Blind Multisignatures for Anonymous Tokens with Decentralized Issuance

Ioanna Karantaidou*
George Mason University
Fairfax, VA, USA
New York University
New York, NY, USA
ikaranta@gmu.edu

Omar Renawi*
CISPA Helmholtz Center for Information Security and Saarland University
Saarbrucken, Germany
omar.renawi@cispa.de

Foteini Baldimtsi
George Mason University
Fairfax, VA, USA
Mysten Labs
New York, NY, USA
foteini@gmu.edu

Nikolaos Kamarinakis
University of Maryland
College Park, MD, USA
Common Prefix
Athens, Greece
k4m4@umd.edu

Jonathan Katz
Google
Washington DC, USA
University of Maryland
College Park, MD, USA
jkatz2@gmail.com

Julian Loss
CISPA Helmholtz Center for Information Security
Saarbrucken, Germany
loss@cispa.de

Abstract

We propose the first constructions of anonymous tokens with decentralized issuance. Namely, we consider a dynamic set of signers/issuers; a user can obtain a token from any subset of the signers, which is publicly verifiable and unlinkable to the issuance process. To realize this new primitive we formalize the notion of blind multi-signatures (BMS), which allow a user to interact with multiple signers to obtain a (compact) signature; even if all the signers collude they are unable to link a signature to an interaction with any of them. We then present two BMS constructions, one based on BLS signatures and a second based on discrete logarithms without pairings. We prove security of both our constructions in the Algebraic Group Model. We also provide a proof-of-concept implementation and show that it has low-cost verification, which is the most critical operation in blockchain applications.

Keywords

Blind Signatures, Anonymous Tokens, Decentralized Issuance

ACM Reference Format:
Ioanna Karantaidou*, Omar Renawi*, Foteini Baldimtsi, Nikolaos Kamarinakis, Jonathan Katz, and Julian Loss. 2024. Blind Multisignatures for Anonymous Tokens with Decentralized Issuance . In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 25 pages. https://doi.org/10.1145/3658644.3690364

*Equal Contribution.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA.

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690364

1 Introduction

In the digital world, *authorization* plays a foundational role. From regulating access to online services to ensuring the integrity of voting systems, effective access-control mechanisms are crucial for maintaining security and trust. User authorization can be implemented via various methods depending on the application scenario. Common approaches include using credentials such as usernames and passwords or relying on third-party services, such as Auth0 or OpenID, to access user accounts. However, these authentication methods raise concerns regarding user privacy. Each time a user logs in, the service learns everything about the user's activities, enabling the creation of a full profile of their habits. While this level of information leakage may be necessary for certain applications, in many other cases it is desirable to avoid it. Consider, for instance, a subscription-based news portal. In a privacy-friendly world, the only thing that the service should learn is *whether* the user has a valid subscription to the service or whether the user has an account and *nothing else*.

One prominent solution to the problem of anonymous user authorization is anonymous tokens. In a nutshell, an anonymous token system includes three types of parties: *issuers*, *users*, and *verifiers*. An issuer provides an anonymous token to a user whose identity is typically known by the issuer at the time of issuance. The user can subsequently present the token to a verifier who can authenticate its validity. Anonymous tokens must be both unforgeable and anonymous, where unforgeability means that a user cannot forge a token and anonymity guarantees unlinkability between token issuance and presentation/verification. Blind signatures are a related notion; one can view anonymous tokens as blind signatures with no message. There are a number of blind signatures and anonymous token schemes with different properties [1, 13, 14, 19, 35, 56], and growing interest in their adoption by companies including Cloudfare,[1] Apple,[2], Google,[3] and Facebook.[4] A recent IETF draft [5] aims to standardize anonymous tokens.

Anonymous tokens can support public or private verifiability. Privately verifiable tokens assume the issuer and the verifier are the same entity, whereas publicly verifable tokens do not. Public verifiability is essential for large heterogeneous systems with a large number of verifiers who do not wish to also serve as token issuers. For blockchain applications, public verifiability is also necessary so tokens can be verified on-chain, possibly via a smart contract.

In all existing anonymous token systems, tokens are issued by a single issuer. This, however, introduces a single point of failure: if the token issuer is compromised it can issue an arbitrary number of tokens to unauthorized users. Furthermore, it is important for certain applications that tokens be issued by multiple issuers who jointly endorse a credential. Consider for example a tokenized anonymous-voting application where the governors of a Decentralized Autonomous Organization (DAO) wish to issue anonymous tokens to external members so they can vote on various issues. The voting policy may demand that a member is only eligible to vote if they receive endorsement from a miminum number of governors. To our knowledge, all prior work that would enable this use-case relies on heavy machinery such as zero-knowledge proofs, timelock encryption, or homomorphic encryption [3].

Publicly verifiable tokens with decentralized issuance. Motivated by this discussion, we propose the concept of *publicly verifiable anonymous tokens with decentralized issuance.* That is, we consider a dynamic set of signers/issuers; a user can obtain a token signed by any subset of the signers, which is publicly verifiable and unlinkable to the issuance process. As a building block toward this primitive, we propose blind multi-signatures (BMS). Multisignatures have the benefit of allowing for a flexible set of issuers that may change frequently, and require no-coordination amongst the issuers for token generation. This can be preferable to primitives like threshold signatures which require a coordinated Distributed Key Generation (DKG) protocol to be executed amongst the set of signers/issuers, and typically assume a static set of signers.

A BMS scheme can directly serve as a publicly verifiable anonymous token with decentralized issuance. Users can interact with each signer separately, collect individual signatures, and then aggregate them to obtain a final signature. As with multisignatures, a BMS reveals the set of signers who issued the token. For certain applications, we consider this to be a feature, as different signers may be responsible for certifying different attributes of a user. Knowing the identities of the signers can also enhance credibility of the tokens. Additionally, it offers some type of "signer accountability." For instance, if a signer is frequently associated with the issuance of tokens that are later misused, that signer may be penalized. At the same time, this raises the valid concern that disclosing the set of signers results in a reduced anonymity set, as a token is only unlinkable within the set of tokens that are signed by the same group of signers. We note, however, that for many applications this is not necessarily a problem. For starters, when the total number of signers is small and the number of users is large, the anonymity set for each user is likely to remain large. In other cases, the set of signers required for a valid token may be fixed (even as that set may change in different epochs); this would be the case in the DAO voting scenario discussed earlier, where a token is valid only when signed by the set of all current governors.

[1] https://blog.cloudflare.com/privacy-pass-standard
[2] https://developer.apple.com/news/?id=huqjyh7k
[3] https://github.com/google/anonymous-tokens, https://developers.google.com/privacy-sandbox/protections/private-state-tokens
[4] https://research.fb.com/privatestats
[5] https://datatracker.ietf.org/wg/privacypass/about/

1.1 Our Contributions

We now briefly summarize our technical contributions.

Blind multisignatures (BMS). The foundational building block at the core of our constructions is *blind multisignatures* (BMS). Multisignatures enable the computation of a joint signature on a message m, by a set of n signers, without requiring any coordination amongst the signers. As already explained, a BMS scheme can directly serve as a anonymous token scheme with decentralized issuance. In Section 3 we provide rigorous definitions for blind multisignatures (BMS) and their corresponding security properties: blindness and one-more unforgeability (OMUF). We then present two BMS constructions with different tradeoffs, described next.

BMS based on BLS. In Section 4 we construct BM_BLS, a blind multisignature based on the Boneh–Lynn–Shacham (BLS) signature scheme [9]. We prove concurrent security of our construction in the Algebraic Group and Random Oracle Models (AGM + ROM) based on the q-dlog assumption. BLS is an efficient signature scheme that uses pairings and has recently seen adoption in the blockchain space (i.e., the Chia Network [16], Celo [12], Filecoin, and PoS Ethereum) due to its efficient support for signature aggregation. An IETF standardization effort for BLS has been ongoing since 2019 [30]. Blind BLS [7] and BLS multisignatures [8] already exist in the literature. However, combining them to obtain a blind multisignature is not trivial. In particular, a significant challenge is to avoid so-called *rogue-key* attacks where an adversary breaks security by choosing a (malformed) public key based on the public keys of honest parties. Our construction is secure against rogue-key attacks in the plain public-key model, i.e., there is no need for signers to prove knowledge of their signing keys. It also supports public-key aggregation.

A pairing-free BMS. In Section 5 we present BM_SB, a pairing-free BMS scheme based on the recent threshold blind-signature scheme Snowblind [18]. We prove concurrent security based on the discrete-logarithm (dlog) assumption in the AGM. Towards taming the complexity of this proof, we follow a similar technique as in recent work [28, 32]. In particular, we first propose a new cryptographic primitive called a multi-identification (mID) scheme and adapt the security notion to fit our new primitive. Then, we construct a multi-identification scheme and prove its security. Finally, we show how this implies security of our BMS scheme.

Compared to our BLS-based construction, our second scheme enjoys more efficient verification (since it avoids pairings) and has very short signatures regardless of the number of signers. As opposed to our BLS construction, however, this scheme requires each (corrupted) signer to submit a proof of possession of its public key, which in turn prevents public key aggregation.

Implementation and evaluation. In Section 6, we present a proof-of-concept Python implementation of our two constructions, and a generic smart-contract library for verifying our anonymous tokens on the Ethereum blockchain. We evaluate the efficiency and cost of our implementations, demonstrating their practicality. Verifying a token on Ethereum costs about 232K gas for BM_BLS, irrespective of the number of signers, and about 280K gas for a BM_SB token issued by 11 signers. As of April 28th, 2024, when the median gas price was approximately 7.4 gwei [36] and the Ethereum closing price[6] was 3,262.77 USD, this translates to a monetary cost of ~\$5.60 and ~\$6.76, respectively. Moreover, BM_BLS tokens can be aggregated, meaning that the verification cost can be amortized across multiple users. The amortized cost for verifying a batch of 32 or more tokens is around 110K gas, or ~\$2.66.

1.2 Related Work

As already noted, although there exist a variety of anonymous token constructions, none of them supports decentralized issuance. We discuss two types of related work: (1) blind signatures with multiple issuers and (2) decentralized anonymous credentials (a primitve more general than anonymous tokens).

Blind multisignatures and threshold signatures. Blind signatures with multiple signers can be found in the form of multisignatures or threshold signatures, with the primary distinction between them being whether the signers generate their keys independently (multisignatures) or whether they need to jointly run a protocol to generate a single public key and individual key shares (threshold signatures). Some blind multisignature schemes have been suggested in the literature [7, 15, 42, 49, 59], but they all lack rigorous security analysis. Several constructions of blind threshold signatures exist [2, 18, 34, 37, 39, 57], but as we have noted these all require coordination between the issuers during key generation and do not immediately support dynamic signing sets.

Decentralized anonymous credentials. Anonymous credential systems are typically *multi-use*, i.e., credentials that encode a set of attributes are issued once and presented multiple times. Compared to anonymous tokens, which can be viewed as a *single-use* credential without attributes, those schemes are therefore much more complex and expensive. The problem of decentralized issuance for anonymous credentials has been addressed using different approaches which we briefly discuss below. We note, however, that converting any of these anonymous credential schemes to an efficient anonymous token scheme is non trivial.

A number of decentralized anonymous-credential schemes use threshold techniques [20, 51, 57, 58]; these all have the drawback of requiring the issuers to coordinate at the time of key generation as discussed above. Another recent line of work [29, 46] constructs decentralised multi-use anonymous credentials from aggregate signatures with randomizable tags. Finally, some work [26] has considered decentralized anonymous credentials based on peer-to-peer anonymous attestation on a bulletin board/blockchain rather than issuing authorities, a setting quite different from the one we consider here.

[6] See https://coinmarketcap.com/currencies/ethereum/historical-data.

2 Preliminaries

We let λ denote the security parameter. PPT means probabilistic polynomial time. We let $\mathsf{poly}(\lambda)$ be an unspecified polynomial function of λ and $\mathsf{negl}(\lambda)$ a negligible function. We let $[t] = \{1, \ldots, t\}$. We use $x \xleftarrow{\$} \mathcal{D}$ to refer to sampling a uniform element x from $\mathcal{D}$. We write $y \leftarrow \mathsf{A}^{\mathcal{O}}(x)$ to denote the randomized output of an algorithm A that takes x as input and has access to an oracle $\mathcal{O}$. Given a game Game parameterized by an adversary A, the success probability of A in Game is $\mathsf{Adv}_A^{\mathsf{Game}}(\lambda) := \Pr[\mathsf{Game}_A = \mathsf{true}]$.

2.1 Cryptographic Assumptions

Assumption 1 ([24]). *Let $\mathbb{G}$ be a cyclic group of order p. The q-discrete-logarithm assumption holds if for every* PPT *algorithm A:*

$$\Pr\left[\begin{array}{c} x \leftarrow \mathbb{Z}_p^* \\ x^* \leftarrow A(g, Y_1 = g^x, \ldots, Y_q = g^{x^q}) \end{array} : x^* = x\right] \leq negl(\lambda).$$

Note that the standard discrete-logarithm assumption is just the 1-dlog assumption.

Definition 1 (Bilinear Pairings). Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of order p. A pairing is an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that for all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$ it holds that $e(P^a, Q^b) = e(P^a, Q)^b = e(P, Q^b)^a = e(P, Q)^{ab}$. If $\mathbb{G}_1 = \mathbb{G}_2$ then we say the pairing is symmetric.

2.2 The Algebraic Group Model (AGM)

The AGM [24] is a formal model for analyzing group-based cryptosystems. In the AGM, the adversary A is assumed to be *algebraic*. Roughly, this means that if $\vec{g} = (g_1, \ldots, g_t)$ are the group elements A has been given at any point in its execution, then if it outputs a group element y it also outputs a representation $\vec{z}$ such that $y = \prod_{i \in [t]} g_i^{z_i}$. We stress that group elements A receives from any oracles it has access to are included in $\vec{g}$, and any time A submits a group element y to one of its oracles it must also output a representation of y.

2.3 Blind Signatures

A blind signature scheme [14] is an interactive protocol between a signer and a user that allows the user to obtain a signature that cannot later be linked to the user by the signer. A blind signature scheme BS consists of the following algorithms:

- $\mathsf{BS.KGen}(1^\lambda) \to (sk, pk)$. Run by a signer to generate keys.
- $\mathsf{BS.Sign}\langle \mathcal{U}(m, pk), \mathcal{S}(sk)\rangle \to \sigma$. This is an interactive protocol between a (stateful) signer $\mathcal{S}$ with input the secret key sk and a (stateful) user $\mathcal{U}$ with input a message m and the signer's public key. $\mathcal{U}$ outputs a signature σ.
- $\mathsf{BS.Ver}(pk, m, \sigma) \to 0/1$. Run by a verifier; outputs 1 iff σ is a valid signature for m under key pk.

Correctness can be formalized in the obvious way. A secure blind signature scheme should satisfy *blindness* (i.e., a signature cannot be linked back to its corresponding signing session, even by the signer itself) and *one-more unforgeability* (i.e., an adversarial user $\mathcal{U}$ making ℓ blind signing queries cannot output $\ell + 1$ valid signatures). We recall the formal definitions in Appendix A.1.

2.4 Multi-Signatures

A multi-signature scheme allows a set of signers to each generate a signature on a message m; those signatures can then be aggregated to form a compact signature of all the signers on m. Some multi-signatures also support public-key aggregation which allows for a compact representation of all the signers' public keys. Our definition roughly follows that of Drijvers et al. [21]. For public parameters pp, a multi-signature scheme MS consists of the following algorithms:

- $\mathsf{MS.KGen}(pp) \rightarrow (sk, pk)$. Run by a signer to obtain a key pair.
- $\mathsf{MS.KAgg}(\vec{K}) \rightarrow \mathsf{apk}$. Given a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$, outputs an aggregate public key apk.
- $\mathsf{MS.Sign}\ (sk_i, m) \rightarrow \sigma_i$. A signer with input[7] a secret key sk_i and a message m outputs a signature σ_i.
- $\mathsf{MS.Comb}(m, \{\sigma_i\}) \rightarrow \sigma$. Individual signatures σ_i can be combined into a signature σ.
- $\mathsf{MS.Ver}(\mathsf{apk}, m, \sigma) \rightarrow 0/1$. Outputs 1 if σ is a valid signature for message m under aggregate key apk.

We recall security definitions for multi-signatures in Appendix A.2. We remark that one challenge in multi-signature schemes is avoiding rogue-key attacks [27, 38, 40, 43, 45, 48], which can occur when an attacker uses a public key that is not generated honestly, but instead depends in some way on an honest signer's public key. One way to avoid such attacks is to rely on the so-called *knowledge-of-secret-key (KOSK)* model which can be implemented by having each signer include a zero-knowledge proof of knowledge (aka a proof of possession) of their secret key along with their public key [7, 43, 53]. Schemes that do not require this extra assumption are said to be in the *plain public-key model* [6, 8, 47].

3 Blind Multi-signatures

A blind multi-signature combines the features of both blind and multi-signature schemes. It resembles a multi-signature in that it is a signature on a message m signed by multiple signers that verifies under the set of public keys of the signers $\vec{K}$ or under an aggregate key apk if scheme supports key aggregation. It also resembles a blind signature, as the signing happens in an interactive fashion between a user $\mathcal{U}$ who knows m and a set of signers who should be unable to link the final signature to the issuance process. Below we provide a rigorous definition.

For public parameters pp, a blind multi-signature scheme BMS consists of the following algorithms:

- $\mathsf{BMS.KGen}(pp) \rightarrow (sk, pk)$. Run by a signer to obtain a key pair.
- $\mathsf{BMS.KAgg}(\vec{K}) \rightarrow \mathsf{apk}$. Outputs an aggregate public key apk for a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$.
- $\mathsf{BMS.Sign}\langle \mathcal{U}(m, \vec{K}), \{\mathcal{S}_i(sk_i)\}_{i \in [n]}\rangle \rightarrow \sigma$. This is an interactive protocol run between a user $\mathcal{U}$ and signers $\mathcal{S}_1, \ldots, \mathcal{S}_n$, where the signers do not directly communicate with each other. Each signer has only its own secret key as input; $\mathcal{U}$ has a message m and the signers' public keys $\vec{K}$ as input, and outputs a signature σ. We assume all keys in $\vec{K}$ are distinct.
- $\mathsf{BMS.Ver}(\mathsf{apk}, m, \sigma) \rightarrow 0/1$. Outputs 1 if σ is a valid signature on m under aggregate key apk.

Correctness requires that if signers honestly generate keys (sk_i, pk_i) and then run $\sigma \leftarrow \mathsf{BMS.Sign}\ \langle \mathcal{U}(m, \vec{K}), \mathcal{S}_i(sk_i)\rangle$, where $\vec{K} = \{pk_i\}$, then $\mathsf{BMS.Ver}(\mathsf{apk}, m, \sigma) = 1$, where $\mathsf{apk} = \mathsf{BMS.KAgg}(\vec{K})$.

Security. A blind multi-signature should satisfy *one-more unforgeability* and *blindness.*

Let apk be the aggregated public key for a set of signers, one of whom is honest. One-more unforgeability requires that an adversarial user $\mathcal{U}$ (possibly colluding with all corrupted signers) should be unable to forge a signature that verifies under apk, unless this signature came from its interaction with the honest signer. Below we give the formal definition in the plain public-key model. (In the KOSK model, the adversary must also output the secret key corresponding to any adversarial public key.) The signing oracle Sign_{sk^*} simulates the honest signer's execution of the signing protocol.

Definition 2 (One-more unforgeability (OMUF)). Given a blind multi-signature scheme BMS = (KGen, KAgg, Sign, Ver), we define the game $\mathsf{OMUF}_A^{\mathsf{BMS}}$ as follows:

- **Setup:** The challenger generates a key pair (sk^*, pk^*) using BMS.KGen, and gives pk^* to A.
- **Queries:** A may repeatedly query a signing oracle Sign_{sk^*}.
- **Output:** A outputs a list of tuples $(\sigma_1^*, m_1^*, \vec{K}_1), \ldots, (\sigma_{\ell+1}^*, m_{\ell+1}^*, \vec{K}_{\ell+1})$; let $\mathsf{apk}_i = \mathsf{BMS.KAgg}(\vec{K}_i)$ for all i. A wins if: (1) pk^* is in each set $\vec{K}_i$, (2) $\mathsf{BMS.Ver}\ (\mathsf{apk}_i, m_i, \sigma_i) = 1$ for all i, and (3) the number of completed interactions with Sign_{sk^*} is at most ℓ. If A wins, the game outputs true.

BMS is one-more unforgeable (OMUF) if for any PPT A,

$$\mathsf{Adv}_{A,\mathsf{BMS}}^{OMUF}(\lambda) := \Pr[\mathsf{OMUF}_A^{\mathsf{BMS}} = \mathsf{true}] = \mathsf{negl}(\lambda).$$

Sequential vs. concurrent security. The above models concurrent security, i.e., the adversary may concurrently run multiple executions with Sign_{sk^*}. To model sequential security, Sign_{sk^*} should not open a new signing session before the previous one is closed.

The next security property of blind multi-signatures is blindness, i.e., even the signers themselves should be unable to link a signature to its corresponding signing session. In the definition we assume that all signers are colluding and we allow for maliciously generated keys. In the blindness game the adversary A starts by choosing all the signers' public keys as well as two messages to be signed. The honest user runs two executions of the signing protocol with A and the given keys, one for each message, in a random order. A is then given the two resulting signatures and asked to guess the order in which the two messages were signed. Formally, given blind multi-signature scheme BMS= (KGen, KAgg, Sign, Ver) let $\mathsf{mBlind}_A^{\mathsf{BMS}}$ be the following game:

Definition 3 (Blindness). The adversary A outputs public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$ and messages m_0, m_1. The challenger picks $b \leftarrow \{0, 1\}$, and runs two signing sessions as the user $\mathcal{U}(m_b, \vec{K})$, $\mathcal{U}(m_{1-b}, \vec{K})$, while A participates in the signing sessions as the n signers. If one or both sessions fail to output a (valid) signature, the game outputs $(\perp, \perp)$. Otherwise, if A closes both sessions successfully, the game outputs the resulting signatures (σ_0, σ_1). Eventually,

[7] In some schemes, the signer additionally needs to know $\vec{K}$.

A outputs a bit b' and wins the game if $b' = b$, and in this case, the game outputs true.

BMS is blind if for any PPT adversary A,

$$\mathsf{Adv}^{mBlind}_{A,\mathsf{BMS}}(\lambda) := \Pr[\mathsf{mBlind}^{\mathsf{BMS}}_{A} = \text{true}] = \frac{1}{2} + \mathsf{negl}(\lambda).$$

4 BLS Blind Multisignatures

In this section we construct a blind multisignature scheme based on blind BLS signatures. As such, we begin by reviewing the latter. We also provide a proof of security for the blind BLS signature scheme in the AGM+ROM, since this will serve as a useful warmup for our eventual proof of security for the blind multisignature scheme.

4.1 Blind BLS Signatures

We start by describing the blind BLS signature scheme [7]. For simplicity, in these sections we present constructions and proofs using symmetric pairings. Let $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, p, g, e)$ denote the system parameters and let $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ be a hash function. The blind BLS scheme consists of the following algorithms:

- $\mathsf{KGen}(1^\lambda)$ outputs $(sk, pk) = (x, X)$, where $x \xleftarrow{\$} \mathbb{Z}_p$ and $X = g^x \in \mathbb{G}$.
- $\mathsf{Sign}\langle \mathcal{U}(m, X), \mathcal{S}(sk)\rangle$ outputs a signature σ as per Fig. 1.
- $\mathsf{Ver}(pk = X, m, \sigma)$: Checks whether $e(\sigma, g) = e(\mathsf{H}(m), X)$.

Correctness is immediate and blindness holds unconditionally [7].

$\mathcal{S}(sk = x)$		$\mathcal{U}(m, X)$
	$\xleftarrow{\bar{m} = \mathsf{H}(m)\cdot g^r}$	$r \xleftarrow{\$} \mathbb{Z}_p$
$\bar{s} = \bar{m}^x$	$\xrightarrow{\bar{s}}$	$\sigma = \bar{s}X^{-r}$

Figure 1: Signing for Blind BLS

Boldyreva [7] showed that blind BLS is one-more unforgeable under the "chosen-target" CDH assumption (or the one-more static CDH assumption) in the ROM. In Appendix B, we prove one-more unforgeability under the q-dlog assumption in the AGM+ROM. While the two sets of assumptions/models are incomparable, we note that our proof gives a tighter reduction. As noted earlier, our main motivation for giving this proof is that it serves as a warmup for the proof of unforgeability for our blind multisignature scheme based on blind BLS.

4.2 BLS-Based Blind Multisignatures

We now present our blind multisignature scheme based on blind BLS, which we denote by BM_BLS. Our main observation is that we can construct a blind multisignature scheme directly from blind BLS; that is, the user can interact with each signer exactly as in the blind BLS scheme, and then combine the signatures it obtains into a single multisignature using an additional hash function.

Let $(\mathbb{G}, \mathbb{G}_T, p, g, e)$ and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ be as in the previous section, and let $\mathsf{H}_{\mathsf{agg}} : \{0,1\}^* \to \mathbb{Z}_p^*$ be another hash function.

- $\mathsf{BM\_BLS.KGen}(1^\lambda)$: As in the blind BLS scheme.
- $\mathsf{BM\_BLS.KAgg}(\vec{K})$: Given keys $\vec{K} = \{X_1, \ldots, X_n\}$, set $a_i = \mathsf{H}_{\mathsf{agg}}(\vec{K}, X_i)$ and output $\mathsf{apk} = \prod_{i=1}^{n} X_i^{a_i}$.
- $\mathsf{BM\_BLS.Sign}\langle \mathcal{U}(m, \vec{K}), \{\mathcal{S}_i(sk_i)\}_{i\in[n]}\rangle$: $\mathcal{U}$ runs the interactive signing protocol with each signer as in Fig. 1, using independent randomness each time, to obtain partial signatures $\{\sigma_i\}$. The final signature is computed as $\sigma = \prod_{i\in[n]} \sigma_i^{a_i}$, where $a_i = \mathsf{H}_{\mathsf{agg}}(\{X_1, \ldots, X_n\}, X_i)$.
- $\mathsf{BM\_BLS.Ver}(\mathsf{apk}, m, \sigma)$: Checks if $e(\sigma, g) = e(\mathsf{H}(m), \mathsf{apk})$.

To see that correctness holds, note first that

$$\begin{aligned}\sigma_i X_i^{-r_i} &= (\mathsf{H}(m) g^{r_i})^{x_i} X_i{}^{-r_i} \\ &= \mathsf{H}(m)^{x_i} g^{r_i x_i} g^{-r_i x_i} \;=\; \mathsf{H}(m)^{x_i}\end{aligned}$$

for all i. Thus,

$$\begin{aligned}e(\sigma, g) &= e(\prod_{i\in[n]} \sigma_i^{a_i}, g) \\ &= e(\mathsf{H}(m)^{\sum_{i\in[n]} x_i a_i}, g) \\ &= e(\mathsf{H}(m), g^{\sum_{i\in[n]} x_i a_i}) \\ &= e(\mathsf{H}(m), \prod_{i\in[n]} g^{x_i a_i}) \\ &= e(\mathsf{H}(m), \prod_{i\in[n]} (g^{x_i})^{a_i}) \\ &= e(\mathsf{H}(m), \prod_{i\in[n]} X_i^{a_i}) \;=\; e(\mathsf{H}(m), \mathsf{apk}),\end{aligned}$$

and BM_BLS.Ver outputs 1.

Discussion. Due of the simple nature of the protocol, $\mathcal{U}$ can contact each signer in parallel to obtain the necessary partial signatures. Moreover, even if some signers are unreachable, $\mathcal{U}$ can compute a multisignature based on the set of signers who respond.

Multisignature aggregation. Multisignatures on multiple, distinct messages with respect to the same aggregate public key can be aggregated. For example, given signatures σ_1 on message m_1 and σ_2 on message m_2, signed by the same set of signers, the aggregate signature $\sigma = \sigma_1\sigma_2$ can verified by checking if $e(\sigma, g) = e(\mathsf{H}(m_1)\mathsf{H}(m_2), \mathsf{apk})$. This also enables more-efficient verification.

Security. Blindness follows by a natural extension of the proof for blind BLS (cf. Appendix C.1). It is more challenging to prove one-more unforgeability. We prove the following in Appendix C.2.

Theorem 4. *Assume the discrete logarithm problem is hard, and model* $\mathsf{H}, \mathsf{H}_{\mathsf{agg}}$ *as random oracles. Then* BM_BLS *is one-more unforgeable for all PPT algebraic adversaries.*

5 A Pairing-Free Construction

In this section we show an alternate construction of blind multisignatures that has the advantage of avoiding pairings. Motivated by prior work [28], we introduce the concept of multi-identification schemes with security against a certain form of man-in-the-middle (MiTM) attacks, and then design such a scheme. Finally, we show how to use such schemes to construct blind multisignatures.

5.1 Multi-Identification Schemes

Hauck et al. [28] prove OMUF security of blind signature schemes built from identification schemes by proving one-more man-in-the-middle (OMMIM) security of the underlying identification scheme.

$\mathcal{S}_i(pk_i = g^{sk_i}, sk_i)$		$\mathcal{U}(\vec{K} = \{pk_1, \ldots, pk_n\}, m)$
$a_i, b_i, y_i \leftarrow \mathbb{Z}_q$ $A_i := g^{a_i}, B_i := g^{b_i} \cdot h^{y_i}$ $com_i := \mathsf{H}_{\mathsf{com}}(pk_i, b_i, y_i)$		
	$\xrightarrow{A_i, B_i, com_i}$	For $i \in [n] : \beta_i \leftarrow \mathbb{Z}_q$ $\alpha, r \leftarrow \mathbb{Z}_q$ $A := \prod_j A_j, B := \prod_j B_j$ $\overline{R} := g^r \cdot \prod_j pk_j^{\alpha^3 \cdot \beta_j} \cdot A^{\alpha^3} \cdot B^{\alpha}$ For $i \in [n] : \overline{c}_i := \mathsf{H}_{\mathsf{sig}}(\vec{K}, pk_i, \overline{R}, m)$ For $i \in [n] : c_i := \overline{c}_i \cdot \alpha^{-3} + \beta_i$
	$\xleftarrow{c_i, \{B_j, com_j\}_{j \in [n]}}$	
	$\xrightarrow{b_i, y_i}$	
	$\xleftarrow{\{b_i, y_i\}_{i \in [n]}}$	
If $\exists i \in [n]\ com_i \neq \mathsf{H}_{\mathsf{com}}(i, y_i, b_i)$: Abort If $\exists i \in [n]\ B_i \neq g^{b_i} \cdot h^{y_i}$: Abort $y := \sum_{j \in [n]} y_j$ $z_i := a_i + (c_i + y^3) \cdot sk_i$		
	$\xrightarrow{z_i}$	$b := \sum_j b_j, y := \sum_j y_j, z := \sum_j z_j$ If $g^z \neq A \cdot \prod_{j \in [n]} pk_j^{c_j + y^3} \vee B \neq g^b \cdot h^y$: Abort $\overline{z} := r + \alpha^3 \cdot z + \alpha \cdot b$ $\overline{y} := \alpha \cdot y$ Output $\sigma := (\overline{R}, \overline{y}, \overline{z})$

Figure 2: Blind multisignature scheme BM_SB**.**

We cannot immediately follow their methodology because it is not clear how to build blind *multi*signatures from standard identification schemes. To address this, we put forth the notion of a multi-identification scheme (mID). In an mID scheme, a set of provers, each of which has its own keys (pk,sk), interact with a verifier to prove knowledge of their secret keys. Intuitively, mID schemes allow provers to prove themselves to a verifier as a *group*. Although not very useful on their own, mID schemes can be used as a technical tool to build blind multi-signature schemes.

Definition 5 (Multi-identification schemes). For public parameters pp, an mID scheme is a tuple $\mathsf{mID} := (\mathsf{mID.KGen}, \mathsf{mID.Idfy})$ where

- $\mathsf{mID.KGen}(pp)$: Outputs a pair of keys (sk, pk).
- $\mathsf{mID.Idfy}\langle \mathcal{P}_i, \mathcal{V} \rangle$: This is an interactive protocol between the verifier $\mathcal{V}$ and multiple provers $\{\mathcal{P}_i\}$, in which the provers do not directly communicate with each other. Each prover has its own secret key as input, while the verifier has the public keys of all the provers. The protocol terminates when $\mathcal{V}$ outputs 1 (ACCEPT) or 0 (REJECT).

Definition 6 (Correctness). Let $\mathsf{mID} := (\mathsf{mID.KGen}, \mathsf{mID.Idfy})$ be an mID scheme with n provers $\mathcal{P}_i$ and a verifier $\mathcal{V}$. We say that mID is correct iff for all pp it holds that

$$\Pr\left[\begin{array}{c}\forall i \in [n] : (sk_i) \leftarrow \mathsf{mID.KGen}(pp) \\ b \leftarrow \mathsf{mID.Idfy}\langle \mathcal{P}_i(sk_i, pk_i), \mathcal{V}(\{pk_1, \ldots, pk_n\})\rangle\end{array} : b = 1\right] = 1.$$

We generalize the security notion introduced by Hauck et al. [28] for mID schemes. Analogous to the standard OMMIM definition, we assume there is an active man-in-the-middle adversary A between the provers and the verifier. We also allow A to control all but one of the provers.

Definition 7 (One-more MiTM (OMMIM) security). Let A be an adversary and let $\mathsf{mID} := (\mathsf{mID.KGen}, \mathsf{mID.Idfy})$ be an mID scheme. Define the game ℓ-OMMIM as follows:

- **Setup.** Generate pp and run $(sk^*, pk^*) \leftarrow \mathsf{mID.KGen}(pp)$. Give pk^* to A.
- **Online phase.** A interacts (concurrently) with an honest prover using sk^*, and an honest verifier. For the latter, it must use a set of public keys containing pk^*.
- **Output.** A succeeds if it successfully completes at least $\ell + 1$ verifier sessions (i.e., by making the verifier output 1) but closes at most ℓ sessions with the honest prover.

We say that mID is ℓ-OMMIM-secure if any PPT A succeeds with negligible probability in the above game.

5.2 Constructing a Multi-Identification Scheme

We provide a construction of a multi-ID scheme, inspired by prior work [18]. The protocol is depicted in Figure 5. Let $pp := (\mathbb{G}, g, q, h)$, where $\mathbb{G}$ is a group of prime order $q = 2 \bmod 3$ with generator g, and $h \in \mathbb{G}$ is a uniform group element. Define the scheme mID = (mID.KGen, mID.Idfy) as follows:

- mID.KGen(pp): sample $sk \leftarrow \mathbb{Z}_q$, set $pk := g^{sk_i}$, and output (sk, pk).
- mID.Idfy works as follows (see Fig. 5):
  - mID.Prove$_1$: Sample $a_i, b_i, y_i \leftarrow \mathbb{Z}_q$ and set $A_i := g^{a_i}$ and $B_i := g^{b_i} \cdot h^{y_i}$. Then send (A_i, B_i).
  - mID.Ver$_1$: After receiving all $\{(A_i, B_i)\}$, choose $c_i \leftarrow \mathbb{Z}_q$ for all i and send c_i and $\{B_j\}$ to the ith prover.
  - mID.Prove$_2$: Send (b_i, y_i).
  - mID.Ver$_2$: After receiving $\{(b_i, y_i)\}$ from all provers, abort if $B_i \neq g^{b_i} \cdot h^{y_i}$ for some i. Otherwise, send $\{b_i, y_i\}$ to all provers.
  - mID.Prove$_3$: Abort if $B_j \neq g^{b_j} \cdot h^{y_j}$ for some j. Otherwise, compute $y := \sum_j y_j$ and send $z_i := a_i + b_i + (c_i + y^3) \cdot sk_i$.
  - mID.Ver$_3$: After receiving $\{z_i\}$ from all provers, compute $A := \prod_i A_i$, $B := \prod_i B_i$, $R := A \cdot B$, $y := \sum_i y_i$, and $z := \sum_i z_i$. Return 1 iff $g^z \cdot h^y = R \cdot \prod_i pk_i^{c_i + y^3}$.

To see that correctness holds, note that

$$
\begin{aligned}
R \cdot \prod_{j\in[n]} pk_j^{c_j+y^3} &= h^{\sum_{j\in[n]} y_j} \cdot \prod_{j\in[n]} A_j \cdot pk_j^{c_j+y^3} \\
&= g^{\sum_{j\in[n]} a_j} \cdot h^{\sum_{j\in[n]} y_j} \cdot g^{\sum_{j\in[n]} sk_j\cdot(c_j+y^3)} \\
&= g^{\sum_{j\in[n]} a_j + sk_j\cdot(c_j+y^3)} \cdot h^{\sum_{j\in[n]} y_j} \\
&= g^{\sum_{j\in[n]} z_j} \cdot h^y \;=\; g^z \cdot h^y.
\end{aligned}
$$

We prove the following in Appendix D:

Theorem 8. *Assume the discrete-logarithm problem is hard. Then* mID *is ℓ-*OMMIM*-secure for all PPT algebraic adversaries.*

5.3 A Pairing-Free BMS

In this section, we introduce a pairing-free blind multi-signature scheme BM_SB. Our scheme is inspired by the blind threshold-signature scheme Snowblind [18]. For the reader's convenience, we illustrate the scheme as an interactive protocol in Figure 2. For $pp = (\mathbb{G}, q, g, h)$ as in the previous section, and for hash functions $\mathsf{H}_{\mathsf{com}} : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathsf{H}_{\mathsf{sig}} : \{0,1\}^* \to \mathbb{Z}_p$ treated as random oracles, we define BM_SB as follows:

- BM_SB.KGen(1^λ): As before.
- BM_SB.Sign is an interactive protocol run by a user Usr and multiple signers. It works as follows (see Figure 2):
  - Sign$_1$(sk_i): Sample $a_i, b_i, y_i \leftarrow \mathbb{Z}_q$, and compute $A_i := g^{a_i}$, $B := g^{b_i} \cdot h^{y_i}$, and $com_i := \mathsf{H}_{\mathsf{com}}(pk_i, b_i, y_i)$. Then send (A_i, B_i, com_i).
  - Usr$_1$($\vec{K} = \{pk_i\}, m$): Upon receiving (A_i, B_i, com_i) from all signers, sample $\alpha, r \leftarrow \mathbb{Z}_q$ and $\beta_i \leftarrow \mathbb{Z}_q$ for all i, and compute $A := \prod_j A_j$, $B := \prod_j B_j$, and $\overline{R} := g^r \cdot A^{\alpha^3} \cdot B^\alpha \cdot \prod_j pk_j^{\alpha^3\cdot\beta_j}$. Then for all i compute $\overline{c}_i := \mathsf{H}_{\mathsf{sig}}(\vec{K}, pk_i, \overline{R}, m)$ and $c_i := \overline{c}_i \cdot \alpha^{-3} + \beta_i$. Send $c_i, \{B_j, com_j\}_j$ to the ith signer.
  - Sign$_2$: Send b_i, y_i.
  - Usr$_2$: Upon receiving b_j, y_j from all signers, abort if $B_j \neq g^{b_j} \cdot h^{y_j}$ or $com_j \neq \mathsf{H}_{\mathsf{com}}(pk_j, b_j, y_j)$ for some j. Otherwise, send $\{b_j, y_j\}_j$ to all signers.
  - Sign$_3$: Compute $y := \sum_j y_j$ and $z_i := a_i + (c_i + y^3) \cdot sk_i$, and send z_i.
  - Usr$_3$: Upon receiving z_j from all signers, compute $b := \sum_j b_j$, $y := \sum_j y_j$, and $z := \sum_j z_j$. Abort if $g^z \neq A \cdot \prod_j pk_j^{c_j+y^3}$ or $B \neq g^b \cdot h^y$. Compute $\overline{z} := r + \alpha^3 \cdot z + \alpha b$ and $\overline{y} := \alpha y$, and output the signature $\sigma = (\overline{R}, \overline{y}, \overline{z})$.
- Vrfy($\vec{K}, m, \sigma$): Parse $\overline{R}, \overline{y}, \overline{z} \leftarrow \sigma$, compute $\overline{c}_i := \mathsf{H}_{\mathsf{sig}}(\vec{K}, pk_i, \overline{R}, m)$ for all i, and output 1 if $\overline{y} \neq 0$ and $\overline{R} \cdot \prod_i pk_i^{\overline{c}_i + \overline{y}^3} = g^{\overline{z}} \cdot h^{\overline{y}}$, and 0 otherwise.

(BM_SB.KAgg is not defined because the scheme does not support key aggregation, and Vrfy takes the set of keys $\vec{K} = \{pk_1, \ldots, pk_n\}$ as input instead of an aggregate key apk.)

To see that correctness holds, note that

$$
\begin{aligned}
g^{\overline{z}} \cdot h^{\overline{y}} &= g^r \cdot g^{\alpha\cdot b} \cdot h^{\overline{y}} \cdot g^{\alpha^3 z} \\
&= g^r \cdot (g^b \cdot h^y)^\alpha \cdot (g^{\sum_j z_i})^{\alpha^3} \\
&= g^r \cdot B^\alpha \cdot (g^{\sum_i a_i + (c_i+y^3)\cdot sk_i})^{\alpha^3} \\
&= g^r \cdot B^\alpha \cdot g^{\alpha^3 \cdot \sum_i a_i} \cdot (g^{\sum_i (c_i+y^3)\cdot sk_i})^{\alpha^3} \\
&= g^r \cdot B^\alpha \cdot A^{\alpha^3} \cdot \left(\prod_i pk_i^{(c_i+y^3)}\right)^{\alpha^3} \\
&= g^r \cdot B^\alpha \cdot A^{\alpha^3} \cdot \prod_i pk_i^{(\overline{c}_i\cdot\alpha^{-3}+\beta_i+(\overline{y}\cdot\alpha^{-1})^3)\cdot\alpha^3} \\
&= g^r \cdot B^\alpha \cdot A^{\alpha^3} \cdot \prod_i pk_i^{\alpha^3\cdot\beta_i} \cdot \prod_i pk_i^{(\overline{c}_i+\overline{y}^3)} \\
&= \overline{R} \cdot \prod_i pk_i^{(\overline{c}_i+\overline{y}^3)}.
\end{aligned}
$$

We prove the following in Appendix E.

Theorem 9. *For all PPT* A, $\mathbf{Adv}^{\mathsf{mBlind}}_{\mathsf{A},\mathsf{BM\_SB}}(\lambda) \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Theorem 10. *Assume the discrete logarithm problem is hard, and model* $\mathsf{H}_{\mathsf{com}}$ *and* $\mathsf{H}_{\mathsf{sig}}$ *as random oracles. Then* BM_SB *is one-more unforgeable for all PPT algebraic adversaries in the KOSK model.*

6 Evaluation

We present proof-of-concept implementations of BM_BLS and BM_SB, written in Python. (Code available at https://github.com/k4m4/bm-poc.) We also implemented a signature-verification smart contract (in Solidity) for each scheme. For our evaluation, we used the BN254 elliptic curve (using an EIP-1964 implementation,[8] with Rust bindings), which is estimated to provide around 100 bits of security [54]. We used BN254 for both schemes since, at the time of writing, it is the only pairing-friendly elliptic curve supported by Ethereum, but also the only curve over which EC addition and EC multiplication can be practically performed on an Ethereum smart contract.

6.1 Implementation Benchmarks

Table 1 shows the sizes of the token (i.e., signature) and the public keys in the signing set for each scheme. A BM_BLS token $\sigma \in \mathbb{G}_1$

[8]https://github.com/matter-labs/eip1962

is 64 bytes long, while a BM_SB token $\sigma = (\overline{R}, \overline{y}, \overline{z}) \in \mathbb{G}_1 \times \mathbb{Z}_p \times \mathbb{Z}_p$ is 128 bytes long. BM_BLS public keys can be aggregated into a single $\mathbb{G}_1$-element that is 64 bytes long, regardless of the number of signers/issuers. Conversely, BM_SB does not support public-key aggregation, meaning that all public keys in the signing set need to be transmitted. Moreover, BM_SB public keys need to be accompanied by corresponding proofs of possession (not reflected in the numbers in Table 1).

Construction	Token size			Public key size	
	$\mathbb{G}_1$	$\mathbb{Z}_p$	Bytes	$\mathbb{G}_1$	Bytes
BM_BLS	1	0	64	1	64
BM_SB	1	2	128	n	$64n$

Table 1: Token and public key sizes, assuming n signers.

Table 2 gives the communication costs for issuance. We record the number of bytes exchanged between the user and a *single* signer, expressed as a function of the total number of signers n. The data transferred from the user to a signer is denoted by $U \rightarrow S$, and $S \rightarrow U$ represents the data transferred from a signer to the user.

Figure 3 shows the execution times for issuance and verification of a single token, averaged over 10 trials. (Measurements were performed on a 2021 MacBook Pro laptop with a 10-core Apple M1 Pro processor and 16 GB of RAM.) We use Python co-routines to simulate communication between the user and signers, hence latency costs are excluded. Issuance costs account for the user's cost plus the cost of all signers. Key aggregation costs are not accounted for. BM_BLS verification involves just a single pairing check, irrespective of number of issuers, while BM_SB verification requires multiple EC additions and EC multiplications, the amount of which grows proportionally to the number of issuers. The main bottleneck for BM_BLS verification is the EC multiplications performed by the hash-to-curve operations (more details in the next section).

6.2 Smart-Contract Implementation

We envision that blockchain applications can leverage blind multisignatures to enable a set of signers to issue tokens off-chain that can then be verified by smart contracts on-chain. An example of such an application is a DAO with tokenized anonymous voting, as discussed in the introduction. As such, we also implemented an Ethereum Solidity smart contract performing token verification for BM_BLS and BM_SB.

For Ethereum compatibility and efficiency, we use the minSig approach [4] (reducing signature size at the expense of an increase in the public-key size). In addition, in our BM_BLS implementation we switch to asymmetric, Type-3 pairings. This means public keys are now of the form $(X_1, X_2) = (g_1^x, g_2^x) \in \mathbb{G}_1 \times \mathbb{G}_2$, as the user needs X_1 to unblind, while verification and key aggregation rely on X_2. A key's validity needs to be verified by checking $e(X_1, g_2) \stackrel{?}{=} e(g_1, X_2)$. When moving to the asymmetric setting, we have to use a version of the AGM for asymmetric pairings [5, 17], and unforgeability will require the co-qdlog assumption [5].

Constr.	r	$U \rightarrow S$		$S \rightarrow U$		Bytes exchanged	
		$\mathbb{G}$	$\mathbb{Z}_p$	$\mathbb{G}$	$\mathbb{Z}_p$	Per round	Total
BM_BLS	1	1	0	1	0	128	128
BM_SB	1	0	n	2	1	$32n + 160$	$64n + 224$
	2	0	$n - 1$	0	2	$32n + 32$	
	3	0	0	0	1	32	

Table 2: Communication overhead for token issuance, measured between the user and a single signer, as a function of the total number of signers n.

For practical on-chain token verification, we use Ethereum's BN254 pre-compiled contracts to perform group operations and asymmetric pairing checks at reduced gas costs [10, 11, 52]. We adopt the hash-to-curve implementation of Fouque and Tibouchi [23, 31, 50], since the constant-time "hash and pray" alternative is vulnerable to a gas griefing attack [41]. Our smart contract maintains a nullifier D that keeps track of which tokens have been verified; upon successful verification of a token σ, it adds $\mathsf{H}(\sigma)$ to D.

In Figure 4, we show the gas costs for verifying a single token via our smart contract. We exclude the one-time cost of public key aggregation, but we include the cost of checking whether the token has already been presented (i.e., checking whether $\mathsf{H}(\sigma) \in D$) and storing the hash of the token in the nullifier. BM_BLS verification requires 2 pairings and a single hash-to-curve operation; it costs $\approx 232K$ gas, irrespective of the number of signers. On the other hand, the BM_SB verification cost grows with the number of signers n. Verifying a BM_SB token requires computing n hashes (SHA-256, in our implementation), $n + 2$ elliptic-curve additions, and $3n + 2$ elliptic-curve multiplications. The cost of verifying a BM_SB token exceeds that of a BM_BLS token for ≥ 11 signers.

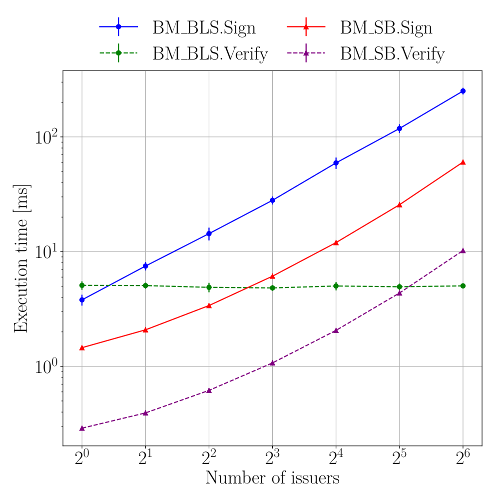

Figure 3: Execution times for issuance and verification of a single token.

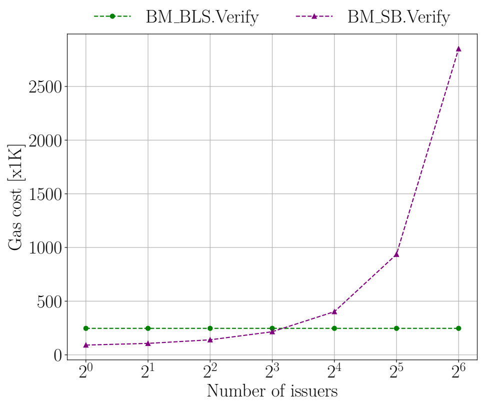

Figure 4: Ethereum gas cost for token verification, as a function of the number of signers.

An important benefit of BM_BLS is that it supports aggregation of tokens that share the same issuer set (as described in Section 4). This can be used to improve verification costs. Table 3 shows the gas costs for verifying aggregate tokens issued by the same set of 11 issuers. (We use a set of size 11 since that is the threshold at which the cost of verifying a BM_SB token exceeds that of a BM_BLS token.) Costs include the fixed transaction base fee (21K gas), hashing to curve, performing curve additions, multiplications, and pairings, and checking/maintaining the nullifier; since key aggregation only needs to be done once, we do not include it in the costs. Note that verifying more than ≈ 260 BM_BLS or ≈ 100 BM_SB tokens will exceed Ethereum's ≈ $30M$ block gas limit.

	BM_BLS.Ver (m)		BM_SB.Ver (m)
Num. of tokens	**Total**	**Amortized**	**Total**
1	232,083	232,083	279,746
2	370,692	185,346	559,492
4	576,120	144,030	1,118,984
8	1,011,144	126,393	2,237,968
16	1,814,572	113,411	4,475,936
32	3,615,310	112,978	8,951,872
64	6,966,217	108,847	17,903,744
128	13,846,324	108,174	35,807,488
256	27,277,340	106,552	71,614,976
512	54,596,438	106,634	143,229,952
1024	110,386,321	107,799	286,459,904

Table 3: Total and amortized gas costs for verifying multiple tokens issued by the same set of 11 issuers.

Excluding public-key aggregation, verifying an aggregate of ℓ BM_BLS tokens requires 2ℓ curve additions, ℓ hash-to-point invocations, and 2 pairings. On the other hand, BM_SB tokens are not aggregatable, so their verification cost grows linearly with the number of tokens being verified. The BM_SB token verification cost also grows with the number of signers, as described earlier.

Acknowledgements

I. Karantaidou was funded by a Protocol Labs fellowship. O. Renawi and J. Loss were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 507237585, and by the European Union, ERC-2023-STG, Project ID: 101116713. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. F. Baldimtsi was funded by NSF #2143287.

References

[1] Ghous Amjad, Kevin Yeo, and Moti Yung. 2023. RSA Blind Signatures with Public Metadata. Available at https://eprint.iacr.org/2023/1199.

[2] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. 2020. Privacy-preserving auditable token payments in a permissioned blockchain system. In *AFT 2020*. ACM, 255–267.

[3] Aragon ZK Research (AZKR). 2023. Nouns Private Voting Research Sprint - Technical Report. https://research.aragon.org/nouns-tech.html.

[4] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, François Garillot, Jonas Lindstrøm, Ben Riva, Arnab Roy, Alberto Sonnino, Pun Waiwitlikhit, and Joy Wang. 2024. Subset-optimized BLS Multi-signature with Key Aggregation. In *Financial Cryptography 2024*. Available at https://eprint.iacr.org/2023/498.

[5] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. 2020. A Classification of Computational Assumptions in the Algebraic Group Model. In *Advances in Cryptology – CRYPTO 2020, Part II (Lecture Notes in Computer Science, Vol. 12171)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 121–151. https://doi.org/10.1007/978-3-030-56880-1_5

[6] Mihir Bellare and Gregory Neven. 2006. Multi-signatures in the plain public-Key model and a general forking lemma. In *ACM CCS 2006: 13th Conference on Computer and Communications Security*, Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati (Eds.). ACM Press, Alexandria, Virginia, USA, 390–399. https://doi.org/10.1145/1180405.1180453

[7] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography (Lecture Notes in Computer Science, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, Heidelberg, Germany, Miami, FL, USA, 31–46. https://doi.org/10.1007/3-540-36288-6_3

[8] Dan Boneh, Manu Drijvers, and Gregory Neven. 2018. Compact Multi-signatures for Smaller Blockchains. In *Advances in Cryptology – ASIACRYPT 2018, Part II (Lecture Notes in Computer Science, Vol. 11273)*, Thomas Peyrin and Steven Galbraith (Eds.). Springer, Heidelberg, Germany, Brisbane, Queensland, Australia, 435–464. https://doi.org/10.1007/978-3-030-03329-3_15

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *Advances in Cryptology – ASIACRYPT 2001 (Lecture Notes in Computer Science, Vol. 2248)*, Colin Boyd (Ed.). Springer, Heidelberg, Germany, Gold Coast, Australia, 514–532. https://doi.org/10.1007/3-540-45682-1_30

[10] Vitalik Buterin and Christian Reitwiessner. 2017. EIP-197: Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128. https://eips.ethereum.org/EIPS/eip-197.

[11] Antonio Salazar Cardozo and Zachary Williamson. 2018. EIP-1108: Reduce alt_bn128 precompile gas costs. https://eips.ethereum.org/EIPS/eip-1108.

[12] Celo. 2021. Celo BLS Snarks. https://github.com/celo-org/celo-bls-snark-rs.

[13] Melissa Chase, F. Betül Durak, and Serge Vaudenay. 2023. Anonymous Tokens with Stronger Metadata Bit Hiding from Algebraic MACs. In *Advances in Cryptology—Crypto 2023, Part II (LNCS, Vol. 14082)*. Springer, 418–449. https://doi.org/10.1007/978-3-031-38545-2_14

[14] David Chaum. 1983. Blind Signature System. In *Advances in Cryptology – CRYPTO'83*, David Chaum (Ed.). Plenum Press, New York, USA, Santa Barbara, CA, USA, 153.

[15] Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. 2003. ID-based multi-proxy signature and blind multisignature from bilinear pairings. *Proc. KIISC* 3 (2003), 11–19.

[16] Chia Network. 2022. BLS Signatures. https://github.com/Chia-Network/bls-signatures.

[17] Geoffroy Couteau and Dominik Hartmann. 2020. Shorter Non-interactive Zero-Knowledge Arguments and ZAPs for Algebraic Languages. In *Advances in Cryptology – CRYPTO 2020, Part III (Lecture Notes in Computer Science, Vol. 12172)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 768–798. https://doi.org/10.1007/978-3-030-56877-1_27

[18] Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. 2023. Snowblind: A Threshold Blind Signature in Pairing-Free Groups. In *Advances in Cryptology – CRYPTO 2023, Part I (Lecture Notes in Computer Science, Vol. 14081)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer,

Heidelberg, Germany, Santa Barbara, CA, USA, 710–742. https://doi.org/10.1007/978-3-031-38557-5_23

[19] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *Proc. PETS* 2018, 3 (2018), 164–180. https://doi.org/10.1515/POPETS-2018-0026

[20] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. 2023. Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance. Available at https://eprint.iacr.org/2023/602.

[21] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. 2019. On the Security of Two-Round Multi-Signatures. In *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, San Francisco, CA, USA, 1084–1101. https://doi.org/10.1109/SP.2019.00050

[22] Marc Fischlin. 2006. Round-Optimal Composable Blind Signatures in the Common Reference String Model. In *Advances in Cryptology – CRYPTO 2006 (Lecture Notes in Computer Science, Vol. 4117)*, Cynthia Dwork (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 60–77. https://doi.org/10.1007/11818175_4

[23] Pierre-Alain Fouque and Mehdi Tibouchi. 2012. Indifferentiable Hashing to Barreto—Naehrig Curves. In *LatinCrypt 2012 (LNCS)*. Springer, 1–17.

[24] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. 2018. The Algebraic Group Model and its Applications. In *Advances in Cryptology – CRYPTO 2018, Part II (Lecture Notes in Computer Science, Vol. 10992)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 33–62. https://doi.org/10.1007/978-3-319-96881-0_2

[25] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. 2020. Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model. In *Advances in Cryptology – EUROCRYPT 2020, Part II (Lecture Notes in Computer Science, Vol. 12106)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, Germany, Zagreb, Croatia, 63–95. https://doi.org/10.1007/978-3-030-45724-2_3

[26] Christina Garman, Matthew Green, and Ian Miers. 2014. Decentralized Anonymous Credentials. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*. The Internet Society, San Diego, CA, USA.

[27] Lein Harn. 1994. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proc. Computers and Digital Techniques* 141, 5 (1994), 307–313.

[28] Eduard Hauck, Eike Kiltz, and Julian Loss. 2019. A Modular Treatment of Blind Signatures from Identification Schemes. In *Advances in Cryptology – EUROCRYPT 2019, Part III (Lecture Notes in Computer Science, Vol. 11478)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, Germany, Darmstadt, Germany, 345–375. https://doi.org/10.1007/978-3-030-17659-4_12

[29] Chloé Hébant and David Pointcheval. 2022. Traceable Constant-Size Multi-authority Credentials. In *SCN 2022 (LNCS, Vol. 13409)*. Springer, 411–434.

[30] IETF. 2019. IETF BLS Signature Scheme. https://datatracker.ietf.org/doc/draft-boneh-bls-signature/.

[31] IETF. 2022. IETF Hashing to Elliptic Curves. https://datatracker.ietf.org/doc/draft-irtf-cfrg-hash-to-curve/.

[32] Julia Kastner, Julian Loss, and Omar Renawi. 2023. Concurrent Security of Anonymous Credentials Light, Revisited. In *ACM CCS 2023*. ACM, 45–59.

[33] Julia Kastner, Julian Loss, and Jiayu Xu. 2022. On Pairing-Free Blind Signature Schemes in the Algebraic Group Model. In *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II (Lecture Notes in Computer Science, Vol. 13178)*, Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe (Eds.). Springer, Heidelberg, Germany, Virtual Event, 468–497. https://doi.org/10.1007/978-3-030-97131-1_16

[34] Jinho Kim, Kwangjo Kim, and Chulsoo Lee. 2002. An Efficient and Provably Secure Threshold Blind Signature. In *ICISC 01: 4th International Conference on Information Security and Cryptology (Lecture Notes in Computer Science, Vol. 2288)*, Kwangjo Kim (Ed.). Springer, Heidelberg, Germany, Seoul, Korea, 318–327.

[35] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2020. Anonymous Tokens with Private Metadata Bit. In *Advances in Cryptology—Crypto 2020, Part I (LNCS, Vol. 12170)*. Springer, 308–336.

[36] Alex Kroeger. 2023. Gas Prices Dashboard. https://dune.com/kroeger0x/gas-prices.

[37] Veronika Kuchta and Mark Manulis. 2014. Rerandomizable threshold blind signatures. In *International Conference on Trusted Systems*. Springer, 70–89.

[38] Susan K. Langford. 1996. Weakness in Some Threshold Cryptosystems. In *Advances in Cryptology – CRYPTO'96 (Lecture Notes in Computer Science, Vol. 1109)*, Neal Koblitz (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 74–82. https://doi.org/10.1007/3-540-68697-5_6

[39] Chin-Laung Lei, Wen-Shenq Juang, and Pei-Ling Yu. 2002. Provably secure blind threshold signatures based on discrete logarithm. *Journal of Information Science and Engineering* 18, 1 (2002), 23–39.

[40] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. 1995. Threshold-Multisignature Schemes where Suspected Forgery Implies Traceability of Adversarial Shareholders. In *Advances in Cryptology – EUROCRYPT'94 (Lecture Notes in Computer Science, Vol. 950)*, Alfredo De Santis (Ed.). Springer, Heidelberg, Germany, Perugia, Italy, 194–204. https://doi.org/10.1007/BFb0053435

[41] Chih Cheng Liang and Kobi Gurkan. 2020. Non-constant time hash to point attack vector. https://github.com/thehubbleproject/hubble-contracts/issues/171. Accessed: 2024-04-26.

[42] Rongxing Lu, Zhenfu Cao, and Yuan Zhou. 2005. Proxy blind multi-signature scheme without a secure channel. *Applied mathematics and computation* 164, 1 (2005), 179–187.

[43] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. 2006. Sequential Aggregate Signatures and Multisignatures Without Random Oracles. In *Advances in Cryptology – EUROCRYPT 2006 (Lecture Notes in Computer Science, Vol. 4004)*, Serge Vaudenay (Ed.). Springer, Heidelberg, Germany, St. Petersburg, Russia, 465–485. https://doi.org/10.1007/11761679_28

[44] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. 2018. Simple Schnorr Multi-Signatures with Applications to Bitcoin. Cryptology ePrint Archive, Report 2018/068. https://eprint.iacr.org/2018/068.

[45] Markus Michels and Patrick Horster. 1996. On the Risk of Disruption in Several Multiparty Signature Schemes. In *Advances in Cryptology – ASIACRYPT'96 (Lecture Notes in Computer Science, Vol. 1163)*, Kwangjo Kim and Tsutomu Matsumoto (Eds.). Springer, Heidelberg, Germany, Kyongju, Korea, 334–345. https://doi.org/10.1007/BFb0034859

[46] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. 2023. Aggregate Signatures with Versatile Randomization and Issuer-Hiding Multi-Authority Anonymous Credentials. In *ACM CCS 2023*. ACM, 30–44.

[47] Jonas Nick, Tim Ruffing, and Yannick Seurin. 2021. MuSig2: Simple Two-Round Schnorr Multi-signatures. In *Advances in Cryptology – CRYPTO 2021, Part I (Lecture Notes in Computer Science, Vol. 12825)*, Tal Malkin and Chris Peikert (Eds.). Springer, Heidelberg, Germany, Virtual Event, 189–221. https://doi.org/10.1007/978-3-030-84242-0_8

[48] Kazuo Ohta and Tatsuaki Okamoto. 1999. Multi-signature schemes secure against active insider attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 82, 1 (1999), 21–31.

[49] H Petersen, P Horster, and M Michels. 1995. Blind multisignature schemes and their relevance to electronic voting. In *11th Annual Computer Security Applications Conference*. IEEE, 149–155.

[50] Hubble Project. 2020. Hubble Project Solidity BLS Implementation. https://github.com/thehubbleproject/hubble-contracts/blob/master/contracts/libs/BLS.sol.

[51] Deevashwer Rathee, Guru Vamsi Policharla, Tiancheng Xie, Ryan Cottone, and Dawn Song. 2023. ZEBRA: SNARK-based Anonymous Credentials for Practical, Private and Accountable On-chain Access Control. Available at https://eprint.iacr.org/2022/1286.

[52] Christian Reitwiessner. 2017. EIP-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128. https://eips.ethereum.org/EIPS/eip-196.

[53] Thomas Ristenpart and Scott Yilek. 2007. The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks. In *Advances in Cryptology – EUROCRYPT 2007 (Lecture Notes in Computer Science, Vol. 4515)*, Moni Naor (Ed.). Springer, Heidelberg, Germany, Barcelona, Spain, 228–245. https://doi.org/10.1007/978-3-540-72540-4_13

[54] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad S. Wahby. 2020. *Pairing-Friendly Curves*. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-07. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves/07/

[55] Claus-Peter Schnorr. 2001. Security of Blind Discrete Log Signatures against Interactive Attacks. In *ICICS 01: 3rd International Conference on Information and Communication Security (Lecture Notes in Computer Science, Vol. 2229)*, Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou (Eds.). Springer, Heidelberg, Germany, Xian, China, 1–12.

[56] Tjerand Silde and Martin Strand. 2022. Anonymous Tokens with Public Metadata and Applications to Private Contact Tracing. In *Financial Cryptography 2022 (LNCS, Vol. 13411)*. Springer, 179–199.

[57] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. 2019. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers. In *NDSS 2019*. The Internet Society.

[58] Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. 2022. Utt: Decentralized ecash with accountable privacy. Available at https://eprint.iacr.org/2022/452.

[59] Raylin Tso, Zi-Yuan Liu, and Yi-Fan Tseng. 2019. Identity-based blind multisignature from lattices. *IEEE Access* 7 (2019), 182916–182923.

A Cryptographic Definitions

A.1 Blind Signatures

Definition 11 (Correctness). A blind signature scheme is correct if for BS.Sign $\langle \mathcal{U}(m, pk), \mathcal{S}(sk) \rangle \rightarrow \sigma$, then BS.Ver $(pk, m, \sigma) = 1$ with overwhelming probability.

For one-more unforgeability, the adversary is the user $\mathcal{U}$ that has to forge a signature on pk^* that did not come out of its interactions with signer that holds the secret to pk^*. The the one-more unforgeability game goes as follows: The challenger is going to fix the honest signer key pair (sk^*, pk^*) and respond to the adversary's (A) signing queries with signatures on messages of the A's choice. After q complete signing sessions during query phase, A has to submit $q + 1$ signatures on distinct messages during the forgery phase. If all signatures verify correctly, it means that a forgery has happened and A wins the game. Since the challenger cannot test whether a message was signed during the query phase (because of blindness), when the number of submitted signatures during the forgery exceeds the number of signatures seen during the query phase, a forgery took place.

Below we give the formal one-more unforgeability definition. We first define the signing oracle Sign_{sk^*}.

Sign_{sk^*}: Its functionality is similar to the oracle defined for multi-signatures in Section A.2. It simulates the honest signer's execution of BS.Sign $\langle \mathcal{S}(sk^*) \rangle$ and it outputs some intermediate value that can be used to compute the final signature.

Definition 12 (One-More Unforgeability). For a blind signature scheme BS, let game $\mathsf{OMUF}_A^{\mathsf{BS}}$ be the following game:

- **Setup:** The challenger generates the parameters and a key pair (sk^*, pk^*). It runs the adversary $A(\mathsf{par}, pk^*)$.
- **Queries:** A interacts with the signing oracle $\mathsf{Sign}_{sk^*}{}^{(sk^*)}$.
- **Output:** A submits a tuple of forgeries $(\sigma_1^*, m_1^*), \ldots, (\sigma_{\ell+1}^*, m_{\ell+1}^*)$ and wins if $\mathsf{BS.Ver}(pk^*, m_i, \sigma_i) = 1\ \forall i \in [\ell + 1]$ and the number of valid signatures received from its interaction with the challenger during the Query phase is not more than ℓ. If A wins, the game outputs true.

Then BS is one-more unforgeable (OMUF) if for any probabilistic polynomial time adversary A,

$$\mathsf{Adv}_{A,\mathsf{BS}}^{\mathsf{OMUF}}(\lambda) := \Pr[\mathsf{OMUF}_A^{\mathsf{BS}} = \text{true}] = \text{negl}(\lambda)$$

The second security property of blind signatures is that of blindness, i.e. the signer or a third party looking at a signing transcript cannot link a signature to its corresponding signing session.

The idea of the blindness game is the following: The adversarial entity A is the signer. In the malicious signer model [22], a key pair (sk, pk) and two messages are picked by the signer A. The challenger of the blindness game can interact with A and outputs two signatures. The signatures correspond to the messages picked by A and A is allowed to keep the transcripts of the signing sessions. In order to win the game, A has to link each transcript to its corresponding message/signature.

Below we give the formal blindness definition.

Definition 13 (Blindness). Given a blind signature scheme BS = (KGen, Sign, Ver) let game $\mathsf{Blind}_A^{\mathsf{BS}}$ be the following game:

The adversary A picks a pair of keys (sk, pk) and messages m_0, m_1. The challenger picks $b \in \{0, 1\}$ and runs two signing sessions as the user. A participates in the signing sessions as the signer S and is given back σ_0, σ_1 by the challenger. A has to output a bit b', and wins if $b' = b$. If A wins, the game outputs true.

BS is blind if for any probabilistic polynomial time adversary A,

$$\mathsf{Adv}_{A,\mathsf{BS}}^{Blind}(\lambda) := \Pr[\mathsf{Blind}_A^{\mathsf{BS}} = \text{true}] \leq \frac{1}{2} + negl(\lambda).$$

A.2 Multi-Signatures

We define the security model for multi-signatures that support key aggregation.

Security Model. A multi-signature should satisfy the properties of *correctness* and *unforgeability* (i.e. an adversarial user should not be able to forge a signature that verifies under apk for a set of signers where at least one signer is honest).

We start with correctness for multi-signatures, which guarantees that if all signers participate honestly, then the final signature will verify under the aggregate key computed on their public keys.

Definition 14 (Correctness). A multi-signature scheme is correct if for every n, $i \in [n]$, $(sk_i, pk_i) \leftarrow \mathsf{MS.KGen}(1^\lambda)$ and for every m, if all signers with public keys in $\vec{K}$ participate in the interactive MS.Sign then the output is a signature σ such that MS.Ver (apk, m, σ) = 1 with overwhelming probability for apk = MS.KAgg $(\vec{K})$.

For unforgeability, even if an adversary has corrupted all but one signer with public key pk^*, the adversary should still not be able to forge a signature that verifies under an apk that includes pk^*. The honest keys (sk^*, pk^*) are generated and stored by the challenger. The unforgeability adversary can query on messages of its choice and see signatures under $\{pk^*\}$ or its supersets. In order for the adversary to win, it has to submit a forgery on a new message m^* signed by a set of public keys that includes pk^*.

Below we give the formal unforgeability definition. The signing oracle Sign_{sk^*} simulates one signer running algorithm Sign. It takes as input the parameters par, the signer's secret key sk^* and the message m. For concurrent security, the oracle runs many open sessions, each one identified by its session number, whereas in the sequential setting, the oracle returns only messages for the current open session and will not initiate a new one before this is complete.

Definition 15 (Unforgeability). For multisignature scheme MS, let $\mathsf{EUF\text{-}CMA}_A^{\mathsf{MS}}$ be the following game:

- **Setup:** The challenger generates a key pair (sk^*, pk^*) for the honest signer. It runs the adversary $A(\mathsf{par}, pk^*)$.
- **Queries:** A picks a message m and queries the signing oracle $\mathsf{Sign}_{sk^*}{}^{(sk^*, \cdot)}$. This step can be repeated multiple times for different inputs m.
- **Output:** A outputs $\sigma^*, m^*, \vec{K} = \{pk_1, \ldots, pk_n\}$ and succeeds if $pk^* \in \vec{K}$, no signing queries were made on m^*, and

$$\mathsf{MS.Ver}(\mathsf{KAgg}(\vec{K}), m^*, \sigma^*) = 1.$$

MS is EUF-CMA-secure (existentially unforgeable under chosen-message attacks) if for any PPT adversary A,

$$\mathsf{Adv}_{A,\mathsf{MS}}^{\mathsf{EUF\text{-}CMA}}(\lambda) := \Pr[\mathsf{EUF\text{-}CMA}_A^{\mathsf{MS}} = \text{true}] = negl(\lambda)$$

A stronger adversary. A stronger definition for unforgeability requires the keys set $\vec{K}$ to be known to the signers and the adversary A has never queried $\mathsf{Sign}_{sk^*}{}^{(sk^*, m^*, \vec{K}^*)}$, where $\vec{K}^*$ was used in the adversary's forgery. It is satisfied by Schnorr-based multi-signature schemes [44, 47]. In these schemes, the set of signers is embedded in the signature share and cannot be easily changed.

A.3 The ROS Problem

Definition 16 (Random Inhomogeneities in an Overdetermined System of Linear Equations (ROS) [25, 55]). Let $\mathbb{G}$ be a group of prime order q with generator g. For a positive integer $\ell \in \mathbb{Z}^+$, an adversary A, a hash function $\mathsf{H}_{\text{ros}} : \mathbb{Z}_q^\ell \times \Omega \to \mathbb{Z}_q$ modeled as a random oracle for an arbitrary set Ω, define the game ℓ-ROS as follows.

- **Setup.** Ais executed with q and ℓ as input, and it gets oracle access to H_{ros}.
- **Online phase.** The game simulates the oracle H_{ros} for Aand responds to its queries.
- **Output.** The game outputs 1 iff Aterminates and outputs (i) pair-wise distinct tuples $(\vec{\rho}_1, aux_1), \ldots, (\vec{\rho}_{\ell+1}, aux_{\ell+1})$, and (ii) $(c_1, \ldots, c_\ell) \in \mathbb{Z}_q^\ell$, such that for all $i \in [\ell+1]$ the equality

$$\sum_{j \in [\ell]} \rho_{i,j} \cdot c_j = \mathsf{H}_{\text{ros}}(\vec{\rho}_i, aux_i).$$

B Unforgeability of Blind BLS

Theorem 17. *Given an algebraic adversary A making $q-1$ parallel signing queries and Q random oracle queries against the blind BLS scheme, there is a* PPT *algorithm that breaks q-dlog in the ROM with probability at least* $(\frac{1}{2} - \frac{1}{p})\,\mathsf{Adv}^{\mathsf{OMUF}}_{A,\mathsf{BLS}}(\lambda)$.

Proof overview. Our proof involves two possible strategies on the challenger side, one of them is picked at random in the beginning: the secret x is either embedded in the random oracle's responses or in the honest signer's public key, the adversary A is unaware of the strategy used and unable to plan their attack accordingly. A successful forgery results in solving for x and in the first case reduces to the discrete logarithm problem and in the second case to the q-dlog problem with overwhelming probability. To picture why, in the second case, $q-1$ queries to the signing oracle are using q powers of the unknown x in the exponent. We use the AGM to exctract q equations from the forgery and we show that not all equations are trivial. Our reduction uses the q-dlog instance to simulate $q-1$ signing queries. The scheme's unforgeability overall relies on q-dlog assumption as the problem's hardness also implies the discrete logarithm assumption and covers both strategies.

Proof. For simplicity we focus first on the case where $q = 2$. So we have an adversary A who makes a single signing query and then outputs two forgeries. Let $X = g^x$ be the signing public key. Let $h_1, \ldots, h_Q$ be the responses A receives to its random-oracle queries. Let $\overline{m}_1$ denote the query A makes to the signing oracle. Note that A must also provide a representation of $\overline{m}_1$ with respect to $g, X, h_1, \ldots, h_Q$, so $\overline{m}_1 = g^\alpha X^\beta \prod_{i\in[Q]} h_i^{\gamma_i}$.

In response to the signing query, A receives $\bar{s}_1 = \overline{m}_1^x$. When A outputs its forgeries (m_1, σ_1) and (m_2, σ_2), we assume w.l.o.g. that m_1, m_2 were queried to the random oracle, and arrange the indices so that $\mathsf{H}(m_1) = h_1$ and $\mathsf{H}(m_2) = h_2$. If these are valid forgeries then $\sigma_j = h_j^x$ for $j = 1, 2$. Note further that A must provide representations of σ_1, σ_2, say $\sigma_j = g^{\alpha_j} X^{\beta_j} \left(\prod_{i\in[Q]} h_i^{\gamma_{i,j}}\right) \cdot \bar{s}_1^{\delta_j}$, for $j = 1, 2$. Thus,

$$\begin{aligned} h_j^x = \sigma_j &= g^{\alpha_j} X^{\beta_j} \left(\prod_i h_i^{\gamma_{i,j}}\right) \cdot \bar{s}_1^{\delta_j} \\ &= g^{\alpha_j} g^{\beta_j x} \left(\prod_i h_i^{\gamma_{i,j}}\right) \cdot \overline{m}_1^{\delta_j \cdot x} \\ &= g^{\alpha_j} g^{\beta_j x} \left(\prod_i h_i^{\gamma_{i,j}}\right) \left(g^\alpha g^{\beta x} \prod_i h_i^{\gamma_i}\right)^{\delta_j \cdot x} \end{aligned} \quad (1)$$

or, by some algebra:

$$g^{\alpha_j + \beta'_j x + \beta''_j x^2} \prod_i h_i^{\gamma_{i,j} + \gamma'_{i,j} x} = 1 \quad (2)$$

for $j = 1, 2$, where $\beta'_j, \beta''_j, \gamma'_{i,j}$ are efficiently computable.

There are now two cases: either for some j, i the exponent of h_i in the jth equation (i.e., $\gamma_{i,j} + \gamma'_{i,j} x$) is non-zero, or for all j, i the exponent of h_i in the jth equation is zero. If A succeeds with probability ϵ, then either the first or second case must happen with probability at least $\epsilon/2$.

Assume the first case happens with probability at least $\epsilon/2$. We can use this to solve the discrete-logarithm problem with probability at least $\epsilon/2 - 1/p$ as follows. Given Y, set the public key to $X = g^x$ for known, uniform $x \in \mathbb{Z}_p$. For the ith hash query, program the response to be $h_i = g^{s_i} Y^{r_i}$ for uniform $s_i, r_i \in \mathbb{Z}_p^*$. (If $h_i = 1$ for some i then we can solve for $\log_g Y$ directly, so we assume this does not happen in what follows.) Since x is known, queries to the signing oracle can be answered easily. If A forges and the first case happens then, except with probability $1/p$, we get an equation of the form $g^A Y^B = 1$ with A, B known and $B \neq 0$, which allows us to solve for $\log_g Y$. To see this is the case, note that all the exponents in (2) are known, and so we have an equation of the form $g^a \cdot \prod_i (g^{s_i} Y^{r_i})^{b_i} = g^{a + \sum_i b_i s_i} \cdot Y^{\sum_i b_i r_i} = 1$, with a and the $\{b_i\}$ known, and at least one of the $\{b_i\}$ non-zero. Letting i be the largest index for which b_i is non-zero and viewing $\{b_j\}_{j<i}$ as fixed, note that r_i is uniform from A's point of view and there is at most one non-zero value of b_i for which $\sum_i b_i r_i = 0$. This concludes the analysis of the first case.

Before continuing, we analyze (2) in more detail. Assume we are in the second case, so for all j, i we have $\gamma_{i,j} + \gamma'_{i,j} x = 0$. Call an equation of this form *trivial* if $\gamma'_{i,j} = 0$ (which implies $\gamma_{i,j} = 0$). We claim that it is not possible for all equations to be trivial. To see this, note that (using equation (1))

$$\gamma'_{i,j} = \begin{cases} \delta_j \cdot \gamma_i & i \neq j \\ \delta_j \cdot \gamma_i - 1 & i = j. \end{cases}$$

Thus, all equations are trivial only if $\delta_1, \delta_2, \gamma_1, \gamma_2$ are such that

$$\begin{aligned} \delta_1 \cdot (\gamma_1,\ \gamma_2) &= (1,\ 0) \\ \delta_2 \cdot (\gamma_1,\ \gamma_2) &= (0,\ 1). \end{aligned}$$

But since the vector $(\gamma_1, \gamma_2) \in \mathbb{Z}_p^2$ spans a vector space of dimension at most 1, this is impossible.

Returning to the main proof, assume the second case happens with probability at least $\epsilon/2$; we use this to solve the 2-dlog assumption with probability at least $\epsilon/2$. Given $Y_1 = g^x, Y_2 = g^{x^2}$, we set the public key equal to $X = Y_1$ and program $h_i = g^{r_i}$ (for uniform $r_i \in \mathbb{Z}_p^*$) for all i. When A makes signing query

$\overline{m}_1 = g^{\alpha} X^{\beta} \prod_i h_i^{\gamma_i}$, we answer it with $\bar{s}_1 = Y_1^{\alpha} Y_2^{\beta} \prod_i Y_1^{r_i \gamma_i}$. When A outputs its forgeries, we derive equations as in (2), e.g.,

$$g^{\alpha_j} Y_1^{\beta'_j} Y_2^{\beta''_j} \prod_i g^{r_i \gamma_{i,j}} Y_1^{r_i \gamma'_{i,j}} = 1$$

for $j = 1, 2$, where $\gamma_{i,j}, \gamma'_{i,j}$ are efficiently computable. Since we are in the second case, we know that $g^{r_i \gamma_{i,j}} Y_1^{r_i \gamma'_{i,j}} = 1$ for all i, j. As we have shown above, it is not possible for all equations to be trivial; thus, for some i, j it must hold that $\gamma'_{i,j} \neq 0$. We can use any such non-trivial equation to solve for $x = \log_g Y_1$. This completes the proof when $q = 2$.

We sketch how to extend the above argument for arbitrary $q > 2$. The key thing that changes is that now A's queries to the signing oracle can also depend on answers to previous signing queries. Thus, in general, when A makes its jth query $\overline{m}_j$ to the signing oracle it now provides a representation in terms of $g, X, \{h_i\}_{i \in [Q]}$, and $\{\bar{s}_i\}_{i<j}$. However, it is easy to show by induction that this allows derivation of a representation of the form

$$\overline{m}_j = g^{\sum_{k=0}^{j} \beta_{j,k}\, x^k} \cdot \prod_{i \in [Q]} h_i^{\sum_{k=0}^{j-1} \gamma_{i,j,k}\, x^k} \tag{3}$$

where the $\{\beta_{j,k}\}$ and $\{\gamma_{i,j,k}\}$ are efficiently computable. Thus, the analogue of (2) for the q forgeries output by A becomes

$$g^{\sum_{k=0}^{q} \beta'_{j,k}\, x^k} \prod_{i \in [Q]} h_i^{\sum_{k=0}^{q-1} \gamma'_{i,j,k}\, x^k} = 1 \tag{4}$$

for $j = 1, \ldots, q$, where the $\{\beta'_{j,k}\}$ and $\{\gamma'_{i,j,k}\}$ are efficiently computable. As before, we have two cases: either for some j, i the exponent of h_i in the jth equation is non-zero, or for all j, i the exponent of h_i in the jth equation is zero. A reduction to the discrete-logarithm problem in the first case is the same as before, so we focus on the second case.

As before, in the second case we have $\sum_{k=0}^{q-1} \gamma'_{i,j,k}\, x^k = 0$ for all j, i, and we call an equation of this form *trivial* if $\gamma'_{i,j,1} = 0$. We again claim that it is impossible for all equations to be trivial. Indeed, define the vectors $\vec{\gamma}_j = (\gamma_{1,j,1}, \ldots, \gamma_{q,j,1}) \in \mathbb{Z}_p^q$ for $j = 1, \ldots, q-1$, where $\vec{\gamma}_j$ corresponds to the vector of exponents of $h_1, \ldots, h_q$ for the jth signing query of A. Then all equations can be trivial only if there exist $\delta_{1,1}, \ldots, \delta_{q,q-1}$ such that $\sum_{j=1}^{q-1} \delta_{i,j} \cdot \vec{\gamma}_j = e_i \in \mathbb{Z}_p^q$, for $i = 1, \ldots, q$, where e_i is the vector that is 1 at position i and 0 everywhere else. But since the $\{\vec{\gamma}_j\}$ span a vector space of dimension at most $q-1$, and the $\{e_i\}$ are q linearly independent vectors, this is clearly impossible.

With this in place, we now show how to solve the q-dlog assumption when the second case happens with probability at least $\epsilon/2$. Given $Y_1 = g^x, \ldots, Y_q = g^{x^q}$, we set the public key equal to $X = Y_1$ and program $h_i = g^{r_i}$, where $r_i \in \mathbb{Z}_p^*$ is uniform. When A makes its jth signing query as in (3), we answer it with

We have $\prod_{k=0}^{j} Y_{k+1}^{\beta_{j,k}} \cdot \prod_{i \in [Q]} \prod_{k=0}^{j-1} Y_{k+1}^{r_i \cdot \gamma_{i,j,k}}$ for $j = 1, \ldots, q-1$. When A outputs its forgeries, we derive equations as in (4) and find i, j for which $\sum_{k=0}^{q-1} \gamma'_{i,j,k}\, x^k = 0$ and $\gamma'_{i,j,1} \neq 0$. We then use that equation to solve for x. □

C Proofs for BM_BLS

C.1 Blindness

We show:

Theorem 18. BM_BLS *is unconditionally blind.*

Proof. Let A be an adversary controlling n signers, picking two messages m_0, m_1, and playing the game of Definition 3. At the end of the game, A holds two transcripts $t_b = \{\overline{m}_1^b, \ldots, \overline{m}_n^b, \bar{s}_1^b, \ldots, \bar{s}_n^b\}$ and $t_{1-b} = \{\overline{m}_1^{1-b}, \ldots, \overline{m}_n^{1-b}, \bar{s}_1^{1-b}, \ldots, \bar{s}_n^{1-b}\}$ and signatures σ_0, σ_1. All elements in t_b, t_{1-b} are independent from $m_0, m_1, \sigma_0, \sigma_1$. □

C.2 One-More Unforgeability of BM_BLS

We sketch a proof of Theorem 4 based on the proof of Theorem 17.

Proof. A's output forgery consists of $(m_1, \sigma_1, \vec{K}_1), \ldots, (m_q, \sigma_q, \vec{K}_q)$. Let $X = g^x$ be the public key of the honest signer, given to A. For the forgery to be valid, X will be included in all key sets $\vec{K}_1, \ldots, \vec{K}_q$. Let $h_1, \ldots, h_Q$ be the responses A receives to its random-oracle queries. Let $\overline{m}_j$ be the jth query to the signing oracle and $\bar{s}_j$ the output. Together with $\overline{m}_j$, A provides a representation in terms of $g, X, \{h_i\}_{i \in [Q]}$, and $\{\bar{s}_i\}_{i<j}$, i.e. all the group elements given so far, including previous signing queries. For every element Z submitted in the forgery ($Z \in \{\sigma_j\}$ for $j \in [q]$, or $Z \in \{X_{j,t}\}$, $X_{j,t} \in \vec{K}_j$ and for $t \in [|\vec{K}_j|]$), it also provides a representation in terms of $g, X, \{h_i\}_{i \in [Q]}$, and $\{\bar{s}_i\}_{i \in [q-1]}$. Z can also be written in the following form $Z = g^{\sum_{k=0}^{q} \beta_{j,k}\, x^k} \cdot \prod_{i \in [Q]} h_i^{\sum_{k=0}^{q} \gamma_{i,j,k}\, x^k}$ where the $\{\beta_{j,k}\}$ and $\{\gamma_{i,j,k}\}$ are efficiently computable.

When A outputs its forgery, we assume it has queried the random oracle $\mathsf{H}_{\mathrm{agg}}$ for every element in $\vec{K}_j$ that outputs an element $a_{j,t}$ in $\mathbb{Z}_p^*$. Without loss of generality, we assume that the honest signer's key appears first in every set $\vec{K}_j$, $X_{j,1} = X$. We also assume w.l.o.g. that $m_1, \ldots, m_q$ were queried to the random-oracle H, and arrange the indices so that $\mathsf{H}(m_1) = h_1, \ldots, \mathsf{H}(m_q) = h_q$.

From the validity of the signatures it holds that

$$\sigma_j = h_j^{a_{j,1} x + \sum_t a_{j,t} (\log_g X_{j,t} + \sum_{i \in [Q]} \log_{h_i} X_{j,t})} \tag{5}$$

for $j \in [q]$, $i \in [Q]$ and $t \in [|\vec{K}_j|]$. Since all $\sigma_j, X_{j,t}$ have the form of Z, we can efficiently move all terms in one side, group the exponents and derive an equation of the form

$$g^{\sum_{k=0}^{q} \beta'_{j,k}\, x^k} \prod_{i \in [Q]} h_i^{\sum_{k=0}^{q} \gamma'_{i,j,k}\, x^k} = 1 \tag{6}$$

for $j = 1, \ldots, q$, where $\{\beta'_{j,k}, \gamma'_{i,j,k}\}$ are efficiently computable.

As before, we have two cases: either for some j, i the exponent of h_i in the jth equation is non-zero, or for all j, i the exponent of h_i in the jth equation is zero. If A succeeds with probability ϵ, then either the first or second case must happen with probability at least $\epsilon/2$. When the second case happens, we derive $\delta_{1,1}, \ldots, \delta_{q,q-1}$ such that $\sum_{j=1}^{q-1} \delta_{i,j} \cdot \vec{\gamma}_j = e_i \in \mathbb{Z}_p^q$ for $i = 1, \ldots, q$, where $\vec{\gamma}_j$ corresponds to the vector of exponents of $h_1, \ldots, h_q$ for the jth signing query of A. $e_i = \vec{0}$ happens when the adversary outputs a signature σ_i such that the exponent of h_i in (5) has the linear term in x equal to zero. Since A does not control the outputs $\vec{a}_i = (a_{i,1}, \ldots, a_{i,|\vec{K}_i|})$

of $\mathsf{H}_{\mathsf{agg}}$, this happens with probability Q/p. From the union bound, the probability that one e_i is the zero vector is less than $q \cdot Q/p$. With probability at least $\epsilon/2 - q \cdot Q/p$, the 2nd case happens and $e_1, \ldots, e_q$ are non-zero vectors. Then, a reduction to the q-dlog problem is the same as before.

We now focus on the first case. When we handle case one the secret key x is picked and known and it holds that the exponent of at least one hash query in Equation 6 is non-zero. We again program the response for the ith hash query to be $h_i = g^{s_i} Y^{r_i}$ for uniform $s_i, r_i \in \mathbb{Z}_p^*$ and from (6), get an equation of the form $g^A Y^B = 1$ with A, B known and $B \neq 0$, except with probability $1/p$.

We can derive an equation $g^A Y^{B_1} (Y^2)^{B_2} = 1$ and solve a quadratic equation for $\log_g Y$. □

D One-More MitM Security of mID

In this section, we prove Theorem 8.

Proof overview. We follow the framework used to prove the OMUF security of $\mathbf{BS}_{[f_1]}$ [18]. In particular, we show that in order for an adversary to win Game ℓ-OMMIM against mID, it must either win Game ℓ-ROS for $\ell = 1$ (see Def. 16), which is shown to be information-theoretically hard in [25], or it must solve the dlog problem. To this end, we proceed via a series of games to rule out *bad* events regarding the algebraic representation of the group elements submitted by the algebraic adversary A upon Ver_1 queries. In $\mathbf{Game}_1$, we show that A cannot use pk_1 in the representation of the group element B, because otherwise, this is equivalent to solving $\mathsf{dlog}_g pk_1$. Then, in $\mathbf{Game}_2$, we show that A cannot use the group elements h and pk_1 in the representations of the public keys of the corrupted provers pk_k for $2 \leq k \leq n$. If h or pk_1 occur in the representation of any pk_k, a reduction wins the dlog game. Next, we show in $\mathbf{Game}_3$ that using the group element A_{pid} of a prover session pid in the representation of R_{vid} of a successful verifier session vid forces A to make a Prove_3 query for the session pid, otherwise a reduction can win the dlog game. Then, we show in $\mathbf{Game}_4$ that the value y_{vid} at the verifier side of successful verifier sessions must satisfy a certain equation; otherwise, a reduction can win the dlog game. Finally, in Games $\mathbf{Game}_5$-$\mathbf{Game}_8$, we show that a reduction wins the dlog game unless A can solve the 1-ROS problem.

Proof. Let A be an adversary running the ℓ-OMMIM game against mID. A behaves as a man-in-the-middle between the prover and the verifier. Assume A wins $\ell\text{-OMMIM}_{\mathsf{mID}}$. It follows that it closes $\ell + 1$ verifier sessions successfully. It follows that each successful verifier session vid^* satisfies the equation

$$R_{\mathsf{vid}^*} \cdot \prod_{j \in [n]} pk_j^{c_{\mathsf{vid}^*,j} + y^3_{\mathsf{vid}^*}} = g^{z_{\mathsf{vid}^*}} \cdot h^{y_{\mathsf{vid}^*}}. \qquad (7)$$

As A is algebraic, it submits a representation for each group element it outputs using the group elements of its input. Specifically, if there are Q_{Prove} open honest prover sessions, the group elements in A's input are $g, pk_1, h, \mathsf{A}_{pid}, \mathsf{B}_{pid}$ for $pid \in [Q_{\mathsf{Prove}}]$, and the group elements in its output are $A_{\mathsf{vid}^*,k}, B_{\mathsf{vid}^*,k}$ on the verifier side and $\{B_j\}_{j \in [n]}$ on the prover side, for $\mathsf{vid}^* \in Q_{\mathsf{Ver}}$ and $k \in [n]$.

Let $(g_{[R_{\mathsf{vid}^*}]}, h_{[R_{\mathsf{vid}^*}]}, pk_{1[R_{\mathsf{vid}^*}]}, \mathsf{A}_{pid[R_{\mathsf{vid}^*}]}, \mathsf{B}_{pid[R_{\mathsf{vid}^*}]})$ be the representations A submits of the group element R_{vid^*} of the verifier session $\mathsf{vid}^* \in [Q_{\mathsf{Ver}}]$ when it makes the Ver_1 query for this session[9]. It follows that

$$R_{\mathsf{vid}^*} = g^{g_{[R_{\mathsf{vid}^*}]}} \cdot h^{h_{[R_{\mathsf{vid}^*}]}} \cdot pk_1^{pk_{1[R_{\mathsf{vid}^*}]}} \cdot \prod_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{A}_{pid}^{\mathsf{A}_{pid}[R_{\mathsf{vid}^*}]} \cdot h^{\mathsf{y}_{pid} \cdot \mathsf{B}_{pid}[R_{\mathsf{vid}^*}]}. \qquad (8)$$

We substitute Equation (8) in Equation (7)

$$g^{g_{[R_{\mathsf{vid}^*}]}} \cdot h^{h_{[R_{\mathsf{vid}^*}]}} \cdot pk_1^{pk_{1[R_{\mathsf{vid}^*}]}} \cdot \prod_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{A}_{pid}^{\mathsf{A}_{pid}[R_{\mathsf{vid}^*}]} \cdot h^{\mathsf{y}_{pid} \cdot \mathsf{B}_{pid}[R_{\mathsf{vid}^*}]} \cdot \prod_{j \in [n]} pk_j^{c_{\mathsf{vid}^*,j} + y^3_{\mathsf{vid}^*}} = g^{z_{\mathsf{vid}^*}} \cdot h^{y_{\mathsf{vid}^*}}. \qquad (9)$$

We proceed to rule out *bad* cases regarding Equation (9) that prevent our reduction R_0 from solving the dlog problem. We define the following series of games:

$\mathbf{Game}_0$. This game is identical to Game ℓ-OMMIM. Let BAD_1 denote the event in which A makes a Prove_2 query of the form $(c, \{B_j\}_{j \in [n]})$, and there exists $j' \in [n]$ such that $pk_{1[B_{j'}]} \neq 0$, and A closes this session later successfully via a Prove_3 query.

$\mathbf{Game}_1$. This game is identical to the previous game, except that it aborts and outputs 0 if the event BAD_1 occurs.

Claim 1. $\Pr[\mathsf{BAD}_1] \leq \mathbf{Adv}^{\mathsf{dlog}}_{\mathsf{R}_1}$.

Proof. We construct a reduction R_1 that takes a dlog challenge U as input, sets $pk_1 := U$, runs A with input pk_1, and simulates the honest prover's and verifier's oracles for the adversary. While the simulation of the verifier's oracles is done as described by the protocol, the prover's oracles are simulated as follows:

- $\mathsf{Prove}_1()$: increment pid, sample $\mathsf{z}_{pid}, u_{pid,1}, u_{pid,2} \leftarrow \mathbb{Z}_q$, compute and return $\mathsf{A}_{pid} := g^{\mathsf{z}_{pid}} \cdot pk_1^{-u_{pid,1}}$, $\mathsf{B}_{pid} := g^{u_{pid,2}}$.
- $\mathsf{Prove}_2(pid, \mathsf{c}_{pid}, \{B_k\}_{k \in [n]})$: convert the representations of B_k for all $k \in [n]$ to the basis (g, h) if the representation does not contain pk_1 component, and let y_k be the respective h component. Define $\tilde{\mathsf{y}}_{pid} := \sum_{k=2}^n y_k$, compute and return $\mathsf{y}_{pid} := (u_{pid,1} - \mathsf{c}_{pid})^{\frac{1}{3}} - \tilde{\mathsf{y}}_{pid}$, and $\mathsf{b}_{pid} := u_{pid,2} - \mathsf{y}_{pid} \cdot w$. If there exists a B_k containing a pk_1 component in its representation, mark it as B_k^* for later processing, and compute and return $\mathsf{y}_{pid} := (u_{pid,1} - \mathsf{c}_{pid})^{\frac{1}{3}}$, and $\mathsf{b}_{pid} := u_{pid,2} - \mathsf{y}_{pid} \cdot w$.
- Prove_3(pid, $\{b_k, y_k\}_{k \in [n]}$): if any B_k was marked as B_k^* in the Prove_2 query, return $\perp$. Abort and return 0 if there is $k \in [n]$ with $g^{b_k} \cdot h^{y_k} \neq B_k$. Return z_{pid}.

The initial value of pid is 0.

We note the following regarding this simulation:

[9]The representation of R_{vid^*} consists of the representation of both A_{vid^*} and B_{vid^*} since $R_{\mathsf{vid}^*} = A_{\mathsf{vid}^*} \cdot B_{\mathsf{vid}^*}$

$\mathcal{P}_i(pk_i = g^{sk_i}, sk_i)$ | $\mathcal{V}(\vec{K} = \{pk_1, \dots, pk_n\})$

$a_i, b_i \leftarrow \mathbb{Z}_q, y_i \leftarrow \mathbb{Z}_q^*$
$A_i := g^{a_i}$

$\xrightarrow{A_i, B_i}$

Wait for all provers to commit
$A := \prod_{j \in [n]} A_j, B := \prod_{j \in [n]} B_j$
For all $j \in [n] : c_j \leftarrow \mathbb{Z}_q$

$\xleftarrow{c_i, \{B_j\}_{j \in [n]}}$

$\xrightarrow{b_i, y_i}$

Wait for all provers to send openings for B_i
If $\exists i \in [n] : B_i \neq g^{b_i} \cdot h^{y_i} \vee y_i = 0$:
  Abort
$y := \sum_{j \in [n]} y_j, R := A \cdot h^y$

$\xleftarrow{\{(b_j, y_j)\}_{j \in [n]}}$

If $\exists j \in [n] : B_j \neq g^{b_j} \cdot h^{y_j}$:
  Abort
$y := \sum_{j \in [n]} y_j$
$z_i := a_i + (c_i + y^3) \cdot sk_i$

$\xrightarrow{z_i}$

Wait for all provers to respond
$z := \sum_{j \in [n]} z_j$
If $R \cdot \prod_{j \in [n]} pk_j^{c_j + y^3} \neq g^z \cdot h^y$:
  Abort

Figure 5: Multi-identification scheme mID **executed by n provers and a verifier.**

(1) The response $(\mathsf{y}_{pid}, \mathsf{b}_{pid})$ to Prove$_2$ queries is a valid opening to B_{pid} of the Prove$_1$ response. In particular, we have

$$\begin{aligned} g^{\mathsf{b}_{pid}} \cdot h^{\mathsf{y}_{pid}} &= g^{u_{pid,2} - \mathsf{y}_{pid} \cdot w} \cdot h^{\mathsf{y}_{pid}} \\ &= g^{u_{pid,2}} \cdot h^{-\mathsf{y}_{pid}} \cdot h^{\mathsf{y}_{pid}} \\ &= g^{u_{pid,2}} \\ &= \mathsf{B}_{pid}. \end{aligned}$$

(2) The simulation satisfies perfect correctness if A computes B_j for all $j \in [n]$ honestly. Let $y^* := \mathsf{y}_{pid} + \tilde{\mathsf{y}}_{pid} = (u_{pid,1} - \mathsf{c}_{pid})^{\frac{1}{3}}$. Then, it must hold that

$$R \cdot pk_1^{\mathsf{c}_{pid} + y^{*3}} = \mathsf{A}_{pid} \cdot h^{\mathsf{y}_{pid}} \cdot pk_1^{\mathsf{c}_{pid} + y^{*3}} = g^{\mathsf{z}_{pid}} \cdot h^{\mathsf{y}_{pid}}.$$

It holds that

$$\begin{aligned} &\mathsf{A}_{pid} \cdot h^{\mathsf{y}_{pid}} \cdot pk_1^{\mathsf{c}_{pid} + y^{*3}} \\ &= g^{\mathsf{z}_{pid}} \cdot pk_1^{-u_{pid,1}} \cdot h^{\mathsf{y}_{pid}} \cdot pk_1^{\mathsf{c}_{pid} + ((u_{pid,1} - \mathsf{c}_{pid})^{\frac{1}{3}})^3} \\ &= g^{\mathsf{z}_{pid}} \cdot pk_1^{-u_{pid,1}} \cdot h^{\mathsf{y}_{pid}} \cdot pk_1^{u_{pid,1}} \\ &= g^{\mathsf{z}_{pid}} \cdot h^{\mathsf{y}_{pid}}. \end{aligned}$$

If there is a marked B_k^*, R_1 transforms its representation to the basis (g, h, pk_1) by computing $g_{[B_k^*]}, h_{[B_k^*]}, pk_{1[B_k^*]}$ such that the equality $g^{g_{[B_k^*]}} \cdot h^{h_{[B_k^*]}} \cdot pk_1^{pk_{1[B_k^*]}} = B_k^*$ holds. Then, it takes the opening of B_k^* sent in the Prove$_3$ query, i.e. b_k^*, y_k^* such that $g^{b_k^*} \cdot h^{y_k^*} = B_k^*$. These two representation of B_k^* allow R_1 to compute $\mathsf{dlog}_g pk_1$ as follows: We know that

$$g^{g_{[B_k^*]}} \cdot h^{h_{[B_k^*]}} \cdot pk_1^{pk_{1[B_k^*]}} = g^{b_k^*} \cdot h^{y_k^*}.$$

By taking the discrete logarithm of both sides we get

$$g_{[B_k^*]} + w \cdot h_{[B_k^*]} + sk_1 \cdot pk_{1[B_k^*]} = b_k^* + w \cdot y_k^*.$$

R_1 computes $sk_1 = \mathsf{dlog}_g pk_1 = \mathsf{dlog}_g U$ as

$$sk_1 = \frac{b_k^* + w \cdot y_k^* - g_{[B_k^*]} + w \cdot h_{[B_k^*]}}{pk_{1[B_k^*]}}.$$

This equation is solvable because $pk_{1[B_k^*]} \neq 0$, hence the claim. □

It follows from this claim that $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_1} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_0} - \mathbf{Adv}_{\mathsf{R}_1}^{\mathsf{dlog}}$; therefore, we assume that A wins $\mathbf{Game}_1$.

Define BAD_2 as the event that there is $2 \leq i \leq n$ with $h_{[pk_i]} \neq 0 \vee pk_{1[pk_i]} \neq 0$.

$\mathbf{Game}_2$. This game is identical to $\mathbf{Game}_1$, except that it aborts and outputs 0 if BAD_2 occurs.

Claim 2. $\Pr[\mathsf{BAD}_2] \leq \frac{1}{2} \cdot \mathbf{Adv}_{\mathsf{R}_2}^{\mathsf{dlog}}$.

Proof. Assuming A submits a proof of possession for each corrupted public key pk_k for $2 \leq k \leq n$ (we assume it submits all the secret keys sk_k), we show that if there is a winning verifier session under a set of public keys $\vec{K} = \{pk_1, \dots, pk_n\}$ and there exists $2 \leq i \leq n$ with $h_{[pk_i]} \neq 0$ or $pk_{1[pk_i]} \neq 0$, then the reduction R_2 wins the dlog game.

Given a dlog challenge U and an adversary A running $\mathbf{Game}_2$, R_2 flips a coin and behaves as follows:

- **On Heads.** R_2 embeds U in h and simulates the honest pro ver's oracles exactly as described by the protocol. It computes $\mathsf{dlog}_g U$ if A submits a set of keys $\vec{K} = \{pk_1, \dots, pk_n\}$, and there exists $2 \leq i \leq n$, such that the representation of pk_i contains an h component or B_{pid} component for some prover session pid (recall that $\mathsf{B}_{pid} = g^{\mathsf{b}_{pid}} \cdot h^{\mathsf{y}_{pid}}$). If $h_{[pk_i]} = \mathsf{B}_{pid\,[pk_i]} = 0$ for all $pid \in Q_{\mathsf{Prove}}$ and all $i \in [n]$, R_2 aborts and outputs $\perp$. Knowing that $pk_i = g^{g_{[pk_i]}} \cdot h^{h_{[pk_i]}}$

(note that R_2 knows the discrete logarithms of all group elements that A may use in the representation of pk_i except h, therefore, it can aggregate all non-h components as a single g component), and hence $sk_i = g_{[pk_i]} + \mathrm{dlog}_g h \cdot h_{[pk_i]}$, R_2 computes and outputs

$$\mathrm{dlog}_g U = \mathrm{dlog}_g h = \frac{sk_i - g_{[pk_i]}}{h_{[pk_i]}}.$$

- **On Tails.** Given a discrete logarithm challenge U, R_2 behaves as follows. First, it chooses a public parameter $pp = (\mathbb{G}, q, g, h)$ with known $w = \mathrm{dlog}_g h$, sets $pk_1 = U$, and runs A with input pk_1 (and access to pp). During the execution of A, R_2 simulates the verifier's and the honest prover's oracles. It simulates the verifier's oracles Ver_1,Ver_2, and Ver_3 exactly as described by the protocol mID. For the simulation of the prover's oracles, it defines $pid := 0$ and simulates the oracles as follows:
  - $\mathsf{Prove}_1()$: Sample $\mathsf{z}_{pid}, u_{pid,1}, u_{pid,2} \leftarrow \mathbb{Z}_q$ and return $\mathsf{A}_{pid} := g^{\mathsf{z}_{pid}} \cdot pk_1^{-u_{pid,1}}$, $\mathsf{B}_{pid} := g^{u_{pid,2}}$. Increment pid.
  - $\mathsf{Prove}_2(pid, \mathsf{c}_{pid}, \{B_k\}_{k\in[n]})$: convert the representations of B_k for all $k \in [n]$ to the basis (g, h) (this is possible because pk_1 is the only group element with unknown discrete logarithm and due to **Game**$_1$ that $pk_{1[B_k]} = 0$ for all B_k), and let y_k be the respective h component. Define $\tilde{\mathsf{y}}_{pid} := \sum_{k=2}^{n} y_k$, compute and return $\mathsf{y}_{pid} := (u_{pid,1} - \mathsf{c}_{pid})^{\frac{1}{3}} - \tilde{\mathsf{y}}_{pid}$, and $\mathsf{b}_{pid} := u_{pid,2} - \mathsf{y}_{pid} \cdot w$.
  - Prove_3(pid, $\{b_k, y_k\}_{k\in[n]}$): abort and return 0 if there is $k \in [n]$ with $g^{b_k} \cdot h^{y_k} \neq B_k$. Return z_{pid}.

R_2 computes $\mathrm{dlog}_g U$ if A outputs $\vec{K} = \{pk_1, \ldots, pk_n\}$, and there exists $2 \le i \le n$, such that the representation of pk_i contains a pk_1 component or A_{pid} component for some prover session pid (recall that $\mathsf{A}_{pid} = g^{\mathsf{z}_{pid}} \cdot pk_1^{y^*}$ for $(y^* = \mathsf{y}_{pid}^3 + c_{pid,1} + \tilde{\mathsf{y}}_{pid})$) . If $pk_{1[pk_i]} = \mathsf{A}_{pid[pk_i]} = 0$ for all $pid \in Q_{\mathsf{Prove}}$ and all $i \in [n]$, R_2 aborts and outputs $\perp$. Knowing that $pk_i = g^{g_{[pk_i]}} \cdot pk_1^{pk_{1[pk_i]}}$ (note that R_2 knows the discrete logarithms of all group elements that A may use in the representation of pk_i except pk_1, therefore, it can aggregate all non-pk_1 components as a single g component), and hence $sk_i = g_{[pk_i]} + \mathrm{dlog}_g h \cdot h_{[pk_i]}$, R_2 computes and outputs $\mathrm{dlog}_g U = \mathrm{dlog}_g pk_1 = \frac{sk_i - g_{[pk_i]}}{pk_{1[pk_i]}}$. □

Per this claim, $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_2} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_1} - \frac{1}{2} \cdot \mathbf{Adv}_{\mathsf{R}_2}^{\mathrm{dlog}}$; therefore, we assume that A must ensure that $h_{[pk_i]} = pk_{1[pk_i]} = 0$ and win **Game**$_2$.

Next, we define BAD_3 as the event that there is a successful verifier session vid* and a prover session $pid \in Q_{\mathsf{Prove}}$ with $\mathsf{A}_{pid[R_{\mathsf{vid}^*}]} \neq 0$ and A did not close pid via a Prove_3 query *successfully.*

Game$_3$. This game is identical to **Game**$_2$, except that it aborts and outputs 0 if BAD_3 occurs.

CLAIM 3. *Let* Q_{Prove_1} *be the number of* Prove_1 *queries made by* A. *Then* $\Pr[\mathsf{BAD}_3] \leq \frac{1}{Q_{\mathsf{Prove}_1}} \cdot \mathbf{Adv}_{\mathsf{R}_3}^{\mathrm{dlog}}$.

PROOF. We show that if there is an accepting verifier session vid* and a prover session pid with $\mathsf{A}_{pid[R_{\mathsf{vid}^*}]} \neq 0$ and pid was not closed via a Prove_3 query, then there is a reduction R_3 that wins the dlog game.

Given a dlog challenge U and an adversary A running **Game**$_0$ and triggering the said event, we construct R_3 as follows: R_3 samples $w \leftarrow \mathbb{Z}_q$, and chooses public parameter $pp = (\mathbb{G}, q, g, h = g^w)$. Then, it samples $sk_1 \leftarrow \mathbb{Z}_q$, computes $pk_1 := g^{sk_1}$, and runs A with input pk_1 and access to pp. It simulates the verifier's oracles for A as described by the protocol mID, initializes $pid := 0$, samples $pid^* \leftarrow [Q_{\mathsf{Prove}_1}]$, and simulates the prover's oracles as follows:

- $\mathsf{Prove}_1()$: increment pid, sample $a_{pid}, \mathsf{b}_{pid} \leftarrow \mathbb{Z}_q, \mathsf{y}_{pid} \leftarrow \mathbb{Z}_q^*$. If $pid \neq pid^*$, compute $\mathsf{A}_{pid} := g^{a_{pid}}$, otherwise set $\mathsf{A}_{pid} := U$. Finally, compute $\mathsf{B}_{pid} := g^{\mathsf{b}_{pid}} \cdot h^{\mathsf{y}_{pid}}$ and return $(pid, \mathsf{A}_{pid}, \mathsf{B}_{pid})$.
- $\mathsf{Prove}_2(pid, c_{pid}, \{B_{k,pid}\}_{k\in[n]})$: return $(\mathsf{b}_{pid}, \mathsf{y}_{pid})$.
- $\mathsf{Prove}_3(pid, \perp)$: abort if there is $k \in [n]$ with $B_k \neq g^{b_k} \cdot h^{y_k}$ or $pid = pid^*$. Otherwise, compute $y := \sum_{k\in[n]} y_k$, $\mathsf{z}_{pid} := a_{pid} + (c_{pid} + y^3) \cdot sk$, and return z_{pid}.

If A terminates and closes $\ell + 1$ verifier sessions successfully, R_3 finds a successful verifier session vid* satisfying $\mathsf{A}_{pid^*[R_{\mathsf{vid}^*}]} \neq 0$. Per correctness, this state satisfies Equation (9). Using the internal view of the reduction, we substitute $\mathsf{A}_{pid} := g^{a_{pid}}, \mathsf{A}_{pid^*} = U = g^{\mathrm{dlog}_g U}$ for $pid \in [Q_{\mathsf{Prove}_1}] \backslash \{pid^*\}$ in Equation (9), which gives the equation

$$g^{g_{[R_{\mathsf{vid}^*}]}} \cdot h^{h_{[R_{\mathsf{vid}^*}]}} \cdot pk_1^{pk_{1[R_{\mathsf{vid}^*}]}} \cdot g^{\mathrm{dlog}_g U \cdot \mathsf{A}_{pid^*[R_{\mathsf{vid}^*}]}} \cdot h^{\mathsf{y}_{pid^*} \cdot \mathsf{B}_{pid^*[R_{\mathsf{vid}^*}]}} \cdot \prod_{\substack{pid\in[Q_{\mathsf{Prove}}]\\ pid\neq pid^*}} g^{a_{pid} \cdot \mathsf{A}_{pid[R_{\mathsf{vid}^*}]}} \cdot h^{\mathsf{y}_{pid} \cdot \mathsf{B}_{pid[R_{\mathsf{vid}^*}]}} \cdot \prod_{j\in[n]} pk_j^{c_{\mathsf{vid}^*,j} + y_{\mathsf{vid}^*}^3} = g^{z_{\mathsf{vid}^*}} \cdot h^{y_{\mathsf{vid}^*}}. \quad (10)$$

By taking the discrete logarithm of both sides, we get

$$g_{[R_{\mathsf{vid}^*}]} + w \cdot h_{[R_{\mathsf{vid}^*}]} + sk_1 \cdot pk_{1[R_{\mathsf{vid}^*}]} + \mathrm{dlog}_g U \cdot \mathsf{A}_{pid^*[R_{\mathsf{vid}^*}]} + w \cdot \mathsf{y}_{pid^*} \cdot \mathsf{B}_{pid^*[R_{\mathsf{vid}^*}]} + \sum_{\substack{pid\in[Q_{\mathsf{Prove}}]\\ pid\neq pid^*}} a_{pid} \cdot \mathsf{A}_{pid[R_{\mathsf{vid}^*}]} + w \cdot \mathsf{y}_{pid} \cdot \mathsf{B}_{pid[R_{\mathsf{vid}^*}]} \cdot \sum_{j\in[n]} sk_j \cdot (c_{\mathsf{vid}^*,j} + y_{\mathsf{vid}^*}^3) = z_{\mathsf{vid}^*} + w \cdot y_{\mathsf{vid}^*}. \quad (11)$$

Note that the discrete logarithms of all group elements that may occur in the representation of pk_j for all $j \in [n]$ are known to the reduction R_3; therefore, any representation of pk_j submitted by A can be transformed to the basis g, which allows R_3 to efficiently compute $sk_j = \mathrm{dlog}_g pk_j$.

Then R_3 computes

$$\mathrm{dlog}_g U = \frac{C}{-\mathsf{A}_{pid^*[R_{\mathsf{vid}^*}]}}, \quad (12)$$

where $C = g_{[R_{\mathsf{vid}^*}]} + w \cdot h_{[R_{\mathsf{vid}^*}]} + sk_1 \cdot pk_{1[R_{\mathsf{vid}^*}]} + w \cdot \mathsf{y}_{pid^*} \cdot \mathsf{B}_{pid^*[R_{\mathsf{vid}^*}]} + \sum_{\substack{pid\in[Q_{\mathsf{Prove}}]\\ pid\neq pid^*}} a_{pid} \cdot \mathsf{A}_{pid[R_{\mathsf{vid}^*}]} + w \cdot \mathsf{y}_{pid} \cdot \mathsf{B}_{pid[R_{\mathsf{vid}^*}]} \cdot \sum_{j\in[n]} sk_j \cdot (c_{\mathsf{vid}^*,j} + y_{\mathsf{vid}^*}^3) - z_{\mathsf{vid}^*} - w \cdot y_{\mathsf{vid}^*}$.

If the event BAD_3 occurs, $\mathsf{A}_{pid^*}{}_{[R_{\mathsf{vid}^*}]} \neq 0$; therefore, this equation is defined, which allows R_3 to compute the $\mathsf{dlog}_g U$ successfully if it guesses pid^* correctly and embeds U in A_{pid^*} (otherwise R_3 aborts), which occurs with probability $\frac{1}{Q_{\mathsf{Prove}_1}}$, hence the claim. □

Per this claim, we have $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_3} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_2} - \frac{1}{Q_{\mathsf{Prove}_1}} \cdot \mathbf{Adv}_{\mathsf{R}_3}^{\mathsf{dlog}}$. Since A wins $\mathbf{Game}_2$ we assume in the following that A wins $\mathbf{Game}_3$.

Let BAD_4 be an event that occurs if there is a successful verifier session vid^* with

$$y_{\mathsf{vid}^*} \neq h_{[R_{\mathsf{vid}^*}]} + \sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]}. \tag{13}$$

$\mathbf{Game}_4$. This game is identical to $\mathbf{Game}_3$, except that it aborts and outputs 0 if event BAD_4 occurs.

Claim 4. $\Pr[\mathsf{BAD}_4] \leq \mathbf{Adv}_{\mathsf{R}_4}^{\mathsf{dlog}}$.

Proof. Assume A closes $\ell + 1$ verifier sessions successfully, and there exists a successful verifier session vid^* satisfying $y_{\mathsf{vid}^*} \neq h_{[R_{\mathsf{vid}^*}]} + \sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]}$. We construct a reduction R_4 that exploits A to win the dlog game. Given a discrete logarithm challenge U, R_4 chooses a public parameter $pp := (\mathbb{G}, q, g, h)$ with $h = U$, samples $sk_1 \leftarrow \mathbb{Z}_q$, sets $pk_1 := g^{sk_1}$, and runs A with input pk_1 and access to pp on $\mathbf{Game}_4$. It simulates the honest prover's oracles and the verifier's oracles exactly as described by the protocol mID.

Finally, when A terminates and closes $\ell + 1$ verifier session successfully, each successful session vid^* satisfies Equation (9). From the internal state of the prover, we know that $\mathsf{A}_{pid} = g^{a_{pid}}$. We substitute this value in Equation (9) and get the equation

$$g^{g_{[R_{\mathsf{vid}^*}]}} \cdot h^{h_{[R_{\mathsf{vid}^*}]}} \cdot pk_1^{pk_{1[R_{\mathsf{vid}^*}]}} \cdot \prod_{pid \in [Q_{\mathsf{Prove}}]} g^{a_{pid} \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]}} \cdot h^{\mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]}} \cdot \prod_{j \in [n]} pk_j^{c_{\mathsf{vid}^*,j} + y^3_{\mathsf{vid}^*}} = g^{z_{\mathsf{vid}^*}} \cdot h^{y_{\mathsf{vid}^*}}. \tag{14}$$

Taking the discrete logarithm of both sides, we get

$$g_{[R_{\mathsf{vid}^*}]} + w \cdot h_{[R_{\mathsf{vid}^*}]} + sk_1 \cdot pk_{1[R_{\mathsf{vid}^*}]} + \sum_{pid \in [Q_{\mathsf{Prove}}]} a_{pid} \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} + w \cdot \mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} + \sum_{j \in [n]} sk_j \cdot (c_{\mathsf{vid}^*,j} + y^3_{\mathsf{vid}^*}) = z_{\mathsf{vid}^*} + w \cdot y_{\mathsf{vid}^*}. \tag{15}$$

Note that R_4 can compute sk_j for $j \in [n]$ such that $pk_j = g_j^{sk}$, because the representation of pk_j only contains group elements with known discrete logarithm to the base g. Particularly, the only group element with unknown discrete logarithm is h, and due to the changes made in $\mathbf{Game}_2$, we know that there is no h component in the representation of any pk_j.

It follows that $w = \frac{N}{D}$, where $N := g_{[R_{\mathsf{vid}^*}]} + sk_1 \cdot pk_{1[R_{\mathsf{vid}^*}]} + \sum_{pid \in [Q_{\mathsf{Prove}}]} a_{pid} \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} + \sum_{j \in [n]} sk_j \cdot (c_{\mathsf{vid}^*,j} + y^3_{\mathsf{vid}^*}) - z_{\mathsf{vid}^*}$, and $D := y_{\mathsf{vid}^*} - h_{[R_{\mathsf{vid}^*}]} - \sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]}$. R_4 can compute w and win the dlog game if $y_{\mathsf{vid}^*} - h_{[R_{\mathsf{vid}^*}]} - \sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} \neq 0 \Rightarrow y_{\mathsf{vid}^*} \neq h_{[R_{\mathsf{vid}^*}]} + \sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{y}_{pid} \cdot \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]}$. □

Thus $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_4} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_3} - \mathbf{Adv}_{\mathsf{R}_4}^{\mathsf{dlog}}$, and thus we assume that A wins $\mathbf{Game}_4$ and that Event BAD_4 does not occur.

Next, we define the event BAD_5 as the event that occurs iff for all successful verifier sessions vid^*, the equation

$$-pk_{1[R_{\mathsf{vid}^*}]} - \sum_{pid \in [Q_{\mathsf{Prove}}]} ((\mathsf{y}_{pid} + \tilde{\mathsf{y}}_{pid})^3 + \mathsf{c}_{pid}) \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} - \left(\sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid} + h_{[R_{\mathsf{vid}^*}]} \right)^3 - c_{\mathsf{vid}^*,1} = 0 \tag{16}$$

holds. We further define the sub-events $\mathsf{BAD}_{5,1}$, $\mathsf{BAD}_{5,2}$, $\mathsf{BAD}_{5,3}$, and $\mathsf{BAD}_{5,4}$ such that

$$\mathsf{BAD}_5 = \mathsf{BAD}_{5,1} \vee \mathsf{BAD}_{5,2} \vee \mathsf{BAD}_{5,3} \vee \mathsf{BAD}_{5,4},$$

where

- $\mathsf{BAD}_{5,1}$ is the event that occurs if the event BAD_5 occurs, and there exists a verifier session vid^* such that for all prover sessions $pid \in [Q_{\mathsf{Prove}}]$, either $\mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} = 0$, or $\mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} \neq 0$ and A makes a Prove_2 query for the session pid before making a Ver_1 query for the session vid^*.
- $\mathsf{BAD}_{5,2}$ is the event that occurs if the event $\mathsf{BAD}_5 \wedge \neg \mathsf{BAD}_{5,1}$ occurs, and there exists prover sessions $pid_1, \ldots, pid_k$ for $2 \leq k \leq Q_{\mathsf{Prove}}$ and a successful verifier session vid^* such that $\mathsf{B}_{pid_j}{}_{[R_{\mathsf{vid}^*}]} \neq 0$ for all j.
- $\mathsf{BAD}_{5,3}$ is the event that occurs if the event $\mathsf{BAD}_5 \wedge \neg(\mathsf{BAD}_{5,1} \vee \mathsf{BAD}_{5,2})$ occurs, and there exists verifier sessions $\mathsf{vid}^*{}_1, \ldots, \mathsf{vid}^*{}_k$ for $2 \leq k \leq Q_{\mathsf{Prove}}$, and a prover session $pid \in [Q_{\mathsf{Prove}}]$ with $\mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*{}_j}]} \neq 0$ for all $j \in [k]$.
- $\mathsf{BAD}_{5,4}$ is the event that occurs if the event $\mathsf{BAD}_5 \wedge \neg(\mathsf{BAD}_{5,1} \vee \mathsf{BAD}_{5,2} \vee \mathsf{BAD}_{5,3})$ occurs. That is, for all successful verifier sessions vid^*, there exists exactly one prover session pid with $\mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} \neq 0$, and for all prover sessions pid, there exists exactly one successful verifier session vid^* with $\mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} \neq 0$. Additionally, if $\mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]} \neq 0$ holds, A makes a Prove_2 query for pid before it makes a Ver_1 query for vid^*.

$\mathbf{Game}_5$. This game is identical to $\mathbf{Game}_4$, except that it aborts and outputs 0 if the event $\mathsf{BAD}_{5,1}$ occurs.

Claim 5. $\Pr[\mathsf{BAD}_{5,1}] \leq \mathsf{negl}(\lambda)$.

Proof. As $\mathsf{BAD}_{5,1}$ occurs, all the variables, except $c_{\mathsf{vid}^*,1}$, in Equation (16) of the session vid^* get fixed by A and the prover before the value $c_{\mathsf{vid}^*,1}$ is sampled uniformly at random by the verifier when A makes a Ver_1 query. Consequently, satisfying Equation (16) is equivalent to guessing the value $c_{\mathsf{vid}^*,1}$, which occurs with probability at most $\frac{1}{q}$. □

This claim implies that $Adv_{\mathsf{A}}^{\mathbf{Game}_5} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_4} - \mathsf{negl}(\lambda)$, and therefore, we assume that A wins $\mathbf{Game}_5$, and that $\mathsf{BAD}_{\mathsf{BAD}_{5,1}}$ does not occur. Consequently, every verifier session vid^* must be linked to at least a prover session pid, i.e., $\mathsf{B}_{pid[R_{\mathsf{vid}^*}]} \neq 0$ and A does not make a Prove_2 query for pid before it makes a Ver_1 query for vid^*.

Game$_6$. This game is identical to **Game**$_5$, except that it aborts and outputs 0 if the event $\mathsf{BAD}_{5,2}$ occurs.

Claim 6. $\Pr[\mathsf{BAD}_{5,2}] \leq \mathsf{negl}(\lambda)$.

Proof. Assume there are $2 \leq k$ prover sessions $\{pid_1, \ldots, pid_k\}$ such that $\mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]} \neq 0$ for all $j \in [k]$, and assume that the session pid_k is the session for which A makes a Prove_2 query last among the sessions pid_j for $j \in [k]$. Per the case assumption, $\mathsf{BAD}_{5,1}$ does not occur; therefore, A makes a Prove_2 query for pid_k after it has made Ver_1 query for the session vid^*.

Equation (16) can be expressed as a polynomial

$$P(\mathcal{X}) := -pk_{1[R_{\mathsf{vid}^*}]} - \sum_{j \in [k-1]} ((\mathsf{y}_{pid_j} + \tilde{\mathsf{y}}_{pid_j})^3 + \mathsf{c}_{pid_j}) \cdot \mathsf{A}_{pid_j[R_{\mathsf{vid}^*}]} - ((\mathcal{X} + \tilde{\mathsf{y}}_{pid_k})^3 + \mathsf{c}_{pid_k}) \cdot \mathsf{A}_{pid_k[R_{\mathsf{vid}^*}]} - (\sum_{j \in [k-1]} \mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid_j} + \mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]} \cdot \mathcal{X} + h_{[R_{\mathsf{vid}^*}]})^3 - c_{\mathsf{vid}^*,1}. \quad (17)$$

We distinguish two cases: The first case is that the polynomial P is non-zero. Since y_{pid_k} is generated by the prover uniformly at random, then per the Schwartz-Zippel lemma, $P(\mathsf{y}_{pid_k}) = 0$ holds with probability at most $\frac{3}{q}$.

The other case is that P is a zero-polynomial. It follows that all the coefficients in P are equal to 0 including the coefficients of $\mathcal{X}^2$ and $\mathcal{X}^3$. Thus, we have

$$3 \cdot \tilde{\mathsf{y}}_{pid_k} \cdot \mathsf{A}_{pid_k[R_{\mathsf{vid}^*}]} = 3 \cdot \mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]}{}^2 \cdot (\sum_{j \in [k-1]} \mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid_j} + h_{[R_{\mathsf{vid}^*}]}), \quad (18)$$

and

$$-\mathsf{A}_{pid_k[R_{\mathsf{vid}^*}]} = \mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]}{}^3. \quad (19)$$

We substitute Equation (19) in Equation (18) (Note that $\mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]} \neq 0$) and get

$$-\tilde{\mathsf{y}}_{pid_k} \cdot \mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]} = \sum_{j \in [k-1]} \mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid_j} + h_{[R_{\mathsf{vid}^*}]}. \quad (20)$$

This equation implies that the value $\tilde{\mathsf{y}}_{pid}$ gets fixed by the time A makes a Ver_1 query for the session vid^* and cannot be influenced afterward. To see this, rearrange the equation as

$$\tilde{\mathsf{y}}_{pid_k} = -\frac{\sum_{j \in [k-1]} \mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid_j} + h_{[R_{\mathsf{vid}^*}]}}{\mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]}}. \quad (21)$$

We see that $\sum_{j \in [k-1]} \mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]}$, $h_{[R_{\mathsf{vid}^*}]}$, and $\mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]}$ are fixed by A when it makes the Ver_1 query for session vid^*, and y_{pid_j} gets fixed by the prover as a response to the Prove_2 queries of the other prover sessions (recall that k is the sessions with the last Prove_2 query). Define the polynomial

$$P'(\mathcal{X}) := \tilde{\mathsf{y}}_{pid_k} \cdot \mathsf{B}_{pid_k[R_{\mathsf{vid}^*}]} - h_{[R_{\mathsf{vid}^*}]} - \sum_{j \in [k-2]} \mathsf{B}_{pid_j[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid_j} - \mathsf{B}_{pid_{k-1}[R_{\mathsf{vid}^*}]} \cdot \mathcal{X}.$$

Again, we distinguish two cases: If the polynomial P' is non-zero, then $P'(\mathsf{y}_{pid_{k-1}})$ evaluates to 0 with probability at most $\frac{1}{q}$, because $\mathsf{y}_{pid_{k-1}}$ gets sampled uniformly after the Prove_2 query of the session pid_{k-1}, and at this time all the values in P' are fixed except $\mathcal{X}$. The other case is that P' is the zero polynomial. It follows that the coefficient of $\mathcal{X}$ is zero, i.e., $\mathsf{B}_{pid_{k-1}[R_{\mathsf{vid}^*}]} = 0$, a contradiction.

Therefore, we assume that this case does not occur, and assume in the following that each verifier session vid^* is linked to exactly one prover session pid. □

We assume in the following that A wins **Game**$_6$, because this claim implies that $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_6} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_5} - \mathsf{negl}(\lambda)$.

Game$_7$. This game is identical to **Game**$_6$, except that it aborts and outputs 0 if the event $\mathsf{BAD}_{5,3}$ occurs.

Claim 7. $\Pr[\mathsf{BAD}_{5,3}] \leq \mathsf{negl}(\lambda)$.

Proof. Assume there are successful verifier sessions $\mathsf{vid}^*_1, \ldots, \mathsf{vid}^*_k$ for $2 \leq k \leq \ell + 1$ and a prover session pid such that $\mathsf{B}_{pid[R_{\mathsf{vid}^*_j}]} \neq 0$ for all $j \in [k]$. As $\mathsf{BAD}_{5,2}$ does not occur, it holds that $\mathsf{B}_{pid'[R_{\mathsf{vid}^*_j}]} = 0$ for all $j \in [k]$ and all $pid' \neq pid$, and since $\mathsf{BAD}_{5,1}$ does not occur, it follows that A makes Ver_1 queries for all vid^*_j before making a Prove_2 query for the session pid.

Assume for the sake of contradiction that the equation

$$-pk_{1[R_{\mathsf{vid}^*}]} - \sum_{pid \in [Q_{\mathsf{Prove}}]} ((\mathsf{y}_{pid} + \tilde{\mathsf{y}}_{pid})^3 + \mathsf{c}_{pid}) \cdot \mathsf{A}_{pid[R_{\mathsf{vid}^*}]} - \left(\sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{B}_{pid[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid} + h_{[R_{\mathsf{vid}^*}]}\right)^3 - c_{\mathsf{vid}^*,1} = 0.$$

holds for all $\mathsf{vid}^* \in \{\mathsf{vid}^*_1, \ldots, \mathsf{vid}^*_k\}$. We construct the polynomial

$$P(\mathcal{X}) := -pk_{1[R_{\mathsf{vid}^*}]} - \sum_{pid' \neq pid} ((\mathsf{y}_{pid'} + \tilde{\mathsf{y}}_{pid'})^3 + \mathsf{c}_{pid'}) \cdot \mathsf{A}_{pid'[R_{\mathsf{vid}^*}]} - ((\mathcal{X} + \tilde{\mathsf{y}}_{pid})^3 + \mathsf{c}_{pid}) \cdot \mathsf{A}_{pid[R_{\mathsf{vid}^*}]} - \left(\mathsf{B}_{pid[R_{\mathsf{vid}^*}]} \cdot \mathcal{X} + h_{[R_{\mathsf{vid}^*}]}\right)^3 - c_{\mathsf{vid}^*,1}. \quad (22)$$

We distinguish two cases: if P is a non-zero polynomial, $P(\mathsf{y}_{pid}) = 0$ occurs with probability at most $\frac{3}{q}$ for a single verifier session vid^* by the Schwartz-Zippel lemma, because y_{pid} gets sampled uniformly after all the coefficients in P get fixed. Since $\mathsf{vid}^* \in \{\mathsf{vid}^*_1, \ldots, \mathsf{vid}^*_k\}$, the probability that this event holds for at least one of the verifier sessions is $\frac{3 \cdot k}{q}$. The other case is that P is the zero polynomial. This implies that the equations

$$-\mathsf{A}_{pid[R_{\mathsf{vid}^*}]} = \mathsf{B}_{pid[R_{\mathsf{vid}^*}]}{}^3, \quad (23)$$

and

$$-pk_{1[R_{\mathsf{vid}^*}]} - (\sum_{pid' \neq pid} ((\mathsf{y}_{pid'} + \tilde{\mathsf{y}}_{pid'})^3 + \mathsf{c}_{pid'}) \cdot \mathsf{A}_{pid'[R_{\mathsf{vid}^*}]} = (\mathsf{y}_{pid}^3 + \mathsf{c}_{pid}) \cdot \mathsf{A}_{pid[R_{\mathsf{vid}^*}]} - h_{[R_{\mathsf{vid}^*}]}{}^3 + c_{\mathsf{vid}^*,1} \quad (24)$$

hold for all $\mathsf{vid}^* \in \{\mathsf{vid}^*_1, \ldots, \mathsf{vid}^*_k\}$. Equation (23) implies that $\mathsf{A}_{pid[R_{\mathsf{vid}^*}]} \neq 0$ because per the case assumption $\mathsf{B}_{pid[R_{\mathsf{vid}^*}]} \neq 0$.

Additionally, it follows from Equation (24) that

$$\mathsf{c}_{pid} = \frac{1}{\mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]}} \cdot (-pk_{1[R_{\mathsf{vid}^*}]} - (\sum_{pid' \neq pid} ((\mathsf{y}_{pid'} + \tilde{\mathsf{y}}_{pid'})^3 + \mathsf{c}_{pid'}) \cdot \mathsf{A}_{pid'}{}_{[R_{\mathsf{vid}^*}]}) - \mathsf{y}_{pid}^3 \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} + h_{[R_{\mathsf{vid}^*}]}{}^3 - c_{\mathsf{vid}^*,1}). \quad (25)$$

Recall that the verifier samples $c_{\mathsf{vid}^*,1}$ uniformly at random. We construct a hash function (treated as a random oracle) $\mathsf{H}_{\mathsf{Ver}}$ as follows: Given input $\xi = (A, B)$ for $A, B \in \mathbb{G}$, it returns $\mathsf{T}_{\mathsf{Ver}}[\xi]$ if $\mathsf{T}_{\mathsf{Ver}}[\xi] \neq \perp$. Otherwise, it makes a Ver_1 query with input A, B and receives c_{vid^*}. It stores $\mathsf{T}_{\mathsf{Ver}}[\xi] = c_{\mathsf{vid}^*}$, and returns c_{vid^*}.

It follows from Equation (25) that

$$c_{\mathsf{vid}^*,1} = \mathsf{H}_{\mathsf{Ver}}(A_{\mathsf{vid}^*}, B_{\mathsf{vid}^*}) = \mathsf{c}_{pid} \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} - (-pk_{1[R_{\mathsf{vid}^*}]} - (\sum_{pid' \neq pid} ((\mathsf{y}_{pid'} + \tilde{\mathsf{y}}_{pid'})^3 + \mathsf{c}_{pid'}) \cdot \mathsf{A}_{pid'}{}_{[R_{\mathsf{vid}^*}]}) - \mathsf{y}_{pid}^3 \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} + h_{[R_{\mathsf{vid}^*}]}{}^3). \quad (26)$$

Let $\mathsf{H}_{\mathsf{ROS}}$ be another hash function (modeled as a random oracle), such that

$$\mathsf{H}_{\mathsf{ROS}}(\mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]}, aux_{\mathsf{vid}^*}) = \mathsf{H}_{\mathsf{Ver}}(A_{\mathsf{vid}^*}, B_{\mathsf{vid}^*}) + (pk_{1[R_{\mathsf{vid}^*}]} - (\sum_{pid' \neq pid} ((\mathsf{y}_{pid'} + \tilde{\mathsf{y}}_{pid'})^3 + \mathsf{c}_{pid'}) \cdot \mathsf{A}_{pid'}{}_{[R_{\mathsf{vid}^*}]}) - \mathsf{y}_{pid}^3 \cdot \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]} + h_{[R_{\mathsf{vid}^*_1}]}{}^3), \quad (27)$$

where $aux_{\mathsf{vid}^*} = (g_{[R_{\mathsf{vid}^*}]}, h_{[R_{\mathsf{vid}^*}]}, pk_{1[R_{\mathsf{vid}^*}]}, (\mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*}]}, \mathsf{B}_{pid}{}_{[R_{\mathsf{vid}^*}]})_{pid \in [Q_{\mathsf{Prove}}]})$.

Thus, for arbitrary $\mathsf{vid}^*{}_i$, $\mathsf{vid}^*{}_j$ for $i, j \in [k]$ with $i \neq j$,

$$\mathsf{H}_{\mathsf{ROS}}(\mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*{}_i}]}, aux_{\mathsf{vid}^*{}_i}) = \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*{}_i}]} \cdot \mathsf{c}_{pid}, \quad (28)$$

$$\mathsf{H}_{\mathsf{ROS}}(\mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*{}_j}]}, aux_{\mathsf{vid}^*{}_j}) = \mathsf{A}_{pid}{}_{[R_{\mathsf{vid}^*{}_j}]} \cdot \mathsf{c}_{pid}. \quad (29)$$

This is the *ROS* problem with $\ell = 1$; therefore, the probability that the event (29) occurs is at most $\frac{\binom{H_q}{\ell+1}+1}{q} = \frac{\binom{H_q}{2}+1}{q}$, where H_q is the number of $\mathsf{H}_{\mathsf{ROS}}$ queries A makes. □

Per this claim, we have $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_7} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_6} - \mathsf{negl}(\lambda)$; therefore, we assume A wins $\mathbf{Game}_7$.

$\mathbf{Game}_8$. This game is identical to $\mathbf{Game}_7$, except that it aborts and outputs 0 if the event $\mathsf{BAD}_{5,4}$ occurs.

Claim 8. $\Pr[\mathsf{BAD}_{5,4}] \leq \mathsf{negl}(\lambda)$.

Proof. As $\mathsf{BAD}_{5,4}$ occurs, there exists a single prover session pid_i per verifier session $\mathsf{vid}^*{}_i$, such that $\mathsf{B}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} \neq 0$ and the Prove_2 query for pid_i was made after the Ver_1 query for $\mathsf{vid}^*{}_i$. In the following, we say that the sessions pid_i and $\mathsf{vid}^*{}_i$ are *linked*. This implicitly assigns the verifier sessions the same order as the chronological order among the prover sessions they are linked to. That is, without loss of generality we assign successful verifier session index i iff it is linked to the prover session for which the ith query to Prove_2 query was made.

We show in the following that this case cannot occur if the adversary makes at most ℓ Prove_3 queries. Let $\mathsf{vid}^*{}_i \in$ success be the *i-th* successful verifier session. Assume Equation (16) holds for all successful verifier sessions vid^*. It follows that

$$-pk_{1[R_{\mathsf{vid}^*{}_i}]} - \sum_{k \in [Q_{\mathsf{Prove}}]} ((\mathsf{y}_{pid_k} + \tilde{\mathsf{y}}_{pid_k})^3 + \mathsf{c}_{pid_k}) \cdot \mathsf{A}_{pid_k}{}_{[R_{\mathsf{vid}^*{}_i}]} - \left(\mathsf{B}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} \cdot \mathsf{y}_{pid_i} + h_{[R_{\mathsf{vid}^*{}_i}]}\right)^3 - c_{\mathsf{vid}^*{}_i,1} = 0. \quad (30)$$

Let $\kappa \leq Q_{\mathsf{Prove}}$ be the last index satisfying $\mathsf{A}_{pid_\kappa}{}_{[R_{\mathsf{vid}^*{}_i}]} \neq 0$. It follows that

$$-pk_{1[R_{\mathsf{vid}^*{}_i}]} - \sum_{k \in [\kappa-1]} ((\mathsf{y}_{pid_k} + \tilde{\mathsf{y}}_{pid_k})^3 + \mathsf{c}_{pid_k}) \cdot \mathsf{A}_{pid_k}{}_{[R_{\mathsf{vid}^*{}_i}]} - ((\mathsf{y}_{pid_\kappa} + \tilde{\mathsf{y}}_{pid_\kappa})^3 + \mathsf{c}_{pid_\kappa}) \cdot \mathsf{A}_{pid_\kappa}{}_{[R_{\mathsf{vid}^*{}_i}]} - \left(\mathsf{B}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} \cdot \mathsf{y}_{pid_i} + h_{[R_{\mathsf{vid}^*{}_i}]}\right)^3 - c_{\mathsf{vid}^*{}_i,1} = 0. \quad (31)$$

We show that $\kappa \leq i$ holds, i.e., for any successful verifier session $\mathsf{vid}^*{}_i$, it holds that $\mathsf{A}_{pid_j}{}_{[R_{\mathsf{vid}^*{}_i}]} = 0$ for all $j > i$. Assume for the sake of contradiction that $\kappa > i$, and define the polynomial

$$P(\mathcal{X}) = -pk_{1[R_{\mathsf{vid}^*{}_i}]} - \sum_{k \in [\kappa-1]} ((\mathsf{y}_{pid_k} + \tilde{\mathsf{y}}_{pid_k})^3 + \mathsf{c}_{pid_k}) \cdot \mathsf{A}_{pid_k}{}_{[R_{\mathsf{vid}^*{}_i}]} - ((\mathcal{X} + \tilde{\mathsf{y}}_{pid_\kappa})^3 + \mathsf{c}_{pid_\kappa}) \cdot \mathsf{A}_{pid_\kappa}{}_{[R_{\mathsf{vid}^*{}_i}]} - \left(\mathsf{B}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} \cdot \mathsf{y}_{pid_i} + h_{[R_{\mathsf{vid}^*{}_i}]}\right)^3 - c_{\mathsf{vid}^*{}_i,1}. \quad (32)$$

If Equation (31) holds, then $P(\mathsf{y}_{pid_\kappa}) = 0$. We distinguish two cases: The first case is that P is non-zero. This case occurs with probability at most $\frac{3}{q}$ per the Schwartz-Zippel lemma, because y_{pid_κ} is uniformly random, and it is sampled after all the values in P are fixed. The other case is that P is the zero-polynomial. This case cannot occur, because it implies that $\mathsf{A}_{pid_\kappa}{}_{[R_{\mathsf{vid}^*{}_i}]} = 0$ since $\mathsf{A}_{pid_\kappa}{}_{[R_{\mathsf{vid}^*{}_i}]}$ is the coefficient of $\mathcal{X}^3$, a contradiction.

Thus, we have that $\kappa \leq i$, i.e., $\mathsf{A}_{pid_j}{}_{[R_{\mathsf{vid}^*{}_i}]} = 0$ for all $j > i$. Consequently, Equation (31) is equivalent to

$$-pk_{1[R_{\mathsf{vid}^*{}_i}]} - \sum_{k \in [i-1]} ((\mathsf{y}_{pid_k} + \tilde{\mathsf{y}}_{pid_k})^3 + \mathsf{c}_{pid_k}) \cdot \mathsf{A}_{pid_k}{}_{[R_{\mathsf{vid}^*{}_i}]} - ((\mathsf{y}_{pid_i} + \tilde{\mathsf{y}}_{pid_i})^3 + \mathsf{c}_{pid_i}) \cdot \mathsf{A}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} - \left(\mathsf{B}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} \cdot \mathsf{y}_{pid_i} + h_{[R_{\mathsf{vid}^*{}_i}]}\right)^3 - c_{\mathsf{vid}^*{}_i,1} = 0. \quad (33)$$

Define the polynomial

$$P'(\mathcal{X}) = -pk_{1[R_{\mathsf{vid}^*{}_i}]} - \sum_{k \in [i-1]} ((\mathsf{y}_{pid_k} + \tilde{\mathsf{y}}_{pid_k})^3 + \mathsf{c}_{pid_k}) \cdot \mathsf{A}_{pid_k}{}_{[R_{\mathsf{vid}^*{}_i}]} - ((\mathcal{X} + \tilde{\mathsf{y}}_{pid_i})^3 + \mathsf{c}_{pid_i}) \cdot \mathsf{A}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} - \left(\mathsf{B}_{pid_i}{}_{[R_{\mathsf{vid}^*{}_i}]} \cdot \mathcal{X} + h_{[R_{\mathsf{vid}^*{}_i}]}\right)^3 - c_{\mathsf{vid}^*{}_i,1}. \quad (34)$$

Again, we distinguish two cases: If P' is a non-zero polynomial, $P'(\mathsf{y}_{pid_i})$ occurs with probability at most $\frac{3}{q}$ per the Schwartz-Zippel lemma since y_{pid_i} is sampled uniformly at random after all values in P' get fixed. If P' is the zero-polynomial, it holds that

$$\mathsf{A}_{pid_i\,[R_{\mathsf{vid}^*_i}]} = \mathsf{B}_{pid_i\,[R_{\mathsf{vid}^*_i}]}{}^3, \tag{35}$$

which implies that, if $\mathsf{B}_{pid_i\,[R_{\mathsf{vid}^*_i}]} \neq 0$, then $\mathsf{A}_{pid_i\,[R_{\mathsf{vid}^*_i}]} \neq 0$. However, there are $\ell+1$ unique successful verifier sessions $\mathsf{vid}^*{}_i$ and $\ell+1$ unique prover sessions pid_i for $i \in [\ell+1]$ with $\mathsf{B}_{pid_i\,[R_{\mathsf{vid}^*_i}]} \neq 0$ (as Event $\mathsf{BAD}_{5,4}$ occurs). Thus, it must hold that $\mathsf{A}_{pid_i\,[R_{\mathsf{vid}^*_i}]} \neq 0$ for all $i \in [\ell+1]$. Per **Game**$_3$, A must make a Prove_3 query for all pid_i. However, this is a contradiction, because A must make $\ell+1$ Prove_3 queries, a query for each pid_i for $i \in [\ell+1]$, while it is allowed to make at most ℓ Prove_3 queries. □

Per this claim, we have $\mathbf{Adv}^{\mathbf{Game}_8}_{\mathsf{A}} \geq \mathbf{Adv}^{\mathbf{Game}_7}_{\mathsf{A}} - \mathsf{negl}(\lambda)$, and since A wins **Game**$_7$, it must win **Game**$_8$, as well. Consequently, it holds that $\mathbf{Adv}^{\ell\text{-OMMIM}}_{\mathsf{A},\mathsf{mID}} \leq \mathbf{Adv}^{\mathbf{Game}_8}_{\mathsf{A}} + \mathsf{negl}(\lambda)$. However, winning **Game**$_8$ implies that $\mathsf{BAD}_{5,1}$, $\mathsf{BAD}_{5,2}$, $\mathsf{BAD}_{5,3}$, $\mathsf{BAD}_{5,4}$ do not occur, and since $\mathsf{BAD}_{5,1} \wedge \mathsf{BAD}_{5,2} \wedge \mathsf{BAD}_{5,3} \wedge \mathsf{BAD}_{5,4} = \mathsf{BAD}_5$, this means BAD_5 does not occur; therefore, we assume in the following that there exists a successful verifier session vid^*, for which

$$-pk_{1\,[R_{\mathsf{vid}^*}]} - \sum_{pid \in [Q_{\mathsf{Prove}}]} ((\mathsf{y}_{pid} + \tilde{\mathsf{y}}_{pid})^3 + \mathsf{c}_{pid}) \cdot \mathsf{A}_{pid\,[R_{\mathsf{vid}^*}]} - \left(\sum_{pid \in [Q_{\mathsf{Prove}}]} \mathsf{B}_{pid\,[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid} + h_{[R_{\mathsf{vid}^*}]}\right)^3 - c_{\mathsf{vid}^*,1} \neq 0. \tag{36}$$

Reducing **Game**$_8$ *to* dlog. We construct a reduction R_4 that, given an adversary A that wins **Game**$_8$, wins the dlog game. Given a discrete logarithm challenge U, R_4 behaves as follows. First, it chooses a public parameter $pp = (\mathbb{G}, q, g, h)$ with known $w = \mathsf{dlog}_g h$, sets $pk_1 = U$, and runs A with input pk_1 (and access to pp). During the execution of A, R_4 simulates the verifier's and the honest prover's oracles. It simulates the verifier's oracles Ver_1, Ver_2, and Ver_3 exactly as described by the protocol mID. For the simulation of the prover's oracles, it defines $pid := 0$ and simulates the oracles as follows:

- $\mathsf{Prove}_1()$: increment pid, sample $\mathsf{z}_{pid}, u_{pid,1}, u_{pid,2} \leftarrow \mathbb{Z}_q$, compute and return $\mathsf{A}_{pid} := g^{\mathsf{z}_{pid}} \cdot pk_1^{-u_{pid,1}}$, $\mathsf{B}_{pid} := g^{u_{pid,2}}$.
- $\mathsf{Prove}_2(pid, \mathsf{c}_{pid}, \{B_k\}_{k\in[n]})$: First, convert the algebraic representations of B_k for all $k \in [n]$ to the basis (g, h). This is possible because pk_1 is the only group element with an unknown discrete logarithm and due to the changes from **Game**$_1$ we know that $pk_{1\,[B_k]} = 0$ for all B_k. Let y_k be the respective h component. Define $\tilde{\mathsf{y}}_{pid} := \sum_{k=2}^{n} y_k$, compute and return $\mathsf{y}_{pid} := (u_{pid,1} - \mathsf{c}_{pid})^{\frac{1}{3}} - \tilde{\mathsf{y}}_{pid}$, and $\mathsf{b}_{pid} := u_{pid,2} - \mathsf{y}_{pid} \cdot w$.
- $\mathsf{Prove}_3(\mathrm{pid}, \{b_k, y_k\}_{k\in[n]})$: abort and return 0 if there is $k \in [n]$ with $g^{b_k} \cdot h^{y_k} \neq B_k$. Return z_{pid}.

When A closes $\ell+1$ verifier sessions successfully, there are $\ell+1$ successful verifier sessions vid satisfying Equation (7). The algebraic coefficients of each R_{vid} submitted by A alongside Ver_1 queries of session vid satisfy Equation (9). Using the internal state of the honest prover (here we rely on the fact that A makes a Prove_3 query due to the changes in **Game**$_3$), we substitute $\mathsf{A}_{pid} = g^{\mathsf{z}_{pid}} \cdot pk_1^{(\mathsf{y}_{pid}+\tilde{\mathsf{y}}_{pid})^3+\mathsf{c}_{pid}}$ in Equation (9), which yields

$$g^{g_{[R_{vid}]}} \cdot h^{h_{[R_{vid}]}} \cdot pk_1^{pk_{1\,[R_{vid}]}} \cdot \prod_{pid\in[Q_{\mathsf{Prove}}]} \left(g^{\mathsf{z}_{pid}} \cdot pk_1^{(\mathsf{y}_{pid}+\tilde{\mathsf{y}}_{pid})^3+\mathsf{c}_{pid}}\right)^{\mathsf{A}_{pid\,[R_{vid}]}} \cdot h^{\mathsf{y}_{pid}\cdot \mathsf{B}_{pid\,[R_{vid}]}} \cdot \prod_{j\in[n]} pk_j^{c_{vid,j}+y_{vid}^3} = g^{z_{vid}} \cdot h^{y_{vid}}. \tag{37}$$

Taking the discrete logarithm of both sides of this equation gives

$$g_{[R_{vid}]} + w \cdot h_{[R_{vid}]} + sk_1 \cdot pk_{1\,[R_{vid}]} + \sum_{pid\in[Q_{\mathsf{Prove}}]} (\mathsf{z}_{pid} + sk_1 \cdot ((\mathsf{y}_{pid}+\tilde{\mathsf{y}}_{pid})^3 + \mathsf{c}_{pid})) \cdot \mathsf{A}_{pid\,[R_{vid}]} + w \cdot \mathsf{y}_{pid} \cdot \mathsf{B}_{pid\,[R_{vid}]} + \sum_{j\in[n]} sk_j \cdot (c_{vid,j} + y_{vid}^3) = z_{vid} + w \cdot y_{vid}. \tag{38}$$

Note that R_4 can compute sk_j for all $1 < j \leq n$ such that $pk_j = g^{sk_j}$, because the representation of pk_j only contains group elements with known discrete logarithm to the base g. More specifically, the only group element with unknown discrete logarithm is pk_1, and due to the changes made in **Game**$_2$, we know that there is no pk_1 component in the representation of any pk_j. Additionally, from **Game**$_8$, we know that there exists a successful verifier session vid^* satisfying Equation (36). It follows that $sk_1 = \frac{N}{D}$, where $N := g_{[R_{\mathsf{vid}^*}]} + w \cdot (h_{[R_{\mathsf{vid}^*}]} - y_{\mathsf{vid}^*}) + \sum_{pid\in[Q_{\mathsf{Prove}}]} \mathsf{z}_{pid} \cdot \mathsf{A}_{pid\,[R_{\mathsf{vid}^*}]} + w \cdot \mathsf{y}_{pid} \cdot \mathsf{B}_{pid\,[R_{\mathsf{vid}^*}]} + \sum_{j=2}^{n} sk_j \cdot (c_{\mathsf{vid}^*,j} + y^3_{\mathsf{vid}^*}) - z_{\mathsf{vid}^*}$, $D := -pk_{1\,[R_{\mathsf{vid}^*}]} - \sum_{pid\in[Q_{\mathsf{Prove}}]} ((\mathsf{y}_{pid}+\tilde{\mathsf{y}}_{pid})^3 + \mathsf{c}_{pid}) \cdot \mathsf{A}_{pid\,[R_{\mathsf{vid}^*}]} - c_{\mathsf{vid}^*,1} - y^3_{\mathsf{vid}^*}$. Per **Game**$_4$, we have $y_{\mathsf{vid}^*} = \sum_{pid\in[Q_{\mathsf{Prove}}]} \mathsf{B}_{pid\,[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid} + h_{[R_{\mathsf{vid}^*}]}$. By substituting y_{vid^*} for $\sum_{pid\in[Q_{\mathsf{Prove}}]} \mathsf{B}_{pid\,[R_{\mathsf{vid}^*}]} \cdot \mathsf{y}_{pid} + h_{[R_{\mathsf{vid}^*}]}$ in Equation 36, we obtain the equation $D \neq 0$, and thus, R_4 can compute sk_1 and win Game dlog from session vid^*. □

E Proofs for BM_SB

E.1 Blindness

Proof. Let A be an adversary playing Game mBlind against BM_SB. Let the transcripts of the two executions be $T_0 := \{T_{0,1}, \ldots, T_{0,n}\}$ and $T_1 := \{T_{1,1}, \ldots, T_{1,n}\}$, where $T_{i,j} = (A_{i,j}, B_{i,j}, c_{i,j}, \{com_{i,k}\}_{k\in[n]}, y_{i,j}, b_{i,j}, \{b_{i,k}, y_{i,k}\}_{k\in[n]}, z_{i,j})$ for $i \in \{0,1\}$ and $j \in [n]$, and let $(m_0, \sigma_0 = (\overline{R}_0, \overline{y}_0, \overline{z}_0))$ and $(m_1, \sigma_1 = (\overline{R}_1, \overline{y}_1, \overline{z}_1))$ be the message-signature pairs. Define $V_{i,k}(\mathsf{A}) := (T_i, m_k, \sigma_k)$ for $i, k \in \{0,1\}$.

First, we show that, for all $i, k \in \{0,1\}$, $V_{i,k}(\mathsf{A})$ determines a valid user state $\mathsf{ust} := (r_{i,k}, \alpha_{i,k}, \beta_{i,k,1}, \ldots, \beta_{i,k,n})$. [10] We construct

[10] Note that the user state is determined via the tuple ust because all other values on the user side are fixed (given the transcripts and the message-signature pair).

a user state $\mathsf{ust}_{i,k}$ by defining the blinding factors

$$\alpha_{i,k} = \frac{\overline{y}_k}{\sum_{j\in[n]} y_{i,j}}, \tag{39}$$

$$r_{i,k} = \overline{z}_k - \alpha_{i,k}^3 \cdot \sum_{j\in[n]} z_{i,j} - \alpha_{i,k} \cdot \sum_{j\in[n]} b_{i,j}, \tag{40}$$

$$\beta_{i,k,j} = c_{i,j} - \mathsf{H}_{\mathsf{sig}}(\vec{K}_k, pk_{k,j}, m_k, \overline{R}_k) \cdot \alpha_{i,k}^{-3}. \tag{41}$$

Next, we show that these values are uniformly distributed in $\mathbb{Z}_q$ (before A gets access to m_k, σ_k). The uniformity of $\alpha_{i,k}$ follows from the uniformity of $\overline{y}_k$, which is computed by the experiment as $\overline{y}_k = \alpha_k \cdot \sum_{j\in[n]} y_{k,j}$, where α_k is the real blinding factor used by the experiment in the k-th user session. Similarly, the uniformity of $r_{i,k}$ follows from the uniformity of $\overline{z}_k$, which is computed by the experiment as $\overline{z}_k = r_k + \alpha_k^3 \cdot z_k + \alpha_k \cdot b_k$, and r_k is chosen uniformly at random. Finally, $\beta_{i,k,j}$ is uniformly distributed as long as A does not query $\mathsf{H}_{\mathsf{sig}}$ on $(\vec{K}_k, pk_{k,j}, m_k, \overline{R}_k)$. Since $\overline{R}_k = g^{r_k} \cdot \sum_j pk_j^{\alpha_k^3 \cdot \beta_{k,j}} \cdot A^{\alpha_{i,k}^3} \cdot B^{\alpha_k}$ is computed by the experiment, and thus, it is uniformly random due to the uniformity of the real blinding factor r_k, the probability that A queries $\mathsf{H}_{\mathsf{sig}}$ on $\overline{R}_k$ is at most $\frac{Q_H}{q}$, where Q_H is the number of hash queries A makes to $\mathsf{H}_{\mathsf{sig}}$.

Finally, we show that such a user state ust defines a valid signature $(\overline{R}_k, \overline{y}_k, \overline{z}_k)$ and hashes $\overline{c}_{k,1}, \ldots, \overline{c}_{k,n}$ such that

$$\overline{R}_k \cdot \prod_{j\in[n]} pk_j^{\overline{c}_{k,j}+\overline{y}_k^3} = g^{\overline{z}_k} \cdot h^{\overline{y}_k},$$

for all $i, k \in \{0, 1\}$. We assume A closes both signing sessions successfully, otherwise, the game outputs $(\perp, \perp)$, and in this case, A's advantage is in winning the game is 0. This implies that both transcripts T_1, and T_2 are valid.

Since the k-th user session outputs a valid signature $(\overline{R}_k, \overline{y}_k, \overline{z}_k)$ and hashes $c_{k,j}$ for all $j \in [n]$, it holds that

$$\overline{R}_k \cdot \prod_{j\in[n]} pk_j^{\overline{c}_{k,j}+\overline{y}_k^3} = g^{\overline{z}_k} \cdot h^{\overline{y}_k},$$

hence

$$\overline{R}_k = g^{\overline{z}_k} \cdot h^{\overline{y}_k} \cdot \prod_{j\in[n]} pk_j^{-\overline{c}_{k,j}-\overline{y}_k^3}.$$

Substituting Equations (39)–(41) into this equation yields

$$\overline{R}_k = g^{r_{i,k}+\alpha_{i,k}^3 \cdot \sum_{j\in[n]} z_{i,j} + \alpha_{i,k} \cdot \sum_{j\in[n]} b_{i,j}} \cdot h^{\alpha_{i,k} \cdot \sum_{j\in[n]} y_{i,j}} \cdot \prod_{j\in[n]} pk_j^{-(c_{i,j}-\beta_{i,k,j}) \cdot \alpha_{i,k}^3 - \alpha_{i,k}^3 \cdot (\sum_{j\in[n]} y_{i,j})^3}.$$

We rearrange the equation as

$$\overline{R}_k = g^{r_{i,k}} \cdot g^{\alpha_{i,k}^3 \cdot \sum_{j\in[n]} z_{i,j}} \cdot \prod_{j\in[n]} pk_j^{\alpha_{i,k}^3 \cdot (-c_{i,j} - (\sum_{j\in[n]} y_{i,j})^3)} \cdot g^{\alpha_{i,k} \cdot \sum_{j\in[n]} b_{i,j}} \cdot h^{\alpha_{i,k} \cdot \sum_{j\in[n]} y_{i,j}} \cdot \prod_{j\in[n]} pk_j^{\alpha_{i,k} \cdot \beta_{i,j,k}}.$$

Since T_i is a valid transcript for all $i \in \{0, 1\}$, it follows that $A_i \cdot \prod_{j\in[n]} pk_j^{c_j + (\sum_{j\in[n]} y_{i,j})^3} = \prod_{j\in[n]} A_{i,j} \cdot pk_j^{c_j+(\sum_{j\in[n]} y_{i,j})^3} = g^{z_i}$, and $B_i = g^{b_i} \cdot h^{y_i}$, where $b_i = \sum_{j\in[n]} b_{i,j}, y_i = \sum_{j\in[n]} y_{i,j}, z_i = \sum_{j\in[n]} z_{i,j}$. Consequently, we have

$$\overline{R}_k = g^{r_{i,k}} \cdot A_i^{\alpha_{i,k}^3} \cdot B_i^{\alpha_{i,k}} \cdot \prod_{j\in[n]} pk_j^{\beta_{i,j,k} \cdot \alpha_{i,k}^3}.$$

It holds that $(\overline{R}_k, \overline{y}_k, \overline{z}_k) = (g^{r_{i,k}} \cdot A_i^{\alpha_{i,k}^3} \cdot B_i^{\alpha_{i,k}} \cdot \prod_{j\in[n]} pk_j^{\beta_{i,j,k} \cdot \alpha_{i,k}^3}, \alpha_{i,k} \cdot \sum_{j\in[n]} y_{i,j}, r_{i,k} + \alpha_{i,k}^3 \cdot \sum_{j\in[n]} z_{i,j} + \alpha_{i,k} \cdot \sum_{j\in[n]} b_{i,j})$, and $\overline{c}_{k,j} = \mathsf{H}_{\mathsf{sig}}(\vec{K}_k, pk_{k,j}, g^{r_{i,k}} \cdot A_i^{\alpha_{i,k}^3} \cdot B_i^{\alpha_{i,k}} \cdot \prod_{j\in[n]} pk_j^{\beta_{i,j,k} \cdot \alpha_{i,k}^3}, m_k)$, which concludes the claim. □

E.2 One-More Unforgeability

We now give the proof of Theorem 10.

Proof outline. We prove one-more unforgeability following prior work [28, 32]. BM_SB is built from a secure multi-ID scheme mID. It remains to show that the OMMIM security of mID implies the OMUF security of BM_SB. To this end, we provide a reduction R_5 that exploits any algebraic forger A winning Game OMUF against our BM_SB scheme to win Game OMMIM against mID. However, for R_5 to function properly, it requires a few conditions to hold; therefore, we first prove that these restrictions indeed hold. In particular, we start by showing that A must make (at least) an $\mathsf{H}_{\mathsf{sig}}$-query for each valid signature it outputs. As A is algebraic, it must submit a representation for each group element in its queries to $\mathsf{H}_{\mathsf{sig}}$. This allows R_5 to learn the representation for each group element $\overline{R}$ that occurs in the forgeries output by A. We then show that a specific relation must hold between the forgeries A outputs and the representation of $\overline{R}$ that A submits in the corresponding query to $\mathsf{H}_{\mathsf{sig}}$. Finally, we provide the reduction R_5 that runs Game OMMIM against mID and uses its challenger's prover and verifier oracles to simulate the signers and the random oracle $\mathsf{H}_{\mathsf{sig}}$, respectively, for A. When A terminates and outputs $\ell + 1$ valid signatures, R_5 crafts responses to close the verifier sessions using those signatures, which allows it to win the game OMMIM.

Proof. Since we provide proofs in the AGM, we assume that all adversaries A are algebraic, i.e., A outputs a representation for each group element it outputs. We use the notation $o_{1[o_2]}$ to denote the exponent of o_1 in the representation of o_2. For simplicity and compactness, we often transform the representations submitted by A to a reduced form [32, 33]. Given a basis $\vec{I} = (o_1, \ldots, o_t)$, a group element $o' \in \vec{I}$ and an arbitrary group element o'', we denote $o'_{[o'']\vec{I}}$ as the exponent of o' in the representation of o'' after reducing the representation of o'' to the basis $\vec{I}$.

We start by defining a series of games:

Game$_0$. This game is the OMUF game against BM_SB.

Let BAD_6 be the event that occurs if A outputs a valid signature $(\overline{R}, \overline{y}, \overline{z})$ under a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$ such that no entry $(\vec{K}, pk_k, \overline{R}, m)$ in the hash table $\mathsf{T}_{\mathsf{sig}}$ of $\mathsf{H}_{\mathsf{sig}}$ for any $k \in [n]$ and any message m

Game$_1$. This game is identical to **Game$_0$**, except that it aborts and outputs 0 if BAD_6 occurs.

Claim 9. $\Pr[\mathsf{BAD}_6] \leq \mathsf{negl}(\lambda)$.

Proof. We assume w.l.o.g. that A only outputs $\ell+1$ signatures. From the verification equation, we know that a signature $(\overline{R}, \overline{y}, \overline{z})$ for a message m under a vector of public keys $\vec{K} = (pk_1, \ldots, pk_n)$ is valid iff $g^{\overline{z}} \cdot h^{\overline{y}} = \overline{R} \cdot \prod_{i\in[n]} pk_i^{\overline{c}_i + \overline{y}^3}$, where $\overline{c}_i = \mathsf{H}_{\mathsf{sig}}(\vec{K}, pk_i, \overline{R}, m)$. Thus, outputting a valid signature without querying $\mathsf{H}_{\mathsf{sig}}$ previously on the input $(\vec{K}, pk_i, \overline{R}, m)$ is equivalent to guessing the output of $\mathsf{H}_{\mathsf{sig}}$ on this input. As $\mathsf{H}_{\mathsf{sig}}$'s output is uniformly random on new inputs, the probability that A outputs a valid signature without querying $\mathsf{H}_{\mathsf{sig}}$ beforehand on $(\vec{K}, pk_i, \overline{R}, m)$ for all $i \in [n]$ is at most $\frac{1}{q^n}$ because it must guess n independent random values $\overline{c}_i$. Since A outputs $\ell+1$ signatures, the probability that it did not make a hash query for at least a valid signature is $\frac{\ell+1}{q^n}$. □

Per this claim, we have that $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_1} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_0} - \mathsf{negl}(\lambda)$; therefore, we assume that A wins $\mathbf{Game}_1$ and BAD_6 does not occur.

Define the event BAD_7 that occurs if A outputs a valid signature $(\overline{R}, \overline{y}, \overline{z})$ under a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$, and there is $i \in [n]$ such that the representation of pk_i contains an h component.

Game$_2$. This game is identical to **Game**$_1$, except that it aborts and outputs 0 if BAD_7 occurs.

Claim 10. $\Pr[\mathsf{BAD}_7] \leq \mathbf{Adv}_{\mathsf{R}_6}^{\mathsf{dlog}}$.

Proof. We construct a reduction R_6 that takes a dlog challenge U as input, embeds U in h, and simulates the signer's oracles exactly as described by the protocol. When A outputs $\vec{K} = \{pk_1, \ldots, pk_n\}$, and there exists $i \in [n]$, such that the representation of pk_i contains an h component or $B_{1,\mathsf{sid}}$ component for some signing session sid (recall that $B_{1,\mathsf{sid}} = g^{b_{1,\mathsf{sid}}} \cdot h^{y_{1,\mathsf{sid}}}$). Knowing that $pk_i = g_{\vec{I}}^{g_{[pk_i]}} \cdot h_{\vec{I}}^{h_{[pk_i]}}$ (note that R_6 knows the discrete logarithms of all group elements that A may use in the representation of pk_i except h, therefore, it can aggregate all non-h components as a single g component and reduce the representation to the basis $I = (g, h)$), and hence $sk_i = g_{[pk_i]_{\vec{I}}} + \mathsf{dlog}_g h \cdot h_{[pk_i]_{\vec{I}}}$, R_6 computes and outputs

$$\mathsf{dlog}_g U = \mathsf{dlog}_g h = \frac{sk_i - g_{[pk_i]_{\vec{I}}}}{h_{[pk_i]_{\vec{I}}}}.$$

This completes the proof. □

It follows that $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_2} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_1} - \mathbf{Adv}_{\mathsf{R}_6}^{\mathsf{dlog}}$. Thus, we assume that A wins this game and that BAD_7 does not occur.

Next, we define the event BAD_8 that occurs if A outputs a valid signature $(\overline{R}, \overline{y}, \overline{z})$ under a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$ with a corresponding $\mathsf{H}_{\mathsf{sig}}$ query $(\vec{K}, pk_k, \overline{R}, m)$ for some $k \in [n]$ and some message m, and the equation $h_{[\overline{R}]_{\vec{I}}} + \sum_{k=1}^n (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot h_{[pk_k]_{\vec{I}}} - \overline{y} \neq 0$ holds, where $I = (g, h)$.

Game$_3$. This game is identical to **Game**$_2$, but it aborts and outputs 0 if BAD_8 occurs.

Claim 11. $\Pr[\mathsf{BAD}_8] \leq \mathbf{Adv}_{\mathsf{R}_7}^{\mathsf{dlog}}$.

Proof. We show that winning **Game**$_2$ while the equation $h_{[\overline{R}]_{\vec{I}}} + \sum_{k=1}^n (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot h_{[pk_k]_{\vec{I}}} - \overline{y} \neq 0$ holds is hard under the dlog problem. We provide a reduction R_7 that given a discrete logarithm challenge U, sets $h = U$ and samples the other variables in pp and simulates the (honest) signers oracles exactly as described by the signing protocol:

- Sign_1: increment the session identifier sid (initially, sid = 0), sample randomly $com_{1,\mathsf{sid}}, z_{1,\mathsf{sid}}, u_{1,\mathsf{sid}}, u_{2,\mathsf{sid}} \leftarrow \mathbb{Z}_q$, compute $A_{1,\mathsf{sid}} := g^{z_{1,\mathsf{sid}}} \cdot pk_1^{-u_{1,\mathsf{sid}}}$, $B_{1,\mathsf{sid}} := g^{u_{2,\mathsf{sid}}}$, and return $A_{1,\mathsf{sid}}, B_{1,\mathsf{sid}}, com_{1,i}$.
- $\mathsf{Sign}_2(\mathsf{sid}, c_{1,\mathsf{sid}}, \{com_i\}_{i\in[n]})$: If there is com_i for $i \in [n]$ with $\mathsf{T}_{\mathsf{com}}[com_i] = \perp$, sample a random $y_i \leftarrow \mathbb{Z}_q$ and set $\mathsf{T}_{\mathsf{com}}[com_i] = y_i$. Extract $y_i := \mathsf{T}_{\mathsf{com}}[com_i]$ for all $i \in [n]$, compute $\tilde{y} := \sum_{i=2}^n y_i$, compute $y_{1,\mathsf{sid}} := (u_{1,\mathsf{sid}} - c_{1,\mathsf{sid}} - \tilde{y})^{1/3}$, $b_{1,\mathsf{sid}} := u_{2,\mathsf{sid}} - y_{1,\mathsf{sid}} \cdot \mathsf{dlog}_g h$, set $\mathsf{T}_{\mathsf{com}}[y_{1,\mathsf{sid}}] = com_{1,\mathsf{sid}}$, and return $y_{1,\mathsf{sid}}, b_{1,\mathsf{sid}}$.
- $\mathsf{Sign}_3(\mathsf{sid})$: return $z_{1\mathsf{sid}}$.
- $\mathsf{H}_{\mathsf{sig}}(\mathsf{Q})$: via lazy sampling. Return $\mathsf{T}_{\mathsf{sig}}[Q]$ if $\mathsf{T}_{\mathsf{sig}}[Q] \neq \perp$; otherwise, sample ξ uniformly at random, set $\mathsf{T}_{\mathsf{sig}}[Q] = \xi$, and return ξ.
- $\mathsf{H}_{\mathsf{com}}(\mathsf{Q})$: via lazy sampling. Return $\mathsf{T}_{\mathsf{com}}[Q]$ if $\mathsf{T}_{\mathsf{com}}[Q] \neq \perp$; otherwise, sample ξ uniformly at random, set $\mathsf{T}_{\mathsf{com}}[Q] = \xi$, and return ξ.

Assume A outputs a valid signature $(\overline{R}, \overline{y}, \overline{z})$ for a message m with a corresponding hash query $(\vec{K}, pk_k, \overline{R}, m)$ for some $k \in [n]$ (due to **Game**$_1$ there is at least one such a query) under a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$. It follows that

$$g^{\overline{z}} \cdot h^{\overline{y}} = \overline{R} \cdot \prod_{k=1}^n pk_k^{\overline{c}_k + \overline{y}^3}. \tag{42}$$

Using the representation of $\overline{R}$ that A submits to $\mathsf{H}_{\mathsf{sig}}$, we write

$$\overline{R} = g^{g_{[\overline{R}]}} \cdot h^{h_{[\overline{R}]}} \cdot \prod_{i=1}^n pk_i^{pk_{i[\overline{R}]}} \cdot \prod_{\mathsf{sid}\in[Q_S]} A_{\mathsf{sid}}^{A_{\mathsf{sid}}[\overline{R}]} \cdot B_{\mathsf{sid}}^{B_{\mathsf{sid}}[\overline{R}]} \cdot \prod_{j=1}^{\nu} h_j^{h_j[\overline{R}]}. \tag{43}$$

Combining both equations yields

$$g^{\overline{z}} \cdot h^{\overline{y}} = g^{g_{[\overline{R}]}} \cdot h^{h_{[\overline{R}]}} \cdot \prod_{\mathsf{sid}\in[Q_S]} A_{\mathsf{sid}}^{A_{\mathsf{sid}}[\overline{R}]} \cdot B_{\mathsf{sid}}^{B_{\mathsf{sid}}[\overline{R}]} \cdot \prod_{j=1}^{\nu} h_j^{h_j[\overline{R}]} \cdot \prod_{i=1}^n pk_i^{\overline{c}_i + \overline{y}^3 + pk_{i[\overline{R}]}}. \tag{44}$$

Given the Signer's internal view of A_{sid} and B_{sid}, we have

$$g^{\overline{z}} \cdot h^{\overline{y}} = g^{g_{[\overline{R}]}} \cdot h^{h_{[\overline{R}]}} \cdot \prod_{\mathsf{sid}\in[Q_S]} g^{a_{\mathsf{sid}} \cdot A_{\mathsf{sid}}[\overline{R}]} \cdot (g^{b_{\mathsf{sid}}} \cdot h^{y_{\mathsf{sid}}})^{B_{\mathsf{sid}}[\overline{R}]} \cdot \prod_{i=1}^n pk_i^{\overline{c}_i + \overline{y}^3 + pk_{i[\overline{R}]}}. \tag{45}$$

By taking the discrete logarithm of both sides we get

$$\overline{z} + w \cdot \overline{y} = g_{[\overline{R}]} + w \cdot h_{[\overline{R}]} + \sum_{\mathsf{sid}\in[Q_{\mathsf{Sign}}]} a_{\mathsf{sid}} \cdot A_{\mathsf{sid}[\overline{R}]} + B_{\mathsf{sid}[\overline{R}]} \cdot (b_{\mathsf{sid}} + w \cdot y_{\mathsf{sid}}) + \sum_{k=1}^n (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot (g_{[pk_k]_{\vec{I}}} + w \cdot h_{[pk_k]_{\vec{I}}}),$$

where $I = (g, h)$. We aggregate the exponents in $\overline{R}$'s representation of the basis of known discrete logarithms as $g_{[\overline{R}]_{\vec{I}}}$, and all h components of $\overline{R}$ as $h_{[\overline{R}]_{\vec{I}}}$

$$\overline{z} + w \cdot \overline{y} = g_{[\overline{R}]_{\vec{I}}} + w \cdot h_{[\overline{R}]_{\vec{I}}} + \sum_{k=1}^{n} (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot (g_{[pk_k]_{\vec{I}}} + w \cdot h_{[pk_k]_{\vec{I}}}).$$

Then, R_7 can compute $w = \mathrm{dlog}_g h = \mathrm{dlog}_g U$ by rearranging the equation as

$$w = \frac{\overline{z} - g_{[\overline{R}]_{\vec{I}}} - \sum_{k=1}^{n} (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot g_{[pk_k]_{\vec{I}}}}{h_{[\overline{R}]_{\vec{I}}} + \sum_{k=1}^{n} (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot h_{[pk_k]_{\vec{I}}} - \overline{y}}, \quad (46)$$

if the denominator is non-zero, hence the claim. □

Per this claim, it holds that $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_3} \geq \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_2} - \mathbf{Adv}_{\mathsf{R}_7}^{\mathrm{dlog}}$, and thus, we assume in the following that A wins **Game**$_3$ and that BAD_8 does not occur.

Next, we define the event BAD_9, which occurs if A outputs a valid signature $(\overline{R}, \overline{y}, \overline{z})$ under a set of public keys $\vec{K} = \{pk_1, \ldots, pk_n\}$ with a corresponding $\mathsf{H}_{\mathsf{sig}}$ query $(\vec{K}, pk_k, \overline{R}, m)$ for some $k \in [n]$ and some message m, and $h_{[\overline{R}]_{\vec{I}}} \neq \overline{y}$, where $\vec{I} = (g, h)$.

Game$_4$. This game is identical to **Game**$_3$, except that it aborts and outputs 0 if BAD_9 occurs.

Claim 12. $\Pr[\mathsf{BAD}_9] = 0$.

Proof. Due to **Game**$_2$, we have that for all public keys of the corrupted signers pk_k for $2 \leq k \leq n$, it must hold that $h_{[pk_k]_{\vec{I}}} = 0$. Furthermore, we know that the honestly-computed public key pk_1 has no h component, i.e., $h_{[pk_1]_{\vec{I}}} = 0$. It follows that $\sum_{k=1}^{n} (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot h_{[pk_k]_{\vec{I}}} = 0$. Since per **Game**$_3$, $h_{[\overline{R}]_{\vec{I}}} + \sum_{k=1}^{n} (\overline{c}_k + \overline{y}^3 + pk_{k[\overline{R}]}) \cdot h_{[pk_k]_{\vec{I}}} - \overline{y} = 0$ holds, it follows that $h_{[\overline{R}]_{\vec{I}}} = \overline{y}$. □

It follows that $\mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_4} = \mathbf{Adv}_{\mathsf{A}}^{\mathbf{Game}_3}$ and A must win this **Game**$_4$, which rules out Event BAD_9.

Reducing ℓ-OMMIM to **Game**$_4$. We describe now our reduction R_1 that runs Game ℓ-OMMIM against mID with access to the oracles Prove = (Prove$_1$, Prove$_2$, Prove$_3$) and Ver = (Ver$_1$, Ver$_2$, Ver$_3$), and aims to close $\ell + 1$ verifier sessions successfully.

Initially, R_1 receives public parameter $pp = (\mathbb{G}, q, g, h)$ from the challenger and initializes an instance of **Game**$_4$ over pp.

Next, R_1 runs A on **Game**$_4$ and ensures that (i) it simulates the game for A perfectly, and (ii) it wins Game ℓ-OMMIM by simulating n provers; the honest prover of the public key pk_1 and $n-1$ corrupted provers (it uses the same public keys A uses to simulates the corrupted signers).

When A queries the honest signer oracles, or the random oracles $\mathsf{H}_{\mathsf{com}}$ and $\mathsf{H}_{\mathsf{sig}}$, R_1 perfectly simulates these oracles as follows:

- $\mathsf{Sign}_1()$: generate sid, $A_{1,\mathsf{sid}}, B_{1,\mathsf{sid}} \leftarrow \mathsf{Prove}_1()$, sample $com_{1,\mathsf{sid}} \leftarrow \mathbb{Z}_q$, and return sid, $A_{1,\mathsf{sid}}, B_{1,\mathsf{sid}}, com_{1,\mathsf{sid}}$.
- $\mathsf{Sign}_2(\mathsf{sid}, c_{1,\mathsf{sid}}, \{com_{i,\mathsf{sid}}\}_{i\in[n]})$: extract $\tilde{\mathsf{b}}_{\mathsf{i,sid}}$ and $\tilde{\mathsf{y}}_{\mathsf{i,sid}}$ from $\mathsf{T}_{\mathsf{com}}$ using $com_{i,\mathsf{sid}}$, compute $\tilde{\mathsf{B}}_{\mathsf{i,sid}} = g^{\tilde{\mathsf{b}}_{\mathsf{i,sid}}} \cdot h^{\tilde{\mathsf{y}}_{\mathsf{i,sid}}}$, make $\mathsf{Prove}_2(\mathsf{sid}, c_{1,\mathsf{sid}}, \{\tilde{\mathsf{B}}_{\mathsf{i,sid}}\}_{i\in[n]})$ query to obtain $b_{1,\mathsf{sid}}, y_{1,\mathsf{sid}}$, set $\mathsf{T}_{\mathsf{com}}[1, b_{1,\mathsf{sid}}, y_{1,\mathsf{sid}}] = com_{1,\mathsf{sid}}$, and return $y_{1,\mathsf{sid}}, b_{1,\mathsf{sid}}$.
- $\mathsf{Sign}_3(\mathsf{sid}, \{b_{i,\mathsf{sid}}, y_{i,\mathsf{sid}}\}_{i\in[n]})$: check and abort if there is a pair $(b_{i,\mathsf{sid}}, y_{i,\mathsf{sid}})$ such that $com_{i,\mathsf{sid}} \neq \mathsf{H}_{\mathsf{com}}(i, b_{i,\mathsf{sid}}, y_{i,\mathsf{sid}})$ or $B_{i,\mathsf{sid}} \neq g^{b_{i,\mathsf{sid}}} \cdot h^{y_{i,\mathsf{sid}}}$. Generate and return $z_{1,\mathsf{sid}} = \mathsf{Prove}_3(\mathsf{sid}, \{b_i, y_i\}_{i\in[n]})$.
- $\mathsf{H}_{\mathsf{sig}}(Q)$: via lazy sampling. Return $\mathsf{T}_{\mathsf{sig}}[Q]$ if $\mathsf{T}_{\mathsf{sig}}[Q] \neq \perp$; otherwise, parse Q as $(\vec{K}, pk_\kappa, \overline{R}, m)$. If $pk_\kappa \notin \vec{K}$ (this implicitly asserts $\kappa \in [n]$), sample $c \leftarrow \mathbb{Z}_q$, store $\mathsf{T}_{\mathsf{sig}}[Q] = c$, and return c. Otherwise, using the representation of $\overline{R}$ A submits, R_1 splits $\overline{R}$ into two group elements A_1^* and B_1^*, where $B_1^* := \prod_{\mathsf{sid}} B_{1,\mathsf{sid}}^{B_{1,\mathsf{sid}}[\overline{R}]} \cdot h^{\hat{h}}, \hat{h} := h_{[\overline{R}]_{\vec{I}}} - \sum_{\mathsf{sid}} B_{\mathsf{sid}[\overline{R}]} \cdot h_{[B_{\mathsf{sid}}]}, A_1^* := \overline{R}/B_1^*$, and sid is an index over all already opened signing sessions.[11] Note that splitting $\overline{R}$ this way ensures that all h components in $\overline{R}$, i.e. $h_{[\overline{R}]_{\vec{I}}}$, are gathered in B_1^*, because the only group elements containing h components are $B_{1,\mathsf{sid}}$ for all sid, and h itself. To see this, we note that the group elements A has seen are $g, h, pk_i, A_{\mathsf{sid}}, B_{\mathsf{sid}}$ for $i \in [n]$ and opened signing sessions sid. Per **Game**$_2$, pk_i does not contain h components, and the other group elements are generated honestly by the experiment OMMIM. It follows that $h_{[\overline{R}]_{\vec{I}}} = \sum_{\mathsf{sid}} h_{[B_{1,\mathsf{sid}}]} \cdot B_{1,\mathsf{sid}[\overline{R}]} + h_{[\overline{R}]}$. Moreover, per **Game**$_4$, this ensures that $y_1^* = h_{[\overline{R}]_{\vec{I}}} = \overline{y}$, where $\overline{y}$ is the field element in the forgery A eventually outputs if it decides to use $\overline{R}$ in its forgeries. Next, sample uniformly at random $a_k^*, b_k^* \leftarrow \mathbb{Z}_q$ and $y_k^* \leftarrow \mathbb{Z}_q^*$ for all $2 \leq k \leq n$, such that, $\sum_{k=2}^{n} a_k^* = 0$, $\sum_{k=2}^{n} b_k^* = 0$, and $\sum_{k=2}^{n} y_k^* = 0$. Generate $A_k^* := g^{a_k^*}, B_k^* := g^{b_k^*} \cdot h^{y_k^*}$ for all $2 \leq k \leq n$, and open n Verifier sessions by calling $\mathsf{Ver}_1(A_k^*, B_k^*)$ and obtain challenges $(\overline{c}_k, \{B_j^*\}_{j\in[n]})$ for $k \in [n]$.
Choosing the exponents with a zero-sum nullifies the effect of A_k^* and B_k^* for all $2 \leq k < n$ at the verifier side because the multiplication of these group elements is the identity 1, and this way, we can make Ver_3 queries for those provers by simply sending $z_k = 0$. Finally, store these values for later processing, set $\mathsf{T}_{\mathsf{sig}}[(\vec{K}, pk_i, \overline{R}, m)] = \overline{c}_i$, and return $\overline{c}_\kappa$.
- $\mathsf{H}_{\mathsf{com}}(Q)$: via lazy sampling. Return $\mathsf{T}_{\mathsf{com}}[Q]$ if $\mathsf{T}_{\mathsf{com}}[Q] \neq \perp$; otherwise, sample uniform ξ, set $\mathsf{T}_{\mathsf{com}}[Q] = \xi$, and return ξ.

Closing the verifier sessions. When A terminates and outputs $\ell+1$ valid signatures $\vec{F} = \{(\overline{R}_i, \overline{y}_i, \overline{z}_i)\}_{i\in[\ell+1]}$, R_1 does the following. First, it makes a Prove_2 query (using arbitrary syntactically correct input) for all prover sessions sid that (i) were opened via Prove_1 queries and no Prove_2 queries were made for them, and (ii) there exists a signature $(\overline{R}, \overline{y}, \overline{z}) \in \vec{F}$, such that, $\overline{R}$ occurs in a hash query $Q \in \mathsf{T}_{\mathsf{sig}}$, and $B_{\mathsf{sid}[\overline{R}]} \neq 0$. Making Prove_2 queries for those sessions grants A the openings for $B_{1,\mathsf{sid}}$. For all $j \in [\ell+1]$, assume the pair $((\overline{R}_j, \overline{y}_j, \overline{z}_j), m_j)$ is a valid signature under the keys set $\vec{K}_j$. It follows per **Game**$_1$ that there exists a hash query to $\mathsf{H}_{\mathsf{sig}}$ of the form $(\vec{K}_j, pk_k, \overline{R}_j, m_j)$ with $pk_k \in \vec{K}$, and thus, there is a verifier session vid_j that was opened using $(A_{1,j}^*, B_{1,j}^*), \ldots, (A_{n,j}^*, B_{n,j}^*)$ with $\overline{R}_j =$

[11] Recall that $\mathsf{B}_{\mathsf{sid}}$ denotes the group element B generated by the OMMIM experiment and was forwarded to A upon a Sign_1 query to the honest signer's oracle.

$A^*_{1,j} \cdot B^*_{1,j}$. R_1 needs to send the openings for B^*_k for all $k \in [n]$. R_1 already knows the openings for $B^*_2, \ldots, B^*_n$; therefore, it can simply make Ver_2 queries with input $(b^*_2, y^*_2), \ldots, (b^*_n, y^*_n)$. It remains for R_1 to compute the opening for B^*_1. It computes $b^*_1 = \sum_{\mathsf{sid}} b_{1,\mathsf{sid}} \cdot B_{1,\mathsf{sid}}[\overline{R}]$, and $y^*_1 = \sum_{\mathsf{sid}} y_{1,\mathsf{sid}} \cdot B_{1,\mathsf{sid}}[\overline{R}] + \hat{h}$ for all sid. Indeed, $B^*_1 = g^{b^*_1} \cdot h^{y^*_1}$ because $B^*_1 = \prod_{\mathsf{sid}} B_{1,\mathsf{sid}}^{B_{1,\mathsf{sid}}[\overline{R}]} \cdot h^{\hat{h}} = g^{\sum_{\mathsf{sid}} b_{\mathsf{sid}} \cdot B_{1,\mathsf{sid}}[\overline{R}]} \cdot h^{\sum_{\mathsf{sid}} y_{\mathsf{sid}} \cdot B_{1,\mathsf{sid}}[\overline{R}]} \cdot h^{\hat{h}} = g^{\sum_{\mathsf{sid}} b_{\mathsf{sid}} \cdot B_{1,\mathsf{sid}}[\overline{R}]} \cdot h^{\sum_{\mathsf{sid}} y_{\mathsf{sid}} \cdot B_{1,\mathsf{sid}}[\overline{R}] + \hat{h}}$. Next, R_1 makes a Ver_2 query with input (b^*_1, y^*_1), and receives n responses of the form $\{(b^*_k, y^*_k)\}_{k \in [n]}$ (one response for every simulated prover), which it can ignore.

Then, R_1 generates $z^*_{i,j} = 0$ for all $2 \le i \le n$ and uses $z^*_{i,j}$ to make a Ver_3 query for the corrupted provers. Finally, R_1 closes the verifier session vid_j by making a Ver_3 query impersonating the honest prover by sending $z^*_{1,j} := \overline{z}_j - b^*_{1,j}$. By repeating this procedure for all $j \in [n]$, A closes $\ell + 1$ verifier sessions.

Winning Game ℓ-$\mathsf{OMMIM}_{\mathsf{mID}}$. For each closed verifier session vid_j, let R_j, y_j, z_j be the values R, y, z at the verifier side and $\vec{K} = \{pk_{1,j}, \ldots, pk_{n,j}\}$ the public keys for which the session verifies. We show that the verification equation of vid_j necessarily holds if the forgery wins the game OMUF. In particular, the mID verification equation implies $g^{z_j} \cdot h^{y_j} = R_j \cdot \prod_{i \in [n]} pk_{i,j}^{\overline{c}_{i,j} + y_j{}^3}$. This means

$$
\begin{aligned}
& g^{\sum_{i=1}^n z^*_{i,j}} \cdot h^{\sum_{i=1}^n y^*_{i,j}} = \prod_{i=1}^n A^*_{i,j} \cdot h^{y^*_{i,j}} \cdot \prod_{i=1}^n pk_{i,j}^{\overline{c}_{i,j} + (\sum_{k=1}^n y^*_{k,j})^3} \\
\Leftrightarrow\quad & g^{z^*_{1,j}} \cdot g^{\sum_{i=2}^n z^*_{i,j}} \cdot h^{y^*_{1,j}} \cdot h^{\sum_{i=2}^n y^*_{i,j}} = A^*_{1,j} \cdot h^{y^*_{1,j}} \\
& \cdot \prod_{i=2}^n A^*_{i,j} \cdot h^{y^*_{i,j}} \cdot \prod_{i=1}^n pk_{i,j}^{\overline{c}_{i,j} + (y^*_{1,j} + \sum_{k=2}^n y^*_{k,j})^3} \\
\Leftrightarrow\quad & g^{z^*_{1,j}} \cdot h^{y^*_{1,j}} = A^*_{1,j} \cdot h^{y^*_{1,j}} \cdot \prod_{i=1}^n pk_{i,j}^{\overline{c}_{i,j} + y^{*3}_{1,j}} \\
\Leftrightarrow\quad & g^{\overline{z}_j} \cdot h^{y^*_{1,j}} \cdot g^{-b^*_{1,j}} = A^*_{1,j} \cdot h^{y^*_{1,j}} \cdot \prod_{i=1}^n pk_{i,j}^{\overline{c}_{i,j} + y^{*3}_{1,j}} \\
\Leftrightarrow\quad & g^{\overline{z}_j} \cdot h^{y^*_{1,j}} = A^*_{1,j} \cdot h^{y^*_{1,j}} \cdot g^{b^*_{1,j}} \cdot \prod_{i=1}^n pk_{i,j}^{\overline{c}_{i,j} + y^{*3}_{1,j}} \\
\overset{\textbf{Game}_4}{\Longleftrightarrow}\quad & g^{\overline{z}_j} \cdot h^{\overline{y}_j} = \overline{R}_j \cdot \prod_{i=1}^n pk_{i,j}^{\overline{c}_{i,j} + \overline{y}^3_j} \quad \mid \text{BM\_SB Ver. Eq.}
\end{aligned}
$$

Since the forgeries are valid, it follows that the verification equation of BM_SB holds. □

$\mathsf{R_1}^{\mathsf{Prove_1,Prove_2,Prove_3}}_{\ell\text{-OMMIM,mID}}(pp)$:

$((m_1,\sigma_1),\ldots,(m_{\ell+1},\sigma_{\ell+1})) \leftarrow \mathsf{A}^{\mathsf{Sign_1,Sign_2,Sign_3,H_{com},H_{sig}}}(pp)$
For all vid:
  $y^*_{vid,1} \coloneqq \sum_{j\in[n]} y_{pid,j} \cdot B_{pid,1}{}_{[\overline{R}]} \cdot h_{[\overline{R}]}$
  $b^*_{vid,1} \coloneqq \sum_{j\in[n]} b_{pid,1} \cdot B_{pid,1}{}_{[\overline{R}]}$
  For all $i \in [n]$:
    $\{(b_{vid,k}, y_{vid,k})\}_{k\in[n]} \leftarrow \mathsf{Ver}_2(vid, b^*_{vid,i}, y^*_{vid,i})$
For $i = 1,\ldots,\ell+1$:
  Parse σ_i as $\overline{R}_i, \overline{y}_i, \overline{z}_i$
  Determine vid by finding $\overline{R}_{vid} = \overline{R}_i$ in $\mathsf{T_{sig}}$
  $z^*_{vid,1} \coloneqq \overline{z}_i - b^*_{vid,1}$
  $b \leftarrow \mathsf{Ver}_3(vid, z_{vid,1})$
  For $j = 2,\ldots,n$:
    $z_{vid,j} \coloneqq 0$
    $b \leftarrow \mathsf{Ver}_3(vid, z_{vid,j})$

Oracle $\mathsf{Sign}_1(\perp)$:
  $(\mathsf{sid}, A_{\mathsf{sid},1}, B_{\mathsf{sid},1}) \leftarrow \mathsf{Prove}_1(\perp)$
  $com_{\mathsf{sid},1} \leftarrow \mathbb{Z}_q$
  Return $(\mathsf{sid}, A_{\mathsf{sid},1}, B_{\mathsf{sid},1}, com_{\mathsf{sid},1})$

Oracle $\mathsf{Sign}_2(\mathsf{sid}, c_{\mathsf{sid},1}, \{com_{\mathsf{sid},k}, B_{\mathsf{sid},k}\}_{k\in[n]})$:
  For all $k \in [n]$:
    Extract $k_{\mathsf{sid}}, b_{\mathsf{sid},k}, y_{\mathsf{sid},k}$ from $\mathsf{T_{com}}$
    $B_{\mathsf{sid},k} \coloneqq g^{b_{\mathsf{sid},k}} \cdot h^{y_{\mathsf{sid},k}}$
  $(b_{\mathsf{sid},1}, y_{\mathsf{sid},1}) \leftarrow \mathsf{Prove}_2(\mathsf{sid}, c_{\mathsf{sid},1}, \{B_{\mathsf{sid},k}\}_{k\in[n]})$
  $\mathsf{T_{com}}[(1_{\mathsf{sid}}, b_{\mathsf{sid},1}, y_{\mathsf{sid},1})] \coloneqq com_{\mathsf{sid},1}$
  Return $(b_{\mathsf{sid},1}, y_{\mathsf{sid},1})$

Oracle $\mathsf{Sign}_3(\mathsf{sid}, \{b_{\mathsf{sid},k}, y_{\mathsf{sid},k}\}_{k\in[n]})$:
  If $\exists k \in [n] : com_{\mathsf{sid},k} \neq \mathsf{H_{com}}(k_{\mathsf{sid}}, b_{\mathsf{sid},k}, y_{\mathsf{sid},k})$
  $\vee\ B_{\mathsf{sid},k} \neq g^{b_{\mathsf{sid},k}} \cdot h^{y_{\mathsf{sid},k}}$:
    Abort
  $z_{\mathsf{sid},1} \leftarrow \mathsf{Prove}_3(pid, \{b_{\mathsf{sid},k}, y_{\mathsf{sid},k}\})_{k\in[n]}$
  Return $z_{\mathsf{sid},1}$

Oracle $\mathsf{H_{com}}(Q)$:
  If $\mathsf{T_{com}}[Q] \neq \perp$:
    Return $\mathsf{T_{com}}[Q]$
  $\xi \leftarrow \mathbb{Z}_q$
  $\mathsf{T_{com}}[Q] \coloneqq \xi$
  Return ξ

Oracle $\mathsf{H_{sig}}(Q)$:
  If $\mathsf{T_{sig}}[Q] \neq \perp$: Return $\mathsf{T_{sig}}[Q]$
  Parse Q as $\vec{K}_{vid}, pk_{vid,\kappa}, \overline{R}_{vid}, m_{vid}$
  If $pk_{vid,\kappa} \notin \vec{K}_{vid}$:
    $c \leftarrow \mathbb{Z}_q$
    $\mathsf{T_{sig}}[Q] \coloneqq c$
    Return c
  $B^*_{vid,1} \coloneqq \prod_{pid\in Q_{\mathsf{Sign}}} B_{pid,1}^{B_{pid,1}[\overline{R}_{vid}]} \cdot h_{[\overline{R}_{vid}]}$
  $A^*_{vid,1} \coloneqq \frac{\overline{R}_{vid}}{B^*_{vid,1}}$
  For $i = 2,\ldots,n-1$:
    $y_{vid,i}, a_{vid,i}, b_{vid,i} \leftarrow \mathbb{Z}_q$
  $y^*_{vid,n} \coloneqq -\sum_{2\leq i\leq n-1} y_{vid,i}$
  $a^*_{vid,n} \coloneqq -\sum_{2\leq i\leq n-1} a_{vid,i}$
  $b^*_{vid,n} \coloneqq -\sum_{2\leq i\leq n-1} b_{vid,i}$
  For $i = 2,\ldots,n$:
    $A^*_{vid,i} \coloneqq g^{a^*_{vid,i}}, B^*_{vid,i} \coloneqq g^{b^*_{vid,i}} \cdot h^{y^*_{vid,i}}$
    $(\overline{c}_{vid,i}, \{B_{vid,k}\}_{k\in[n]}) \leftarrow \mathsf{Ver}_1(i, A^*_{vid,i}, B^*_{vid,i})$
  For all $i \in [n]$: $\mathsf{T_{sig}}[(\vec{K}_{vid}, pk_{vid,i}, \overline{R}_{vid}, m_{vid})] \coloneqq \overline{c}_{vid,i}$
  Return $\overline{c}_{vid,\kappa}$

Figure 6: Reduction R_1, which reduces Game_4 to Game ℓ-OMMIM against mID.