, March 2006. (Cited on Page 10.)

[KKZM23] George Kadianakis, Dmitry Khovratovich, Zhenfei Zhang, and Mary Maller. Signature merging for large-scale consensus. Ethereum Research Forum, November 2023. Accessed: 2026-04-08. (Cited on Page 4, 6.)

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for Turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015. (Cited on Page 14.)

[KZHB25] George Kadianakis, Arantxa Zapico, Hossein Hafezi, and Benedikt Bünz. Kzh-fold: Accountable voting from sublinear accumulation. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 409–422, 2025. (Cited on Page 4, 6.)

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 74–90. Springer, Berlin, Heidelberg, May 2004. (Cited on Page 5.)

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

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to bitcoin. *DCC*, 87(9):2139–2164, 2019. (Cited on Page 3.)

[Nev08] Gregory Neven. Efficient sequential aggregate signed data. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 52–69. Springer, Berlin, Heidelberg, April 2008. (Cited on Page 5.)

[QLH12] Haifeng Qian, Xiangxue Li, and Xinli Huang. Tightly secure non-interactive multisignatures in the plain public key model. *Informatica*, 23(3):443–460, 2012. (Cited on Page 5.)

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

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 228–245. Springer, Berlin, Heidelberg, May 2007. (Cited on Page 3, 7.)

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

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 433–463. Springer, Cham, August 2022. (Cited on Page 6.)