# Chain Reductions for Multi-Signatures

Mihir Bellare[1] Wei Dai[2]

May 27, 2021

**Abstract**

Existing proofs for Discrete Log (DL) based multi-signature schemes give essentially no guarantee if the schemes are implemented, as they are in practice, in 256-bit groups. This is because the current reductions, which are in the standard model and from DL, are loose. We show that relaxing either the model or the assumption suffices to obtain tight reductions. Namely we give (1) tight proofs from DL in the Algebraic Group Model, and (2) tight, standard-model proofs from well-founded assumptions other than DL. We first do this for the classical 3-round schemes, namely BN and MuSig. Then we give a new 2-round multi-signature scheme, HBMS, as efficient as prior ones, for which we do the same. These multiple paths to security for a single scheme are made possible by a framework of chain reductions, in which a reduction is broken into a chain of sub-reductions involving intermediate problems. Overall our results improve the security guarantees for DL-based multi-signature schemes in the groups in which they are implemented in practice.

[1] Department of Computer Science & Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@eng.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grant CNS-1717640 and a gift from Microsoft.

[2] Department of Computer Science & Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `weidai@eng.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~weidai/`. Supported in part by a Powell Fellowship and grants of first author.

## Contents

## 1 Introduction

Usage in cryptocurrencies has lead to interest in practical, Discrete-Log-based multi-signature schemes. Proposals exist, are efficient, and are supported by proofs, BUT, the bound on adversary advantage in the proofs is so loose that the proofs fail to support use of the schemes in the 256-bit groups in which they are implemented in practice. This leaves the security of in-practice schemes unclear.

We ask, is it possible to bridge this gap to give some valuable support, in the form of tight reductions, for in-practice schemes? As long as we stay in the current paradigm, namely standard-model proofs from DL, the answer is likely NO. To make progress, we need to be willing to change either the model or the assumption. We show that in fact changing either suffices. Our approach is to give, for any scheme, many different paths to security. In particular we give (1) tight reductions from DL in the Algebraic Group Model (AGM) [9], and (2) tight, standard-model reductions from well-founded assumptions other than DL. We obtain these results via a framework in which a reduction is "factored" into a chain of sub-reductions involving intermediate problems.

We implement this approach first with classical 3-round schemes, giving chain reductions yielding (1) and (2) above for the BN [3] and MuSig [13] schemes. Then, in the space of 2-round schemes, we give a new, efficient scheme, called HBMS, for which we do the same. We now look at all this in more detail.

B<span style="font-variant:small-caps"></span>ACKGROUND. A multi-signature $\sigma$ on a message $m$ can be thought of as affirming that "We, the members of this group, all, jointly, endorse $m$." The group is indicated by the vector $\mathbf{vk} = (\mathbf{vk}[1], \ldots, \mathbf{vk}[n])$ of individual public verification keys of its members, and can be dynamic, changing from one signature to another. Signing is done via an interactive protocol between group members; each member $i$ begins with its own public verification key $\mathbf{vk}[i]$, its matching private signing key $\boldsymbol{sk}[i]$, and the message $m$, and, at the end of the interaction, they output the multi-signature $\sigma$. The latter should be compact (of size independent of the size of the group), precluding the trivial solution in which $\sigma$ is a list of the individual signatures of the group members on $m$.

Following its suggestion in the 1980s [?], the primitive has seen much evolution [?, ?, ?, ?, 3]. Early schemes assumed all signers in the signing protocol picked their verification keys honestly. "Rogue-key attacks," in which a malicious signer picked its verification key as function of that of an honest signer, lead to an upgraded target, schemes that retain security even in the presence of adversarially-chosen verification keys. Towards this challenging end we first saw schemes either using interactive key-generation [?] or making the "knowledge of secret key" assumption [?, ?]. Finally, BN [3] gave an efficient, Schnorr-based scheme in the "plain public-key" model, where security was provided even in the face of maliciously-chosen verification keys, yet no more was assumed about these keys than their having certificates as per a standard PKI.

The BN model and definition have become the preferred target; it is the one used in the schemes we discuss next, and in our scheme as well. We denote the security goal as MS-UF. In Section 4 we define it via a game, and define the ms-uf advantage of an adversary as its probability of winning this game.

A NEW WAVE. Applications in blockchains and cryptocurrencies —see [6] for details— have fueled a resurgence of interest in multi-signatures. The desire here is MS-UF-secure, DL-based schemes that work over standard elliptic curves such as Secp256k1 or Curve25519. (Pairing-based schemes [6] are thus precluded.) The natural candidate is BN. But the new application arena has lead to a desire for the following further features, not possessed by BN: (1) Key aggregation. There should be a way to aggregate a set of verification keys into a single, short aggregate key, relative to which signatures are verified. (2) Two rounds. A signing protocol using only 2 rounds of interaction, as

| Scheme MS | Previous | | Ours | |
|---|---|---|---|---|
| | $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$ | $p \approx 2^{256}$ | $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$ | $p \approx 2^{256}$ |
| $\mathsf{BN}$ [3] | $\sqrt{(q \cdot t^2)/p}$ | $2^{-8}$ | $t^2/p$ | $2^{-96}$ |
| $\mathsf{MuSig}$ [6, 13] | $\sqrt[4]{(q^3 \cdot t^2)/p}$ | 1 | $t^2/p$ | $2^{-96}$ |

Figure 1: **Bounds on ms-uf advantage for the 3-round schemes BN and MuSig.** First we show prior bounds, then ours. In each case we first show the upper bound $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$ as a formula, where $t, q, q_s$ are, respectively the adversary running time, the number of its RO queries and the number of executions of the signing protocol, while prime $p$ is the size of the underlying group $\mathbb{G}$. We then show the evaluation with $t = q = 2^{80}$, $q_s = 2^{30}$ and $p \approx 2^{256}$, to capture security over 256-bit curves Secp256k1 or Curve25519. Our bounds are tight and optimal, matching those for the DL problem itself.

opposed to the 3 used by BN.

MuSig [13, 6] broke ground by adapting BN to add key aggregation. Now the effort moved to reducing the number of rounds. This proved challenging. Early proposals of two-round schemes — [2, 12, 18] as well as an early, two-round version of MuSig [13]— were broken by DEFKLNS [8]. To fill the gap, DEFKLNS gave a new two-round scheme, mBCJ. Other proposals followed: MuSig2 [14], MuSig-DN [?] and DWMS [1]. All these support key aggregation.

All the schemes discussed here come with proofs of MS-UF security based on the hardness of the DL (Discrete Log) problem in the underlying group $\mathbb{G}$, up to variations in the model (standard or AGM [9]) or the type of DL problem (plain or OMDL [?]).

Current bounds. On being informed that a scheme has a proof of security based on the hardness of the DL problem in an underlying elliptic-curve group $\mathbb{G}$, the expectation of a practitioner is that the probability that a time $t$ attacker can violate MS-UF security is no more than the probability of successfully computing a discrete logarithm in $\mathbb{G}$, which, as per [17], is $t^2/p$, where $p$, a prime, is the size of $\mathbb{G}$. Concretely, with the 256-bit curves Secp256k1 or Curve25519 —$p \approx 2^{256}$— they would expect that a time $t \approx 2^{80}$ attacker has ms-uf advantage at most $2^{160-256} = 2^{-96}$.

But this expectation is only correct if the reduction in the proof is tight. Current proofs for DL-based multi-signature schemes are loose. With the 256-bit curves Secp256k1 or Curve25519, and for a $2^{80}$-time attacker, the proof of [3] for BN can preclude only a $2^{-8}$ ms-uf advantage, while the proof of [13, 6] for MuSig cannot even preclude a ms-uf advantage of 1, meaning there may be, per the proof, no security at all (cf. Figure 1). For 2-round schemes, the advantage precluded by current proofs is $2^{-16}$ in one case, and again just 1 for the others (cf. Figure 2). Overall, the proofs fail, by big margins, to support the parameter choices and expectations of practice.

Before continuing, let us expand on the above estimates. A proof of MS-UF security for a multi-signature scheme MS gives a formula $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$ that upper bounds the ms-uf advantage of an adversary as a function of its running time $t$, the number $q$ of its queries to the random oracle, and the number $q_s$ of executions of the signing protocol in the chosen-message attack in the ms-uf game. They are shown in Figures 1 and 2. We assume that $t \geq q \geq q_s$. To get these formulas, we first assume that the best attack against the DL problem is generic, so that a time $t$ attacker has success probability at most $t^2/p$ [17]. Next, we use the concrete-security results, in theorems in the papers, that give reductions from the DL problem to the MS-UF security of their scheme. The square-roots in the formulas arise from uses of forking lemmas [?, 3, 2]; the fourth-roots from nested use. The bounds in our Figures are approximate, dropping negligible additive terms. See

| **Scheme** | **Security** | | **Efficiency** | |
|---|---|---|---|---|
| | $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$ | $p \approx 2^{256}$ | Sign | Vf |
| mBCJ | $(q_s^3 \cdot q^2 \cdot t^2)/p$ | 1 | $T_2^{\text{me}} + T_3^{\text{me}}$ | $3T_2^{\text{me}}$ |
| MuSig-DN | $\sqrt[4]{(q^3 \cdot t^2)/p}$ | 1 | NIZK | $T_2^{\text{me}}$ |
| MuSig2, $\nu \geq 4$ | $\sqrt[4]{(q^3 \cdot t^2)/p}$ | 1 | $T_\nu^{\text{me}}$ | $T_2^{\text{me}}$ |
| MuSig2, $\nu = 2$ | $(t^2 + q^3)/p$ | $2^{-16}$ | $T_2^{\text{me}}$ | $T_2^{\text{me}}$ |
| DWMS | $(q_s^2 q^2 \cdot t^2)/p$ | 1 | $T_2^{\text{me}} + T_{2N}^{\text{me}}$ | $T_2^{\text{me}}$ |
| HBMS | $t^2/p$ | $2^{-96}$ | $T_2^{\text{me}}$ | $T_3^{\text{me}}$ |

Figure 2: **Bounds on ms-uf advantage for 2-round schemes.** First we show bounds for prior schemes, then the bounds for our new scheme HBMS. As before, we first show the upper bound formula $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$, where $t, q, q_s$ are, respectively the adversary running time, the number of its RO queries and the number of executions of the signing protocol, while prime $p$ is the size of the underlying group $\mathbb{G}$. We then show the evaluation with $t = q = 2^{80}$, $q_s = 2^{30}$ and $p \approx 2^{256}$, to capture security over 256-bit curves Secp256k1 or Curve25519. For MuSig2, results differ depending on a parameter $\nu$ of the scheme. We also show estimates of signing time (per signer) and verification time. Here $T_n^{\text{me}}$ is the time to compute one $n$-multi-exponentiation in $\mathbb{G}$. The "NIZK" for MuSig-DN indicates that signing requires computation and verification of a NIZKs, which is (much) more expensive then other operations shown.

---

Appendix A for details.

<u>Towards better bounds.</u> Our thesis is that proofs should provide, not merely a qualitative guarantee, but one whose bounds quantitatively support parameter choices made in practice and the indications of cryptanalysis. Accordingly we want multi-signature schemes for which we can prove tight bounds on ms-uf advantage. How are we to reach this end? Impossibility results for Schnorr signatures [**?**, 11], on which the multi-signature schemes under consideration are based, indicate that a search for tight reductions that are both (1) in the standard model, and (2) from DL, is unlikely to succeed. We need to be flexible, and relax either (1) or (2). In fact we show that relaxing either suffices: We give (1) tight reductions from DL in the Algebraic Group Model (AGM) [9], and (2) tight, standard-model reductions from assumptions other than DL. Together, these provide valuable theoretical support for the use of practical multi-signature schemes in 256-bit groups.

<u>AGM.</u> The AGM considers a limited, but still large and inclusive, class of adversaries, called algebraic. When such an adversary queries a group element to an oracle, it provides also its representation in terms of prior group elements that the adversary has seen. Intuitively, the assumption is that the adversary "knows" how group elements it creates are represented. For elliptic curve groups, this appears to be a realistic assumption, so that the AGM captures all natural and known attack strategies.

When considering the merits of the AGM, an important one to keep in mind is that a proof in the AGM immediately implies a proof in the well-accepted Generic Group Model (GGM) of [17]. (So the AGM is only "better" than the GGM.) In more detail, a tight AGM reduction from DL to some problem X immediately yields a GGM bound on adversary advantage, for X, that matches the GGM bound for DL [9]. Thus, overall, tight AGM reductions provide a valuable guarantee. This is recognized by Fuchsbauer, Plouviez and Seurin [10] who use the AGM to give a tight reduction from DL to the UF security of the Schnorr signature scheme. Their result gives hope, realized here,

$$\begin{tikzcd} \mathrm{DL} \arrow[d, "1"'] & \\ \mathrm{IDL} \arrow[r, "2"] \arrow[d, "3"'] & \mathsf{BN} \\ \mathrm{XIDL} \arrow[r, "4"] \arrow[dr, "5"] & \mathsf{MuSig} \\ & \mathsf{HBMS} \end{tikzcd}$$

| | Reduction | SM | AGM |
|---|---|---|---|
| 1 | DL $\to$ IDL | Th. 3.2 | **Th. 3.1** |
| 2 | IDL $\to$ $\mathsf{BN}$ | **Th. 5.1** | – |
| 3 | IDL $\to$ XIDL | Th. 3.4 | **Th. 3.3** |
| 4 | XIDL $\to$ $\mathsf{MuSig}$ | **Th. 6.1** | – |
| 5 | XIDL $\to$ $\mathsf{HBMS}$ | Th. 7.2 | **Th. 7.1** |

Figure 3: **Chain reductions for multi-signatures.** SM stands for "Standard Model" and AGM for "Algebraic Group Model." An arrow P $\to$ Q means a reduction from P to Q; i.e. a proof that P implies Q. A boldface **Theorem Number** indicates the reduction is **tight**. A blank appears in the AGM column when a (tight) SM reduction to its left makes the AGM reduction unnecessary. Writing a MS scheme like $\mathsf{BN}, \mathsf{MuSig}, \mathsf{HBMS}$ as a point in a chain refers to MS-UF security of the scheme in question.

---

that such reductions are possible for multi-signatures as well.

CHAIN REDUCTIONS. We achieve the above ends, and more, as follows. For each multi-signature scheme $\mathsf{MS}$ we consider, we give a chain of reductions, starting from DL, that we depict as

$$\mathrm{DL} = \mathrm{P}_0 \to \mathrm{P}_1 \to \cdots \to \mathrm{P}_{m-1} \to \mathrm{P}_m = \mathsf{MS}\ ,$$

where $\mathrm{P}_1, \ldots, \mathrm{P}_{m-1}$ are intermediate computational problems. We refer to $m \geq 1$ as the length of the chain. For each step $\mathrm{P}_{i-1} \to \mathrm{P}_i$ we provide one of the following.

**1.** A tight, standard-model reduction. This is the ideal and done for as many steps as possible.

**2.** When **1.** is not possible, we give, instead, two things:

**2.1** A tight AGM reduction, AND ALSO

**2.2** A non-tight standard-model reduction.

Since a tight standard-model reduction implies a tight AGM one, this yields a tight AGM reduction from DL to $\mathsf{MS}$, the first of our goals stated above. (A bit better, since some sub-reductions are standard-model.) For $i$ such that the chain $\mathrm{P}_i \to \cdots \to \mathsf{MS}$ consists only of tight standard-model reductions, we have a tight, standard model proof of $\mathsf{MS}$ from assumption $\mathrm{P}_i$, realizing our second goal, stated above, of tight standard-model reductions from assumptions other than DL. (Of course how interesting or valuable this is depends on the choice of $\mathrm{P}_i$, but as discussed below, we are able to make well-founded choices.)

Finally, something not yet mentioned, that follows from **1** and **2.2** of the chain reductions, is that we always have a standard model (even if non-tight) reduction DL $\to$ $\mathsf{MS}$. This means that, while adding tight AGM reductions that are valuable in practice, we are not lowering the theoretical or qualitative guarantees, these remaining as one would expect or desire.

Chain reductions can be seen as a way to implement a modular proof framework in the style of [11], in which steps are reused across proofs for different schemes.

NEW BOUNDS FOR CLASSICAL SCHEMES. We start by revisiting the classical 3-round schemes, namely

BN and MuSig. Figure 3 illustrates our chains, that we now discuss.

IDL, formulated in [11] —they call it IDLOG, which we have abbreviated— is a purely group-based problem that is equivalent to the security against parallel impersonation under key-only attack (PIMP-KOA) of the Schnorr ID scheme. A tight GGM bound for IDL was shown by [11], but an AGM reduction DL $\rightarrow$ IDL does not seem to be in the literature; we fill this gap by providing it in Theorem 3.1. A (non-tight) standard model DL $\rightarrow$ IDL reduction is in [11], but we slightly improve it in Theorem 3.2.

Now our chain for BN is DL $\rightarrow$ IDL $\rightarrow$ BN. This chain has length 2. Our main result for BN is Theorem 5.1, which shows IDL $\rightarrow$ BN with a *tight, standard model* reduction. Putting this together with our above-mentioned tight DL $\rightarrow$ IDL AGM-reduction of Theorem 3.1, we get a tight DL $\rightarrow$ BN AGM-reduction. Also our tight, standard-model IDL $\rightarrow$ BN reduction says that BN is as secure as the Schnorr identification scheme, which is valuable in its own right since the latter has withstood cryptanalysis for many years.

We introduce an intermediate, purely group-based problem we call XIDL. We show IDL $\rightarrow$ XIDL with a tight AGM reduction (Theorem 3.3) and a (non-tight) standard-model reduction (Theorem 3.2).

Our chain for MuSig is DL $\rightarrow$ IDL $\rightarrow$ XIDL $\rightarrow$ MuSig. This chain has length 3. Our main result for MuSig is Theorem 6.1, which shows XIDL $\rightarrow$ MuSig with a *tight, standard model* reduction. Putting this together with the rest of the chain, we get a tight DL $\rightarrow$ MuSig AGM-reduction. If we are willing to view XIDL as an assumption extending IDL, we can also view MuSig as based tightly on that.

This means we show that $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p) \leq t^2/p$ for both schemes, matching the DL bound. This is tight and optimal, since the multi-signature schemes can be broken by taking discrete-logs. Figure 1 compares our results with the prior ones.

<u>New 2-round scheme.</u> Turning to 2-round schemes, we give a new scheme, called HBMS. HBMS supports key aggregation, in line with other 2-round schemes. Our chain for our new 2-round HBMS scheme is DL $\rightarrow$ IDL $\rightarrow$ XIDL $\rightarrow$ HBMS. This chain has length 3. We show XIDL $\rightarrow$ HBMS with a tight AGM reduction (Theorem 7.1) and a (non-tight) standard-model reduction (Theorem 7.2). Putting this together with the rest of the chain, we get a tight DL $\rightarrow$ HBMS AGM-reduction, in particular showing $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p) \leq t^2/p$, matching the DL bound. We also get a (non-tight) DL $\rightarrow$ HBMS standard-model-reduction.

Figure 2 compares HBMS with prior 2-round schemes. It shows that our improvement in security is not at the cost of efficiency. (Signing in HBMS is as efficient, or more so, than in prior schemes. For verification, MuSig-DN [**?**] is slightly faster, but signing in the latter is prohibitive due to the use of NIZKs.)

As the above shows, we reuse steps across different chains. Thus XIDL is an intermediate point for both MuSig and HBMS, and IDL for both BN and XIDL. This simplifies proofs and reduces effort. It also shows common elements and relations across schemes.

<u>Equivalences.</u> As discussed above, Theorem 5.1 shows IDL $\rightarrow$ BN with a tight, standard model reduction. We also give, in Theorem 5.2, a converse, namely a tight, standard-model reduction showing BN $\rightarrow$ IDL. This shows that IDL and BN are, security-wise, equivalent. Similarly, as discussed above, Theorem 6.1 shows MuSig $\rightarrow$ XIDL with a tight, standard model reduction, and we also give, in Theorem 6.2, a converse, namely a tight, standard-model reduction showing XIDL $\rightarrow$ MuSig. This shows that XIDL and MuSig are equivalent. Overall, this shows that IDL and XIDL are not arbitrary choices, but characterizations of the schemes whose consideration is necessary.

<u>Related work.</u> The interest for blockchains and cryptocurrencies, and thus our focus, is DL-

based schemes over elliptic curves. There are many other multi-signature schemes, based on other hard problems. Aggregate signatures [**?**, **?**] yield multi-signatures, but these use pairings (bilinear maps). A pairing-based multi-signature scheme is also given in [6]. Lattice-based multi-signature schemes include [**?**, **?**].

As noted above, IDL [11] captures the security against parallel impersonation under key-only attack (PIMP-KOA) of the Schnorr ID scheme and thus, given the ZK property of the scheme, also its security against parallel impersonation under passive attack (PIMP-PA). "Parallel" means multiple impersonation attempts are allowed. IMP-PA, traditional security against impersonation under passive attack, is the case where just one impersonation attempt is allowed. The Reset Lemma [4] gives a standard model DL $\rightarrow$ IMP-PA reduction. This uses rewinding and is non-tight, with a square-root loss. BD [**?**] introduce the Multi-Base Discrete Logarithm (MBDL) problem. They give a tight standard-model MBDL $\rightarrow$ IMP-PA reduction, and also show that, in the GGM, the security of MBDL is the same as that of DL. An interesting open question is whether MBDL can be used as a starting point for tight reductions for multi-signature schemes.

## 2 Preliminaries

Notation. If $n$ is a positive integer, then $\mathbb{Z}_n$ denotes the set $\{0, \ldots, n-1\}$ and $[n]$ or $[1..n]$ denote the set $\{1, \ldots, n\}$. If $\boldsymbol{x}$ is a vector then $|\boldsymbol{x}|$ is its length (the number of its coordinates), $\boldsymbol{x}[i]$ is its $i$-th coordinate and $[\boldsymbol{x}] = \{\, \boldsymbol{x}[i] \, : \, 1 \leq i \leq |\boldsymbol{x}| \,\}$ is the set of all its coordinates. A string is identified with a vector over $\{0,1\}$, so that if $x$ is a string then $x[i]$ is its $i$-th bit and $|x|$ is its length. By $\varepsilon$ we denote the empty vector or string. The size of a set $S$ is denoted $|S|$.

Let $S$ be a finite set. We let $x \leftarrow\$ S$ denote sampling an element uniformly at random from $S$ and assigning it to $x$. We let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; \rho)$ denote executing algorithm $A$ on inputs $x_1, \ldots$ and coins $\rho$ with access to oracles $\mathrm{O}_1, \ldots$, and letting $y$ be the result. We let $\rho \leftarrow\$ \mathrm{rand}(A)$ denote sampling random coins for algorithm $A$ and assigning it to variable $\rho$. We let $y \leftarrow\$ A^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ be the result of $\rho \leftarrow\$ \mathrm{rand}(A)$ followed by $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots; \rho)$. We let $[A^{\mathrm{O}_1,\cdots}(x_1, \ldots)]$ denote the set of all possible outputs of $A$ when invoked with inputs $x_1, \ldots$ and oracles $\mathrm{O}_1, \ldots$. Algorithms are randomized unless otherwise indicated. Running time is worst case.

Games. We use the code-based game playing framework of [5]. (See Fig. 4 for an example.) Games have procedures, also called oracles. Amongst these are Init and a Fin. In executing an adversary $\mathcal{A}$ with a game Gm, procedure Init is executed first, and what it returns is the input to $\mathcal{A}$. The latter may now call all game procedures except Init, Fin. When the adversary terminates, its output is viewed as the input to Fin, and what the latter returns is the game output. By $\mathrm{Gm}(\mathcal{A}) \Rightarrow y$ we denote the event that the execution of game Gm with adversary $\mathcal{A}$ results in output $y$. We write $\Pr[\mathrm{Gm}(\mathcal{A})]$ as shorthand for $\Pr[\mathrm{Gm}(\mathcal{A}) \Rightarrow \mathsf{true}]$, the probability that the game returns $\mathsf{true}$. In writing game or adversary pseudocode, it is assumed that boolean variables are initialized to $\mathsf{false}$, integer variables are initialized to 0 and set-valued variables are initialized to the empty set $\emptyset$.

A procedure (oracle) with a certain name O may appear in several games. (For example, Ch appears in two games in Figure 4.) To disambiguate, we may write Gm.O for the one in game Gm.

When adversary $\mathcal{A}$ is executed with game Gm, we consider the running time of $\mathcal{A}$ as the running time of the execution of $\mathrm{Gm}(\mathcal{A})$, which includes the time taken by game procedures. By $\mathrm{Q}^{\mathrm{O}}_{\mathcal{A}}$ we denote the number of queries made by $\mathcal{A}$ to oracle O in the execution. These counts are both worst case.

Groups. Throughout, $\mathbb{G}$ is a group whose order, assumed prime, we denote by $p$. We will use multiplicative notation for the group operation, and we let $1_{\mathbb{G}}$ denote the identity element of $\mathbb{G}$. We let $\mathbb{G}^* = \mathbb{G} \setminus \{1_{\mathbb{G}}\}$ denote the set of non-identity elements, which is the set of generators of $\mathbb{G}$

since the latter has prime order. If $g \in \mathbb{G}^*$ is a generator and $X \in \mathbb{G}$, then $\mathsf{DL}_{\mathbb{G},g}(X) \in \mathbb{Z}_p$ denotes the discrete logarithm of $X$ in base $g$.

Algebraic algorithms. We recall the definition of algebraic algorithms [9]. As above, fix a group $\mathbb{G}$ of prime order $p$, and let $g$ be a generator. In all of our security games involving $\mathbb{G}$ and $g$, we assume that any inputs and outputs of game oracles that are group elements (meaning, in $\mathbb{G}$) are distinguished. In particular, it will be clear from the game pseudocode definition which components of inputs and outputs are such group elements. We say that an adversary, against game Gm, is algebraic, if, whenever it submits a group element $Y \in \mathbb{G}$ as an oracle query, it also provides, alongside, a representation of $Y$ in terms of group elements previously returned by the game oracles (the latter including Init). Specifically, suppose during an execution of adversary $\mathcal{A}$ with game Gm, the adversary submits a group element $Y \in G$ to game oracle O. Then, alongside, it must provide a vector $(v_0, v_1, \ldots, v_m) \in \mathbb{Z}_p^m$, called a representation of $Y$, such that $Y = g^{v_0} \cdot h_1^{v_1} \cdots h_m^{v_m}$, where $h_1, \ldots, h_m$ are the group elements that have been returned to the adversary by game oracles of Gm, so far. When considering an execution of game Gm with an adversary $\mathcal{A}$ that is not algebraic, we omit the writing of representations in the oracle calls.

Hedging. Not all attacks are algebraic. The thesis of [9] is that natural ones are, and thus proving security relative to algebraic adversaries gives meaningful guarantees in practice. We adopt this here but add hedging. Recall this means that, for the same scheme, we seek both (1) A tight AGM reduction from DL, and (2) a standard-model (even if non-tight) reduction from DL. The former is used to guide and support parameter choices. The latter is viewed as at least qualitatively ruling out non-algebraic attacks.

Reductions. All our standard-model reductions are black-box and preserve algebraic-ness of adversaries, meaning, if the starting adversary is algebraic, so is the constructed one. This means that we can chain standard-model reductions with AGM-reductions to get overall AGM reductions.

## 3 Hardness of problems in groups

Our chain reductions exploit three computational problems related to groups: standard discrete log (DL); IDL [11]; and a new problem XIDL that we introduce. Here we give the definitions. We then show the length-2 chain DL $\rightarrow$ IDL $\rightarrow$ XIDL. We give reductions that are tight in the AGM and also give (non-tight) standard-model reductions, a total of four results. Referring to Figure 3, we are establishing the four theorems, shown in the table, that correspond to arrows 1 and 3. For the rest of the section, we fix a group $\mathbb{G}$ of prime order $p$, and a generator $g \in \mathbb{G}$.

DL. We recall the standard discrete logarithm (DL) problem via game $\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}$ in Figure 4. Init provides the adversary, as input, a random challenge group element $X$, and to win it must output $x' = \mathsf{DL}_{\mathbb{G},g}(X)$ to Fin. We let $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A})]$ be the discrete-log advantage of adversary $\mathcal{A}$.

IDL. The identification discrete logarithm (IDL) problem, introduced by KMP [11], characterizes the hardness of parallel impersonation under key-only attack (PIMP-KOA) security [11] of the Schnorr identification scheme [16]. Formally, consider the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$ given in Fig. 4, where parameter $q$ is a positive integer. The IDL-adversary receives a random target point $X$ from Init. It is additionally given access to a challenge oracle Ch that can be called at most $q$ times. The oracle takes as query a group element $R$ (representing the commitment sent by the prover in Schnorr identification), stores it as $R_i$, and responds with a random challenge $c_i \leftarrow\$\, \mathbb{Z}_p$ (representing the one sent by the verifier). The adversary wins if it can produce the discrete log $z$ (representing the final prover response) of the group element $R_i \cdot X^{c_i}$, for a choice of $i$, denoted $I$, made by the

```
Game $\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}$

INIT:
1 $x \leftarrow\!\!\$\ Z_{|\mathbb{G}|}$ ; $X \leftarrow g^x$ ; Return $X$

FIN($x'$):
2 Return $(x = x')$
```

```
Game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$

INIT:
1 $x \leftarrow \mathbb{Z}_{|\mathbb{G}|}$ ; $X \leftarrow g^x$
2 Return $X$

CH($R$): // At most $q$ queries.
3 $i \leftarrow i+1$ ; $R_i \leftarrow R$
4 $c_i \leftarrow\!\!\$\ \mathbb{Z}_{|\mathbb{G}|}$ ; Return $c_i$

FIN($I, z$):
5 Return $(g^z = R_I \cdot X^{c_I})$
```

```
Game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}$

INIT:
1 $x \leftarrow \mathbb{Z}_{|\mathbb{G}|}$ ; $X \leftarrow g^x$
2 Return $X$

NWTAR($S$): // At most $q_1$ queries.
3 $j \leftarrow j+1$ ; $S_j \leftarrow S$
4 $e_j \leftarrow\!\!\$\ \mathbb{Z}_{|\mathbb{G}|}$ ; $T_j \leftarrow S_j \cdot X^{e_j}$
5 Return $e_j$

CH($j_{\mathrm{sel}}, R$): // At most $q_2$ queries.
6 $i \leftarrow i+1$ ; $R_i \leftarrow R$ ; $Y_i \leftarrow T_{j_{\mathrm{sel}}}$
7 $c_i \leftarrow\!\!\$\ \mathbb{Z}_{|\mathbb{G}|}$ ; Return $c_i$

FIN($I, z$):
8 Return $(g^z = R_I \cdot {Y_I}^{c_I})$
```

Figure 4: Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. Let $q, q_1, q_2$ be positive integers. Top: Game defining discrete logarithm (DL) problem. Bottom left: Game defining identification logarithm (IDL) problem. Bottom right: Game defining random-target identification logarithm (XIDL) problem.

---

adversary. We define the IDL-advantage of $\mathcal{A}$ to be $\mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A})]$.

KMP [11] study IDL in the Generic Group Model (GGM) [17] and prove a bound matching that for DL. Here, we strengthen this to give a tight AGM reduction DL $\to$ IDL. This could be seen as implicit in part of the AGM proof of security for the Schnorr signature scheme given in [10], although they make no connection to IDL.

**Theorem 3.1** [DL $\to$ IDL, AGM] *Let $\mathbb{G}$ be a group of prime order $p$ with generator $g$. Let $q$ be a positive integer. Let $\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}$ be an algebraic adversary against $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$. Then, adversary $\mathcal{A}_{\mathrm{dl}}$ can be constructed so that*

$$\mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}) \le \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) + \frac{q}{p} \ .$$

*Furthermore, the running time of $\mathcal{A}_{\mathrm{dl}}$ is about that of $\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}$.*

The idea of the proof is as follows. Since $\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}$ is algebraic, its query $R$ to CH is accompanied by $(r_1, r_2)$ such that $R = g^{r_1}X^{r_2}$. Our adversary $\mathcal{A}_{\mathrm{dl}}$, who is running $\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}$, records these as $R_i, r_{i,1}, r_{i,2}$, and responds with a random $c_i$. Eventually, $\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}$ outputs $I, z$. Assuming it succeeds, we have $g^z = R_I \cdot X^{c_I} = g^{r_{I,1}}X^{r_{I,2}}X^{c_I}$, or $g^{z-r_{I,1}} = X^w$ where $w = (r_{I,2}+c_I) \bmod p$. Now $\mathsf{DL}_{\mathbb{G},g}(X)$ can be obtained as long as $w$ has an inverse modulo $p$, meaning is non-zero. But $c_I$ was chosen at random *after the adversary supplied* $r_{I,2}$, so the probability that $w$ is 0 is at most $1/p$. The factor of $q$ accounts for the adversary's having a choice of $I$ made after receiving challenges. The full proof is given in Appendix C.

By $q$-IDL, we refer to IDL with parameter $q$. 1-IDL corresponds to IMP-KOA security of the Schnorr identification scheme, and a reduction DL $\to$ 1-IDL is obtained via the Reset Lemma of [4]. KMP show that 1-IDL $\to$ $q$-IDL. Overall this gives a standard model (very non-tight) DL $\to$ $q$-IDL reduction. However, a somewhat tighter (but still non-tight) result can be obtained when the forking lemma of [3] (which we recall as Lemma B.1) is applied directly instead. Concretely, we give the following theorem, improving the prior reduction by a $\sqrt{q}$ factor. The proof is in Appendix D.

**Theorem 3.2** [DL $\to$ IDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. Let $q$ be a positive integer. Let $\mathcal{A}_{\mathrm{idl}}$ be an adversary against the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$. The proof constructs an adversary $\mathcal{A}_{\mathrm{dl}}$ (explicitly given in Fig. 12) such that*

$$\mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}}) \leq \sqrt{q \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{idl}})} + \frac{q}{p} \ . \qquad (1)$$

*Additionally, the running time of $\mathcal{A}_{\mathrm{dl}}$ is approximately $\mathrm{T}_{\mathcal{A}_{\mathrm{dl}}} \approx 2 \cdot \mathrm{T}_{\mathcal{A}_{\mathrm{idl}}}$.*

<ins>XIDL.</ins> We define a new problem, random target identification discrete logarithm, abbreviated XIDL. It abstracts out the algebraic core of MuSig, and we will show that its security is equivalent to the MS-UF security of MuSig. It will also be an intermediate point in our reduction chain reaching our new HBMS scheme, thereby serving multiple purposes.

With $\mathbb{G}, p, g$ fixed as usual, XIDL is parameterized by positive integers $q_1, q_2$. Formally, consider the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}$ given in Fig. 4. The adversary receives a randomly chosen group element $X$ from Init. The game maintains a list $T_1, \ldots, T_{q_1}$ of "targets." The adversary can create a target by querying the New Target oracle NwTar with a group element $S$ of its choosing, whence $T_j = S \cdot X^{e_j}$ is added to the list of targets, for $e_j$ chosen randomly from $\mathbb{Z}_p$ by the game and returned to the adversary. The adversary can query the challenge oracle Ch$(j_{\mathrm{sel}}, R)$ by supplying an index $j_{\mathrm{sel}}$ and a group element $R$. The oracle records $T_{j_{\mathrm{sel}}}$ as $Y_i$, and $R$ as $R_i$, based on the counter $i$ it maintains. Intuitively, Ch is similar to the challenge oracle Ch in IDL game, besides that our adversary here needs to specify the target $T_{j_{\mathrm{sel}}}$ it is trying to impersonate against. The adversary wins the game if it can produce the discrete log $z$ of $R_I \cdot Y_I^{c_I}$, for an index $I$ of its choice. The oracles NwTar and Ch are allowed to be called at most $q_1$ and $q_2$ times, respectively. We define the XIDL advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A})]$.

We show hardness of XIDL in both the AGM and the standard model, starting with the former. The theorem actually establishes the stronger DL $\to$ XIDL, tightly in the AGM.

**Theorem 3.3** [IDL $\to$ XIDL, AGM] *Let $\mathbb{G}$ be a group of order $p$ with generator $g$. Let $q_1, q_2$ be positive integers. Let $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$ be an algebraic adversary against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}$. Then, adversary $\mathcal{A}_{\mathrm{dl}}$ can be constructed so that*

$$\mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}) \leq \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) + \frac{q_1 + q_2}{p} \ .$$

*Furthermore, the running time of $\mathcal{A}_{\mathrm{idl}}$ is about that of $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$.*

The full proof is given in Appendix F. Here we sketch the intuition. Since $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$ is algebraic, the $j$-th query to NwTar is of the form $S_j, s_{j,1}, s_{j,2}$ such that $S_j = g^{s_{j,1}} X^{s_{j,2}}$, and the $i$-th query to Ch is of the form $j_{\mathrm{sel}}, R_i, r_{i,1}, r_{i,2}$ such that $R_i = g^{r_{i,1}} X^{r_{i,2}}$. Let $e_j, c_i$ denote, respectively, the responses to the $j$-th query to NwTar and the $i$-th query to Ch. Eventually, $\mathcal{A}_{\mathrm{xidl}}$ outputs $I, z$. Assuming it succeeds, the equation $g^z = R_I \cdot T_J^{c_I} = R_I \cdot (S_J \cdot X^{e_J})^{c_I}$ must hold, where $J$ was the selected index $j_{\mathrm{sel}}$ in the $I$-th query to Ch. This means that $g^z = g^{r_{I,1}} X^{r_{I,2}} (g^{s_{J,1}} X^{s_{J,2}} X^{e_J})^{c_I}$, whence $g^{z-r_{I,1}-s_{J,1}\cdot c_I} = X^w$, where $w = r_{I,2} + (s_{J,2} + e_J)c_I$. As long as $w$ is non-zero modulo $p$, one can solve for the value of $\mathsf{DL}_{\mathbb{G},g}(X)$. But $e_J$ and $c_I$ were independently chosen after the adversary supplied $s_{J,2}$ and $r_{I,2}$, respectively. The probability that there exists $j$ such that $(s_{j,2} + e_j) = 0 \bmod p$ is at most $q_1/p$ over $q_1$ queries to NwTar. Assuming there is no such $j$, the probability that $w = 0$ is at most $q_2/p$, due to the $q_2$ queries to Ch that $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$ can make.

In the standard model, techniques in the security proof of MuSig [6, 13] could be used to show DL → XIDL, but this would use two applications of the Forking Lemma, leading to a fourth-root in the bound. Instead, we show IDL → XIDL, using a single application of the forking lemma, so that there is only a square-root in the bound.

**Theorem 3.4** [IDL → XIDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p$ with generator $g$. Let $q_1, q_2$ be positive integers. Let $\mathcal{A}_{\mathrm{xidl}}$ be an adversary against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}$. Then, an adversary $\mathcal{A}_{\mathrm{idl}}$ can be constructed so that*

$$\mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}}) \le \sqrt{q_2 \cdot \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q_1}(\mathcal{A}_{\mathrm{idl}})} + \frac{q_2}{p} .$$

*Furthermore, the running time of $\mathcal{A}_{\mathrm{idl}}$ is about twice of that of $\mathcal{A}_{\mathrm{xidl}}$.*

The full proof is given in Appendix E. We now sketch the intuition. Adversary $\mathcal{A}_{\mathrm{idl}}$ receives $X$ from game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$ and runs adversary $\mathcal{A}_{\mathrm{xidl}}$, forwarding it $X$ as the target point. It answers queries to $\mathcal{A}_{\mathrm{xidl}}$'s NwTar oracle using its own $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$.Ch oracle. Specifically, the $j$-th query $S$ to NwTar is responded to with $e_j \leftarrow\$ \mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}.\mathrm{Ch}(S)$, and $\mathcal{A}_{\mathrm{idl}}$ additionally records the group element $T_j \leftarrow S \cdot X^{e_j}$. It simulates adversary $\mathcal{A}_{\mathrm{xidl}}$'s Ch oracle locally, meaning the $i$-th query $\mathrm{Ch}(j_{\mathrm{sel}}, R)$ is responded to with a fresh challenge $c_i \leftarrow\$ \mathbb{Z}_p$. Eventually, adversary $\mathcal{A}_{\mathrm{xidl}}$ gives a response $I, z$. Our $\mathcal{A}_{\mathrm{idl}}$ adversary wins game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$ if it can produce the discrete log of $T_j$ for any $j$ of its choice. To do so, $\mathcal{A}_{\mathrm{idl}}$ uses rewinding, the analysis of which uses the Forking Lemma [3] that we recall as Lemma B.1. Rewinding is used to produce another response, $(I', z')$, from a forked execution of $\mathcal{A}_{\mathrm{xidl}}$. The Forking Lemma applies to an execution of an algorithm making queries to one oracle, but adversary $\mathcal{A}_{\mathrm{xidl}}$ has two oracles NwTar and Ch. We only "fork" $\mathcal{A}_{\mathrm{xidl}}$ on its queries to Ch. Specifically, we program oracle NwTar to behave identically compared to the first run (meaning we use previously recorded values of $e_1, \ldots$ as long as they are defined). In the second run, oracle Ch is replied with $c_1, \ldots, c_{I-1}, c'_I, \ldots$, where $c'_I, \ldots$ are randomly chosen from $\mathbb{Z}_p$. Let us assume that $\mathcal{A}_{\mathrm{idl}}$ has derived two valid responses from $\mathcal{A}_{\mathrm{xidl}}$ using the Forking Lemma. Then it is guaranteed that $I = I'$ and $c_I \neq c'_I$. Moreover, we know the two executions of $\mathcal{A}_{\mathrm{xidl}}$ only differ *after* the response of the $I$-th query to Ch, so the $I$-th query to Ch in both runs is some $J, R_I$. This allows our adversary to solve the equations $g^z = R_I \cdot T_J^{c_I}$ and $g^{z'} = R_I \cdot T_J^{c'_I}$ (which are guaranteed to be true if both runs succeed) to compute $\mathsf{DL}_{\mathbb{G},g}(T_J)$ and thus win the IDL game.

## 4 Definitions for multi-signatures

DEFKLNS [8] found subtle gaps in some prior proofs of security for some multi-signature schemes [2, 12, 18]. Some of the latter schemes had been around for quite a long time before this happened. This suggests that, in the domain of multi-signatures, we need more care and careful analyses. We suggest that this needs to begin with *definitions.* The ones in prior work, stemming mostly from [3], suffer from some lack of detail and precision. In particular, the very *syntax* of a multi-signature scheme is not specified in detail. This results in scheme descriptions that lack somewhat in precision, and to proofs that stay at a high level in part due to lack of technical language in which to give details. This in turn can lead to bugs.

To address these issues, we revisit the definitions. We start with a detailed syntax that formalizes the signing protocol as a stateful algorithm, run separately by each player. (The state will be maintained by the overlying game.) Details addressed include that a player knows its position in the signer list, that player identities are separate from public keys, and integration of the ROM through a parameter describing the type of ideal hash functions needed. Then we give a security definition written via a code-based game.

SYNTAX. A multi-signature scheme $\mathsf{MS}$ specifies algorithms $\mathsf{MS.Kg}$, $\mathsf{MS.Vf}$, $\mathsf{MS.Sign}$, as well as a set $\mathsf{MS.HF}$ of functions, and an integer $\mathsf{MS.nr}$, whose intent and operation is as follows. *Key generation.* Via $(pk, sk) \leftarrow\$ \mathsf{MS.Kg}$, the key generation algorithm generates public signature-verification key $pk$ and secret signing key $sk$ for a user. (Each user is expected to run this independently to get its keys.) *Hash functions.* $\mathsf{MS.HF}$ is a set of functions, from which, via $\mathsf{h} \leftarrow\$ \mathsf{MS.HF}$, one is drawn and provided to scheme algorithms (except key generation) and the adversary as the random oracle. Specifying this as part of the scheme allows the domain and range of the random oracle to be scheme-dependent. *Verification.* Via $d \leftarrow \mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma)$, the verification algorithm deterministically outputs a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ indicating whether or not $\sigma$ is a valid signature on message $m$ under a vector $\boldsymbol{pk}$ of verification keys. *Signing.* The signing protocol is specified by signing algorithm $\mathsf{MS.Sign}$. In each round, each party, applies this algorithm to its current state $\mathsf{st}$ and the vector $\mathbf{in}$ of received messages from the other parties, to compute an outgoing message $\sigma$ (viewed as broadcast to the other parties) and an updated state $\mathsf{st}'$, written $(\sigma, \mathsf{st}') \leftarrow \mathsf{MS.Sign}^{\mathrm{H}}(\mathbf{in}, \mathsf{st})$. In the last round, $\sigma$ is the signature that this party outputs. (See Figure 5.) *Rounds.* The interaction consists of a fixed number $\mathsf{MS.nr}$ of rounds. (We number the rounds $0, \ldots, \mathsf{MS.nr}$. The final broadcast of the signature is not counted as in practice it is a local output.) *Key Aggregation.* We say that a multi-signature scheme $\mathsf{MS}$ supports key aggregation if $\mathsf{MS}$ has additional two algorithms $\mathsf{MS.Ag}$ and $\mathsf{MS.VfAg}$ such that: (1) Via $apk \leftarrow\$ \mathsf{MS.Ag}^{\mathrm{H}}(pk_1, \ldots, pk_n)$, $\mathsf{MS.Ag}$ generates an aggregate public key, (2) Via $d \leftarrow \mathsf{MS.VfAg}^{\mathrm{H}}(apk, m, \sigma)$, the aggregate verification algorithm deterministically outputs a decision $d \in \{\mathsf{true}, \mathsf{false}\}$, and (3) the verification algorithm $\mathsf{MS.Vf}$ is defined exactly as $\mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma) := \mathsf{MS.VfAg}^{\mathrm{H}}(\mathsf{MS.Ag}^{\mathrm{H}}(\boldsymbol{pk}), m, \sigma)$.

Some conventions will aid further definitions and scheme descriptions. A party's state $\mathsf{st}$ has several parts: $\mathsf{st.n}$ is the number of parties in the current execution of the protocol; $\mathsf{st.me} \in [1..\mathsf{st.n}]$ is the party's own identity; $\mathsf{st.rnd} \in [0..\mathsf{MS.nr}]$ is the current round number; $\mathsf{st.sk}$ is the party's own signing key; $\mathsf{st.pk}$ is the $\mathsf{st.n}$-vector of all verification keys; $\mathsf{st.msg}$ is the message being signed; $\mathsf{st.rej} \in \{\mathsf{true}, \mathsf{false}\}$ is the decision to reject (not produce a signature) or accept. It is assumed and required that each invocation of $\mathsf{MS.Sign}$ leaves all of these unchanged except for $\mathsf{st.rnd}$, which it increments by 1, and $\mathsf{st.rej}$, which is assumed initialized to $\mathsf{false}$ and may at some point be set to $\mathsf{true}$. The state can, beyond these, have other components that vary from protocol to protocol. (For example, Figure 6 describing the BN scheme has $\mathsf{st}.\boldsymbol{R}[j], \mathsf{st}.\boldsymbol{t}[j], \mathsf{st}.\boldsymbol{z}[j], \mathsf{st}.R, \ldots$.) We write $\mathsf{st} \leftarrow \mathsf{StInit}(j, sk, \boldsymbol{pk}, m)$ to initialize $\mathsf{st}$ by setting $\mathsf{st.n} \leftarrow |\boldsymbol{pk}|$ ; $\mathsf{st.me} \leftarrow j$ ; $\mathsf{st.rnd} \leftarrow 0$ ; $\mathsf{st.sk} \leftarrow sk$ ; $\mathsf{st.pk} \leftarrow \boldsymbol{pk}$ ; $\mathsf{st.msg} \leftarrow m$ ; $\mathsf{st.rej} \leftarrow \mathsf{false}$. If an execution $(\sigma, \mathsf{st}') \leftarrow \mathsf{MS.Sign}^{\mathrm{H}}(\mathbf{in}, \mathsf{st})$ returns $\sigma = \perp$ then it is assumed and required that further executions starting from $\mathsf{st}'$ all return $\perp$ as the output message.

CORRECTNESS. Algorithm $\mathsf{Exec}_{\mathsf{MS}}$, shown in the left column of Fig. 5, executes the signing protocol of $\mathsf{MS}$ on input a vector $\boldsymbol{sk}$ of signing keys, a vector $\boldsymbol{pk}$ of matching verification keys with $|\boldsymbol{sk}| = |\boldsymbol{pk}|$, and a message $m$ to be signed, and with access to random oracle $\mathsf{h} \in \mathsf{MS.HF}$. The number of parties $n$ at line 1 is the number of coordinates (length) of $\boldsymbol{pk}$. The state $\mathsf{st}_j$ of party $j$ at line 3 is initialized using the function $\mathsf{StInit}$ defined above. The loop at line 5 executes $\mathsf{MS.nr}$ rounds. Here $\boldsymbol{b}$ denotes the $n$-vector of currently-broadcast messages, meaning $\boldsymbol{b}[i]$ was broadcast by party $i$ in the prior round, and the entire vector is the input to party $j$ for the current round. At line 8, $\boldsymbol{b}$

```
Algorithm $\mathsf{Exec}^{\mathsf{h}}_{\mathsf{MS}}(\boldsymbol{sk}, \boldsymbol{pk}, m)$:
$n \leftarrow |\boldsymbol{pk}|$
For $j = 1, \ldots, n$ do
    $\mathsf{st}_j \leftarrow \mathsf{StInit}(j, \boldsymbol{sk}[j], \boldsymbol{pk}, m)$
$\boldsymbol{b} \leftarrow (\varepsilon, \ldots, \varepsilon)$ // $n$-vector
For $i = 1, \ldots, \mathsf{MS.nr}$ do
    For $j = 1, \ldots, n$ do
        $(\sigma_j, \mathsf{st}_j) \leftarrow\$ \mathsf{MS.Sign}^{\mathsf{h}}(\boldsymbol{b}, \mathsf{st}_j)$
    $\boldsymbol{b} \leftarrow (\sigma_1, \ldots, \sigma_n)$
Return $\sigma_1$
```

```
Game $\mathbf{G}^{\text{ms-cor}}_{\mathsf{MS},n}$

FIN:
$\mathsf{h} \leftarrow\$ \mathsf{MS.HF}$
For $i = 1, \ldots, n$ do
    $(\boldsymbol{pk}[i], \boldsymbol{sk}[i]) \leftarrow\$ \mathsf{MS.Kg}$
$\sigma \leftarrow\$ \mathsf{Exec}^{\mathsf{h}}_{\mathsf{MS}}(\boldsymbol{sk}, \boldsymbol{pk}, m)$
$d \leftarrow \mathsf{MS.Vf}^{\mathsf{h}}(\boldsymbol{pk}, m, \sigma)$
Return $d$
```

```
Game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$

INIT:
1 $\mathsf{h} \leftarrow\$ \mathsf{MS.HF}$ ; $(pk, sk) \leftarrow\$ \mathsf{MS.Kg}$ ; Return $pk$

NS$(k, \boldsymbol{pk}, m)$:
2 $\boldsymbol{pk}[k] \leftarrow pk$ ; $u \leftarrow u + 1$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $m_u \leftarrow m$ ; $\mathsf{st}_u \leftarrow \mathsf{StInit}(k, sk, \boldsymbol{pk}, m)$
3 $\boldsymbol{b} \leftarrow (\varepsilon, \ldots, \varepsilon)$ ; $(\sigma, \mathsf{st}_u) \leftarrow\$ \mathsf{MS.Sign}^{\mathrm{H}}(\boldsymbol{b}, \mathsf{st}_u)$ ; Return $\sigma$

SIGN$_j(s, \boldsymbol{b})$: // $1 \le j \le \mathsf{MS.nr}$
4 $(\sigma, \mathsf{st}_s) \leftarrow\$ \mathsf{MS.Sign}^{\mathrm{H}}(\boldsymbol{b}, \mathsf{st}_s)$ ; Return $\sigma$

H$(x)$:
5 Return $\mathsf{h}(x)$

FIN$(k, \boldsymbol{pk}, m, \sigma)$:
6 If $(\boldsymbol{pk}[k] \ne pk)$ then Return false
7 If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \le i \le u\}$ then Return false
8 Return $\mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma)$
```

Figure 5: **Top left:** Procedure specifying an honest execution of the signing protocol associated with multi-signature scheme MS. **Top right:** Correctness game. **Bottom:** Unforgeability game.

now holds the next round of broadcasts.

The correctness game $\mathbf{G}^{\text{ms-cor}}_{\mathsf{MS},n}$ shown in the right column of Fig. 5 has only one procedure, namely FIN. We say that MS satisfies (perfect) correctness if for all positive integers $n$ we have $\Pr[\mathbf{G}^{\text{ms-cor}}_{\mathsf{MS},n}] = 1$.

UNFORGEABILITY. Game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ in Fig. 5 captures a notion of unforgeability for multi-signatures that slightly extends [3]. There is one honest player whose keys are picked at line 1, the adversary controlling all the other players. A new instance of the signing protocol is initialized by calling NS with an index $k$ and a vector $\boldsymbol{pk}$ of verification keys that the adversary can choose, possibly dishonestly, subject only to $\boldsymbol{pk}[k]$ being the verification key $pk$ of the honest player, as enforced by line 2. The first message of the honest player is sent out, and at this point $\mathsf{st}_u.\mathsf{rnd} = 1$. Now the adversary can run multiple concurrent instances of the signing protocol with the honest signer. Oracle H is the random oracle, simply calling $\mathsf{h}$. Eventually the adversary calls FIN with a forgery index $k$, a vector of verification keys (subjected to $\boldsymbol{pk}[k]$ being the public key of the honest signer), a message and a claimed signature. It wins if verification succeeds and the forgery was non-trivial.

The ms-uf-advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A})]$.

It is convenient for (later) proofs to have a separate signing oracle $\text{Sign}_j$ for each round $j \in [1..\mathsf{MS.nr}]$. It is required that any $\text{Sign}_j(s, \cdot)$ satisfy $s \in [1..u]$, and that the prior round queries $\text{Sign}_k(s, \cdot)$ for $k < j$ have already been made. It is required that for each $j, s$, at most one $\text{Sign}_j(s, \cdot)$ query is ever made.

Remarks. Our syntax and security notions for multi-signatures view a group of signers as captured by the vector (rather than the set) of their public keys. So for example, a forgery $((pk_1, pk_2), m, \sigma)$ is considered to be non-trivial even if there was a previous signing session under public keys $(pk_2, pk_1)$ and message $m$. This differs from previous formalizations that work instead with sets of public keys. However, previous definition can be recovered if a canonical encoding of sets of public keys into vectors of public keys is fixed in the usage of a scheme.

## 5 Analysis of the BN scheme

BN scheme. Let $\mathbb{G}$ be a group of prime order $p$. Let $g$ be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. The associated BN [3] multi-signature scheme $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ is shown in detail, in our syntax, in Fig. 6. The set $\mathsf{MS.HF}$ consists of all functions $\mathsf{h}$ such that $\mathsf{h}(0, \cdot) : \{0,1\}^* \to \{0,1\}^\ell$ and $\mathsf{h}(1, \cdot) : \{0,1\}^* \to \mathbb{Z}_p$. For $b \in \{0,1\}$ we write $\mathrm{H}_b(\cdot)$ for $\mathrm{H}(b, \cdot)$, so that scheme algorithms, and an ms-uf adversary, will have access to oracles $\mathrm{H}_0, \mathrm{H}_1$ rather than just $\mathrm{H}$.

The signing protocol has 3 rounds. In round 0, player $j$ picks $r \leftarrow_{\$} \mathbb{Z}_p$, stores $g^r$ in its state as $\mathsf{st}.\boldsymbol{R}[j]$, computes, and stores in its state, a value $\mathsf{st}.\boldsymbol{t}[j] \leftarrow \mathrm{H}_0((j, \mathsf{st}.\boldsymbol{R}[j]))$ that we call the BN-commitment, and broadcasts the BN-commitment. (Per our syntax, what is returned is the message to be broadcast and the updated state to be retained.) Since each player does this, in round 1, player $j$ receives the BN-commitments of the other players, storing them in vector $\mathsf{st}.\boldsymbol{t}$, and now broadcasting $\mathsf{st}.\boldsymbol{R}[j]$. In round 2, these broadcasts are received, so player $j$ can form the vector $\mathsf{st}.\boldsymbol{R}$. At line 20, it returns $\perp$ if one of the received values fails to match its commitment. As per our conventions, when this happens, this player will always broadcast $\perp$ in the future, so for round 3 we assume lines 21 and 22 are executed. These lines create the second component $\mathsf{st}.\boldsymbol{z}[j]$ of a Schnorr signature relative to the Schnorr-commitment $\mathsf{st}.\boldsymbol{R}[j]$ defined at line 13, and the player's own secret key, the computations being modulo $p$. This $\mathsf{st}.\boldsymbol{z}[j]$ is broadcast, so that, in round 3, our player receives the corresponding values from the other players. At line 27 it forms their modulo-$p$ sum $z$ and then forms the final signature $(\mathsf{st}.R, z)$.

Our description of the signing protocol differs, from that in [3], in some details that are brought out by our syntax, for example in using explicit party identities rather than seeing these as implicit in public keys.

Prior bounds. We recall the prior result of [3]. Let $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ and let $\mathcal{A}_{\text{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\text{ms}}$ has at most $q$ distinct queries across $\mathrm{H}_0, \mathrm{H}_1$ and at most $q_{\text{s}}$ queries to NS. Suppose the number of parties (length of verification-key vector) in queries to NS and Fin is at most $n$. Let $a = 8q_{\text{s}} + 1$ and $b = 2q + 16n^2 q_{\text{s}}$. Let $p = |\mathbb{G}|$. Then BN [3] give a DL-adversary $\mathcal{A}_{\text{dl}}$ such that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\text{ms}}) \leq \sqrt{(q + q_{\text{s}}) \cdot \left(\mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{A}_{\text{dl}}) + \frac{a}{p} + \frac{b}{2^\ell}\right)} \,. \tag{2}$$

The running time of $\mathcal{A}_{\text{dl}}$ is twice that of the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\text{ms}}$. BN obtain this result via their general forking lemma, which uses rewinding and accounts for the square-root in the bound.

```
Kg:
$sk \leftarrow\$ \mathbb{Z}_p$ ; $pk \leftarrow g^{sk}$
Return $(pk, sk)$
```

```
$\mathsf{Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma)$:
$(R, z) \leftarrow \sigma$ ; $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$
BN :
    For $i = 1, \ldots, n$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$
    Return ( $g^z = R \cdot \prod_{i=1}^{n} pk_i^{c_i}$ )
MuSig :
    $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
    $c \leftarrow \mathrm{H}_1((R, apk, m))$
    Return ( $g^z = R \cdot apk^c$ )
```

```
$\mathsf{Sign}^{\mathrm{H}}(\boldsymbol{b}, \mathsf{st})$:
$j \leftarrow \mathsf{st.me}$ ; $n \leftarrow \mathsf{st.n}$ ; $m \leftarrow \mathsf{st.msg}$ ; $sk \leftarrow \mathsf{st.sk}$ ; $\boldsymbol{pk} \leftarrow \mathsf{st.pk}$
If ($\mathsf{st.rnd} = 0$) then
    $\mathsf{st}.r \leftarrow\$ \mathbb{Z}_p$ ; $\mathsf{st}.\boldsymbol{R}[j] \leftarrow g^r$ ; $\mathsf{st}.\boldsymbol{t}[j] \leftarrow \mathrm{H}_0((j, \mathsf{st}.\boldsymbol{R}[j]))$ ; $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$
    Return $(\mathsf{st}.\boldsymbol{t}[j], \mathsf{st})$
If ($\mathsf{st.rnd} = 1$) then
    For all $i \neq j$ do $\mathsf{st}.\boldsymbol{t}[i] \leftarrow \boldsymbol{b}[i]$
    $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$ ; Return $(\mathsf{st}.\boldsymbol{R}[j], \mathsf{st})$
If ($\mathsf{st.rnd} = 2$) then
    For all $i \neq j$ do $\mathsf{st}.\boldsymbol{R}[i] \leftarrow \boldsymbol{b}[i]$
    If ( $\exists\, i : \mathrm{H}_0((i, \mathsf{st}.R[i])) \neq \mathsf{st}.\boldsymbol{t}[i]$ ) then Return $(\bot, \mathsf{st})$
    $\mathsf{st}.R \leftarrow \prod_{i=1}^{n} \mathsf{st}.R[i]$
    BN : $c_j \leftarrow \mathrm{H}_1((j, R, \boldsymbol{pk}, m))$ ; $\mathsf{st}.\boldsymbol{z}[j] \leftarrow sk \cdot c_j + \mathsf{st}.r$
    MuSig :
        $apk \leftarrow \prod_{i=1}^{n} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$ ; $c \leftarrow \mathrm{H}_1((R, apk, m))$
        $\mathsf{st}.\boldsymbol{z}[j] \leftarrow sk \cdot \mathrm{H}_2((\mathsf{st.me}, \boldsymbol{pk})) \cdot c + \mathsf{st}.r$
    $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$ ; Return $(\mathsf{st}.\boldsymbol{z}[j], \mathsf{st})$
If ($\mathsf{st.rnd} = 3$) then
    For all $i \neq j$ do $\mathsf{st}.\boldsymbol{z}[i] \leftarrow \boldsymbol{b}[i]$
    $z \leftarrow \sum_{i=1}^{n} \mathsf{st}.\boldsymbol{z}[i]$ ; Return $((\mathsf{st}.R, z), \mathsf{st})$
```

Figure 6: Algorithms of the multi-signature scheme $\mathsf{BN}[\mathbb{G}, g, \ell]$ and $\mathsf{MuSig}[\mathbb{G}, g, \ell]$, where $\mathbb{G}$ is a group of prime order $p$ with generator $g$. Code that differs between the two schemes is marked explicitly. Oracle $\mathrm{H}_i(\cdot)$ is defined to be $\mathrm{H}(i, \cdot)$ for $i = 0, 1$ (BN) and $i = 0, 1, 2$ (MuSig).

---

<u>Security of BN from IDL.</u> We give a IDL $\rightarrow$ BN reduction that is *tight* and in the *standard model.* Combining this with our tight AGM reduction DL $\rightarrow$ IDL of Theorem 3.1 we conclude a tight AGM reduction DL $\rightarrow$ BN. However, the standard model tight IDL $\rightarrow$ BN reduction is also interesting in its own right. It says that BN is just as secure as the Schnorr identification scheme. Since the latter has been around and resisted cryptanalysis for quite some time, this is good support for the security of BN.

**Theorem 5.1** [IDL $\rightarrow$ BN, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p$. Let $g$ be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ be the associated BN multi-signature scheme. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 5. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_0, q_1, q_{\mathrm{s}}$ distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{NS}$, respectively, and the*

*number of parties (length of verification-key vector) in queries to* NS *and* FIN *is at most $n$. Let $\alpha = q_s(4q_0 + 2q_1 + q_s)$ and $\beta = q_0(q_0 + n)$. Then we construct an adversary $\mathcal{A}_{\mathrm{id}}$ for game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$ (shown explicitly in Figure 18) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \leq \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q_1}(\mathcal{A}_{\mathrm{idl}}) + \frac{\alpha}{2p} + \frac{\beta}{2^\ell} \ . \tag{3}$$

*The running time of $\mathcal{A}_{\mathrm{idl}}$ is about that of the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$. Furthermore, adversary $\mathcal{A}_{\mathrm{idl}}$ is algebraic if adversary $\mathcal{A}_{\mathrm{ms}}$ is.*

Above, $q_0$ is the number of distinct queries to $\mathrm{H}_0$ made, not directly by the adversary, but across the execution of the adversary in game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$, and similarly for $q_1$. A lower bound on $q_1$ is the length of $\boldsymbol{pk}$ in $\mathcal{A}_{\mathrm{ms}}$'s FIN query, so we can assume it is positive. With the above theorem, we can now derive an upperbound $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p)$ of the advantage of any MS adversary with running time $t$, making $q$ queries to H, and $q_s$ signing interactions. We take $\ell \approx \log_2(p)$ and assume that $q_s \leq q \leq t \leq p$. Additionally, we assume that the advantage of any IDL adversary with running time $t$ is at most $t^2/p$ (as justified by Theorem 3.2). We obtain $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p) \leq t^2/p$ as shown in Fig. 1.

The full proof of Theorem 5.1 is given in Appendix G. Here we give a sketch. The reduction adversary $\mathcal{A}_{\mathrm{idl}}$ receives a group element $X$ from $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$ and forwards it to adversary $\mathcal{A}_{\mathrm{ms}}$ as the target public key. In order to run adversary $\mathcal{A}_{\mathrm{ms}}$, our adversary needs to be able to simulate the signing oracles $\mathrm{NS}, \mathrm{SIGN}_1, \mathrm{SIGN}_2$ as well as random oracles $\mathrm{H}_0$ and $\mathrm{H}_1$ without knowing $\mathsf{DL}_{\mathbb{G},g}(X)$. We first describe how the reduction proceeds if $\mathcal{A}_{\mathrm{ms}}$ makes no queries to $\mathrm{NS}, \mathrm{SIGN}_1$ or $\mathrm{SIGN}_2$, as this steps constitutes the main difference between our proof and the original proof of security for BN [3]. Adversary $\mathcal{A}_{\mathrm{idl}}$ uses the challenge oracle $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}.\mathrm{CH}$ to program the random oracle $\mathrm{H}_1$ (hence CH needs to be able to be queried upto the number of times $\mathrm{H}_1$ is evaluated). In particular, for each query $\mathrm{H}_1((k, R, \boldsymbol{pk}, m))$ where $\boldsymbol{pk}[k] = X$, our adversary first computes $T \leftarrow R \cdot \prod_{j \neq k} \mathrm{H}_1((j, R, \boldsymbol{pk}, m))$, then obtains $c \leftarrow\!\$\ \mathrm{CH}(T)$ before returning $c$ as the return value for the query $\mathrm{H}_1((k, R, \boldsymbol{pk}, m))$. By construction, a valid forgery for $\boldsymbol{pk}, m$ is some signature $\sigma = (R, z)$ such that

$$g^z = R \cdot \prod_{i=1}^{n} \boldsymbol{pk}[i]^{\mathrm{H}_1((i,R,\boldsymbol{pk},m))} = T \cdot X^c \ ,$$

where the first equality is by the verification equation of $\mathsf{BN}$ and the second equality is by the way $\mathrm{H}_1$ is programmed. This means that adversary $\mathcal{A}_{\mathrm{idl}}$ can simply forward the value of $z$ from a valid forgery, along with the index of the CH query corresponding to the $\mathrm{H}_1$ query of the forgery, to break game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$. Moreover, adversary $\mathcal{A}_{\mathrm{idl}}$ succeeds as long as the forgery given by $\mathcal{A}_{\mathrm{ms}}$ is valid.

It remains to show that oracles $\mathrm{NS}, \mathrm{SIGN}_1, \mathrm{SIGN}_2$ can be simulated without knowledge of the secret key, $\mathsf{DL}_{\mathbb{G},g}(X)$. Roughly, this is done using the zero-knowledge property of the underlying Schnorr identification scheme as well as by programming the random oracles $\mathrm{H}_0$ and $\mathrm{H}_1$. The original proof by [3] constructs an adversary and argues that it simulates these oracles faithfully if certain bad events do not happen. We take a more careful approach and do this formally via a sequence of seven games and use the code-base game playing framework of [5]. This game sequence incurs the additive loss as indicated in Equation (3).

CONVERSE. IDL is not merely some group problem that can be used to justify security of $\mathsf{BN}$ tightly; the hardness of IDL is, in fact, tightly equivalent to the MS-UF security of $\mathsf{BN}$. Formally, we give below a reduction turning any adversary against IDL into a forger $\mathcal{A}_{\mathrm{ms}}$ against $\mathsf{BN}$. This means that any security justification for $\mathsf{BN}$ must also justify the hardness of IDL.

```
$\underline{\mathcal{A}_{\mathrm{ms}}^{\mathrm{H}_1}(pk)}$:
$X \leftarrow pk$ ; $(pk', sk') \leftarrow\!\!\$\ \mathsf{MS.Kg}()$
$(I, z) \leftarrow \mathcal{A}_{\mathrm{xidl}}^{\mathrm{CH}}(pk)$ // $g^z = R_I \cdot pk^{c_{I,1}}$
$\sigma \leftarrow (R_I, z + sk' \cdot c_{I,2} \mod p)$ ; Return $((pk, pk'), m_I, \sigma)$
$\underline{\mathrm{CH}(R)}$:
$i \leftarrow i + 1$ ; $R_i \leftarrow R$ ; $m_i \leftarrow \langle i \rangle$
$c_{i,1} \leftarrow\!\!\$\ \mathrm{H}_1((1, R_i, (pk, pk'), m_i))$ ; $c_{i,2} \leftarrow\!\!\$\ \mathrm{H}_1((2, R_i, (pk, pk'), m_i))$
Return $c_{i,1}$
```

Figure 7: Adversary $\mathcal{A}_{\mathrm{ms}}$ for Theorem 6.1. For an integer $i$, $\langle i \rangle$ denote the binary representation of $i$.

---

**Theorem 5.2** [$\mathsf{BN} \to$ IDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p$. Let $g$ be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ be the associated BN multi-signature scheme. Let $q$ be a positive integer and $\mathcal{A}_{\mathrm{idl}}$ be an adversary against $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$. Then, we can construct an adversary $\mathcal{A}_{\mathrm{ms}}$ for game $\mathbf{G}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}$, making no queries to* NS*, and at most $2q$ queries to* $\mathrm{H}_1$*, such that*

$$\mathbf{Adv}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \geq \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}}) \ . \tag{4}$$

*The running time of $\mathcal{A}_{\mathrm{ms}}$ is about that of $\mathcal{A}_{\mathrm{idl}}$.*

**Proof of Theorem 5.2:** Consider the adversary given in Fig. 7. The adversary receives the target public key $pk$ from the MS-UF game and samples a key pair $(pk', sk') \leftarrow\!\!\$\ \mathsf{MS.Kg}$. The adversary will attempt to forge a signature against the vector of public keys $(pk, pk')$. Adversary $\mathcal{A}_{\mathrm{ms}}$ forwards $X = pk$ as the target point and runs IDL adversary $\mathcal{A}_{\mathrm{idl}}$. For each query $\mathrm{CH}(R)$ of $\mathcal{A}_{\mathrm{idl}}$, adversary $\mathcal{A}_{\mathrm{ms}}$ simulates the response as per line 4 to 6. If $\mathcal{A}_{\mathrm{idl}}$ succeeds, it must be that

$$g^z = R_I \cdot pk^{c_{I,1}} \ .$$

The value of $z$ can be used to construct a forgery signature (line 3). ▮

## 6 Analysis of the MuSig scheme

The current three-round version of MuSig has been proposed and analyzed by both [13] and [6]. Roughly, it is the $\mathsf{BN}$ scheme with added key aggregation.

Let $\mathbb{G}$ be a group of prime order $p$. And let $g$ be a generator of $g$ and $\ell \geq 1$ be an integer. The formal specification of $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ in our syntax is shown in Fig. 6. There are minimal differences between $\mathsf{MuSig}$ and $\mathsf{BN}$ and we only highlight the differences. The set $\mathsf{MS.HF}$ consists of all functions $h$ such that $h(0, \cdot) : \{0,1\}^* \to \{0,1\}^\ell$ and $h(i, \cdot) : \{0,1\}^* \to \mathbb{Z}_p$ for $i = 1, 2$. Verification is done as follows. First, an aggregate key $apk$ for the list of keys $\boldsymbol{pk} = (pk_1, \ldots, pk_n)$ is computed as $apk \leftarrow pk_1^{\mathrm{H}_2((1,\boldsymbol{pk}))} \cdots pk_n^{\mathrm{H}_2((n,\boldsymbol{pk}))}$ (line 8). Next, a single challenge is derived from the commitment $R$ and aggregate key $apk$ (line 9). The signature $(R, z)$ is valid if $g^z = R \cdot apk^c$. The second round of signing also changes accordingly to generate a valid signature (line 24 and 25).

The following gives a tight, standard-model reduction XIDL $\to \mathsf{MuSig}$. Combining this with our tight AGM chain DL $\to$ IDL $\to$ XIDL from Theorems 3.1 and 3.3, we get a tight AGM reduction DL $\to \mathsf{MuSig}$.

**Theorem 6.1** [XIDL $\to$ $\mathsf{MuSig}$, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p$. Let $g$ be a generator of $\mathbb{G}$ and $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ be the associated MuSig multi-signature scheme. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 5. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_0, q_1, q_2, q_{\mathrm{s}}$ distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$, respectively, and the number of parties (length of verification-key vector) in queries to* NS *and* Fin *is at most $n$. Let $\alpha = q_{\mathrm{s}}(4q_0 + 2q_1 + q_{\mathrm{s}}) + 2q_1q_2$ and $\beta = q_0(q_0 + n)$. Then we construct an adversary $\mathcal{A}_{\mathrm{xidl}}$ for game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$ (shown explicitly in Figure 24) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \leq \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}}) + \frac{\alpha}{2p} + \frac{\beta}{2^{\ell}} \ . \tag{5}$$

*The running time of $\mathcal{A}_{\mathrm{xidl}}$ is about that of the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$. Furthermore, adversary $\mathcal{A}_{\mathrm{xidl}}$ is algebraic if adversary $\mathcal{A}_{\mathrm{ms}}$ is.*

We remark that the values of $q_1$ and $q_2$ above arise from the number of queries to $\mathrm{H}_1$ and $\mathrm{H}_2$ made in the execution of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}})$. As a result, the appearance of $q_1$ and $q_2$ has their orders "switched" compared to in Section 3. With the above theorem, we can now derive an upperbound $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_{\mathrm{s}}, p)$ of the advantage of any MS adversary with running time $t$, making $q$ queries to H, and $q_{\mathrm{s}}$ signing interactions. We take $\ell \approx \log_2(p)$ and assume that $q_{\mathrm{s}} \leq q \leq t \leq p$. Additionally, we assume that the advantage of any XIDL adversary with running time $t$ is at most $t^2/p$ (as justified by Theorem 3.4). We obtain $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p) \leq t^2/p$ as shown in Fig. 1.

We again describe the reduction at a high level and defer the full proof to Appendix H. First, the reduction adversary $\mathcal{A}_{\mathrm{xidl}}$ receives group element $X$ from game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G}g,q_2,q_1}$ and runs $\mathcal{A}_{\mathrm{ms}}$ with the target public key set to $X$. Similar to the proof of Theorem 5.1, our adversary needs to simulate the signing oracles $\mathrm{NS}, \mathsf{Sign}_1, \mathsf{Sign}_2$ as well as $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ without knowing $\mathsf{DL}_{\mathbb{G},g}(X)$ in order to run $\mathcal{A}_{\mathrm{ms}}$. This again relies on the zero-knowledge property of the underlying Schnorr identification scheme and the programming of $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$. This step is done formally in a game sequence in the full proof and incurs the additive loss in Equation (5). To turn a forgery into a break against XIDL, our adversary programs $\mathrm{H}_1$ and $\mathrm{H}_2$ as follows. For the $j$-th query of $\mathrm{H}_2((k, \boldsymbol{pk}))$ where $\boldsymbol{pk}[k] = X$, the adversary first computes $S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$, then obtains $e_j \leftarrow\$ \ \textsc{NwTar}(S)$ before returning $e_j$ as the response for the query. We remark that this particular query of $\mathrm{H}_2$ have created an aggregate public key $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))} = S \cdot X^{e_j}$, which is also the value of $T_j$ that is recorded in the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. For each $i$-th query of $\mathrm{H}_1((R, apk, m))$, the adversary first finds the index $j_{\mathrm{sel}}$ of the $\mathrm{H}_2$-query that corresponds to the input $apk$, then obtains $c_i \leftarrow\$ \ \textsc{Ch}(j_{\mathrm{sel}}, R)$ before returning $c_i$ as the response for the query. If the eventual forgery is given for these two particular queries to $\mathrm{H}_1$ and $\mathrm{H}_2$, meaning forgery is $\boldsymbol{pk}, m, (R, z)$ for some $z$, then the verification equation of the signature scheme says that $g^z = R \cdot apk^{\mathrm{H}_1((R,apk,m))}$. But this matches exactly the winning condition of $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$, since $apk = T_{j_{\mathrm{sel}}}$ and $c_i = \mathrm{H}_1((R, apk, m))$. Hence, our adversary $\mathcal{A}_{\mathrm{xidl}}$ can simply return $(i, z)$ to break XIDL, as long as the forgery provided by $\mathcal{A}_{\mathrm{ms}}$ is valid.

Similar to the relation between IDL and $\mathsf{BN}$, XIDL is also tightly equivalent to the MS-UF security of $\mathsf{MuSig}$. In particular, we turn any adversary breaking XIDL into a forger against $\mathsf{MuSig}$. This means that any security justification for $\mathsf{MuSig}$ must also justify the hardness of XIDL.

**Theorem 6.2** [$\mathsf{MuSig}$ $\to$ XIDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p$. Let $g$ be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ be the associated MuSig multi-signature scheme. Let $q_1, q_2$ be a positive integers and $\mathcal{A}_{\mathrm{xidl}}$ be an adversary against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. Then, we can construct an adversary $\mathcal{A}_{\mathrm{ms}}$ for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$, making no queries to* NS*, and at most $2q_1$ and $2q_2$ queries to $\mathrm{H}_1$ and $\mathrm{H}_2$ respectively, such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \geq \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}}) \ . \tag{6}$$

```
$\underline{\mathcal{A}_{\text{ms}}^{\text{H}_1,\text{H}_2}(pk)}$:
$X \leftarrow pk$ ; $(I, z) \leftarrow \mathcal{A}_{\text{xidl}}^{\text{NwTar},\text{Ch}}(pk)$ ; $J \leftarrow \text{TI}[I]$
$\sigma \leftarrow (R_I, z)$ ; Return $((pk, S_J), m_I, \sigma)$
$\underline{\text{NwTar}(S)}$:
$j \leftarrow j + 1$ ; $S_j \leftarrow S$
$e_{j,1} \leftarrow\$ \text{H}_2((1, (pk, S)))$ ; $e_{j,2} \leftarrow\$ \text{H}_2((2, (pk, S)))$ ; $e_j \leftarrow e_{j,2}/e_{j,1} \mod p$
$apk_j \leftarrow pk^{e_{j,1}} S^{e_{j,2}}$ ; $T_j \leftarrow pk \cdot S^{e_j}$ ; Return $e_j$
$\underline{\text{Ch}(j_{\text{sel}}, R)}$:
$i \leftarrow i + 1$ ; $R_i \leftarrow R$ ; $m_i \leftarrow \langle i \rangle$ ; $\text{TI}[i] \leftarrow j_{\text{sel}}$
$c_i \leftarrow \text{H}_1((apk_{j_{\text{sel}}}, R, m_i)) \cdot e_{j_{\text{sel}},1}$ ; Return $c_i$
```

Figure 8: Adversary $\mathcal{A}_{\text{ms}}$ for Theorem 6.1. For an integer $i$, $\langle i \rangle$ denote the binary representation of $i$.

---

*The running time of $\mathcal{A}_{\text{ms}}$ is about that of $\mathcal{A}_{\text{idl}}$.*

**Proof of Theorem 6.2:** Consider the adversary given in Fig. 8. The adversary receives the target publick key $pk$ from the MS-UF game. Adversary $\mathcal{A}_{\text{ms}}$ forwards $X = pk$ as the target point and runs XIDL adversary $\mathcal{A}_{\text{idl}}$. For each query $\text{NwTar}(S)$ of $\mathcal{A}_{\text{xidl}}$, adversary $\mathcal{A}_{\text{ms}}$ uses $S$ as a public key to generate the aggregate key $apk$ for the list $(pk, S)$. By construction, the $j$-th target $T_j$ for the XIDL game is related to $apk_j$ by $apk_j = T_j^{e_{j,1}}$. For each $\text{Ch}(j_{\text{sel}}, R)$ query of $\mathcal{A}_{\text{xidl}}$, adversary $\mathcal{A}_{\text{ms}}$ programs in the $\text{H}_1$ outputs corresponding to a forgery agaisnt the aggregate key $apk_{j_{\text{sel}}}$ (line 6 and 7). By construction, if $\mathcal{A}_{\text{xidl}}$ succeeds, it must be that

$$g^z = R_I \cdot T_J^{c_I} = R_I \cdot T_J^{\text{H}_1((apk_J, R, m_i)) \cdot e_{J,1}} = R_I \cdot apk_J^{\text{H}_1((apk_J, R, m_i))} .$$

Hence, adversary $\mathcal{A}_{\text{ms}}$ produces a valid forgery at line 2. ▌

## 7 HBMS: Our new two-round multi-signature scheme

Recall that BN and MuSig are three-round schemes, and two-round schemes are desired due to blockchain applications. In this section, we introduce our new, efficient two-round multi-signature scheme supporting key-aggregation, HBMS. We first demonstrate its tight security against algebraic adversaries (Theorem 7.1), before justifying its security in the standard model (Theorem 7.2). Referring to Fig. 3, these results establish arrow 5. We refer to Fig. 2 for comparisons of HBMS against other two-round schemes.

<u>Two-round MS scheme HBMS.</u> The formal definition of our scheme is given in Fig. 9. HBMS has the same key generation algorithm Kg and key aggregation Ag algorithm as MuSig. We describe informally the process involved to sign a message $m$ under a vector of public keys $\boldsymbol{pk}$. In the first round, each signer $i$ samples $s_i$ and $r_i$ uniformly from $\mathbb{Z}_p$ and computes a commitment

$$T_i \leftarrow \text{H}_0((\boldsymbol{pk}, m))^{s_i} \cdot g^{r_i} ,$$

which is sent to every other signer. In the second round, each signer receives the list of commitments $T_1, \ldots, T_n$ from each signer, and compute the aggregate value $T \leftarrow \prod_i T_i$. Each signer then computes the challenge value as $c \leftarrow \text{H}_1((T, apk, m))$. To compute the reply, each signer $i$ computes $z_i \leftarrow r_i + sk \cdot c \cdot \text{H}_2((i, \boldsymbol{pk}))$ and sends $(s_i, z_i)$ to every other signer. Finally, any signer can

```
MS.Kg:
    $sk \leftarrow\$ \mathbb{Z}_p$ ; $pk \leftarrow g^{sk}$
    Return $(pk, sk)$
```

```
MS.Vf^{H_0,H_1,H_2}(pk, m, σ):
    $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$ ; $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
    $(T, s, z) \leftarrow \sigma$ ; $c \leftarrow \mathrm{H}_1((T, apk, m))$
    $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$ ; Return $(g^z h^s = T \cdot apk^c)$
```

```
MS.Sign^{H_0,H_1,H_2}(b, st):
    $j \leftarrow$ st.me ; $n \leftarrow$ st.n ; $m \leftarrow$ st.msg ; $sk \leftarrow$ st.sk ; $\boldsymbol{pk} \leftarrow$ st.pk
    $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$ ; $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
    If (st.rnd $= 0$) then
        st.$r[j] \leftarrow\$ \mathbb{Z}_p$ ; st.$s[j] \leftarrow\$ \mathbb{Z}_p$
        $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$ ; st.$\boldsymbol{R}[j] \leftarrow g^{\mathsf{st}.r[j]}$ ; st.$\boldsymbol{T}[j] \leftarrow$ st.$\boldsymbol{R}[j] \cdot h^{\mathsf{st}.s[j]}$
        st.rnd $\leftarrow$ st.rnd $+ 1$ ; Return (st.$\boldsymbol{T}[j]$, st)
    If (st.rnd $= 1$) then
        For all $i \neq j$ do st.$\boldsymbol{T}[i] \leftarrow \boldsymbol{b}[i]$
        st.$T \leftarrow \prod_{i=1}^n$ st.$\boldsymbol{T}[i]$ ; st.$c \leftarrow \mathrm{H}_1(($st.$T, apk, m))$ ; $e_j \leftarrow \mathrm{H}_2((j, \boldsymbol{pk}))$
        st.$z[j] \leftarrow sk \cdot c \cdot e_j +$ st.$r[j]$ ; st.$\boldsymbol{t}[j] \leftarrow$ (st.$s[j]$, st.$z[j]$)
        st.rnd $\leftarrow$ st.rnd $+ 1$ ; Return (st.$\boldsymbol{t}[j]$, st)
    If (st.rnd $= 2$) then
        For all $i \neq j$ do st.$\boldsymbol{t}[i] \leftarrow \boldsymbol{b}[i]$
        $(s, z) \leftarrow \sum_i^n \boldsymbol{t}[i]$ ; Return ((st.$T, s, z$), st)
```

Figure 9: Two-round multi-signature scheme $\mathsf{MS} = \mathsf{HBMS}[\mathbb{G}, g]$ parameterized by a group $\mathbb{G}$ of prime order $p$ with generator $g$.

---

now compute the final signature as $(T, s, z)$ where $s = \sum_i s_i$ and $z = \sum_i z_i$. To verify a signature $(T, s, z)$ on $(\boldsymbol{pk}, m)$, the equation

$$g^z \cdot \mathrm{H}_0((\boldsymbol{pk}, m))^s = T \cdot apk^{\mathrm{H}_1((T, apk, m))} ,$$

must hold, where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$. Compared to MuSig, the verification equation of HBMS involves an additional power of $\mathrm{H}((\boldsymbol{pk}, m))$ (hence the name HBMS, or "Hash-Base Multi-Signature").

Tight security against algebraic adversaries. We first show that HBMS is tightly MS-UF-secure against algebraic adversaries.

**Theorem 7.1** [DL $\rightarrow$ HBMS, AGM] *Let $\mathbb{G}$ be a group of prime order $p$ with generator $g$. Let* MS *be the* $\mathsf{HBMS}[\mathbb{G}, g]$ *scheme. Let $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$ be an algebraic adversary for game $\mathbf{G}_{\mathsf{MS}}^{\text{ms-uf}}$ of Figure 5. Assume the execution of game $\mathbf{G}_{\mathsf{MS}}^{\text{ms-uf}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_1, q_2$ distinct queries to $\mathrm{H}_1, \mathrm{H}_2$, respectively. Then we construct an adversary $\mathcal{A}_{\mathrm{dl}}$ for game $\mathsf{DL}_{\mathbb{G},g}$ (shown explicitly in Figure 26) such that*

$$\mathbf{Adv}_{\mathsf{MS}}^{\text{ms-uf}}(\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}) \leq \mathbf{Adv}_{\mathbb{G},g}^{\mathrm{dl}}(\mathcal{A}_{\mathrm{dl}}) + \frac{(q_1 + 1)q_2}{p} . \tag{7}$$

*The running time of $\mathcal{A}_{\mathrm{dl}}$ is about that of the execution of game $\mathbf{G}_{\mathsf{MS}}^{\text{ms-uf}}$ with $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$.*

Above, a reduction is given directly from DL, and there is no multiplicative loss. As before, assuming $q_{\mathrm{s}} \leq q \leq t \leq p$ and the generic hardness of DL (advantage of $t$-time adversary to be at most $t^2/p$), we derive that $\mathbf{UB}_{\mathsf{MS}}^{\text{ms-uf}}(t, q, q_{\mathrm{s}}, p) \leq t^2/p$, as shown in Fig. 2.

We give the highlevel proof sketch here and defer the full proof to Appendix I. Let $\mathcal{A}_{\mathrm{ms}}$ be the algebraic adversary against $\mathsf{HBMS}$. Our reduction adversary $\mathcal{A}_{\mathrm{dl}}$ sets its own target point $X$ (which it needs to obtain the discrete log of) as the target public key for $\mathcal{A}_{\mathrm{ms}}$. In order to run $\mathcal{A}_{\mathrm{ms}}$, our adversary $\mathcal{A}_{\mathrm{dl}}$ needs to be able to simulate oracles $\mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2$ (oracles representing the honest signer) as well as random oracles $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$. We first tackle the problem of simulating the honest signer without knowledge of the corresponding secret key. This is done by programming of random oracle $\mathrm{H}_0$. Suppose for $\boldsymbol{pk}, m$, we set $\mathrm{H}_0((\boldsymbol{pk}, m))$ to be $h = g^{\alpha} pk^{\beta}$ for some $\alpha, \beta \neq 0 \in \mathbb{Z}_p$ (whose exact distribution will be specified later). When the adversary interacts with the honest signer, the honest signer must first provide some commitment $T \in G$ (in the output of NS), then later produce $z, s \in \mathbb{Z}_p$ (in the output of $\textsc{Sign}_1$) such that

$$g^z h^s = T \cdot pk^c \ , \tag{8}$$

where $c \in \mathbb{Z}_p$ is some challenge value (that is derived using the random oracle and the responses of the adversary). To do this, our adversary set commitment $T = g^a h^b$ for $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$. It shall be convenient to express $pk$ in terms of $g$ and $h$ as well. Note that as long as $\beta \neq 0$, $pk = h^{(\beta^{-1})} g^{-\alpha(\beta^{-1})}$. Since both $T$ and $pk$ are known to be of the form $g^{\star} h^{\star}$ (where $\star$ denotes some element of $\mathbb{Z}_p$), so is the group element $T \cdot pk^c$ (for any known value of $c$). Hence, the right-hand side of Equation (8) is of the form $g^z h^s$ for some values $z$ and $s$ that our adversary can compute, and our adversary can return them as response in the second round. Above, we noted that this works as long as $\beta \neq 0$. To guarantee this, we sample $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\beta \leftarrow\!\!\$\ \mathbb{Z}_p^*$ in $\mathrm{H}_0$. It remains to check that such way of simulating the honest signer is indistinguishable from the behavior of an honest signer holding the secrete key and executing the protocol. Roughly, this is because in both cases, the triple $(T, z, s)$ is uniformly distributed over $\mathbb{G} \times \mathbb{Z}_p^2$, subjected to the condition that Equation (8) holds.

Now, our adversary $\mathcal{A}_{\mathrm{dl}}$ can move onto turning a forgery from $\mathcal{A}_{\mathrm{ms}}$ into a discrete logarithm for target point $X$. Suppose adversary $\mathcal{A}_{\mathrm{ms}}$ returns forgery $(\boldsymbol{pk}, m, (T, s, z))$. Then,

$$g^z h^s = T \cdot apk^c \ , \tag{9}$$

where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$ and $c = \mathrm{H}_1((T, apk, m))$. Since $\mathcal{A}_{\mathrm{ms}}$ is algebraic, our adversary $\mathcal{A}_{\mathrm{dl}}$ can rewrite Equation (9) to the form $g^{\alpha_g} = X^{\alpha_X}$, which allows us to compute the discrete log of $X$ as $\alpha_g \alpha_x^{-1} \mod p$, as long as $\alpha_X$ is not zero. The full proof upperbounds the probability that $\alpha_X = 0$ to be at most $q_1 q_2 / p$. Outside of this bad event, our adversary $\mathcal{A}_{\mathrm{dl}}$ will successfully compute the value of $\mathsf{DL}_{\mathbb{G},g}(X)$ from a valid forgery.

<u>STANDARD MODEL SECURITY OF HBMS.</u> We reduce the security of $\mathsf{HBMS}$ to the hardness of XIDL, with factor $q_{\mathrm{s}}$ loss. For applications, the number of signing queries $q_{\mathrm{s}}$ is much less than adversarial hash function evaluations. As a result, even though our reduction here is non-tight, the reduction loss is smaller compared to previous results for $\mathsf{BN}$, $\mathsf{MuSig}$ or other two round schemes (cf. Figure 1 and 2), at the expense of assuming the hardness of XIDL. Interestingly, due to Theorem 6.2, our results also state that $\mathsf{HBMS}$ is secure as long as $\mathsf{MuSig}$ is (via the reduction chain $\mathsf{MuSig} \rightarrow$ XIDL $\rightarrow \mathsf{HBMS}$), and this reduction again only losses a factor of $q_{\mathrm{s}}$ in the advantage.

**Theorem 7.2** [XIDL $\rightarrow$ $\mathsf{HBMS}$, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p$ with generator $g$. Let $\mathsf{MS}$ be the $\mathsf{HBMS}[\mathbb{G}, g]$ scheme given in Fig. 9. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 5. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_0, q_1, q_2, q_{\mathrm{s}}$ distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$, respectively. Then we construct an adversary $\mathcal{A}_{\mathrm{xidl}}$ for game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$ (shown explicitly in Figure 28) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \leq e(q_{\mathrm{s}} + 1) \cdot \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}}) + \frac{q_1 q_2}{p} \ , \tag{10}$$

*where $e$ is the base of the natural logarithm. Adversary $\mathcal{A}_{\mathrm{xidl}}$ makes $q_2$ queries to* $\textsc{NwTar}$ *and $q_1$*

*queries to* $\text{Ch}$. *The running time of* $\mathcal{A}_{\text{xidl}}$ *is about that of the execution of game* $\mathbf{G}_{\mathsf{MS}}^{\text{ms-uf}}$ *with* $\mathcal{A}_{\text{ms}}$.

Concretely, if we assume that XIDL is quantitatively as hard as DL, then against *any* adversary with running time $t$, making $q$ evaluations of the random oracle and making at most $q_{\text{s}}$ signing queries, $\mathsf{HBMS}$ has security $(q_{\text{s}}t^2 + q^2)/p \approx q_{\text{s}}t^2/p$.

We sketch the highlevel proof here and give the full proof in Appendix J. Our adversary receives the target point $X$ from the XIDL game and sets it as the target public key for adversary $\mathcal{A}_{\text{ms}}$. As before, in order to run $\mathcal{A}_{\text{ms}}$, we need to simulate oracles $\text{NwTar}, \text{Sign}_1, \text{Sign}_2$ as well as $\text{H}_0, \text{H}_1, \text{H}_2$. Recall that in the AGM proof, we can simulate the honest signer for $\boldsymbol{pk}, m$ if we set $\text{H}_0((\boldsymbol{pk}, m)) = g^\alpha h^\beta$. However, this way of programming $\text{H}_0$ does not facilitate in turning a forgery into a break for XIDL. Instead, we would like to program $\text{H}_0((\boldsymbol{pk}, m)) = g^\alpha$ for the forgery $\boldsymbol{pk}, m$. To do this, we use a technique of Coron [7], which programs $\text{H}_0((\boldsymbol{pk}, m))$ randomly in one of these two ways depending on a biased coin flip (with probability $\rho$ of giving 1). The reduction only succeeds if correct "guesses" are made. Specifically, we need that for every $\boldsymbol{pk}, m$ that is queried to the honest signer (in NS) then $\text{H}_0((\boldsymbol{pk}, m))$ must have been programmed to be $g^\alpha pk^\beta$ (for some $\alpha$ and $\beta$), and for the forgery $\boldsymbol{pk}, m$, it must be that $\text{H}_0((\boldsymbol{pk}, m)) = g^\alpha$ (for some $\alpha$). We can then optimizes for the value of $\rho$, resulting in a multiplicative loss of $e(1 + q_{\text{s}})$.

Suppose adversary $\mathcal{A}_{\text{ms}}$ returns a forgery $(\boldsymbol{pk}, m, (T, s, z))$ where we have previously programmed $\text{H}_0((\boldsymbol{pk}, m)) = g^\alpha$. The verification equation say that $g^z h^s = T \cdot apk^c$. Since $h$ is just a power of $g$, the left-hand side of the verification equation is also a known power of $g$ (specifically $g^{z+\alpha \cdot s}$). This means that our adversary $\mathcal{A}_{\text{xidl}}$ can proceed exactly as the reduction for $\mathsf{MuSig}$. In particular, for the $j$-th query of $\text{H}_2((k, \boldsymbol{pk}))$ where $\boldsymbol{pk}[k] = X$, the adversary first computes $S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\text{H}_2((i, \boldsymbol{pk}))}$, then obtains $e_j \leftarrow\$\ \text{NwTar}(S)$ before returning $e_j$ as the response for the query. We remark that this particular query of $\text{H}_2$ have created an aggregate public key $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\text{H}_2((i, \boldsymbol{pk}))} = S \cdot X^{e_j}$, which is also the value of $T_j$ that is recorded in the game $\text{Gm}_{\mathbb{G}, g, q_2, q_1}^{\text{xidl}}$. For each $i$-th query of $\text{H}_1((T, apk, m))$, the adversary first finds the index $j_{\text{sel}}$ of the $\text{H}_2$-query that corresponds to the input $apk$, then obtains $c_i \leftarrow\$\ \text{Ch}(j_{\text{sel}}, T)$ before returning $c_i$ as the response for the query. If the eventual forgery is given for these two particular queries to $\text{H}_1$ and $\text{H}_2$, meaning forgery is $\boldsymbol{pk}, m, (T, s, z)$, then the verification equation of the signature scheme says that $g^{z+\alpha \cdot s} = T \cdot apk^{\text{H}_1((T, apk, m))}$ (if we programmed $\text{H}_0((\boldsymbol{pk}, m))$ to be $g^\alpha$). Hence, our adversary $\mathcal{A}_{\text{xidl}}$ can simply return $(i, z + \alpha \cdot s)$ to break XIDL, as long as the forgery provided by $\mathcal{A}_{\text{ms}}$ is valid and we have made the right guesses in programming $\text{H}_0$.

## References

[1] H. K. Alper and J. Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. Cryptology ePrint Archive, Report 2020/1245, 2020. https://eprint.iacr.org/2020/1245. 4, 26

[2] A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, Oct. 2008. 4, 12

[3] M. Bellare and W. Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In K. Bhargavan, E. Oswald, and M. Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 529–552. Springer, Heidelberg, Dec. 2020. 8

[4] M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 411–422. Springer, Heidelberg, July 2007. 8

[5] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. 4

[6] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, Oct. / Nov. 2006. 3, 4, 11, 12, 14, 15, 17, 25, 26, 28, 31

[7] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, Aug. 2002. 8, 11

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 8, 17, 33, 40

[9] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, Jan. 2003. 3

[10] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Heidelberg, Dec. 2018. 3, 4, 8, 12, 18, 25

[11] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003. 8

[12] J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, Aug. 2000. 23

[13] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round $n$-out-of-$n$ and multi-signatures and trapdoor commitment from lattices. Cryptology ePrint Archive, Report 2020/1110, 2020. https://eprint.iacr.org/2020/1110. 8

[14] M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019. 4, 12, 25

[15] R. El Bansarkhani and J. Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In S. Foresti and G. Persiano, editors, *CANS 16*, volume 10052 of *LNCS*, pages 140–155. Springer, Heidelberg, Nov. 2016. 8

[16] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, Aug. 2018. 3, 4, 5, 9

[17] G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020. 5, 10

[18] L. Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proceedings-Computers and Digital Techniques*, 141(5):307–313, 1994. 3

[19] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983. 3

[20] E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, Aug. 2016. 5, 6, 7, 8, 9, 10

[21] C.-M. Li, T. Hwang, and N.-Y. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 194–204. Springer, Heidelberg, May 1995. 3

[22] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Heidelberg, May / June 2006. 3

[23] C. Ma, J. Weng, Y. Li, and R. Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Designs, Codes and Cryptography*, 54(2):121–133, 2010. 4, 12

[24] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019. 3, 4, 12, 18, 25

[25] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In M. K. Reiter and P. Samarati, editors, *ACM CCS 2001*, pages 245–254. ACM Press, Nov. 2001. 3

[26] J. Nick, T. Ruffing, and Y. Seurin. MuSig2: Simple two-round schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261, 2020. https://eprint.iacr.org/2020/1261. 4, 25

[27] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 20*, pages 1717–1731. ACM Press, Nov. 2020. 4, 7

[28] J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057, 2020. https://eprint.iacr.org/2020/1057. 25

[29] K. Ohta and T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *ASIACRYPT'91*, volume 739 of *LNCS*, pages 139–148. Springer, Heidelberg, Nov. 1993. 3

[30] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In B. K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, Dec. 2005. 5

[31] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. 4

[32] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, Jan. 1991. 9

[33] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997. 4, 5, 10

[34] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy*, pages 526–545. IEEE Computer Society Press, May 2016. 4, 12

## A Security bounds of multi-signature schemes

We survey previous results on discrete-log-based multi-signature schemes, with a focus on their reduction loss. We restate these results in the same notation and framework to facilitate comparisons. We have used this to obtain the estimates in Figures 1 and 2.

For the rest of the section, fix a group $\mathbb{G}$ of prime order $p$ that shall be used by each of the schemes of interest. Additionally, we assume that we fix adversaries $\mathcal{A}_{\mathrm{ms}}$ attacking each multi-signature scheme of interest, with running time $t$ (this is the total execution time of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}})$ and includes the running time of all oracles), making $q$ queries to the random oracle, $q_{\mathrm{s}}$ queries to

NS involving maximum of $N$-signers while achieving success advantage of $\epsilon$. For convenience, we let $q_{\mathrm{T}} = 1 + q + q_{\mathrm{s}}$.

BN. Bellare and Neven [3] gave a 3-round MS scheme that is based on the DL problem. In particular, they showed that given an MS-UF adversary $\mathcal{A}$, there exists DL-adversary with running time $t'$ achieving success advantage $\epsilon'$:

$$\epsilon' \geq \frac{\epsilon^2}{q + q_{\mathrm{s}}} - \frac{2q + 16N^2 q_{\mathrm{s}}}{2^{\ell}} - \frac{8Nq_{\mathrm{s}}}{p} \ , \tag{11}$$

$$t' \approx 2t \ , \tag{12}$$

where $\ell$ is a parameter, describing the output lengths of the random oracle used for commitments.

MuSig. BDN [6] and MPSW [13] gave a 3-round MS scheme that adds key aggregation on-top of BN. For security, BDN showed [6][Theorem 4] that given an MS-UF adversary $\mathcal{A}$, there exists DL-adversary with running time $t'$ achieving success advantage $\epsilon'$ where

$$\epsilon' = \frac{\epsilon - \delta}{64} \ , \tag{13}$$

$$t' = 512 \cdot t \cdot q_{\mathrm{T}}^2 (\epsilon - \delta)^{-1} \ln^{-2}(64/(\epsilon - \delta)) \ , \tag{14}$$

$$\delta = \frac{4Nq_{\mathrm{T}}}{p} \ , \tag{15}$$

as long as $p > 8q/\epsilon$. MPSW gave a tighter result by two direct applications of the forking lemma. In particular, they showed that [13][Theorem 1] given an MS-UF adversary $\mathcal{A}_{\mathrm{ms}}$, there exists DL-adversary with running time $t'$ achieving success advantage $\epsilon'$ where

$$\epsilon' = \frac{\epsilon^4}{q_{\mathrm{T}}^3} - \frac{16q_{\mathrm{s}}(q + N \cdot q_{\mathrm{s}})}{p} - \frac{16(q + Nq_{\mathrm{s}})^2 + 3}{2^{\ell}} \ , \tag{16}$$

$$t' \approx 4t \ . \tag{17}$$

mBCJ. DEFKLNS [8] gave a 2-round MS scheme mBCJ. For security, they showed that given an MS-UF adversary $\mathcal{A}$, there exists DL-adversary with running time $t'$ achieving success advantage $\epsilon'$ where

$$\epsilon' = \frac{\epsilon}{8e(q_{\mathrm{s}} + 1)} \ , \tag{18}$$

$$t' = t \cdot 64(N+1)^2 q_{\mathrm{T}}(q_{\mathrm{s}} + 1)\epsilon^{-1} \ln^{-1}(8e(N+1)(q_{\mathrm{s}} + 1)/\epsilon) \ , \tag{19}$$

as long as $p > 64e(N+1)q_{\mathrm{T}}(q_{\mathrm{s}} + 1)/\epsilon$.

MuSig-DN. NRSW [15] gave a 2-round MS scheme that has deterministic signing. For security, their result [15][Theorem 1] roughly translates to: given an an adversary attack MuSig-DN, there exists OMDL adversary attacking DL with success advantage approximately

$$\epsilon' \geq \left( \epsilon - q_{\mathrm{s}}\delta - \frac{q_{\mathrm{T}}^2}{2^{\lambda - 2}} - \frac{2}{2^{\lambda/4}} \right)^4 q_{\mathrm{T}}^{-3} \ , \tag{20}$$

$$t' \approx 4t \ , \tag{21}$$

where $\lambda$ is a parameter of the scheme and $\delta$ is a small constant associated with the group.

MuSig2. NRS [14] gave a 2-round MS scheme, parameterized by $\nu$. For $\nu \geq 4$, they showed that if there exists $\mathcal{A}$ attacking their scheme, they [14][Theorem 1] can build $\nu q_{\mathrm{s}}$-OMDL adversary with

```
Game Gm_0

FIN:
  $x \leftarrow\$ \mathsf{IG}$
  $c_1, \ldots, c_q \leftarrow\$ C$
  $(I, \sigma) \leftarrow\$ \mathcal{A}(x, c_1, \ldots, c_q)$
  Return $(I > 0)$
```

```
Game Gm_1

FIN:
  $x \leftarrow\$ \mathsf{IG}$
  $\rho \leftarrow\$ \mathrm{rand}(\mathcal{A})$ ; $c_1, \ldots, c_q \leftarrow\$ C$
  $(I, \sigma) \leftarrow\$ \mathcal{A}(x, c_1, \ldots, c_q)$
  If $(I = 0)$ then return $(0, \epsilon, \epsilon)$
  $c'_I, \ldots, c'_q \leftarrow\$ C$
  $(I', \sigma') \leftarrow\$ \mathcal{A}(x, c_1, \ldots, c_{I-1}, c'_I, \ldots, c'_q)$
  Return $((I = I')$ and $(c_I \neq c'_I))$
```

Figure 10: Games referred to in Lemma B.1. Both games have just one procedure, FIN, which does not take any input. These games run an algorithm $\mathcal{A}$ internally.

running time $t'$ achieving success advantage $\epsilon'$ where

$$\epsilon' \geq \frac{\epsilon^4}{m^3} - \frac{11}{p} - \frac{43m^4}{(p-1)^{\nu-3}} \ , \tag{22}$$

$$t' \approx 4t \ , \tag{23}$$

$$m = (\nu - 1)(q + q_\mathrm{s}) + 1 \ . \tag{24}$$

For $\nu = 2$, they give a tighter proof against algebraic adversaries. In particular, given an algebraic adversary $\mathcal{A}$ attacking their scheme for $\nu = 2$, they build adversary $\mathcal{B}$ against $q_\mathrm{s}$-OMDL that runs in time $t'$ to achieve success advantage $\epsilon'$ with

$$\epsilon' \geq \epsilon - 14\frac{q^3}{p} \ ,$$

$$t' \approx t + O(q^3) \ .$$

DWMS. Alper and Burdges [1] gave a 2-round MS scheme DWMS similar MuSig2 that is also proved secure from OMDL in AGM. Their proof as given is non-concrete. However, tracing through their reduction, we obtained the following reduction loss: given an algebraic MS-UF adversary $\mathcal{A}^{\mathrm{alg}}_{\mathrm{ms}}$, an $q_\mathrm{s}$-OMDL adversary can be constructed with advantage $\epsilon'$ with running time $t'$ where

$$\epsilon' \geq \epsilon - \frac{q_\mathrm{s}^3 q^2}{\sqrt{p}} \ ,$$

and $t' \approx t$.

## B Forking lemma

We recall the general forking lemma of [3]. We restate it using the games of Figure 10. Each game has just one procedure, FIN, which takes no inputs. The games are parameterized by an algorithm $\mathcal{A}$ that is executed inside the game, and also by an algorithm IG called an input generator.

**Lemma B.1** [3] *Let $q \geq 1$ be an integer. Let $C$ be a set of size $|C| \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on inputs $x, c_1, \ldots, c_q$ returns a pair, the first element of which is an integer in the range $0, \ldots, q$, and the second element of which we refer to as a side output. Let* IG *be a randomized algorithm that, as above, we call the input generator. Consider* $\mathrm{Gm}_0$ *(called the single run) and*

```
Game Gm_0, Gm_1, Gm_2

Init:
x ←$ Z_p ; X ← g^x
(I, z) ←$ A_idl^Ch(X)
b ← (g^z = R_I · X^{c_I})
Gm_0: Return b
w ← (r_{I,2} + c_I)
If (w = 0) then bad ← true
    Gm_1:
        If b then x' ← x
        Else x' ← ⊥
    Gm_2: x' ←$ Z_p
Else
    v ← (z − r_{I,1})
    x' ← v · w^{-1} mod p
Gm_1, Gm_2: Return (x = x')

Ch(R, (r_1, r_2)):
i ← i + 1 ; R_i ← R
r_{i,1} ← r_1 ; r_{i,2} ← r_2
c_i ←$ Z_p ; Return c_i
```

```
Adversary A_dl(X):
(I, z) ←$ A_idl^{NwTar,Ch}(X)
w ← (r_{I,2} + c_I)
If (w = 0) then x' ←$ Z_p
Else
    v ← (z − r_{I,1})
    x' ← v · w^{-1} mod p
Return x'

Ch(R, (r_1, r_2)):
i ← i + 1 ; R_i ← R
r_{i,1} ← r_1 ; r_{i,2} ← r_2
c_i ←$ Z_p ; Return c_i
```

Figure 11: Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$ and adversary $\mathcal{A}_{\mathrm{dl}}$ the proof of Theorem 3.1.

$\mathrm{Gm}_1$ *(called the forked run) given in Fig. 10. Then:*

$$\Pr[\mathrm{Gm}_0] \le \frac{q}{|C|} + \sqrt{q \cdot \Pr[\mathrm{Gm}_1]} \,. \tag{25}$$

## C Proof of Theorem 3.1

**Proof of Theorem 3.1:** Consider game $\mathrm{Gm}_0$ given in the left panel of Fig. 11. By construction, it is the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}})$. Next, consider game $\mathrm{Gm}_1$, where the winning condition has been changed to checking that $(x = x')$, where $x'$ is either computed on line 8 or 9 depending on whether $w = 0$. We claim that regardless of whether $w = 0$, game $\mathrm{Gm}_1$ returns true as long as $\mathrm{Gm}_0$ does. Assume $\mathrm{Gm}_0$ returns true, then $b$ is set to true. If $w = 0$, then the game $\mathrm{Gm}_1$ sets $x'$ to $x$ at line 8, so $\mathrm{Gm}_1$ alsot returns true. If $w \neq 0$, then the game $\mathrm{Gm}_1$ computes $x'$ as per line 12 and 13. Observe that if $b$ is true, then

$$g^z = R_I \cdot X^{c_I} \,.$$

Expanding this equation using the fact that $R_i = g^{r_{i,1}} X^{r_{i,1}}$, we get

$$g^z = g^{r_{I,1}} X^{r_{I,2}} \cdot X^{c_I} \,,$$

which means that

$$g^x = X = g^{(z - r_{I,1}) w^{-1}} = g^{x'} \,.$$

So game $\mathrm{Gm}_1$ must return true in this case as well. Hence

$$\Pr[\mathrm{Gm}_0] = \Pr[\mathrm{Gm}_1] \; . \tag{26}$$

Next, consider game $\mathrm{Gm}_2$, which sets $x'$ differently if $w = 0$. We have

$$\begin{aligned}\Pr[\mathrm{Gm}_1] &\le \Pr[\mathrm{Gm}_2] + \Pr[\mathrm{Gm}_2 \text{ sets } \mathsf{bad}] \\ &\le \Pr[\mathrm{Gm}_2] + \frac{q}{p} \; .\end{aligned} \tag{27}$$

Above, the calculation of $\Pr[\mathrm{Gm}_2$ sets bad$]$ is justified as follows. For each Ch query, there is $1/p$ chance that $r_{i,2} + c_i = 0$, since $c_i$ is uniform and independent of $r_{i,2}$. Hence, the probability that there is a choice of $i$ to make $w = r_{i,2} + c_i$ zero is at most $q/p$ using the union bound. Finally, we construct adversary $\mathcal{A}_{\mathrm{dl}}$, given in Fig. 11 such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) \; . \tag{28}$$

This is straight-forward, as $\mathcal{A}_{\mathrm{dl}}$ simulates Ch and computes $x'$ exactly as $\mathrm{Gm}_2$. ▌

## D Proof of Theorem 3.2

**Proof of Theorem 3.2:** Consider games $\mathrm{Gm}_0$ given in Fig. 12. Game $\mathrm{Gm}_0$ pre-samples all the $c_1, \dots, c_q$ values at line 2, but the game behaves otherwise exactly as $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}})$. We define $\Pr[\mathrm{Gm}_0]$ to be the probablity that the first component of the return value of $\mathrm{Gm}_0$ is non-zero. Hence,

$$\Pr[\mathrm{Gm}_0] = \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}}) \; . \tag{29}$$

Next, consider $\mathrm{Gm}_1$, which executes line 6 to 13 in addition to those executed by game $\mathrm{Gm}_0$. Similar to $\mathrm{Gm}_0$, we define $\Pr[\mathrm{Gm}_1]$ to be the probablity that the first component of the return value of $\mathrm{Gm}_1$ is non-zero. We have constructed $\mathrm{Gm}_1$ so that it is a forked run of $\mathrm{Gm}_0$ (with $c_1, \dots, c_q$ viewed as inputs) as defined by the forking lemma [3]. Specifically, line 8 to 10 freshly samples challenges $c'_I, \dots, c'_{q_2}$ after the selected forgery index $I$ before invoking $\mathcal{A}_{\mathrm{idl}}$ with these values programmed into $\mathrm{Ch}_2$. By the forking lemma, we have

$$\Pr[\mathrm{Gm}_0] \le \frac{q_2}{p} + \sqrt{q_2 \cdot \Pr[\mathrm{Gm}_1]} \; . \tag{30}$$

We now move onto game $\mathrm{Gm}_2$, which rewrites the winning condition of $\mathrm{Gm}_1$ into line 15 to 18. We claim that game $\mathrm{Gm}_2$ returns true as long as game $\mathrm{Gm}_1$ returns true. This is because if both flags $b$ and $b'$ are ture, then

$$\begin{aligned} g^z &= R_i X^{c_i} \\ g^{z'} &= R_{i'} X^{c_{i'}} \; , \end{aligned}$$

where $i = i' > 0$. Notice that we also have $R_i = R_{i'}$, this is because the two runs of $\mathcal{A}_{\mathrm{idl}}$ has not diverged when $R_i$ and $R_{i'}$ are supplied (since the first different value of $c_{i'}$ is only supplied afte $R_{i'}$ is given). Hence, putting the two equation together, we have

$$X^{c_I - c'_I} = g^{z - z'} \; ,$$

which implies the the computed value of $w = (c_I - c'_I)^{-1}(z - z')$ (line 17) is the correct discrete log of $X$ base $g$. As a result, $\mathrm{Gm}_2$ must return true as well, and

$$\Pr[\mathrm{Gm}_2] \ge \Pr[\mathrm{Gm}_1] \; . \tag{31}$$

```
Game Gm_0, Gm_1, Gm_2

INIT:
x ←$ $\mathbb{Z}_p$ ; $X \leftarrow g^x$
$\rho$ ←$ rand($\mathcal{A}_{\text{idl}}$) ; $c_1, \ldots, c_q$ ←$ $\mathbb{Z}_p$
$(I, z) \leftarrow \mathcal{A}_{\text{idl}}^{\text{CH}_1}(X; \rho)$
$b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
If not $b$ then $I \leftarrow 0$
Gm_0: Return $(I > 0)$
For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
$c'_I, c'_{I+1}, \ldots, c'_q$ ←$ $\mathbb{Z}_p$
$i \leftarrow 0$ ; $(I', z')$ ←$ $\mathcal{A}_{\text{idl}}^{\text{CH}_2}(X; \rho)$
$b' \leftarrow (g^{z'} = R_{I'} \cdot Y_{I'}^{c_{I'}})$
If not $b'$ then $I' \leftarrow 0$
Gm_1:
    Return $((I = I' > 0)$ and $(c_I \neq c'_I))$
Gm_2:
    If $((I \neq I')$ or $(c_I = c'_I))$ then
        Return $\perp$
    $w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
    Return $(g^w = X)$

CH_1(R):
$i \leftarrow i + 1$ ; $R_i \leftarrow R$
Return $c_i$

CH_2(R):
$i \leftarrow i + 1$ ; $R'_i \leftarrow R$
Return $c'_i$
```

```
Adversary $\mathcal{A}_{\text{dl}}(X)$:
$c_1, \ldots, c_q$ ←$ $\mathbb{Z}_p$ ; $\rho$ ←$ rand($\mathcal{A}_{\text{idl}}$)
$(I, z)$ ←$ $\mathcal{A}_{\text{idl}}^{\text{CH}_1}(X; \rho)$
$b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
If not $b'$ then Return $\perp$
For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
$c'_I, c'_{I+1}, \ldots, c'_{q_2}$ ←$ $\mathbb{Z}_p$
$i \leftarrow 0$ ; $(I', z')$ ←$ $\mathcal{A}_{\text{idl}}^{\text{CH}_2}(X; \rho)$
$b' \leftarrow (g^{z'} = R_{i'} \cdot Y_{i'}^{c_{i'}})$
If not $b'$ then Return $\perp$
If $((I \neq I')$ or $(c_I = c'_I))$ then
    Return $\perp$
$w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
Return $w$

CH_1(R):
$i \leftarrow i + 1$ ; $R_i \leftarrow R$
Return $c_i$

CH_2(R):
$i \leftarrow i + 1$ ; $R'_i \leftarrow R$
Return $c'_i$
```

Figure 12: Games $\text{Gm}_0, \text{Gm}_1, \text{Gm}_2$ and adversary $\mathcal{A}_{\text{dl}}$ for proof of Theorem 3.2. $\rho \leftarrow\$ \text{rand}(\mathcal{A}_{\text{idl}})$ denotes sampling the random coins of $\mathcal{A}_{\text{idl}}$ and assigning it to $\rho$.

Finally, we construct adversary $\mathcal{A}_{\text{dl}}$, given in Fig. 12, such that

$$\Pr[\text{Gm}_2] = \mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{A}_{\text{dl}}) . \tag{32}$$

Adversary $\mathcal{A}_{\text{dl}}$ forwards its target point $X$ to $\mathcal{A}_{\text{idl}}$ and simulates $\text{Gm}_2$, starting from line 2 of $\text{Gm}_2$ and ending at line 17 of $\text{Gm}_2$, before outputting the computed value of $w$ as the discrete log of target point $X$. Putting the above equations together, we obtain the claim in the theorem. ▌

## E Proof of Theorem 3.3

**Proof of Theorem 3.3:** We recall the convention that representation of each of the group elements $S$ and $R$ are additionally supplied when oracles NWTAR and CH are called. Specifically,

```
Game $\mathrm{Gm}_0$, $\mathrm{Gm}_1$, $\mathrm{Gm}_2$

INIT:
1  $x \leftarrow\$ \mathbb{Z}_p$ ; $X \leftarrow g^x$
2  $e_1, \ldots, e_{q_1} \leftarrow\$ \mathbb{Z}_p$ ; $c_1, \ldots, c_{q_2} \leftarrow\$ \mathbb{Z}_p$
3  $(I, z) \leftarrow\$ \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{Ch}}(X)$
4  $b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
5  $\underline{\mathrm{Gm}_0}$: Return $b$
6  $w \leftarrow (r_{I,2} + (s_{I,2} + e_J) \cdot c_I)$
7  If $(w = 0)$ then bad $\leftarrow$ true
8     $\underline{\mathrm{Gm}_1}$:
9        If $b$ then $x' \leftarrow x$
10       Else $x' \leftarrow \bot$
11    $\underline{\mathrm{Gm}_2}$: $x' \leftarrow\$ \mathbb{Z}_p$
12 Else
13    $v \leftarrow (z - r_{I,1} - s_{I,1} \cdot c)$
14    $x' \leftarrow v \cdot w^{-1} \mod p$
15 $\underline{\mathrm{Gm}_1, \mathrm{Gm}_2}$: Return $(x = x')$

NWTAR$(S, (s_1, s_2))$:
16 $j \leftarrow j + 1$ ; $S_j \leftarrow S$
17 $s_{j,1} \leftarrow s_1$ ; $s_{j,2} \leftarrow s_2$
18 $e_j \leftarrow\$ \mathbb{Z}_p$ ; $T_j \leftarrow S_j \cdot X^{e_j}$
19 Return $e_j$

CH$(j_{\mathrm{sel}}, R, (r_1, r_2))$:
20 Requires $1 \leq j_{\mathrm{sel}} \leq j$
21 $i \leftarrow i + 1$ ; $R_i \leftarrow R$
22 $r_{i,1} \leftarrow r_1$ ; $r_{i,2} \leftarrow r_2$
23 $Y_i \leftarrow T_{j_{\mathrm{sel}}}$ ; TJ$[i] \leftarrow j_{\mathrm{sel}}$
24 $c_i \leftarrow\$ \mathbb{Z}_p$ ; Return $c_i$
```

```
Adversary $\mathcal{A}_{\mathrm{dl}}(X)$:
1  $(I, z) \leftarrow\$ \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{Ch}}(X)$
2  $J \leftarrow$ TJ$[I]$
3  $w \leftarrow (r_2 + s_2 \cdot e_J \cdot c_I)$
4  If $(w = 0)$ then $x' \leftarrow\$ \mathbb{Z}_p$
5  Else
6     $v \leftarrow (z - r_{I,1} - s_{I,1} \cdot c)$
7     $x' \leftarrow v \cdot w^{-1} \mod p$
8  Return $x'$

NWTAR$(S, (s_1, s_2))$:
9  $j \leftarrow j + 1$ ; $S_j \leftarrow S$
10 $s_{j,1} \leftarrow s_1$ ; $s_{j,2} \leftarrow s_2$
11 $e_j \leftarrow\$ \mathbb{Z}_p$ ; $T_j \leftarrow S_j \cdot X^{e_j}$
12 Return $e_j$

CH$(j_{\mathrm{sel}}, R, (r_1, r_2))$:
13 Requires $1 \leq j_{\mathrm{sel}} \leq j$
14 $i \leftarrow i + 1$ ; $R_i \leftarrow R$
15 $r_{i,1} \leftarrow r_1$ ; $r_{i,2} \leftarrow r_2$
16 $Y_i \leftarrow T_{j_{\mathrm{sel}}}$ ; TJ$[i] \leftarrow j_{\mathrm{sel}}$
17 $c_i \leftarrow\$ \mathbb{Z}_p$ ; Return $c_i$
```

Figure 13: Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$ and adversary $\mathcal{A}_{\mathrm{dl}}$ the proof of Theorem 3.3.

each of its NWTAR queries must be of the form

$$\mathrm{NwTar}(S, (s_1, s_2)) \ ,$$

such that $S = g^{s_1} X^{s_2}$. And each CH query must be of the form

$$\mathrm{Ch}(j_{\mathrm{sel}}, R, (r_1, r_2)) \ ,$$

such that $R = g^{r_1} X^{r_2}$.

Consider game $\mathrm{Gm}_0$ given in the left panel of Fig. 13. By construction, it is the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}})$. Next, consider game $\mathrm{Gm}_1$, where the winning condition has been changed to checking that $(x = x')$, where $x'$ is either computed on line 9 or 10 depending on whether $w = 0$. We claim that regardless of the value of $w$, game $\mathrm{Gm}_1$ returns true as long as $\mathrm{Gm}_0$ does ($\mathrm{Gm}_0$ returns the boolean value $b$). We check this by cases. First, if $w = 0$, then the games sets $x'$ to $x$ if $b$ is true, so $\mathrm{Gm}_1$ also returns true. If $w \neq 0$, then observe that if $b$ is true, then

$$g^z = R_I \cdot (S_J \cdot X^{e_J})^{c_I} .$$

Expanding this equation using the fact that $R_i = g^{r_{i,1}} X^{r_{i,1}}$ and $S_j = g^{s_{j,1}} X^{s_{j,2}}$, we get

$$g^z = g^{r_{I,1}} X^{r_{I,2}} \cdot (g^{s_{J,1}} X^{s_{J,2}} \cdot X^{e_J})^{c_I} ,$$

which means that

$$g^x = X = g^{(z - r_{I,1} - s_{J1} \cdot c_I) w^{-1}} = g^{x'} .$$

Hence

$$\Pr[\mathrm{Gm}_0] = \Pr[\mathrm{Gm}_1] . \tag{33}$$

Next, consider game $\mathrm{Gm}_2$, which sets $x'$ differently if $w = 0$. We have

$$\begin{aligned} \Pr[\mathrm{Gm}_1] &\leq \Pr[\mathrm{Gm}_2] + \Pr[\mathrm{Gm}_2 \text{ sets } \mathsf{bad}] \\ &\leq \Pr[\mathrm{Gm}_2] + \frac{q_1 + q_2}{p} . \end{aligned} \tag{34}$$

Above, the calculation of $\Pr[\mathrm{Gm}_2$ sets $\mathsf{bad}]$ is justified as follows. First, the probability that $s_{j,2} + e_j = 0$ for any $j$ is at most $q_1/p$, since $e_j$ is uniform and independnet of $s_{j,2}$. Second, assuming $s_{j,2} + e_j \neq 0$ for all $j$, then the probability that $r_{i,2} + (s_{\mathrm{TJ}[i],2} + e_{\mathrm{TJ}[i]}) \cdot c_i = 0$ for some $i$ is at most $q_2/p$, since $c_i$ is uniform and independent of $r_{i,2}$. Finally, we construct adversary $\mathcal{A}_{\mathrm{dl}}$, given in the right panel of Fig. 13 such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) . \tag{35}$$

This is straight-forward, as $\mathcal{A}_{\mathrm{dl}}$ simulates NwTar, Ch and computes $x'$ exactly as $\mathrm{Gm}_2$. ▌

## F Proof of Theorem 3.4

**Proof of Theorem 3.4:** Consider games $\mathrm{Gm}_0$ given in Fig. 14. Game $\mathrm{Gm}_0$ pre-samples all the $e_j$ and $c_i$ values at line 2 and 3, but the game behaves otherwise exactly as $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}})$. We define $\Pr[\mathrm{Gm}_0]$ to be the probablity that the first component of the return value of $\mathrm{Gm}_0$ is non-zero. Hence,

$$\Pr[\mathrm{Gm}_0] = \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}}) . \tag{36}$$

Next, consider $\mathrm{Gm}_1$, which executes line 6 to 14 addition to those executed by game $\mathrm{Gm}_0$. Similar to $\mathrm{Gm}_0$, we define $\Pr[\mathrm{Gm}_1]$ to be the probablity that the first component of the return value of $\mathrm{Gm}_1$ is non-zero. We have constructed $\mathrm{Gm}_1$ so that it is a forked run of $\mathrm{Gm}_0$ (with $c_1, \ldots, c_{q_2}$ viewed as inputs) as defined by the forking lemma [3]. Specifically, line 8 to 10 freshly samples challenges $c'_i, \ldots, c'_{q_2}$ after the selected forgery index $i$ before invoking $\mathcal{A}_{\mathrm{xidl}}$ with these values reprogrammed into Ch. We remark that the values of $e_1, \ldots, e_{q_1}$, which are outputs of NwTar are *not* resampled across the two runs of $\mathcal{A}_{\mathrm{xidl}}$. By the forking lemma, we have

$$\Pr[\mathrm{Gm}_0] \leq \frac{q_2}{p} + \sqrt{q_2 \cdot \Pr[\mathrm{Gm}_1]} . \tag{37}$$

We now move onto game $\mathrm{Gm}_2$, which rewrites the winning condition of $\mathrm{Gm}_1$ into line 16 to 19. We claim that game $\mathrm{Gm}_2$ returns true as long as game $\mathrm{Gm}_1$ returns true. This is because if both flags $b$ and $b'$ are ture, then

$$\begin{aligned} g^z &= R_I Y_I^{c_I} \\ g^{z'} &= R_{I'} Y_{I'}^{c_{I'}} , \end{aligned}$$

```
Game $\mathrm{Gm}_0$, $\mathrm{Gm}_1$, $\mathrm{Gm}_2$

INIT:
1  $x \leftarrow\$ \mathbb{Z}_p$ ; $X \leftarrow g^x$ ; $\rho \leftarrow\$ \mathrm{rand}(\mathcal{A}_{\mathrm{xidl}})$
2  $e_1, \ldots, e_{q_1} \leftarrow\$ \mathbb{Z}_p$ ; $c_1, \ldots, c_{q_2} \leftarrow\$ \mathbb{Z}_p$
3  $(I, z) \leftarrow \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{Ch}}(X; \rho)$
4  $b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
5  If not $b$ then $I \leftarrow 0$
6  $\mathrm{Gm}_0$: Return $(I > 0)$
7  For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
8  $i, j \leftarrow 0$ ; $c'_I, c'_{I+1}, \ldots, c'_q \leftarrow\$ \mathbb{Z}_p$
9  $(I', z') \leftarrow\$ \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{ChSim}}(X; \rho)$
10 $b' \leftarrow (g^{z'} = R_{I'} \cdot Y_{I'}^{c'_{I'}})$
11 If not $b'$ then $i' \leftarrow 0$
12 $j \leftarrow \mathrm{TJ}[I]$ ; $j' \leftarrow \mathrm{TJ}[I']$
13 $\mathrm{Gm}_1$:
14    Return $((I = I' > 0)$ and $(c_I \neq c'_I))$
15 $\mathrm{Gm}_2$:
16    If $((I \neq I')$ or $(c_I = c'_I))$ then
17       Return $\perp$
18    $w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
19    Return $(g^w = T_j)$

NWTAR$(S)$:
20 $j \leftarrow j+1$ ; $S_j \leftarrow S$ ; $T_j \leftarrow S_j \cdot X^{e_j}$
21 Return $e_j$

$\mathrm{ChSim}_i(j_{\mathrm{sel}}, R)$: // $i \in \{1,2\}$
22 $i \leftarrow i+1$ ; $R_i \leftarrow R$
23 $Y_i \leftarrow T_{j_{\mathrm{sel}}}$ ; $\mathrm{TJ}[i] \leftarrow j_{\mathrm{sel}}$
24 $\mathrm{ChSim}_1$ : Return $c_i$
25 $\mathrm{ChSim}_2$ : Return $c'_i$
```

```
Adversary $\mathcal{A}_{\mathrm{idl}}^{\mathrm{Ch}}(X)$:
1  $c_1, \ldots, c_{q_2} \leftarrow\$ \mathbb{Z}_p$ ; $\rho \leftarrow\$ \mathrm{rand}(\mathcal{A}_{\mathrm{xidl}})$
2  $(I, z) \leftarrow\$ \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar}_1,\mathrm{ChSim}_1}(X; \rho)$
3  $b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
4  If not $b$ then Return $\perp$
5  For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
6  $i, j \leftarrow 0$ ; $c'_I, c'_{I+1}, \ldots, c'_q \leftarrow\$ \mathbb{Z}_p$
7  $(I', z') \leftarrow\$ \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar}_2,\mathrm{ChSim}_2}(X; \rho)$
8  $b' \leftarrow (g^{z'} = R_{I'} \cdot Y_{I'}^{c_{I'}})$
9  If not $b'$ then Return $\perp$
10 If $((I \neq I')$ or $(c_I = c'_I))$ then
11    Return $\perp$
12 $j \leftarrow \mathrm{TJ}[I]$ ; $j' \leftarrow \mathrm{TJ}[I']$
13 $w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
14 Return $(j, w)$

$\mathrm{NwTar}_1(S)$:
15 $j \leftarrow j+1$ ; $e_j \leftarrow\$ \mathrm{Ch}(S)$ ; $S_j \leftarrow S$
16 $T_j \leftarrow S_j \cdot X^{e_j}$ ; Return $e_j$

$\mathrm{NwTar}_2(S)$:
17 $j \leftarrow j+1$
18 If not $e_j$ then $e_j \leftarrow \mathrm{Ch}(S)$
19 Return $e_j$

$\mathrm{ChSim}_i(j_{\mathrm{sel}}, R)$ // $i \in \{1,2\}$:
20 $i \leftarrow i+1$ ; $R_i \leftarrow R$
21 $Y_i \leftarrow T_{j_{\mathrm{sel}}}$ ; $\mathrm{TJ}[i] \leftarrow j_{\mathrm{sel}}$
22 $\mathrm{ChSim}_1$ : Return $c_i$
23 $\mathrm{ChSim}_2$ : Return $c'_i$
```

Figure 14: Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$ and adversary $\mathcal{A}_{\mathrm{idl}}$ for proof of Theorem 3.4.

where $I = I' > 0$. Notice that we also have $R_I = R_{I'}$, this is because the two runs of $\mathcal{A}_{\mathrm{xidl}}$ has not diverged when $R_I$ and $R_{I'}$ are supplied (since the first different value of $c_{\rceil iforge'}$ is only supplied afte $R_{i'}$ is given). Via simila reasoning, $Y_I = Y_{I'} = T_J$. Hence, putting the two equation toether, we have

$$Y_i^{c_i - c_{i'}} = g^{z - z'} ,$$

which implies the the computed value of $w$ (line 18) is the correct discrete log of $T_J$ base $g$. As a result, $\mathrm{Gm}_2$ must return true as well, and

$$\Pr[\mathrm{Gm}_2] \geq \Pr[\mathrm{Gm}_1] . \tag{38}$$

Finally, we construct adversary $\mathcal{A}_{\mathrm{idl}}$, given in Fig. 14, such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q_1}(\mathcal{A}_{\mathrm{idl}}) . \tag{39}$$

Crucially, in the construction of $\mathcal{A}_{\mathrm{idl}}$, NwTar oracle need to be simulated differently for the two runs of $\mathcal{A}_{\mathrm{xidl}}$. In the first run, the oracle $\textsc{NwTar}_1$ forwards the queries to Ch (that is given to our reduction adversary from the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$), while recording the responses $e_1, \ldots, e_j$. Then, in the second run, the oracle $\textsc{NwTar}_2$ will return previously recorded values of $e_1, \ldots, e_j$ as long as they are available, and only starts to forward queries when it runs out of previously recorded ones. This is to simulate the behavior of $\mathrm{Gm}_2$, where there is one single fixed sequence of values $e_1, \ldots, e_{q_1}$, used by the oracle NwTar. Putting the above equations together, we obtain the claim in the theorem. ▌

## G Proof of Theorem 5.1

**Proof of Theorem 5.1:** The proof uses a game sequence. Our games will implement $\mathrm{H}_0, \mathrm{H}_1$ with lazy sampling, maintaining tables $\mathrm{HF}_0, \mathrm{HF}_1$ for this purpose. They will provide oracles $\textsc{Sign}_1, \textsc{Sign}_2$ for the first two rounds, but omit $\textsc{Sign}_3$, since this round returns to the adversary only a quantity it could itself compute already. In Fin (for example Figure 15) we assume the query is non-trivial, meaning lines 6,7 of Figure 5 return true, and these lines are thus omitted. We start with games $\mathrm{Gm}_0, \mathrm{Gm}_1$ in Figure 15. Game $\mathrm{Gm}_0$ includes the boxed code, and we claim that

$$\mathbf{Adv}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \ . \tag{40}$$

Let us explain. We wish to move to a game where signing queries are answered without using the secret key *sk*. Naturally, we expect, for this, to use the zero-knowledge property of the Schnorr scheme. But certain obstacles must be removed before we can do this, and this will take a few steps. The first obstacle we address is that the BN-commitment $t_{u,k} = \mathrm{H}_0((k, R_{u,k}))$ may leak information about $R_{u,k}$. Rather than define $t_{u,k}$ in this way, games $\mathrm{Gm}_0, \mathrm{Gm}_1$ accordingly pick it at random at line 3. The reason for the boxed code at line 4 is that, under the "true" assignment $t_{u,k} = \mathrm{H}_0((k, R_{u,k}))$, having $R_{u,k_u} = R_{u',k_{u'}}$ would imply $t_{u,k_u} = t_{u',k_{u'}}$. At line 8, now that the BN-commitments $\boldsymbol{t}$ of all players are known, the games ensure that $t_{u,k}$ indeed equals $\mathrm{H}_0((k, R_{u,k}))$. This is consistent with the real game only if the hash function was not already defined at this point, captured by setting bad at line 17. The boolean CommitStage ensures that bad is only set prior to the release of $R_{s,k}$, since the adversary can set it with probability one if it knows $R_{s,k}$. This justifies Eq. (40).

Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [5]

$$\Pr[\mathrm{Gm}_0(\mathcal{A})] \leq \Pr[\mathrm{Gm}_1(\mathcal{A})] + \Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \ .$$

The probability of setting bad at line 4 is at most $(0 + 1 + \cdots + q_{\mathrm{s}} - 1)/p$, and the probabilities of setting bad at line 5 and line 17 are at most $q_{\mathrm{s}} q_0/p$, so

$$\Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_{\mathrm{s}}(q_{\mathrm{s}} - 1)}{2p} + \frac{2 q_{\mathrm{s}} q_0}{p} = \frac{q_{\mathrm{s}}(4q_0 + q_{\mathrm{s}} - 1)}{2p} \ .$$

Game $\mathrm{Gm}_2$ changes the $\mathrm{NS}, \textsc{Sign}_0, \mathrm{H}_0$ oracles as shown in Figure 16, maintaining the other oracles of $\mathrm{Gm}_1$ from Figure 15. It drops redundant code, which allows it to move the choice of $R_{s,k}$ to line 28. At line 40, it also introduces a table HI to maintain an inverse of the hash function, but does not use this. We have

$$\Pr[\mathrm{Gm}_1(\mathcal{A})] = \Pr[\mathrm{Gm}_2(\mathcal{A})] \ .$$

Game $\mathrm{Gm}_3$ (oracles shown across Figures 16 and 15) aims to figure out the $R_{s,j}$-values of parties $j \neq k$ before having to supply $R_{s,k}$, because we will later need these to program $\mathrm{H}_1$ values. It does

```
Init: // Games Gm_0–Gm_7
  $(pk, sk) \leftarrow\$ \mathsf{MS.Kg}$ ; Return $pk$

NS$(k, \boldsymbol{pk}, m)$: // Games $\boxed{\text{Gm}_0}$, Gm_1
  $u \leftarrow u + 1$ ; $k_u \leftarrow k$ ; $\boldsymbol{pk}[1] \leftarrow pk$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{pk}|$
  $\mathsf{CommitStage}_u \leftarrow \mathsf{true}$ ; $r_{u,k} \leftarrow\$ \mathbb{Z}_p$ ; $R_{u,k} \leftarrow g^{r_{u,k}}$ ; $t_{u,k} \leftarrow\$ \{0,1\}^\ell$
  If ( $\exists\, u' < u : R_{u,k_u} = R_{u',k_{u'}}$ ) then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{t_{u,k_u} \leftarrow t_{u',k_{u'}}}$
  If $(\mathrm{HF}_0[(k, R_{u,1})] \neq \bot)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{t_{u,k} \leftarrow \mathrm{HF}_0[(k, R_{u,k})]}$
  Return $t_{u,k}$

Sign_0$(s, \boldsymbol{t})$: // Games Gm_0, Gm_1
  $k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $\mathsf{CommitStage}_s \leftarrow \mathsf{false}$
  $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$ ; Return $R_{s,k}$

Sign_1$(s, \boldsymbol{R})$: // Games Gm_0, Gm_1, Gm_2
  $k \leftarrow k_s$ ; $\boldsymbol{R}[k] \leftarrow R_{s,k}$
  For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
  If ( $\exists\, i : y_i \neq \boldsymbol{t}_s[i]$ ) then Return $\bot$
  $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$ ; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$ ; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$
  Return $z_{s,k}$

H_0$(x)$: // Games $\boxed{\text{Gm}_0}$, Gm_1
  If $(\mathrm{HF}_0[x] \neq \bot)$ then Return $\mathrm{HF}_0[x]$
  $\mathrm{HF}_0[x] \leftarrow\$ \{0,1\}^\ell$
  If $(\exists\, u' : x = (k_{u'}, R_{u',k_{u'}})$ and $\mathsf{CommitStage}_{u'})$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{\mathrm{HF}_0[x] \leftarrow t_{u',k_{u'}}}$
  Return $\mathrm{HF}_0[x]$

H_1$(x)$: // Games Gm_0–Gm_7
  If $(\mathrm{HF}_1[x] \neq \bot)$ then Return $\mathrm{HF}_1[x]$
  $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

Fin$(\boldsymbol{pk}, m, (R, z))$: // Games Gm_0–Gm_7
  $n \leftarrow |\boldsymbol{pk}|$
  For $i = 1, \ldots, n$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$
  $X \leftarrow \prod_{i=1}^{n} \boldsymbol{pk}[i]^{c_i}$ ; Return $(g^z = RX)$
```

Figure 15: Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ for proof of Theorem 5.1. Some procedures will be included in later games, as indicated. A box around the name of a game following an oracle means the boxed code in that oracle is included in the game.

```
NS$(k, \boldsymbol{pk}, m)$: // Games $\mathrm{Gm}_2$–$\mathrm{Gm}_7$
24 $u \leftarrow u+1$ ; $k_u \leftarrow k$ ; $\boldsymbol{pk}[1] \leftarrow pk$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{pk}|$
25 $t_{u,1} \leftarrow\$ \{0,1\}^\ell$ ; Return $t_{u,1}$

SIGN$_0(s, \boldsymbol{t})$: // Game $\mathrm{Gm}_2$
26 $\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,1} \leftarrow\$ \mathbb{Z}_p$ ; $R_{s,1} \leftarrow g^{r_{s,1}}$ ; $\mathrm{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$
27 Return $R_{s,1}$

SIGN$_0(s, \boldsymbol{t})$: // Games $\mathrm{Gm}_3, \mathrm{Gm}_4$
28 $k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,k} \leftarrow\$ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{r_{s,1}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
29 For $i = 1, \ldots, n_s$ do
30     If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}^*_s[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$
31     Else $\boldsymbol{R}^*_s[i] \leftarrow\$ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}^*_s[i]))$
32 Return $R_{s,k}$

SIGN$_1(s, \boldsymbol{R})$: // Games $\mathrm{Gm}_3$, $\boxed{\mathrm{Gm}_4}$
33 $k \leftarrow k_s$ ; $\boldsymbol{R}[k] \leftarrow R_{s,k}$
34 For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
35 If ( $\exists\, i : y_i \neq \boldsymbol{t}_s[i]$ ) then Return $\perp$
36 If $(\boldsymbol{R} \neq \boldsymbol{R}^*_s)$ then bad $\leftarrow$ true ; $\boxed{\boldsymbol{R} \leftarrow \boldsymbol{R}^*_s}$
37 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$ ; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$ ; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$
38 Return $z_{s,k}$

$\mathrm{H}_0(x)$: // Games $\mathrm{Gm}_2$–$\mathrm{Gm}_7$
39 If $(\mathrm{HF}_0[x] \neq \perp)$ then Return $\mathrm{HF}_0[x]$
40 $\mathrm{HF}_0[x] \leftarrow\$ \{0,1\}^\ell$ ; $(i, R) \leftarrow x$ ; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$ ; Return $\mathrm{HF}_0[x]$
```

Figure 16: Games for proof of Theorem 5.1.

this by "inverting" the BN-commitments, meaning at line 30 it seeks inputs to $\mathrm{H}_0$ that result in the BN-commitments in $\boldsymbol{t}$. If these cannot be found, then random values are chosen instead at line 31. (Not finding the inverses is not yet a bad event. It can happen with high probability. It becomes a bad event only at line 36 when the BN-commitments are verified.) The computation of $t$ at that line is only to ensure that $\mathrm{H}_0$ has been called; this variable will not be used. These steps do not change what the oracles return compared to $\mathrm{Gm}_2$, so we have

$$\Pr[\mathrm{Gm}_2(\mathcal{A})] = \Pr[\mathrm{Gm}_3(\mathcal{A})] .$$

Moving to game $\mathrm{Gm}_4$, the change is only at line 36, which now includes the boxed code. The hope here is that the $\boldsymbol{R}^*_s$ obtained at lines 30,31 is correct with high probability. The boxed code ensures that in $\mathrm{Gm}_4$, it is always correct. Since $\mathrm{Gm}_3, \mathrm{Gm}_4$ are identical-until-bad we have

$$\Pr[\mathrm{Gm}_3(\mathcal{A})] \leq \Pr[\mathrm{Gm}_4(\mathcal{A})] + \Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets bad}] .$$

Line 36 can only set bad if $y_i = \boldsymbol{t}_s[i]$ for all $i$, due to line 35. So it is set only if there is a collision in $\mathrm{H}_0$-values, or no query hashing to $\boldsymbol{t}_s[i]$ was made prior to the latter being provided, but is made later. Thus

$$\Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets bad}] \leq \frac{q_0^2 + nq_0}{2^\ell} . \tag{41}$$

In game $\mathrm{Gm}_4$, the $\boldsymbol{R}$ queried to SIGN$_1$ is the same as the $\boldsymbol{R}^*$ determined in SIGN$_0$, allowing game

```
$\text{SIGN}_0(s, \boldsymbol{t})$: // Game $\text{Gm}_5$
41 $k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{r_{s,k}}$ ; $\text{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
42 For $i = 1, \ldots, n_s$ do
43     If $(\text{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}^*_s[i] \leftarrow \text{HI}_0[i, \boldsymbol{t}_s[i]]$
44     Else $\boldsymbol{R}^*_s[i] \leftarrow\!\!\$\ \mathbb{G}$ ; $t \leftarrow \text{H}_0((i, \boldsymbol{R}^*_s[i]))$
45 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}^*_s[i]$ ; $c_{s,k} \leftarrow \text{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$ ; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$
46 Return $R_{s,k}$

$\text{SIGN}_1(s, \boldsymbol{R})$: // Game $\text{Gm}_5, \text{Gm}_6, \text{Gm}_7$
47 $k \leftarrow k_s$ ; $\boldsymbol{R}[k] \leftarrow R_{s,k}$
48 For $i = 1, \ldots, n_s$ do $y_i \leftarrow \text{H}_0((i, \boldsymbol{R}[i]))$
49 If ( $\exists\, i : y_i \neq \boldsymbol{t}_s[i]$ ) then Return $\perp$ else Return $z_{s,k}$

$\text{SIGN}_0(s, \boldsymbol{t})$: // Game $\boxed{\text{Gm}_6}$, $\text{Gm}_7$
50 $k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
51 $c_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $z_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$ ; $\text{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
52 For $i = 1, \ldots, n_s$ do
53     If $(\text{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}^*_s[i] \leftarrow \text{HI}_0[i, \boldsymbol{t}_s[i]]$
54     Else $\boldsymbol{R}^*_s[i] \leftarrow\!\!\$\ \mathbb{G}$ ; $t \leftarrow \text{H}_0((i, \boldsymbol{R}^*_s[i]))$
55 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}^*_s[i]$
56 If $(\text{HF}_1((k, R_s, \boldsymbol{pk}_s, m_s)) \neq \perp)$ then bad $\leftarrow$ true ; $\boxed{c_{s,k} \leftarrow \text{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)]}$
57 $\text{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)] \leftarrow c_{s,k}$ ; Return $R_{s,k}$
```

Figure 17: Games for proof of Theorem 5.1.

---

$\text{Gm}_5$ (Figure 17) to move line 37 into $\text{SIGN}_0$ as line 45 and to simplify $\text{SIGN}_1$. We have

$$\Pr[\text{Gm}_4(\mathcal{A})] = \Pr[\text{Gm}_5(\mathcal{A})] \; .$$

Now that $R_s$ is determined prior to the release of $R_{s,k_s}$, it becomes possible to successfully program $\text{H}_1$ via the zero-knowledge simulation. Game $\text{Gm}_6$ of Figure 17 does this, setting bad at line 56 if the programming was precluded by the hash value already being defined, and including the boxed code to correct. We have

$$\Pr[\text{Gm}_5(\mathcal{A})] = \Pr[\text{Gm}_6(\mathcal{A})] \; .$$

Games $\text{Gm}_6, \text{Gm}_7$ (Figure 17) are identical-until-bad, so

$$\Pr[\text{Gm}_6(\mathcal{A})] \leq \Pr[\text{Gm}_7(\mathcal{A})] + \Pr[\text{Gm}_7(\mathcal{A}) \text{ sets bad}] \; . \tag{42}$$

When line 56 is executed, the adversary has as yet no information about $R_s$, which means

$$\Pr[\text{Gm}_7(\mathcal{A}) \text{ sets bad}] \leq \frac{q_s q_1}{p} \; . \tag{43}$$

We now build an adversary $\mathcal{A}_{\text{idl}}$ so that

$$\mathbf{Adv}^{\text{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\text{idl}}) \geq \Pr[\text{Gm}_7(\mathcal{A}_{\text{ms}})] \; . \tag{44}$$

We specify $\mathcal{A}_{\text{idl}}$ in Figure 18. It forwards the public key $pk$ to $\mathcal{A}_{\text{ms}}$. Simulating signatures without knowing the secret key, as $\mathcal{A}_{\text{idl}}$ needs to do, is now easy because the oracles of games $\text{Gm}_7$ already did this, and $\mathcal{A}_{\text{idl}}$ can just use the same code. Line 17 to 19 programs the challenge $c_k$ of the target public key by first deriving commitment $R_k$, which is then submitted to CH to derive $c_k$. Since $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q}$ game also samples the challenge uniformly at random, this does not change the behavior

```
Adversary $\mathcal{A}_{\mathrm{idl}}^{\mathrm{CH}}(pk)$:
$(\boldsymbol{pk}, m, (R, z)) \leftarrow\$ \mathcal{A}^{\mathrm{NS},\textsc{Sign}_0,\textsc{Sign}_1,\mathrm{H}_0,\mathrm{H}_1}(pk)$ ; Return $(\mathrm{TJ}[R], z)$
NS$(k, \boldsymbol{pk}, m)$:
$u \leftarrow u+1$ ; $k_u \leftarrow k$ ; $\boldsymbol{pk}[1] \leftarrow pk$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{pk}|$
$t_{u,k} \leftarrow\$ \{0,1\}^{\ell}$ ; Return $t_{u,k}$
$\textsc{Sign}_0(s, \boldsymbol{t})$:
$k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
$c_{s,k} \leftarrow\$ \mathbb{Z}_p$ ; $z_{s,k} \leftarrow\$ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
For $i = 1, \ldots, n_s$ do
    If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \bot)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$
    Else $\boldsymbol{R}_s^*[i] \leftarrow\$ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$ ; $\mathrm{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)] \leftarrow c_{s,k}$ ; Return $R_{s,k}$
$\textsc{Sign}_1(s, \boldsymbol{R})$:
$k \leftarrow k_s$ ; $\boldsymbol{R}[1] \leftarrow R_{s,k}$
For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
If $( \exists\, i : y_i \neq \boldsymbol{t}_s[i] )$ then Return $\bot$ else Return $z_{s,k}$
$\mathrm{H}_0(x)$:
If $(\mathrm{HF}_0[x] \neq \bot)$ then Return $\mathrm{HF}_0[x]$
$\mathrm{HF}_0[x] \leftarrow\$ \{0,1\}^{\ell}$ ; $(i, R) \leftarrow x$ ; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$ ; Return $\mathrm{HF}_0[x]$
$\mathrm{H}_1(x)$:
If $(\mathrm{HF}_1[x] \neq \bot)$ then Return $\mathrm{HF}_1[x]$
$(k, R, \boldsymbol{pk}, m) \leftarrow x$ ; $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$
If $(\boldsymbol{pk}[k] = pk)$ then
    $j \leftarrow j+1$ ; For $i = 2, \ldots, |\boldsymbol{pk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$
    $R_{j,k} \leftarrow R \cdot \prod_{i \neq k} \boldsymbol{pk}[i]^{c_i}$ ; $\mathrm{HF}_1[x] \leftarrow c_k \leftarrow \textsc{Ch}(R_{j,k})$ ; $\mathrm{TJ}[R] \leftarrow j$
Return $\mathrm{HF}_1[x]$
```

Figure 18: Adversary $\mathcal{A}_{\mathrm{idl}}$ for Theorem 5.1.

of $\mathrm{H}_1$. However, if a forgery $(\boldsymbol{pk}, m, (R, z))$, then it must be that

$$g^z = R \cdot \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_1(i,R,\boldsymbol{pk},m)} = R_{j,k} \cdot pk^{c_{j,k}} \ .$$

So $\mathcal{A}_{\mathrm{idl}}$ wins game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$. Eq. (3) is obtained by putting the above all together. ▮

## H Proof of Theorem 6.1

Let $\mathbb{G}$ be a group of prime order $p$ with generator $g$. Let $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ be the associated MuSig multi-signature scheme. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 5. We shall fix these quantities for the rest of the proof. The first lemma relates the advantage of $\mathcal{A}_{\mathrm{ms}}$ against $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ to a simplied game $\mathrm{Gm}_{\mathrm{simp}}$ (given in Fig. 19).

```
Init:
$(pk, sk) \leftarrow\!\!\$\ \mathsf{MS.Kg}$ ; Return $pk$

NS$(k, \boldsymbol{pk}, m)$:
$u \leftarrow u+1$ ; $k_u \leftarrow k$ ; $\boldsymbol{pk}[1] \leftarrow pk$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{pk}|$
$t_{u,1} \leftarrow\!\!\$\ \{0,1\}^\ell$ ; Return $t_{u,1}$

Sign$_1(s, \boldsymbol{R})$:
$k \leftarrow k_s$ ; $\boldsymbol{R}[1] \leftarrow R_{s,k}$
For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
If ( $\exists\, i : y_i \neq \boldsymbol{t}_s[i]$ ) then Return $\perp$ else Return $z_{s,k}$

Sign$_0(s, \boldsymbol{t})$:
$k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
$c_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $z_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
For $i = 1, \ldots, n_s$ do
    If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}^*_s[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$
    Else $\boldsymbol{R}^*_s[i] \leftarrow\!\!\$\ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}^*_s[i]))$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}^*_s[i]$
$\mathrm{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)] \leftarrow c_{s,k}$ ; Return $R_{s,k}$

H$_0(x)$:
If $(\mathrm{HF}_0[x] \neq \perp)$ then Return $\mathrm{HF}_0[x]$
$\mathrm{HF}_0[x] \leftarrow\!\!\$\ \{0,1\}^\ell$ ; $(i, R) \leftarrow x$ ; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$ ; Return $\mathrm{HF}_0[x]$

H$_1(x)$:
If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$
$(R, apk, m) \leftarrow x$ ; $\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$
$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

H$_2(x)$:
If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$
$(k, \boldsymbol{pk}) \leftarrow x$ ; For $i = 1, \ldots, |\boldsymbol{pk}|$ do $\mathrm{HF}_2[(i, \boldsymbol{pk})] \leftarrow e_i \leftarrow\!\!\$\ \mathbb{Z}_p$
$apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i}$ ; For $y \in \mathrm{TV}[apk]$ do $\mathrm{HF}_1[y] \leftarrow \perp$
Return $\mathrm{HF}_2[x]$

Fin$(\boldsymbol{pk}, m, (R, z))$:
For $i = 1, \ldots, |\boldsymbol{pk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$ ; $e_i \leftarrow \mathrm{H}_2((i, \boldsymbol{pk}))$
$X \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i \cdot c_i}$ ; Return $(g^z = RX)$
```

Figure 19: Game $\mathrm{Gm}_{\mathrm{simp}}$ for proof of Theorem 6.1.

---

**Lemma H.1** *Assume the execution of game* $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ *with* $\mathcal{A}_{\text{ms}}$ *has at most* $q_0, q_1, q_2, q_{\text{s}}$ *distinct queries to* $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$*, respectively, and the number of parties (length of verification-key vector) in queries to* NS *and* Fin *is at most* $n$*. Let* $\alpha = q_{\text{s}}(4q_0 + 2q_1 + q_{\text{s}}) + 2q_1q_2$ *and* $\beta = q_0(q_0 + n)$*. Then,*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\text{ms}}) \leq \Pr[\mathrm{Gm}_{\mathrm{simp}}(\mathcal{A}_{\text{ms}})] + \frac{\alpha}{2p} + \frac{\beta}{2^\ell} \ . \tag{45}$$

The second lemma constructs the reduction adversary against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$.

**Lemma H.2** *Assume the execution of game* $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ *with* $\mathcal{A}_{\text{ms}}$ *has at most* $q_0, q_1, q_2, q_{\text{s}}$ *distinct queries to* $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$*, respectively. We construct an adversary* $\mathcal{A}_{\text{xidl}}$ *for game* $\mathrm{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$ *(shown explicitly in Figure 24) such that*

$$\Pr[\mathrm{Gm}_{\text{simp}}(\mathcal{A}_{\text{ms}})] \leq \mathbf{Adv}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\text{xidl}}) \ . \tag{46}$$

**Proof of Lemma H.1:**

```
INIT: // Games $\mathrm{Gm}_0$–$\mathrm{Gm}_9$
  $(pk, sk) \leftarrow\$ \mathsf{MS.Kg}$ ; Return $pk$

NS$(k, \boldsymbol{pk}, m)$: // Games $\boxed{\mathrm{Gm}_0}$, $\mathrm{Gm}_1$
  $u \leftarrow u+1$ ; $k_u \leftarrow k$ ; $\boldsymbol{pk}[k] \leftarrow pk$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$
  $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{pk}|$ ; $\mathsf{CommitStage}_u \leftarrow \mathsf{true}$
  $r_{u,k} \leftarrow\$ \mathbb{Z}_p$ ; $R_{u,k} \leftarrow g^{r_{u,k}}$ ; $t_{u,k} \leftarrow\$ \{0,1\}^\ell$
  If ( $\exists u' < u : R_{u,k_u} = R_{u',k_{u'}}$ ) then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{t_{u,k_u} \leftarrow t_{u',k_{u'}}}$
  If $(\mathrm{HF}_0[(k, R_{u,k})] \neq \perp)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{t_{u,k} \leftarrow \mathrm{HF}_0[(k, R_{u,k})]}$
  Return $t_{u,k}$

SIGN$_0(s, \boldsymbol{t})$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1$
  $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $\mathsf{CommitStage}_s \leftarrow \mathsf{false}$
  $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$ ; Return $R_{s,k}$

SIGN$_1(s, \boldsymbol{R})$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$
  $\boldsymbol{R}[k] \leftarrow R_{s,k}$
  For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
  If ( $\exists i : y_i \neq \boldsymbol{t}_s[i]$ ) then Return $\perp$
  $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$ ; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$ ; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$
  Return $z_{s,k}$

H$_0(x)$: // Games $\boxed{\mathrm{Gm}_0}$, $\mathrm{Gm}_1$
  If $(\mathrm{HF}_0[x] \neq \perp)$ then Return $\mathrm{HF}_0[x]$
  $\mathrm{HF}_0[x] \leftarrow\$ \{0,1\}^\ell$ ; If $(\exists u' : x = (k_{u'}, R_{u',k_{u'}})$ and $\mathsf{CommitStage}_{u'})$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{\mathrm{HF}_0[x] \leftarrow t_{u',k_{u'}}}$
  Return $\mathrm{HF}_0[x]$

H$_1(x)$: // Games $\mathrm{Gm}_0$–$\mathrm{Gm}_7$
  If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$
  $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

H$_2(x)$: // Games $\mathrm{Gm}_0$–$\mathrm{Gm}_7$
  If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$
  $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

FIN$(k, \boldsymbol{pk}, m, (R, z))$: // Games $\mathrm{Gm}_0$–$\mathrm{Gm}_9$
  For $i = 1, \ldots, |\boldsymbol{pk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$ ; $e_i \leftarrow \mathrm{H}_2((i, \boldsymbol{pk}))$
  $X \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i \cdot c_i}$ ; Return $(g^z = RX)$
```

Figure 20: Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ for proof of Theorem 6.1. Some procedures will be included in later games, as indicated. A box around the name of a game following an oracle means the boxed code in that oracle is included in the game.

```
$\mathrm{NS}(k, \boldsymbol{pk}, m)$: // Games $\mathrm{Gm}_2$–$\mathrm{Gm}_9$
25 $u \leftarrow u+1$ ; $k_u \leftarrow k$ ; $\boldsymbol{pk}[.] \leftarrow pk$ ; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{pk}|$
26 $t_{u,k} \leftarrow\!\!\$ \{0,1\}^\ell$ ; Return $t_{u,k}$

$\textsc{Sign}_0(s, \boldsymbol{t})$: // Game $\mathrm{Gm}_2$
27 $k \leftarrow k_s$ ; $\boldsymbol{t}[.] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,k} \leftarrow\!\!\$ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{r_{s,k}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
28 Return $R_{s,k}$

$\textsc{Sign}_0(s, \boldsymbol{t})$: // Games $\mathrm{Gm}_3, \mathrm{Gm}_4$
29 $k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,k} \leftarrow\!\!\$ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{r_{s,k}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
30 For $i = 1, \ldots, n_s$ do
31     If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \bot)$ then $\boldsymbol{R}^*_s[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$
32     Else $\boldsymbol{R}^*_s[i] \leftarrow\!\!\$ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}^*_s[i]))$
33 Return $R_{s,k}$

$\textsc{Sign}_1(s, \boldsymbol{R})$: // Games $\mathrm{Gm}_3$, $\boxed{\mathrm{Gm}_4}$
34 $\boldsymbol{R}[k] \leftarrow R_{s,k}$
35 For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
36 If ( $\exists\, i : y_i \neq \boldsymbol{t}_s[i]$ ) then Return $\bot$
37 If $(\boldsymbol{R} \neq \boldsymbol{R}^*_s)$ then bad $\leftarrow$ true ; $\boxed{\boldsymbol{R} \leftarrow \boldsymbol{R}^*_s}$
38 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$ ; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$ ; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$
39 Return $z_{s,k}$

$\mathrm{H}_0(x)$: // Games $\mathrm{Gm}_2$–$\mathrm{Gm}_9$
40 If $(\mathrm{HF}_0[x] \neq \bot)$ then Return $\mathrm{HF}_0[x]$
41 $\mathrm{HF}_0[x] \leftarrow\!\!\$ \{0,1\}^\ell$ ; $(i, R) \leftarrow x$ ; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$ ; Return $\mathrm{HF}_0[x]$
```

Figure 21: Games for proof of Theorem 6.1.

The proof uses a game sequence. Our games will implement $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ with lazy sampling, maintaining tables $\mathrm{HF}_0, \mathrm{HF}_1, \mathrm{HF}_2$ for this purpose. They will provide oracles $\textsc{Sign}_0, \textsc{Sign}_1$ while omitting $\textsc{Sign}_2$, since this round returns to the adversary only a quantity it could itself compute already. In $\textsc{Fin}$ (for example Figure 20) we assume the query is non-trivial, meaning lines 6,7 of Figure 5 return true, and these lines are thus omitted. We start with games $\mathrm{Gm}_0, \mathrm{Gm}_1$ in Figure 20. Game $\mathrm{Gm}_0$ includes the boxed code, and we claim that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \,. \tag{47}$$

Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [5]

$$\Pr[\mathrm{Gm}_0(\mathcal{A})] \leq \Pr[\mathrm{Gm}_1(\mathcal{A})] + \Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets bad}] \,.$$

The probability of setting bad at line 4 is at most $(0 + 1 + \cdots + q_{\mathrm{s}} - 1)/p$, while the probabilities of setting it at line 5 and 15 are at most $q_{\mathrm{s}} q_0 / p$ so

$$\Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets bad}] \leq \frac{q_{\mathrm{s}}(q_{\mathrm{s}} - 1)}{2p} + 2 \cdot \frac{q_{\mathrm{s}} q_0}{p} = \frac{q_{\mathrm{s}}(4q_0 + q_{\mathrm{s}} - 1)}{2p} \,.$$

Game $\mathrm{Gm}_2$ changes the $\mathrm{NS}, \textsc{Sign}_0, \mathrm{H}_0$ oracles as shown in Figure 21, maintaining the other oracles of $\mathrm{Gm}_1$ from Figure 20. It drops redundant code, which allows it to move the choice of $R_{s,1}$ to line 29. At line 31, it also introduces a table HI to maintain an inverse of the hash function, but does not yet use this. We have

$$\Pr[\mathrm{Gm}_1(\mathcal{A})] = \Pr[\mathrm{Gm}_2(\mathcal{A})] \,.$$

Game $\mathrm{Gm}_3$ (oracles shown across Figures 21 and 20) aims to figure out the $R_{s,j}$-values of parties $j \neq k$ before having to supply $R_{s,k}$, because we will later need these to program $\mathrm{H}_1$ values. It does this by "inverting" the BN-commitments, meaning at line 27 it seeks inputs to $\mathrm{H}_0$ that result in the BN-commitments in $\boldsymbol{t}$. If these cannot be found, then random values are chosen instead at line 37. (Not finding the inverses is not yet a bad event. It can happen with high probability. It becomes a bad event only at line 37 when the BN-commitments are verified.) The computation of $t$ at that line is only to ensure that $\mathrm{H}_0$ has been called; this variable will not be used. These steps do not change what the oracles return compared to $\mathrm{Gm}_2$, so we have

$$\Pr[\mathrm{Gm}_2(\mathcal{A})] = \Pr[\mathrm{Gm}_3(\mathcal{A})] \,.$$

Moving to game $\mathrm{Gm}_4$, the change is only at line 33, which now includes the boxed code. The hope here is that the $\boldsymbol{R}_s^*$ obtained at lines 32,33 is correct with high probability. The boxed code ensures that in $\mathrm{Gm}_4$, it is always correct. Since $\mathrm{Gm}_3, \mathrm{Gm}_4$ are identical-until-$\mathsf{bad}$ we have

$$\Pr[\mathrm{Gm}_3(\mathcal{A})] \leq \Pr[\mathrm{Gm}_4(\mathcal{A})] + \Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets } \mathsf{bad}] \,.$$

Line 38 can only set $\mathsf{bad}$ if $y_i = \boldsymbol{t}_s[i]$ for all $i$, due to line 37. So it is set only if there is a collision in $\mathrm{H}_0$-values, or no query hashing to $\boldsymbol{t}_s[i]$ was made prior to the latter being provided, but is made later. Thus

$$\Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_0^2 + nq_0}{2^\ell} \,. \tag{48}$$

In game $\mathrm{Gm}_4$, the $\boldsymbol{R}$ queried to $\textsc{Sign}_1$ is the same as the $\boldsymbol{R}^*$ determined in $\textsc{Sign}_0$, allowing game $\mathrm{Gm}_5$ (Figure 22) to move line 38 into $\textsc{Sign}_0$ as line 46 and to simplify $\textsc{Sign}_1$. We have

$$\Pr[\mathrm{Gm}_4(\mathcal{A})] = \Pr[\mathrm{Gm}_5(\mathcal{A})] \,.$$

Now that $R_s$ is determined prior to the release of $R_{s,k_s}$, it becomes possible to successfully program $\mathrm{H}_1$ via the zero-knowledge simulation. Game $\mathrm{Gm}_6$ of Figure 22 does this, setting $\mathsf{bad}$ at line 57 if the programming was precluded by the hash value already being defined, and including the boxed code to correct. We have

$$\Pr[\mathrm{Gm}_5(\mathcal{A})] = \Pr[\mathrm{Gm}_6(\mathcal{A})] \,.$$

Games $\mathrm{Gm}_6, \mathrm{Gm}_7$ (Figure 22) are identical-until-$\mathsf{bad}$, so

$$\Pr[\mathrm{Gm}_6(\mathcal{A})] \leq \Pr[\mathrm{Gm}_7(\mathcal{A})] + \Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets } \mathsf{bad}] \,. \tag{49}$$

When line 57 is executed, the adversary has as yet no information about $R_s$, which means

$$\Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_{\mathrm{s}} q_1}{p} \,. \tag{50}$$

Moving on, let us consider games $\mathrm{Gm}_8$ and $\mathrm{Gm}_9$ in Fig. 23, which differ from $\mathrm{Gm}_7$ in modifications to oracles $\mathrm{H}_1$ and $\mathrm{H}_2$. Oracle $\mathrm{H}_1$ now keeps track of a table TV, that stores for each aggregate key $apk$ the set of $\mathrm{H}_1$ queries that contain it. It otherwise behave identically to $\mathrm{Gm}_7.\mathrm{H}_1$. Oracle $\mathrm{Gm}_8.\mathrm{H}_2$ does not contain the boxed code, which makes the oracle behave identically to $\mathrm{Gm}_7.\mathrm{H}_2$. So, we have

$$\Pr[\mathrm{Gm}_7(\mathcal{A})] = \Pr[\mathrm{Gm}_8(\mathcal{A})] \,. \tag{51}$$

By construction, $\mathrm{Gm}_7$ and $\mathrm{Gm}_8$ are identical-until-bad, hence

$$\Pr[\mathrm{Gm}_8(\mathcal{A})] \leq \Pr[\mathrm{Gm}_9(\mathcal{A})] + \Pr[\mathrm{Gm}_8 \text{ sets } \mathsf{bad}] \tag{52}$$

```
$\textsc{Sign}_0(s, \boldsymbol{t})$: // Game $\mathrm{Gm}_5$
$k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,k} \leftarrow\!\$\ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{r_{s,k}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
For $i = 1, \ldots, n_s$ do
    If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \bot)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$
    Else $\boldsymbol{R}_s^*[i] \leftarrow\!\$\ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$ ; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$ ; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$
Return $R_{s,k}$

$\textsc{Sign}_1(s, \boldsymbol{R})$: // Game $\mathrm{Gm}_5$–$\mathrm{Gm}_9$
$k \leftarrow k_s$ ; $\boldsymbol{R}[k] \leftarrow R_{s,k}$
For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$
If $(\exists\, i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\bot$ else Return $z_{s,k}$

$\textsc{Sign}_0(s, \boldsymbol{t})$: // Game $\boxed{\mathrm{Gm}_6}$, $\mathrm{Gm}_7$–$\mathrm{Gm}_9$
$k \leftarrow k_s$ ; $\boldsymbol{t}[k] \leftarrow t_{s,k}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
$c_{s,k} \leftarrow\!\$\ \mathbb{Z}_p$ ; $z_{s,k} \leftarrow\!\$\ \mathbb{Z}_p$ ; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$ ; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$
For $i = 1, \ldots, n_s$ do
    If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \bot)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$
    Else $\boldsymbol{R}_s^*[i] \leftarrow\!\$\ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$
If $(\mathrm{HF}_1((k, R_s, \boldsymbol{pk}_s, m_s)) \neq \bot)$ then bad $\leftarrow$ true ; $\boxed{c_{s,k} \leftarrow \mathrm{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)]}$
$\mathrm{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)] \leftarrow c_{s,k}$ ; Return $R_{s,k}$
```

Figure 22: Games for proof of Theorem 6.1.

```
$\mathrm{H}_1(x)$: // Game $\mathrm{Gm}_8, \mathrm{Gm}_9$
If $(\mathrm{HF}_1[x] \neq \bot)$ then Return $\mathrm{HF}_1[x]$
$(R, apk, m) \leftarrow x$ ; $\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$
$\mathrm{HF}_1[x] \leftarrow\!\$\ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$: // Game $\mathrm{Gm}_8$, $\boxed{\mathrm{Gm}_9}$
If $(\mathrm{HF}_2[x] \neq \bot)$ then Return $\mathrm{HF}_2[x]$
$(\cdot, \boldsymbol{pk}) \leftarrow x$ ; For $i = 1, \ldots, |\boldsymbol{pk}|$ do $\mathrm{HF}_2[(i, \boldsymbol{pk})] \leftarrow e_i \leftarrow\!\$\ \mathbb{Z}_p$
$apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i}$
If $\mathrm{TV}[apk] \neq \bot$ then
    bad $\leftarrow$ true ; $\boxed{\text{For } y \in \mathrm{TV}[apk] \text{ do } \mathrm{HF}_1[y] \leftarrow \bot}$
Return $\mathrm{HF}_2[x]$
```

Figure 23: Games for proof of Theorem 6.1.

$$\leq \Pr[\mathrm{Gm}_9(\mathcal{A})] + \frac{q_1 q_2}{p}\ , \tag{53}$$

where the last inequality is by the fact that each $\mathrm{H}_2$ query has probability at most $q_1/p$ of setting bad. Lastly, we note that $\mathrm{Gm}_9$ and $\mathrm{Gm}_{\mathrm{simp}}$ are identical. This completes the proof of Lemma H.1. ▌

**Proof of Lemma H.2:** Consider $\mathcal{A}_{\mathrm{xidl}}$ in Figure 24. It forwards the public key $pk$ to $\mathcal{A}_{\mathrm{ms}}$. Simulating signatures without knowing the secret key can be done exactly as $\mathrm{Gm}_{\mathrm{simp}}$. To break

```
Adversary $\mathcal{A}_{\text{xidl}}^{\text{Ch}}(pk)$:
 $(\boldsymbol{pk}, m, (R, z)) \leftarrow\!\!\$\ \mathcal{A}^{\text{NS},\text{Sign}_0,\text{Sign}_1,\text{H}_0,\text{H}_1,\text{H}_2}(pk)$
 $apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\text{H}_2((i,\boldsymbol{pk}))}$ ; Return $(\text{TI}[(apk, R, m)], z)$
$\text{H}_1(x)$:
 If $(\text{HF}_1[x] \neq \bot)$ then Return $\text{HF}_1[x]$
 $(R, apk, m) \leftarrow x$ ; $\text{TV}[apk] \leftarrow \text{TV} \cup \{x\}$
 If $(\text{TJ}[apk] = \bot)$ then Return $\text{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$
 $\iota \leftarrow \iota + 1$ ; $\text{TI}[x] \leftarrow \iota$
 $\text{HF}_1[x] \leftarrow c_\iota \leftarrow\!\!\$\ \text{Ch}(\text{TJ}[apk], R)$ ; Return $\text{HF}_1[x]$

$\text{H}_2(x)$:
 If $(\text{HF}_2[x] \neq \bot)$ then Return $\text{HF}_2[x]$
 $(\cdot, \boldsymbol{pk}) \leftarrow x$ ; If $(pk \notin \boldsymbol{pk})$ then Return $\text{HF}_2[x] \leftarrow\!\!\$\ \mathbb{Z}_p$
 $j \leftarrow j + 1$ ; $k \leftarrow \mathsf{minInd}(pk, \boldsymbol{pk})$ ; If $(x \neq (k, \boldsymbol{pk}))$ then Return $\text{HF}_2[x] \leftarrow\!\!\$\ \mathbb{Z}_p$
 $S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\text{H}_2((i,\boldsymbol{pk}))}$
 $\text{HF}_2[x] \leftarrow e_j \leftarrow \text{NwTar}(S)$ ; $apk \leftarrow S \cdot pk^{e_j}$ ; $\text{TJ}[apk] \leftarrow j$
 For $y \in \text{TV}[apk]$ do $\text{HF}_1[y] \leftarrow \bot$
 Return $\text{HF}_2[x]$
```

Figure 24: Adversary $\mathcal{A}_{\text{xidl}}$ for Theorem 6.1. Oracles $\text{NS}, \text{Sign}_0, \text{Sign}_1, \text{H}_0$ are copied from game $\text{Gm}_{\text{simp}}$ (Fig. 19).

---

$\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$, our adversary $\mathcal{A}_{\text{xidl}}$ needs to program $\text{H}_1$ and $\text{H}_2$. For each $\text{H}_2$ query, Line 10 to 12 programs the response $e_j$ for the target public key by first deriving commitment $S = \prod_{i \neq k} \boldsymbol{pk}[i]^{e_i}$, which is then submitted to NwTar to derive $e_k$ that is returned as the response. By construction, the corresponding aggregate public key $apk = S \cdot pk^{e_k}$ is exactly the target $T_j$ recorded by $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$ for this NwTar query. For each $\text{H}_1$ query, our adversary first uses the aggregate public key $apk$ find the corresponding $\text{H}_2$ query via table TJ. If possible, then the adversary proceeds to program in a challenge using the challenge oracle Ch of XIDL. If this is not possible, the advesary simply simulates $\text{H}_1$ honestly. If a forgery $(\boldsymbol{pk}, m, (R, z))$ is valid, then it must be that

$$g^z = R \cdot \prod_{i=1}^{|\boldsymbol{pk}|} apk^{\text{H}_1((R,apk,m))} \ ,$$

where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\text{H}_2((i,\boldsymbol{pk}))}$. Observe that call involving a fresh vector $\boldsymbol{pk}$ to oracle $\text{H}_2$ erases the table $\text{HF}_1$ at every entry associated with the derived $apk$. Hence, our adversary can use the above relation to directly break XIDL. In other words, the value of $z$ included in the forgery makes the following equation true in game $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$, $g^z = R \cdot T_j^{c_i}$, where $j = \text{TJ}[apk]$ and $i = \text{TI}[(R, apk, m)]$. This justifies Equation (46). ▮

## I Proof of Theorem 7.1

The first step in the proof is to move from the security game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ to a game where the signing oracles can be simulated without the target secret key. We encapsulate this in the lemma below, which works strictly in the standard model, meaning it does not require adversaries involved to be algebraic. This allows our latter standard model proof of security for HBMS to also rely on this lemma.

**Lemma I.1** *Let* $\mathbb{G}$ *be a group of prime order* $p$ *with generator* $g$*. Let* $\mathsf{MS} = \mathsf{HBMS}[\mathbb{G}, g]$ *be the scheme specified in Fig. 9. Let* $\mathcal{A}_{\mathrm{ms}}$ *be an adversary for game* $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ *of Fig. 5. Assume the execution of game* $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ *with* $\mathcal{A}_{\mathrm{ms}}$ *has at most* $q_0, q_1, q_2$ *distinct queries to* $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ *respectively. Let* $\rho \in [0, 1]$ *be a real number. Consider games* $\mathrm{Gm}_0$ *and* $\mathrm{Gm}_{1,\rho}$ *give in Fig. 25. Then,*

$$\begin{aligned}\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) &= \Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})] &(54)\\ &= \Pr[\mathrm{Gm}_{1,\rho}(\mathcal{A}_{\mathrm{ms}}) \mid \mathrm{Gm}_{1,\rho}(\mathcal{A}_{\mathrm{ms}}) \textit{ does not abort }] \,. &(55)\end{aligned}$$

*Moreover, the probability that game* $\mathrm{Gm}_1$ *does not abort is*

$$\Pr[\mathrm{Gm}_{1,\rho}(\mathcal{A}_{\mathrm{ms}}) \textit{ does not abort }] = \rho^{q_0} \;, \tag{56}$$

*which is* 1 *if* $\rho = 1$.

**Proof of of Lemma I.1:** Consider games $\mathrm{Gm}_0$ and $\mathrm{Gm}_{1,\rho}$ given in Fig. 25. Game $\mathrm{Gm}_0$ is simply a rewrite of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$, where $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ are lazily sampled. We fix the given adversary $\mathcal{A}_{\mathrm{ms}}$ for the rest of the proof and omit writing it in expression such as $\Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})]$ for simplicity. Game $\mathrm{Gm}_{1,\rho}$ is parameterized by a real number $\rho \in [0, 1]$, and changes the code of NS, $\textsc{Sign}_1$ and $\mathrm{H}_0$. The changes are made so that $\textsc{Sign}_1$ does not use the secret key $sk$, but will however preserve the output distribution of all oracles when it does not abort, as we will show below. In particular, for each $\mathrm{H}_0$ query, game $\mathrm{Gm}_1$ makes a guess, by flipping a biased coin $\mathsf{Coin}(\rho)$, which has probability $\rho$ of returning 1 and probability $1 - \rho$ of returning 0. If the coin flip returns 1, then we set the output of $\mathrm{H}_0(x)$ to be $g^{\beta_g} pk^{\beta_{pk}}$, otherwise we set the output of $\mathrm{H}_0(x)$ to be $g^{\beta_g}$. In either case, $\beta_g$ and $\beta_{pk}$ are uniformly chosen at random as per line 25.

Looking ahead, $\mathrm{Gm}_{1,\rho}$ will be able to simulate signatures for $\boldsymbol{pk}, m$ when $\mathrm{H}_0(\boldsymbol{pk}, m)$ is set to $g^{\beta_g} pk^{\beta_{pk}}$ (when the coin toss returns 1). In fact, $\rho$ is set to 1 in deriving the AGM result and the coin toss never returns 0. However, for the standard model result, we will need to make sure that the $\mathrm{H}_0$ query corresponding to the forgery $pk, m$ is programmed differently, namely that $\mathrm{H}_0((\boldsymbol{pk}, m)) = g^{\beta_g}$.

Game $\mathrm{Gm}_{1,\rho}$ could abort at line 16 (it is assumed that the adversary losses the game if $\mathrm{Gm}_1$ is aborted). By construction, we have

$$\Pr[\mathrm{Gm}_1 \text{ does not abort}] = \rho^{q_0} \;. \tag{57}$$

We claim that, for any value of $\rho$, if game $\mathrm{Gm}_1$ does not abort, then it is indistinguishable from $\mathrm{Gm}_0$ to the adversary. In particular, we claim

$$\Pr[\mathrm{Gm}_1 \mid \mathrm{Gm}_1 \text{ does not abort}] = \Pr[\mathrm{Gm}_0] \;. \tag{58}$$

Showing this amounts to showing that the outputs of $\textsc{Sign}_1$ oracle in either games are distributed identically. Observe that, in game $\mathrm{Gm}_0$, the return value $T_v$ of NS and $(s_v, z_v)$ of $\textsc{Sign}_1$ are uniformly distributed subjected to the constraint that

$$g^{z_v} \mathrm{H}_0((\boldsymbol{pk}_v, m))^{s_v} = T_{v,k} \cdot pk^{e_v c_v} \;.$$

We will show that this is also true in $\mathrm{Gm}_{1,\rho}$, namely that $\textsc{Sign}_0$ and $\textsc{Sign}_1$ in $\mathrm{Gm}_{1,\rho}$ also returns $T_{v,k}$ and $(s_v, z_v)$ that are uniformly distributed subjected to the above equation. In game $\mathrm{Gm}_{1,\rho}$, if $w = pk$ at line 15, then $h = \mathrm{H}_0((\boldsymbol{pk}_v, m)) = g^{\beta_g} pk^{\beta_{pk}}$, by construction of $\mathrm{H}_0$ (line 27). Hence, for a query $\textsc{Sign}_1(v, (T_{v,1}, \ldots, T_{v,n}))$ of game $\mathrm{Gm}_{1,\rho}$, it holds that

$$T_{v,k_v} \cdot pk^{e_v c_v} = g^{a_v} \cdot h^{b_v} \cdot pk^{e_v c_v} = g^{a_v} \cdot (g^{\beta_g} pk^{\beta_{pk}})^{b_v} \cdot pk^{e_v c_v}$$

```
Game $\mathrm{Gm}_0$, $\mathrm{Gm}_{1,\rho}$, $\mathrm{Gm}_{2,\rho}$

INIT:
1  $(pk, sk) \leftarrow\$ \mathsf{MS.Kg}$ ; Return $pk$

NS$(k, \boldsymbol{pk}, m)$:
2  $\boldsymbol{pk}[k] \leftarrow pk$ ; $u \leftarrow u + 1$
3  $k_u \leftarrow k$ ; $m_u \leftarrow m$
4  $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$ ; $h \leftarrow\$ \mathrm{H}_0((\boldsymbol{pk}, m))$
5  $apk_u \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
6  $a_u, b_u \leftarrow\$ \mathbb{Z}_p$ ; $T_{u,k} \leftarrow g^{a_u} h^{b_u}$
7  Return $T_{u,k}$

SIGN$_1(v, \mathbf{in})$:
8  $(T_{v,1}, \ldots, T_{v,n}) \leftarrow \mathbf{in}$ ; $T_v \leftarrow \prod_{i=1}^n T_{v,i}$
9  $c_v \leftarrow \mathrm{H}_1((T_v, apk_v, m_v))$
10 $e_v \leftarrow \mathrm{H}_2((k_v, \boldsymbol{pk}))$
11 $\underline{\mathrm{Gm}_0}$:
12     $z_v \leftarrow a_v + sk \cdot e_v \cdot c_v \mod p$
13     $s_v \leftarrow b_v$
14 $\underline{\mathrm{Gm}_{1,\rho}, \mathrm{Gm}_{2,\rho}}$:
15     $(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[(\boldsymbol{pk}_v, m_v)]$
16     If $(w \neq pk)$ then abort
17     $s_v \leftarrow b_v + e_v \cdot c_v \cdot \beta_{pk}^{-1} \mod p$
18     $z_v \leftarrow a_v + \beta_g \cdot b_v - \beta_g \cdot s_v \mod p$
19 Return $(s_v, z_v)$

SIGN$_2(v, \mathbf{in})$:
20 $(t_1, \ldots, t_n) \leftarrow \mathbf{in}$ ; $t \leftarrow \sum_i t_i$
21 $(s, z) \leftarrow t$ ; Return $(T_v, s, z)$

$\mathrm{H}_0(x)$: // $\mathrm{Gm}_0$
22 If $\mathrm{HF}_0[x] = \perp$ then $\mathrm{HF}_0[x] \leftarrow\$ \mathbb{G}$
23 Return $\mathrm{HF}_0[x]$

$\mathrm{H}_0(x)$: // $\mathrm{Gm}_{1,\rho}, \mathrm{Gm}_{2,\rho}$
24 If $\mathrm{HF}_0[x] \neq \perp$ then Return $\mathrm{HF}_0[x]$
25 $\beta_g \leftarrow\$ \mathbb{Z}_p$ ; $\beta_{pk} \leftarrow\$ \mathbb{Z}_p^*$
26 If $(\mathsf{Coin}(\rho) = 1)$ then
27     $\mathrm{HF}_0[x] \leftarrow g^{\beta_g} pk^{\beta_{pk}}$
28     $\mathrm{TH}[x] \leftarrow (pk, \beta_g, \beta_{pk})$
29 Else
30     $\mathrm{HF}_0[x] \leftarrow g^{\beta_g}$
31     $\mathrm{TH}[x] \leftarrow (g, \beta_g, \beta_{pk})$
32 Return $\mathrm{HF}_0[x]$

$\mathrm{H}_i(x)$: // $i \in \{1, 2\}$
33 If $(\mathrm{HF}_i[x] = \perp)$ then $\mathrm{HF}_i[x] \leftarrow\$ \mathbb{Z}_p$
34 Return $\mathrm{HF}_i[x]$

FIN$(\boldsymbol{pk}, m, (T, s, z))$:
35 If $(\boldsymbol{pk}[k] \neq pk)$ then return false
36 If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return false
37 $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$
38 $\underline{\mathrm{Gm}_{2,\rho}}$:
39     $(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$
40     If $(w \neq g)$ then abort
41 $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$
42 $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
43 $c \leftarrow \mathrm{H}_1((T, apk, m))$
44 Return $(g^z h^s = T \cdot apk^c)$
```

Figure 25: Games $\mathrm{Gm}_0$, $\mathrm{Gm}_{1,\rho}$, and $\mathrm{Gm}_{2,\rho}$, where $\rho \in [0, 1]$ is a real number, used in Lemma I.1 and proof of Theorem 7.2. Notation $\mathsf{Coin}(\rho)$ denotes flipping of a biased coin with probability $\rho$ of giving 1 and $1 - \rho$ of giving 0.

---

$$= g^{a_v + \beta_g \cdot b_v} \cdot pk^{\beta_{pk} \cdot b_v + e_v c_v} .$$

We claim that the above is also equal to $g^{z_v} \cdot h^{s_v}$. In fact, we set $z_v, s_v$ on line 17 and 18 exactly to make this true. To verify this, check that

$$g^{z_v} h^{s_v} = g^{a_v + \beta_g \cdot b_v - \beta_g \cdot s_v} (g^{\beta_g} pk^{\beta_{pk}})^{s_v} = g^{a_v + \beta_g \cdot b_v} pk^{\beta_{pk} \cdot s_v}$$
$$= g^{a_v + \beta_g \cdot b_v} \cdot pk^{\beta_{pk} \cdot b_v + e_v c_v} .$$

Additionally, notice that $s_v, z_v$ are both marginally uniform over $\mathbb{Z}_p$ by construction. This means the outputs of SIGN$_0$, SIGN$_1$ oracle from $\mathrm{Gm}_{1,\rho}$ has the same output distribution compared to that of $\mathrm{Gm}_0$. This justifies Equation (58). ▌

Equipped with Lemma I.1, we move on to prove Theorem 7.1. The proof constructs adversary $\mathcal{A}_{\mathrm{dl}}$ that simulates $\mathrm{Gm}_{1,1}$ (with $\rho$ set to 1).

```
Adversary $\mathcal{A}_{\mathrm{dl}}(X)$:
$pk \leftarrow X$ ; $(k, \boldsymbol{pk}, m, (T, s, z)) \leftarrow\!\!\$\; \mathcal{A}_{\mathrm{ms}}^{\mathrm{NS},\textsc{Sign}_1,\textsc{Sign}_2,\mathrm{H}_0,\mathrm{H}_1,\mathrm{H}_2}(pk)$
If $(\boldsymbol{pk}[k] \neq X)$ then return $\perp$
If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return $\perp$
If not $\mathsf{MS.Vf}^{\mathrm{H}_0,\mathrm{H}_1,\mathrm{H}_2}(\boldsymbol{pk}, m, \sigma)$ then return $\perp$
$(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$ ; $apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
$c \leftarrow \mathrm{H}_1((T, apk, m))$ ; For $i = 1, \ldots, |\boldsymbol{pk}|$ do $e_i \leftarrow \mathrm{H}_2((i, \boldsymbol{pk}))$
$\alpha_g \leftarrow z + \beta_g - \mathsf{Ext}(T, g) - c \cdot \sum_{i \neq k} \mathsf{Ext}(\boldsymbol{pk}[i], g) \cdot e_i$
$\alpha_X \leftarrow -s \cdot \beta_{pk} + \mathsf{Ext}(T, X) + c \cdot (e_k + \sum_{i \neq k} \mathsf{Ext}(\boldsymbol{pk}[i], X) \cdot e_i)$
If $(\alpha_X = 0)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $x' \leftarrow\!\!\$\; \mathbb{Z}_p$
Else $x' \leftarrow \alpha_g \alpha_X^{-1} \mod p$
Return $x'$
```

Figure 26: Adversary $\mathcal{A}_{\mathrm{dl}}$ for Theorem 7.1, oracles $\mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2, \mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ are implemented using the exact code as those in $\mathrm{Gm}_{1,1}$. Notation $\mathsf{Ext}(\cdot, g)$ and $\mathsf{Ext}(\cdot, X)$ are defined in the proof of Lemma 7.2. Computation of $\alpha_g$ and $\alpha_X$ are done modulo $p$.

---

**Proof of Theorem 7.1:** Consider the games $\mathrm{Gm}_0$ and $\mathrm{Gm}_{1,1}$ (with $\rho = 1$) in Fig. 25. We know that,

$$\Pr[\mathrm{Gm}_0] = \Pr[\mathrm{Gm}_{1,1} \mid \mathrm{Gm}_{1,1} \text{ does not abort}] \ .$$

Moreover,

$$\Pr[\mathrm{Gm}_{1,\rho} \text{ does not abort}] = \rho^{q_0} = 1,$$

when $\rho = 1$. Hence, game $\mathrm{Gm}_{1,1}$ never aborts and $\Pr[\mathrm{Gm}_0] = \Pr[\mathrm{Gm}_{1,1}]$ . We shall construct an adversary $\mathcal{A}_{\mathrm{dl}}$, using the fact that given adversary $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$ is algebraic, directly against game $\mathrm{Gm}_{\mathbb{G},g}^{\mathrm{dl}}$.

We first analyze the group elements involved in the inputs and outputs of oracles of $\mathrm{Gm}_{1,1}$. The $u$-th NS query takes in a list of group elements $\boldsymbol{pk}_u$. The $v$-th $\mathsf{Sign}_1$ query takes in a list of group elements $(T_{v,1}, \ldots, T_{v,n})$. The $i$-th $\mathrm{H}_2$ query take in a list of group elements $\boldsymbol{pk}_{\mathrm{H}_2,i}$. The $i$-th $\mathrm{H}_1$ query $(T, apk, m)$ takes in group elements $T_{\mathrm{H}_1,i}$ and $apk_{\mathrm{H}_1,i}$. Above are the exhaustive list of group elements that are given to $\mathrm{Gm}_{1,1}$, let us denote this list by **out**, since they are the output of the adversary. The initial query to $\textsc{Init}$ outputs a group element $pk$. The $u$-th NS query gives out a group element $T_{u,k_u}$. The $i$-th $\mathrm{H}_0$ query gives out a group element $h_i$. The last query to $\textsc{Fin}$ gives group elements $T$ (first component of the forged signature) and $pk$. Above (plus the group generator $g$) are the exhaustive list of group elements that are given out to the adversary $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$. Let us denote this list as **in**. Hence, the algebraic adversary $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$ gives, for each group element in the list **out**, a vector that is of dimension $|\mathbf{in}|$ which is a valid representation of the corresponding group element. Note that every group element in the list **in** is derived using only group operations on two group elements: $g$ and $pk$ (this is by the construction of game $\mathrm{Gm}_{1,1}$). As a result, every group element in the list **out** can be represent using $g$ and $pk$ only. For any $Y \in \mathbf{out}$, we use $\mathsf{Ext}(Y, g)$ and $\mathsf{Ext}(Y, pk)$ to denote this representation, i.e.

$$Y = g^{\mathsf{Ext}(Y,g)} \cdot pk^{\mathsf{Ext}(Y,pk)} \ .$$

We forego writing explicit code deriving these representations, with the understanding that they are well-defined and can be computed easily from the oracle queries of $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$. We will use this notation freely in simulations of $\mathrm{Gm}_{1,1}$.

We move on to giving adversary $\mathcal{A}_{\text{dl}}$, which simulates $\text{Gm}_{1,1}$ for $\mathcal{A}_{\text{ms}}^{\text{alg}}$. Our adversary $\mathcal{A}_{\text{dl}}$ is given in Fig. 26. Our adversary $\mathcal{A}_{\text{dl}}$ simulates oracles $\text{NS}, \text{SignStage}_1, \text{SignStage}_2, \text{H}_0, \text{H}_1$ exactly as $\text{Gm}_{1,1}$, hence their code are omitted. As stated above, since $\mathcal{A}_{\text{dl}}$ simulates $\text{Gm}_{1,1}$, the representation of any group element $Y \in \textbf{out}$ are available via scalars $\mathsf{Ext}(Y, g)$ and $\mathsf{Ext}(g, pk)$. Our adversary uses these scalars to compute the discrete log $x'$.

If $\mathcal{A}_{\text{ms}}^{\text{alg}}$ gives a valid forgery $(\boldsymbol{pk}, m, (T, s, z))$[1] then the verification equation says that

$$g^z \text{H}_0((\boldsymbol{pk}, m))^s = T \cdot apk^{\text{H}_1((T, apk, m))} ,$$

where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\text{H}_2((i, \boldsymbol{pk}))}$. Since every group element in the above equation can be represented using $g$ and $X$, one can solve for $\mathsf{DL}_{\mathbb{G},g}(X)$. Our adversary $\mathcal{A}_{\text{dl}}$ implements this intuition, computing value $\alpha_g$ and $\alpha_X$ (line 7 and 8) such that $g^{\alpha_g} = X^{\alpha_X}$. The only caveat is that $\alpha_X$ could be 0, in which case $\mathsf{DL}_{\mathbb{G},g}(X)$ cannot be solved for. When $\alpha_X = 0$ adversary $\mathcal{A}_{\text{dl}}$ sets bad, and we would like to upperbound the probability of this event. First, note that the view of adversary $\mathcal{A}_{\text{ms}}$ is independent of the value of $\beta_{pk}$. This is because the adversary is only given the value of $h = g^{\beta_g} pk^{\beta_{pk}}$. So, if the forgery is such that $s \neq 0$, then $\alpha_X = 0$ with probability at most $1/p$. If $s = 0$, then we need to make sure that $\mathsf{Ext}(T, X) + c \cdot (e_k + \sum_{i \neq k} \mathsf{Ext}(\boldsymbol{pk}[i], X) \cdot e_i)$ is not zero. We first bound the probability that there exists some query $\text{H}_2((\cdot, \boldsymbol{pk}'))$ (which defines the values of $e'_1, \ldots, e'_{|\boldsymbol{pk}'|}$) such that $e'_k + \sum_{i \neq k} \mathsf{Ext}(\boldsymbol{pk}'[i], X) \cdot e'_i = 0$ (call this quantity $\gamma_{\boldsymbol{pk}'}$). This happens with probability at most $q_2/p$. Suppose the above does not happen, then for each query $\text{H}_1((T', apk', m'))$ (which defines the value of $c'$), where $apk'$ is the aggregate key of some vector $\boldsymbol{pk}'$, the probability that $\mathsf{Ext}(T', X) + c' \cdot \gamma_{\boldsymbol{pk}'} = 0$ is at most $q_2/p$, accounting for at most $q_2$ non-zero values that $\gamma_{\boldsymbol{pk}'}$ could take. This results in an overall bad probability of $q_2/p + q_1 q_2/p = (q_1 + 1) q_2/p$. This justifies Equation (7). ▌

## J Proof of Theorem 7.2

**Proof of of Theorem 7.2:** We will start by considering $\text{Gm}_{1,\rho}$ given in Fig. 25. By Lemma I.1,

$$\mathbf{Adv}_{\mathsf{MS}}^{\text{ms-uf}}(\mathcal{A}_{\text{ms}}) = \Pr[\text{Gm}_{1,\rho}(\mathcal{A}_{\text{ms}}) \mid \text{Gm}_{1,\rho}(\mathcal{A}_{\text{ms}}) \text{ does not abort}] .$$

Towards construction of an adversary against XIDL, consider game $\text{Gm}_{2,\rho}$ (Fig. 25), differ from $\text{Gm}_{1,\rho}$ only at line 40–it aborts if the coin flip corresponding to the forgery target $(\boldsymbol{pk}, m)$ results in $w = g$. Marginally, $\text{Gm}_{2,\rho}$ does not abort at line 40 with probability $(1 - \rho)$. We need to lower bound the probability of $\text{Gm}_{2,\rho}$ not aborting overall, at either line 16 or line 40. Since there are overall $q_{\text{s}}$ *unique* queries to NS in the execution of $\text{Gm}_0$ with $\mathcal{A}_{\text{ms}}$, then the probability that $\text{Gm}_1$ does not abort is exactly

$$\Pr[\text{Gm}_2(\mathcal{A}_{\text{ms}}) \text{ does not abort}] = \rho^{q_{\text{s}}}(1 - \rho) .$$

Setting $\rho = (1 - (1 + q_{\text{s}})^{-1})$, we have that

$$\Pr[\text{Gm}_2(\mathcal{A}_{\text{ms}}) \text{ does not abort}] = (1 - (1 + q_{\text{s}})^{-1})^{q_{\text{s}}}(1 + q_{\text{s}})^{-1} \geq \frac{1}{e(1 + q_{\text{s}})} ,$$

where we applied the fact that $(1 - (1 + n)^{-1})^n \geq e^{-1}$ for positive $n$. Since game $\text{Gm}_2$ can only abort more often than $\text{Gm}_1$ and that the aborting at line 40 is an event independent of whether

[1]Note that for the fogery $\boldsymbol{pk}, m, (T, s, z)$ returend, the corresponding random oracles queries $\text{H}_0((\boldsymbol{pk}, m))$, $\text{H}_1((T, apk, m))$, and $\text{H}_2((i, \boldsymbol{pk}))$ are made in line 4 to 6, even if these points were previously unqueried during the execution of $\mathcal{A}_{\text{ms}}^{\text{alg}}$.

```
$\mathrm{H}_1(x)$: // Game $\mathrm{Gm}_3, \mathrm{Gm}_4$
If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$
$(T, apk, m) \leftarrow x$ ; $\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$
$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$: // Game $\mathrm{Gm}_3, \mathrm{Gm}_4$
If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$
$(\cdot, \boldsymbol{pk}) \leftarrow x$ ; For $i = 1, \ldots, |\boldsymbol{pk}|$ do $\mathrm{HF}_2[(i, \boldsymbol{pk})] \leftarrow e_i \leftarrow\!\!\$\ \mathbb{Z}_p$
$apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i}$
If $\mathrm{TV}[apk] \neq \perp$ then $\mathbf{BadSet} \leftarrow \mathbf{BadSet} \cup \mathrm{TV}[apk]$
Return $\mathrm{HF}_2[x]$

$\textsc{Fin}(\boldsymbol{pk}, m, (T, s, z))$: // Game $\mathrm{Gm}_3$, $\boxed{\mathrm{Gm}_4}$
If $(\boldsymbol{pk}[k] \neq pk)$ then return false
If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return false
$(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$ ; If $(w \neq g)$ then abort
$(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$ ; $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
If ( $(T, apk, m) \in \mathbf{BadSet}$ ) then bad $\leftarrow$ true ; $\boxed{\mathrm{HF}_1[(T, apk, m)] \leftarrow \perp}$
$c \leftarrow \mathrm{H}_1((T, apk, m))$ ; $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$
Return $(g^z h^s = T \cdot apk^c)$
```

Figure 27: Games $\mathrm{Gm}_3$ and $\mathrm{Gm}_4$ for proof of Theorem 7.2. Oracles $\mathsf{Init}, \mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2, \mathrm{H}_0$ are the same as those in $\mathrm{Gm}_{2,\rho}$. Parameter $\rho$ is set to $(1 - (1 + q_s)^{-1})$ in oracle $\mathrm{H}_0$.

$\mathcal{A}_{\mathrm{ms}}$ succeeds, Equation (58) gives us that

$$\Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})] = \Pr[\mathrm{Gm}_2(\mathcal{A}_{\mathrm{ms}}) \mid \mathrm{Gm}_2(\mathcal{A}_{\mathrm{ms}}) \text{ does not abort}] .$$

Hence,

$$\Pr[\mathrm{Gm}_{2,\rho}(\mathcal{A}_{\mathrm{ms}})] \geq \frac{1}{e(1 + q_s)} \cdot \Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})] . \tag{59}$$

For the rest of the proof, we set $\rho = (1 - (1 + q_s)^{-1})$ and omit writing them in the subscript for games. Next, we need to further modify oracles $\mathrm{H}_1$ and $\mathrm{H}_2$ so that whenever $\mathrm{H}_2$ derives a fresh aggregate key $apk$, it must not have been queried to $\mathrm{H}_1$ (in the form of $(T, apk, m)$ for *any* $T$ and $m$). Formally, consider games $\mathrm{Gm}_3$ and $\mathrm{Gm}_4$ given in Fig. 27. These games also keep track of a set **BadSet**, which contains those $\mathrm{H}_1$ queries $(T, apk, m)$ such that the aggregate key $apk$ is later derived in $\mathrm{H}_2$ (line 51). By construction, if any $\mathrm{H}_1$ query $(T, apk, m)$ is not in **BadSet** (at the end of the game execution), the aggregate key $apk$ is either previously derived in $\mathrm{H}_2$, or it has never been derived in any $\mathrm{H}_2$ query. Game $\mathrm{Gm}_3.\textsc{Fin}$ does not contain the boxed code, which makes the oracle behave identically to $\mathrm{Gm}_2.\mathrm{H}_2$. So, we have

$$\Pr[\mathrm{Gm}_2(\mathcal{A})] = \Pr[\mathrm{Gm}_3(\mathcal{A})] . \tag{60}$$

Oracle $\mathrm{Gm}_4.\mathrm{H}_2$ contains the boxed code, which reset the oracle $\mathrm{H}_1$ at the chosen forgery point $(T, apk, m)$ if it is part of **BadSet**. This ensures the value $\mathrm{HF}_1[(T, apk, m)]$ to always be defined *after* the $\mathrm{H}_2$ query that derives aggregate key $apk$. By construction, $\mathrm{Gm}_3$ and $\mathrm{Gm}_4$ are identical-until-bad. So,

$$\Pr[\mathrm{Gm}_3(\mathcal{A})] \leq \Pr[\mathrm{Gm}_4(\mathcal{A})] + \Pr[\mathrm{Gm}_4 \text{ sets bad}] . \tag{61}$$

```
Adversary $\mathcal{A}_{\mathrm{xidl}}^{\textsc{NwTar},\textsc{Ch},\textsc{Fin}}(X)$:
1  $pk \leftarrow X$ ; $(k, \boldsymbol{pk}, m, \sigma) \leftarrow\!\!\$\ \mathcal{A}_{\mathrm{ms}}^{\mathrm{NS},\textsc{Sign}_1,\textsc{Sign}_2,\mathrm{H}_0,\mathrm{H}_1,\mathrm{H}_2}(pk)$
2  If $(\boldsymbol{pk}[k] \neq pk)$ then return $\perp$
3  If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return $\perp$
4  $(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$ ; If $(w \neq g)$ then abort
5  $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$ ; $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$ ; $(T, s, z) \leftarrow \sigma$
6  If ( $(T, apk, m) \in \textbf{BadSet}$ ) then $\mathrm{HF}_1[(T, apk, m)] \leftarrow \perp$
7  $c \leftarrow \mathrm{H}_1((T, apk, m))$ ; $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$ ; $i \leftarrow \mathrm{TI}[(T, apk, m)]$
8  Return $(i, (z + s \cdot \beta_g) \mod p)$
```

```
$\mathrm{H}_1(x)$:
9  If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$
10 $(T, apk, m) \leftarrow x$
11 $\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$
12 If $(\mathrm{TJ}[apk] = \perp)$ then
13     Return $\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$
14 $\iota \leftarrow \iota + 1$ ; $\mathrm{TI}[x] \leftarrow \iota$
15 $\mathrm{HF}_1[x] \leftarrow c_\iota \leftarrow\!\!\$\ \textsc{Ch}(\mathrm{TJ}[apk], T)$
16 Return $\mathrm{HF}_1[x]$
```

```
$\mathrm{H}_2(x)$:
17 If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$
18 $(\cdot, \boldsymbol{pk}) \leftarrow x$ ; If $(pk \notin \boldsymbol{pk})$ then
19     Return $\mathrm{HF}_2[x] \leftarrow\!\!\$\ \mathbb{Z}_p$
20 $j \leftarrow j + 1$ ; $k \leftarrow \mathsf{minInd}(pk, \boldsymbol{pk})$
21 If $(x \neq (k, \boldsymbol{pk}))$ then
22     Return $\mathrm{HF}_2[x] \leftarrow\!\!\$\ \mathbb{Z}_p$
23 $S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
24 $\mathrm{HF}_2[(k, \boldsymbol{pk})] \leftarrow e_j \leftarrow \textsc{NwTar}(S)$
25 $apk \leftarrow S \cdot pk^{e_j}$ ; $\mathrm{TJ}[apk] \leftarrow j$
26 If $\mathrm{TV}[apk] \neq \perp$ then
27     $\textbf{BadSet} \leftarrow \textbf{BadSet} \cup \mathrm{TV}[apk]$
28 Return $\mathrm{HF}_2[x]$
```

Figure 28: Adversary $\mathcal{A}_{\mathrm{xidl}}$ used in Theorem 7.2. Oracles $\mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2, \mathrm{H}_0$ are simulated exactly per code from Fig. 25.

---

We first compute that probability that **BadSet** is non-empty at line 57. Since each $\mathrm{H}_2$ query has probability at most $q_1/p$ probability of adding elements to **BadSet**, we can bound

$$\Pr[\textbf{BadSet} \neq \emptyset \text{ at line 57 }] \leq \frac{q_1 q_2}{p} \ . \tag{62}$$

Note that flag $\mathsf{bad}$ can only be set if $\mathrm{Gm}_4$ did not abort (in oracle $\mathrm{H}_0$ or line 55), which happens with probability $1/(e(1 + q_{\mathrm{s}}))$ by previous analysis. Furthermore, the view of the adversary is *independent* of whether game $\mathrm{Gm}_4$ aborts. Hence,

$$\Pr[\mathrm{Gm}_4(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_1 q_2}{ep(1 + q_{\mathrm{s}})} \ . \tag{63}$$

We now move on to the construction of the adversary, given in Fig. 28. The adversary $\mathcal{A}_{\mathrm{xidl}}$ runs $\mathcal{A}_{\mathrm{ms}}$ while giving it simulated oracle $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}, \textsc{SignStage}_1, \textsc{SignStage}_2$. Code for $\mathrm{H}_0, \mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2$ are copied from game $\mathrm{Gm}_4$. The only new code here is in $\mathrm{H}_1$ and $\mathrm{H}_2$, which we now explain.

For each $j$-th $\mathrm{H}_2$ query $x = (\cdot, \boldsymbol{pk})$, where $\mathrm{HF}_2[x]$ is not yet defined the adversary will sample $\mathrm{HF}_2[(i, \boldsymbol{pk})]$ for each $i = 1, \ldots, |\boldsymbol{pk}|$ as follows. If the target public key $X$ is not in $\boldsymbol{pk}$, then these values are sampled honestly (line 15). Otherwise, let $k$ be the smallest index such that $\boldsymbol{pk}[k] = X$. Our adversary will query the $\textsc{NwTar}$ oracle from $\mathrm{Gm}_{\mathbb{G},g,q_2,q_1}^{\mathrm{xidl}}$ game so that the resulting aggregate public key $apk$ is the target point $T_j$ generated by the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. This is done by first computing the partial aggregation value of $S$ (line 17), before submitting it to the NwTar oracle to obtain response $e_j$ which is set as the output of $\mathrm{H}_2$ (line 19).

For each $\mathrm{H}_1$ query $(T, apk, m)$, the adversary will submit the commitment to the oracle Ch, at the index that corresponds to the aggregate public key $apk$. This is done so that a forgery $(T, s, z)$ corresponding to this $\mathrm{H}_1$ query can be turned into a break against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. Here, we are also utilizing the fact that a successful forgery $(\boldsymbol{pk}, m, (T, s, z))$ is such that $\mathrm{H}_0((\boldsymbol{pk}, m))$ is a known power of $g$. Hence, the verification equation

$$g^z h^s = T \cdot apk^{\mathrm{H}_1((T,apk,m))} ,$$

of the signature scheme implies that the computed response $z + \beta_g s$, against the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$, is valid, i.e. $g^{z+\beta_g s} = T \cdot T_j^{\mathrm{H}_1((T,apk,m))}$, where $T_j = apk$ is the $j$-th target point generated by NwTar oracle. Hence,

$$\Pr[\mathrm{Gm}_4(\mathcal{A}_{\mathrm{ms}})] = \Pr[\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}})] \, . \tag{64}$$

Putting Equation (59), (60), (61) and (64) together, we obtain the result claimed in the theorem. ▮