The Multi-Base Discrete Logarithm Problem: Concrete Security Improvements for Schnorr Identification, Signatures and Multi-Signatures

Mihir Bellare[1] Wei Dai[2]

April 2020

Abstract

We introduce the Multi-Base Discrete Logarithm (MBDL) problem. We show that, unlike the basic Discrete Logarithm (DL) problem, MBDL allows a *tight* (rewinding-avoiding) proof for the IMP-PA security of the Schnorr identification scheme, leading to a proof from MBDL for the Schnorr signature scheme that loses only a factor of the number of adversary hash queries, which is essentially optimal. We show that not only is the MBDL problem hard in the generic group model, but with a bound that matches that for DL, so that our new reductions for Schnorr allow implementations, for the same level of proven security, to use smaller groups, which increases efficiency. We then give an MBDL-based, non-rewinding proof for the BN multi-signature scheme (seeing renewed interest in crypto-currencies) that is tight up to the number of hash queries, improving on the prior DL-based proof, and leading again to improved efficiency by allowing the use of smaller groups.

[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

Many discrete-log based schemes have evaded proofs of security based on the assumed hardness of the *basic* version of the discrete log problem —recall that, cyclic group $\mathbb{G}$ with generator $g \in \mathbb{G}$ being public, the basic problem, denoted DL, is for an adversary, given $X = g^x$, to find the underlying, random exponent x— while continuing to resist attack. The understanding that this arises from strengths of the discrete-log problem not captured in the basic version has lead to the introduction of other versions of the problem, and proofs of security based on them, that not only settle the security of canonical schemes, but also increase understanding of, and add structure and unity to, the area.

This paper suggests a new variant, called the Multi-Base Discrete Logarithm (MBDL) problem. A twist with regard to the applications we give is that the novelty is not (always) in the *existence* of a reduction but in its *tightness*— we will settle long-standing questions in this regard. Crucial to the meaningfulness and practical applicability of this is that we show the *quantitative* hardness of MBDL in the generic group model to match that of DL, so group sizes can be chosen as if DL itself were the starting point.

<u>PRIOR DISCRETE-LOG VARIANTS.</u> A classical variant of the basic DL problem is of course the Diffie-Hellman problem, which allows security proofs of the Diffie-Hellman secret-key exchange [20] and the El Gamal public-key encryption scheme [23]. An example closer to our work is the One-More Discrete Logarithm (OMDL) problem [3], which has seen numerous applications [5, 19, 42, 26, 21]. Fix a cyclic group $\mathbb{G}$ and generator g of $\mathbb{G}$. The adversary is given a list of random challenges $X_1, X_2, \ldots \in \mathbb{G}$ and access to a discrete log oracle $\mathsf{DL}_{\mathbb{G},g}(\cdot)$ that, on input $W \in \mathbb{G}$, returns w such that $g^w = W$. To win, the adversary must return the discrete logarithms to base g of $n+1$ items in the list, *while making at most n calls to its discrete log oracle*, the integer $n \geq 0$ parametrizing the problem so that one can refer to the n-OMDL problem or assumption.

<u>MBDL.</u> Continue to fix a cyclic group $\mathbb{G}$ and generator g of $\mathbb{G}$. In our Multi-Base Discrete Logarithm (MBDL) problem, the adversary is given a challenge $Y \in \mathbb{G}$, a list $X_1, X_2, \ldots, X_n \in \mathbb{G}^*$ of generators of $\mathbb{G}$, and access to an oracle DLO that, on input i, W, returns $\mathsf{DL}_{\mathbb{G},X_i}(W)$, the discrete logarithm of W, *not in base g, but in base X_i*. To win it must find $\mathsf{DL}_{\mathbb{G},g}(Y)$, the discrete logarithm of the challenge Y to base g, *while making at most one call to* DLO *overall*, meaning it is allowed to take the discrete log of at most one group element. (But this element, and the base X_i, can be chosen as it wishes.) The number of bases n is a parameter of the problem, so that one can refer to the n-MBDL problem or assumption. (Our results will rely only on 1-MBDL, but we keep the definition general for possible future applications.) The restriction to at most one DLO call is necessary, for if even two are allowed, $\mathsf{DL}_{\mathbb{G},g}(Y)$ can be obtained as $\mathrm{DLO}(1, Y) \cdot \mathrm{DLO}(1, g)^{-1} \bmod p$ where $p = |\mathbb{G}|$ is the (known) order of the group $\mathbb{G}$.

<u>HOW HARD IS MBDL?</u> As with any new problem, a basic question about MBDL is of course what evidence one can give for its hardness. We follow the conventional and accepted route of proving hardness in the Generic Group Model (GGM) [46]. Theorem 5 says that if an adversary has running time t, then its advantage $\epsilon^{\text{gg-mbdl}}(t)$ in breaking the n-MBDL problem in a generic group of order p is at most $\mathcal{O}(t^2/p)$. (This assumes $n \leq t$. Running time, in the GGM, is captured as number of queries to a group-operation oracle.)

Beyond the qualitative evidence of hardness this gives, the quantitative element is important. The $\mathcal{O}(t^2/p)$ bound on $\epsilon^{\text{gg-mbdl}}(t)$ from Theorem 5 is the same as the bound on the advantage $\epsilon^{\text{gg-dl}}(t)$ of a time t adversary in breaking the *basic* DL problem in the GGM [46]. That is, n-MBDL has the same *quantitative* difficulty as DL itself. This is crucial, not for giving reductions from MBDL that are tighter than from DL, but for these new reductions to have practical value, meaning allow reducing the group size, and thus increasing efficiency, while preserving security. (Thus, looking ahead, our tight reductions of the security of Schnorr identification and signatures to 1-MBDL are valid regardless of what is true or not true about MBDL in the GGM, but had our bound on $\epsilon^{\text{gg-mbdl}}(t)$ been say, $\mathcal{O}(t^4/p)$, rather than the $\mathcal{O}(t^2/p)$ that it is, we would not obtain the speedups shown in Figure 5.)

<u>Schnorr background.</u> Let $\mathbb{G}$ be a group of prime order p, and $g \in \mathbb{G}$ a generator of $\mathbb{G}$. The iconic Schnorr signature scheme $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$ [44] is derived from the Schnorr identification scheme $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ [44] via the Fiat-Shamir transform [25]. The Schnorr signature scheme is widely used. Ed25519, an implementation of it over twisted Edwards curves [10], is in OpenSSL, OpenSSH, GnuPG and many other places [32]. In these uses, existing security proofs, due to their non-tightness, are ignored in picking parameters. Our work fills this gap, justifying security for parameters in actual use.

The heart of the ROM proof of UF-security of the signature scheme from DL, and the root cause of the notorious looseness of that reduction, lies in the rewinding proof of IMP-PA-security of the identification scheme from DL, and the corresponding looseness of that reduction. This makes identification, and tighter reductions for it, the first and most basic problem.

<u>Schnorr identification.</u> IMP-PA (impersonation under passive attack [24]) security of $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ is proven under DL via a rewinding argument. The simplest analysis uses the Reset Lemma of [5]. It shows that, roughly:

$$\epsilon^{\text{imp-pa}}(t) \leq \sqrt{\epsilon^{\text{dl}}(t)} + \frac{1}{p} \,, \tag{1}$$

where $\epsilon^{\text{imp-pa}}(t)$ is the probability of breaking IMP-PA security of ID in time t and $\epsilon^{\text{dl}}(t)$ is the probability of breaking DL in time t. There is, however, no attack showing that the large security loss arising from the square-root in Eq. (1) is inherent. Picking the group size to ensure a desired level of security according to Eq. (1), accordingly, would result in efficiency losses that practitioners deem unnecessary. Accordingly the proof is largely viewed as a qualitative rather than quantitative guarantee, group sizes being chosen in ad hoc ways. Improving the reduction of Eq. (1) to bring the theory more in line with the indications of cryptanalysis has been a long-standing open question.

We suggest, following the theme above, that, manifesting itself here is an unformalized strength of the discrete logarithm problem. We will show that this strength is captured by the MBDL problem, via a proof of IMP-PA security of the Schnorr identification scheme $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ with a *tight* reduction to 1-MBDL: letting $\epsilon^{\text{1-mbdl}}(t)$ be the probability of breaking the 1-MBDL problem in time t, Theorem 1 says that, roughly:

$$\epsilon^{\text{imp-pa}}(t) \leq \epsilon^{\text{1-mbdl}}(t) + \frac{1}{p} \,. \tag{2}$$

Having decided on a security target, namely to ensure $\epsilon^{\text{imp-pa}}(t) \leq \epsilon$ for some chosen values of ϵ, t, one wants to pick a group $\mathbb{G}$ that guarantees it. Eq. (2) allows this group to be smaller than allowed by Eq. (1). Since scheme algorithms have running time cubic in the log of $p = |\mathbb{G}|$, Eq. (2) allows a performance improvement. Figure 5 says that this improvement can range from 1.9x to 3x. To apply the two equations to obtain the Figure 5 cost analysis, we use the estimates $\epsilon^{\text{dl}}(t) \approx \epsilon^{\text{1-mbdl}}(t) \approx t^2/p$, meaning assume we are in a group (like an elliptic curve one) where generic algorithms are the best known, so that the GGM bound is tight. This is where we rely crucially on the quantitative GGM-bound of Theorem 5 for 1-MBDL being the same as the one for DL itself.

<u>Reduction approach.</u> The proof of the prior Eq. (1) is by a classical rewinding argument that exploits the special soundness property of the Schnorr identification scheme, namely that from

two compatible transcripts —this means they are accepting and have the same commitment but different challenges— one can extract the secret key. To find the discrete log, in base g, of a given challenge Y, the discrete log adversary $\mathcal{B}$ plants the challenge as the public key X and performs two, related runs of the given IMP-PA adversary, hoping to get two compatible transcripts —in which case it can extract the secret key and solve its DL instance— which the Reset Lemma [5] says happens with probability roughly the square of the IMP-PA advantage of $\mathcal{A}$, leading to the square-root in Eq. (1).

Recall that our 1-mbdl adversary $\mathcal{B}$ gets input a challenge Y whose discrete logarithm *in the usual base g it must find*, just like a DL adversary. To get Eq. (2) we must avoid rewinding. The question is how and why the ability to take one discrete logarithm in some random base X_1 helps to do this and get a tight reduction. Our reduction deviates from prior ones by *not* setting Y to the public key. Instead, it sets X_1 to the public key. Then, it performs a *single* execution of the given IMP-PA adversary $\mathcal{A}$, "planting" Y in the communication in such a way that success of $\mathcal{A}$ in impersonating the prover yields $\mathsf{DL}_{\mathbb{G},g}(Y)$. This planting step makes one call to $\mathrm{DLO}(1,\cdot)$, meaning asks for a discrete logarithm in base X_1 of some W that depends on the execution. The full proof is in Section 4.

<u>Schnorr signatures.</u> The Schnorr signature scheme $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$ has a proof of UF-security in the ROM under the basic DL assumption [43, 41, 1, 34]. The reduction with the best known bound is via the Forking Lemma [43, 4]. Roughly:

$$\epsilon^{\mathrm{uf}}(t,q) \leq \sqrt{q \cdot \epsilon^{\mathrm{dl}}(t)} + \frac{q^2}{p} \ , \tag{3}$$

where $\epsilon^{\mathrm{uf}}(t,q)$ is the probability of breaking UF security of DS in time t with q queries to the random oracle (RO) and at most q signing queries, and $\epsilon^{\mathrm{dl}}(t)$ is the probability of breaking DL in time t. We give a reduction from 1-MBDL which says, roughly:

$$\epsilon^{\mathrm{uf}}(t,q) \leq q \cdot \epsilon^{\text{1-mbdl}}(t) + \frac{q^2}{p} \ . \tag{4}$$

The first term of our bound from Eq. (4) is the square of the corresponding term from Eq. (3) and thus our bound is always better (smaller). We obtain Eq. (4) by combining Eq. (2) with a reduction of the UF-security of DS to the IMP-PA security of ID from [1]. The full statement appears as Theorem 3. Figure 5 shows speedup factors of 1.6x to 2.5x for Schnorr signatures resulting from this result.

<u>Extensions.</u> Our results extend to tightly reduce the multi-user IMP-PA security of $\mathsf{SchID}[\mathbb{G}, g]$ to 1-MBDL, and to reduce the multi-user UF security of $\mathsf{SchSig}[\mathbb{G}, g]$ to 1-MBDL with only a factor q loss, just as for the single-user case discussed above. This can be shown directly, but is also a consequence of general results of Kiltz, Masny and Pan (KMP) [34]. Our results apply also to Schnorr-derived schemes such as the widely-deployed Ed25519 signature scheme [10].

<u>Multi-signatures from MBDL.</u> Consider parties $1, \ldots, n$, each party i having its own secret signing key sk_i and matching public verification key vk_i. A multi-signature (MS) scheme allows them, given a common message M, to interactively produce a *short, joint* signature σ of M. Multi-signatures have been a subject of research in the cryptographic community for some time [40, 38, 11, 4, 35, 2] but are receiving renewed attention and interest due to applications in cryptocurrencies and blockchains, where the quest is for (secure and) efficient schemes [49, 14, 37, 21]. Both discrete-log based MS schemes [4, 2, 49, 37, 21] and pairing-based ones [11, 14] are being developed. Pairing-based ones can require less communication [14], but this must be traded off with efficiency loss due to larger group sizes, and, more importantly, integration into Bitcoin is hard because it requires schemes compatible with the NIST `secp256k` curve, which is not pairing friendly. This leads to a

current focus on DL-based MS schemes.

BN [4] is a DL-based MS scheme that in particular has attracted interest in this application domain. However, the reduction establishing its ROM security under DL is based on the General Forking Lemma [4] and is appreciably loose. (See Eq. (14).) If, to guarantee a certain level of security (like 128 bits), group sizes were chosen according to this reduction, they would be quite large, and the scheme correspondingly slow. We, instead, prove security with a reduction from 1-MBDL that is tight up to a factor of the number of RO queries. The result is Theorem 4 in Section 5. The speedup factors obtained via reduced group sizes are about the same as for Schnorr signatures.

BN is a 3-round scheme. Two-round schemes were given in [2, 36, 49, 37]. However, DEFKLNS [21] showed that the security proofs of many of these 2-round MS schemes are flawed, and furthermore gave sub-exponential attacks against the schemes. The errors in the proofs arose, at least in part, from the use of complex rewinding techniques that often involved multiple rewindings and associated Forking Lemmas. A benefit of proofs from MBDL is that neither rewinding nor Forking Lemmas are needed. This makes the proofs simpler and less error prone. (In addition to allowing the reduction to be tighter.)

Prior work and discussion. A natural comment is that our result and bound are not, in a strict sense, improvements over prior one, but rather are incomparable, for we improve the concrete security of the reductions, but at the cost of making a new and potentially stronger assumption, namely the hardness of 1-MBDL. Indeed, MBDL, as with any new assumption, should be treated with caution. However, it also seems that improving Eq. (1) to something like Eq. (2) under the basic DL assumption is out of reach and perhaps not even possible, and thus that, as indicated above, the apparent strength of the Schnorr schemes indicated by cryptanalysis is arising from stronger hardness properties of the discrete log problem not captured in the basic version. We are trying to understand and formalize this hardness via new problems that tightly imply security of the Schnorr primitives. And Eq. (2) is somewhat non-trivial in the sense that MBDL does not represent simply assuming the ends we aim to reach.

Shoup [46] proved IMP-PA security of the Schnorr identification scheme in the GGM with a tight reduction to DL. Fuchsbauer, Plouviez and Seurin [28] give a tight reduction of the UF-security of Schnorr signatures to DL in the Algebraic Group Model (AGM) of [27], where it is assumed that, whenever an adversary provides a group element Z, it also provides the representation of Z relative to prior group elements. Both the GGM and the AGM represent security against limited classes of adversaries. Our tight reductions from MBDL, in contrast, are in the standard model, and make no GGM or AGM-like assumptions on adversaries. It is true that we justify MBDL in the GGM, but overall our work can be seen as the following re-factoring of the security analysis: (1) Limit GGM use to verify a general, number-theoretic problem (namely MBDL) and (2) Prove many schemes (Schnorr ID and signatures, BN multi-signatures) in the standard model from MBDL. This seems to us to yield greater understanding and guarantees, and opens up the door to further proofs from MBDL. This kind of re-factoring has been profitably employed in the past in bilinear-map-based cryptography.

Kiltz, Masny and Pan (KMP) [34] measure tightness via ratios of advantage to running time. Under this metric, they claim a tight reduction from DL to IMP-PA of the Schnorr identification scheme $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$. They get this however only when they make the advantage of the DL adversary a constant close to 1. Their results do not seem to provide group-size choices that yield improvements (scheme speedups) over the classical Eq. (1) for arbitrary, given advantage and running time. Our reductions, in contrast, are tight for advantage and time taken individually, and across the full range for these values, and numerical estimates (Figure 5) show clear improvements over the prior Eq. (1). We elaborate in Appendix A.

While new assumptions (like MBDL) should of course be treated with caution, cryptographic research has a history of progress through introducing them. For example, significant advances were obtained by moving from the CDH assumption to the stronger DDH one [39, 18]. Pairing-based cryptography has seen a host of assumptions that have had many further applications, including the bilinear Diffie-Hellman (BDH) assumption of [16] and the DLIN assumption of [13]. The RSA Φ-Hiding assumption of [17] has since found many applications. This suggests that the introduction and exploration of new assumptions, which we continue, is an interesting and productive line of research.

There is some feeling that "interactive" or "non-falsifiable" assumptions are undesirable. However, it depends on the particular assumption; we have seen interactive assumptions that are unbroken and successful, like OMDL [3], and also seen many non-interactive ones that have been broken. It is important that it be possible to show an assumption is false, but this is possible even for assumptions that are classified as "non-falsifiable;" for example, knowledge-of-exponent assumptions have successfully been shown to be false through cryptanalysis [6]. MBDL is similarly amenable to cryptanalytic evaluation.

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 $\{1,\ldots,n\}$. We denote the number of coordinates of a vector $\boldsymbol{x}$ by $|\boldsymbol{x}|$. 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 $[\boldsymbol{x}] = \{\boldsymbol{x}[i] : 1 \le i \le |\boldsymbol{x}|\}$ 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 ε we denote the empty vector or string. The size of a set S is denoted $|S|$. For sets D, R let $\mathrm{FNS}(D, R)$ denote the set of all functions $f : D \to R$.

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 $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1,\ldots;r)$ denote executing algorithm A on inputs $x_1,\ldots$ and coins r with access to oracles $\mathrm{O}_1,\ldots$ and letting y be the result. We let $y \leftarrow\$ A^{\mathrm{O}_1,\cdots}(x_1,\ldots)$ be the resulting of picking r at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1,\ldots;r)$. 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 [9]. (See Fig. 2 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 true.

In writing game or adversary pseudocode, it is assumed that boolean variables are initialized to false, integer variables are initialized to 0 and set-valued variables are initialized to the empty set $\emptyset$.

When adversary $\mathcal{A}$ is executed with game Gm, we consider two running times. The time of the execution, denoted $\mathrm{T}_{\mathrm{Gm}(\mathcal{A})}$, includes the time taken by game procedures, while the time of the adversary, denoted $\mathrm{T}_{\mathcal{A}}$, assumes game procedures take unit time to respond. By $\mathrm{Q}_{\mathcal{A}}^{\mathrm{O}}$ we denote the number of queries made by $\mathcal{A}$ to oracle O in the execution. These counts are all worst case.

GROUPS. Let $\mathbb{G}$ be a group of order 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

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

INIT:
  $p \leftarrow |\mathbb{G}|$ ; $y \leftarrow\!\!\$\ Z_p$ ; $Y \leftarrow g^y$
  Return $Y$

FIN($y'$):
  Return $(y = y')$
```

```
Game $\mathbf{G}^{\mathrm{mbdl}}_{\mathbb{G},g,n}$

INIT:
  $p \leftarrow |\mathbb{G}|$ ; $y \leftarrow\!\!\$\ Z_p$ ; $Y \leftarrow g^y$
  For $i = 1, \ldots, n$ do
      $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$ ; $X_i \leftarrow g^{x_i}$
  Return $Y, X_1, \ldots, X_n$

DLO($i, W$):  // One query
  Return $\mathsf{DL}_{\mathbb{G},X_i}(W)$

FIN($y'$):
  Return $(y = y')$
```

Figure 1: Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. Left: Game defining standard discrete logarithm problem. Right: Game defining (n, m)-multi-base discrete logarithm problem. Recall $\mathsf{DL}_{\mathbb{G},X}(W)$ is the discrete logarithm of $W \in \mathbb{G}$ to base $X \in \mathbb{G}^*$.

elements, which is the set of generators of $\mathbb{G}$ if the latter has prime order. If $g \in \mathbb{G}^*$ is a generator and $X \in \mathbb{G}$, the discrete logarithm base g of X is denoted $\mathsf{DL}_{\mathbb{G},g}(X)$, and it is in the set $\mathbb{Z}_{|\mathbb{G}|}$.

3 The Multi-Base Discrete-Logarithm Problem

We introduce the multi-base discrete-logarithm (MBDL) problem. It is similar in flavor to the one-more discrete-logarithm (OMDL) problem [3], which has found many applications, in that it gives the adversary the ability to take discrete logarithms. For the rest of this Section, we fix a group $\mathbb{G}$ of prime order $p = |\mathbb{G}|$, and we fix a generator $g \in \mathbb{G}^*$ of $\mathbb{G}$. Recall that $\mathsf{DL}_{\mathbb{G},g} : \mathbb{G} \to \mathbb{Z}_p$ is the discrete logarithm function in $\mathbb{G}$ with base g.

DL AND OMDL. We first recall the standard discrete logarithm (DL) problem via game $\mathbf{G}^{\mathrm{dl}}_{\mathbb{G},g}$ on the left of Figure 1. INIT provides the adversary, as input, a random challenge group element Y, and to win it must output $y' = \mathsf{DL}_{\mathbb{G},g}(Y)$ to FIN. We let $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A})]$ be the discrete-log advantage of $\mathcal{A}$.

In the OMDL problem [3], the adversary can obtain many random challenges $Y_1, Y_2, \ldots, Y_n \in \mathbb{G}$. It has access to a discrete log oracle that given $W \in \mathbb{G}$ returns $\mathsf{DL}_{\mathbb{G},g}(W)$. For better comparison with MBDL, let's allow just one query to this oracle. To win it must compute the discrete logarithms of two group elements from the given list $Y_1, Y_2, \ldots, Y_n \in \mathbb{G}$. The integer $n \geq 2$ is a parameter of the problem.

MBDL. In the MBDL problem we introduce, we return, as in DL, to there being a single random challenge point Y whose discrete logarithm in base g the adversary must compute. It has access to an oracle DLO to compute discrete logs, but rather than in base g as in OMDL, to bases that are public, random group elements $X_1, X_2, \ldots, X_n$. It is allowed *just one* query to DLO. (As we will see, this is to avoid trivial attacks.) The integer $n \geq 1$ is a parameter of the problem.

Proceeding formally, consider game $\mathbf{G}^{\mathrm{mbdl}}_{\mathbb{G},g,n}$ on the right in Fig. 1, where $n \geq 1$ is an integer parameter called the number of bases. The adversary's input, as provided by INIT, is a random challenge group element Y together with random generators $X_1, X_2, \ldots, X_n$. It can call oracle DLO with an index $i \in [n]$ and any group element $W \in \mathbb{G}$ of its choice to get back $\mathsf{DL}_{\mathbb{G},X_i}(W)$. Just one such call is allowed. At the end, the adversary wins the game if it outputs $y' = \mathsf{DL}_{\mathbb{G},g}(Y)$

```
$\mathsf{Exec}_{\mathsf{ID}}(vk, sk)$:
1 $(R, \mathsf{st}) \leftarrow\$ \mathsf{ID.Cmt}(vk)$
2 $c \leftarrow\$ \mathsf{ID.Ch}$
3 $z \leftarrow \mathsf{ID.Rsp}(sk, c, \mathsf{st})$
4 $b \leftarrow \mathsf{ID.Vf}(vk, R, c, z)$
5 $\mathsf{tr} \leftarrow (R, c, z)$
6 Return $(b, \mathsf{tr})$
```

```
Game $\mathrm{Gm}^{\text{imp-pa}}_{\mathsf{ID}}$

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

TR:
2 $(b, \mathsf{tr}) \leftarrow\$ \mathsf{Exec}_{\mathsf{ID}}(vk, sk)$ ; Return $\mathsf{tr}$

CH$(R_*)$: // One query
3 $c_* \leftarrow\$ \mathsf{ID.Ch}$ ; Return $c_*$

FIN$(z_*)$:
4 Return $\mathsf{ID.Vf}(vk, R_*, c_*, z_*)$
```

Figure 2: Left: Algorithm defining an honest execution of the canonical identification scheme ID given key pair (sk, vk). Right: Game defining IMP-PA security of ID.

to FIN. We define the mbdl-advantage of $\mathcal{A}$ by

$$\mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,n}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{mbdl}}_{\mathbb{G},g,n}(\mathcal{A})] \ .$$

DISCUSSION. By n-MBDL we will refer to the problem with parameter n. It is easy to see that if n-MBDL is hard then so is n'-MBDL for any $n' \leq n$. Thus, the smaller the value of n, the weaker the assumption. For our results, 1-MBDL, the weakest assumption in the series, suffices.

We explain why at most one DLO query is allowed. Suppose the adversary is allowed two queries. It could compute $a = \text{DLO}(1, Y) = \mathsf{DL}_{\mathbb{G},X_1}(Y)$ and $b = \text{DLO}(1, g) = \mathsf{DL}_{\mathbb{G},X_1}(g)$, so that $X_1^a = Y$ and $X_1^b = g$. Now the adversary returns $y' \leftarrow ab^{-1} \bmod p$ and we have $g^{y'} = (g^{b^{-1}})^a = X_1^a = Y$, so the adversary wins.

The problem can be generalized to allow multiple DLO queries with the restriction that at most one query is allowed per base, meaning for each i there can be at most one $\text{DLO}(i, \cdot)$ query. We do not consider this further in this paper because we do not need it for our results, but it may be useful for future applications.

As evidence for the hardness of MBDL, Theorem 5 proves good bounds on the adversary advantage in the generic group model (GGM). It is also important to consider non-generic approaches to the discrete logarithm problem over elliptic curves, including index-calculus methods and Semaev polynomials [47, 45, 48, 33, 29], but, to the best of our assessment, these do not yield attacks on MBDL that beat the GGM bound of Theorem 5.

4 Schnorr Identification and Signatures from MBDL

In this section, we give a *tight* reduction of the IMP-PA security of the Schnorr identification scheme to the 1-MBDL problem and derive a corresponding improvement for Schnorr signatures.

IDENTIFICATION SCHEMES. We recall that a (canonical) identification scheme [1] ID (see Figure 3 for an example) is a 3-move protocol in which the prover sends a first message called a commitment, the verifier sends a random challenge, the prover sends a response that depends on its secret key, and the verifier makes a decision to accept or reject based on the conversation transcript and the prover's public key. Formally, ID is specified by algorithms ID.Kg, ID.Cmt, ID.Rsp, and ID.Vf, as well as a set ID.Ch of challenges Via $(vk, sk) \leftarrow\$ \mathsf{ID.Kg}$, the key generation algorithm generates public verification key vk and associated secret key sk. Algorithms ID.Cmt and ID.Rsp are the prover algorithms. The commitment algorithm $\mathsf{ID.Cmt}$ takes input the public key vk and returns a commitment message R to send to the verifier, as well as a state st for the prover to retain. The deterministic response algorithm $\mathsf{ID.Rsp}$ takes input the secret key sk, a challenge $c \in \mathsf{ID.Ch}$ sent by the verifier, and a state st, to return a response z to send to the verifier. The deterministic verification algorithm $\mathsf{ID.Vf}$ takes input the public key and a conversation transcript R, c, z to return a decision $b \in \{\mathsf{true}, \mathsf{false}\}$ that is the outcome of the protocol.

An honest execution of the protocol is defined via procedure $\mathsf{Exec}_{\mathsf{ID}}$ shown in the upper left of Fig. 2. It takes input a key pair $(vk, sk) \in [\mathsf{ID.Kg}]$ to return a pair (b, tr) where $b \in \{\mathsf{true}, \mathsf{false}\}$ denotes the verifier's decision whether to accept or reject and $\mathsf{tr} = (R, c, z)$ is the transcript of the interaction. We require that ID schemes satisfy *(perfect) completeness*, namely that for any $(vk, sk) \in [\mathsf{ID.Kg}]$ and any $(b, \mathsf{tr}) \in [\mathsf{Exec}_{\mathsf{ID}}(sk, vk)]$ we have $b = \mathsf{true}$.

Impersonation under passive attack (IMP-PA) [24] is a security metric asking that an adversary not in possession of the prover's secret key be unable to impersonate the prover, even given access to honestly generated transcripts. Formally, consider the game $\mathrm{Gm}^{\text{imp-pa}}_{\mathsf{ID}}$ given in the right column of Fig. 2. An adversary has input the public key vk returned by INIT. It then has access to honest transcripts via the oracle Tr. When it is ready to convince the verifier, it submits its commitment R_* to oracle CH. We allow only one query to CH. In response the adversary obtains a random challenge c_*. It must now output a response z_* to FIN, and the game returns true iff the transcript is accepted by $\mathsf{ID.Vf}$. The R_*, c_* at line 4 are, respectively, the prior query to CH, and the response chosen at line 3. We define the IMP-PA advantage of $\mathcal{A}$ against ID as $\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A})]$, the probability that the game returns true.

<u>SCHNORR IDENTIFICATION SCHEME AND PRIOR RESULTS.</u> Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, and $g \in \mathbb{G}^*$ a generator of $\mathbb{G}$. We recall the Schnorr identification scheme [44] $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ in Fig. 3. The public key $vk = X = g^x \in \mathbb{G}$ where $sk = x \in \mathbb{Z}_p$ is the secret key. The commitment is $R = g^r \in \mathbb{G}$, and r is returned as the prover state by the commitment algorithm. Challenges are drawn from $\mathsf{ID.Ch} = \mathbb{Z}_p$, and the response z and decision b are computed as shown.

The IMP-PA security of $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ based on DL is proven by a rewinding argument. The simplest analysis is via the Reset Lemma of [5]. It leads to the following (cf. [5, Theorem 2], [7, Theorem 3]). Let $\mathcal{A}$ be an adversary attacking the IMP-PA security of ID. Then there is a discrete log adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \sqrt{\mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{B})} + \frac{1}{p} \, . \tag{5}$$

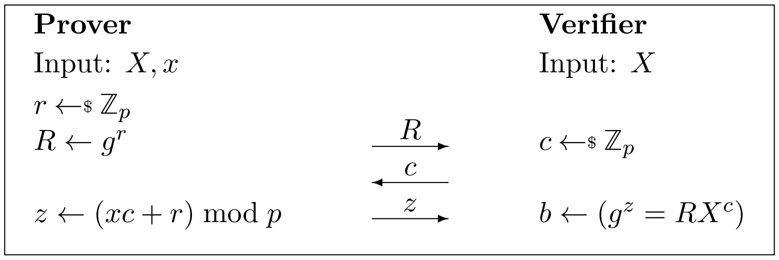
Additionally, the running time $\mathrm{T}_{\mathcal{B}}$ of $\mathcal{B}$ is roughly $2\,\mathrm{T}_{\mathcal{A}}$ plus simulation overhead of the form $\mathcal{O}(\mathrm{Q}^{\mathrm{Tr}}_{\mathcal{A}} \cdot T^{\exp}_{\mathbb{G}})$, where $T^{\exp}_{\mathbb{G}}$ is the time for an exponentiation in $\mathbb{G}$.

<u>OUR RESULT.</u> We show that the IMP-PA-security of the Schnorr identification scheme reduces *tightly* to the 1-MBDL problem. The reduction does *not* use rewinding. Our mbdl-adversary $\mathcal{B}$ solves the 1-MBDL problem by running the given imp-pa adversary $\mathcal{A}$ just once, so the mbdl-advantage, and running time, of the former, are about the same as the imp-pa advantage, and running time, of the latter.

Theorem 1 *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 $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ be the Schnorr identification scheme. Let $\mathcal{A}$ be an adversary attacking the imp-pa security of ID. Then we can construct an adversary $\mathcal{B}$ (shown explicitly in Figure 4) such that*

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) + \frac{1}{p} \, . \tag{6}$$

Additionally, $\mathrm{T}_{\mathcal{B}}$ is roughly $\mathrm{T}_{\mathcal{A}}$ plus simulation overhead of the form $\mathcal{O}(\mathrm{Q}^{\mathrm{Tr}}_{\mathcal{A}} \cdot T^{\exp}_{\mathbb{G}})$.


Prover
Input: X, x
$r \leftarrow\$ \mathbb{Z}_p$
$R \leftarrow g^r$
$z \leftarrow (xc + r) \bmod p$
Verifier
Input: X
$c \leftarrow\$ \mathbb{Z}_p$
$b \leftarrow (g^z = RX^c)$
R
c
z

```
ID.Kg:
    $x \leftarrow\$ \mathbb{Z}_{|G|}$ ; $X \leftarrow g^x$ ; Return $(X, x)$
ID.Cmt:
    $r \leftarrow\$ \mathbb{Z}_{|G|}$ ; $R \leftarrow g^r$ ; Return $(R, r)$
ID.Rsp$(x, c, r)$:
    $z \leftarrow (xc + r) \mod |G|$
    Return $z$
ID.Vf$(X, R, c, z)$:
    $b \leftarrow (g^z = X^c R)$ ; Return $b$
```

```
DS.Kg:
    $x \leftarrow\$ \mathbb{Z}_{|G|}$ ; $X \leftarrow g^x$
    Return $(X, x)$
DS.Sign$^{\mathrm{H}}(sk, m)$:
    $r \leftarrow\$ \mathbb{Z}_{|G|}$ ; $R \leftarrow g^r$
    $c \leftarrow \mathrm{H}(R, m)$
    $z \leftarrow (xc + r) \mod |\mathbb{G}|$
    Return $(R, z)$
DS.Vf$^{\mathrm{H}}(X, m, \sigma)$:
    $(R, z) \leftarrow \sigma$
    $c \leftarrow \mathrm{H}(R, m)$
    Return $(g^z = X^c R)$
```

Figure 3: Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$ and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. The Schnorr ID scheme $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ is shown pictorially at the top and algorithmically at the bottom left. At the bottom right is the Schnorr signature scheme $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$, using $\mathrm{H} : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$.

Proof of of Theorem 1: Recall that, when reducing IMP-PA security of Schnorr to DL, the constructed dl adversary $\mathcal{B}$ sets the target point Y to be the public key X. It is natural to take the same approach in our case. The question is how to use the discrete logarithm oracle DLO to avoid rewinding and get a tight reduction. But this is not clear and indeed the DLO oracle does not appear to help towards this.

Our reduction deviates from prior ones by *not* setting the target point Y to be the public key. Instead we look at a successful impersonation by $\mathcal{A}$. (Simulation of $\mathcal{A}$'s transcript oracle Tr is again via the honest-verifier zero-knowledge property of the scheme.) Adversary $\mathcal{A}$ provides R_*, receives c_* and then returns z_* satisfying $g^{z_*} = R_* X^{c_*}$, where X is the public key. Thus, $\mathcal{A}$ effectively computes the discrete logarithm of $R_* X^{c_*}$. We make this equal our mbdl challenge Y, meaning $\mathcal{B}$, on input Y, arranges that $Y = R_* X^{c_*}$. If it can do this successfully, the z_* returned by $\mathcal{A}$ will indeed be $\mathsf{DL}_{\mathbb{G},g}(Y)$, which it can output and win.

But how can we arrange that $Y = R_* X^{c_*}$? This is where the DLO oracle enters. Adversary $\mathcal{B}$ gives X as input to $\mathcal{A}$, meaning the public key is set to the group generator relative to which $\mathcal{B}$ may compute discrete logarithms. Now, when $\mathcal{A}$ provides R_*, our adversary $\mathcal{B}$ returns a challenge c_* that ensures $Y = R_* X^{c_*}$. This means $c_* = \mathsf{DL}_{\mathbb{G},X}(Y R_*^{-1})$, and this is something $\mathcal{B}$ can compute via its DLO oracle.

Some details include that the X returned by INIT is a generator, while the public key is a random group element, so they are not identically distributed, and that the challenge computed via DLO

```
Adversary $\mathcal{B}^{\text{DLO}}$:
$(Y, X) \leftarrow\!\$\ \textsc{Init}()$ ; $z_* \leftarrow\!\$\ \mathcal{A}^{\textsc{Ch},\textsc{Tr}}(X)$ ; Return $z_*$
$\textsc{Ch}(R_*)$:
$W \leftarrow R_*^{-1} \cdot Y$ ; $c_* \leftarrow \text{DLO}(1, W)$ ; Return $c_*$
Tr:
$z \leftarrow\!\$\ \mathbb{Z}_p$ ; $c \leftarrow\!\$\ \mathbb{Z}_p$ ; $R \leftarrow g^z \cdot X^{-c}$ ; Return $(R, c, z)$
```

```
Game $\text{Gm}_0$ / $\text{Gm}_1$ / $\text{Gm}_2$

$\textsc{Init}$: // Games $\text{Gm}_0$, $\boxed{\text{Gm}_1}$
1 $p \leftarrow |\mathbb{G}|$ ; $y \leftarrow\!\$\ \mathbb{Z}_p$ ; $Y \leftarrow g^y$ ; $x \leftarrow\!\$\ \mathbb{Z}_p$
2 If $(x = 0)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{x \leftarrow\!\$\ \mathbb{Z}_p^*}$
3 $X \leftarrow g^x$ ; Return $(Y, X)$

$\textsc{Init}$: // Game $\text{Gm}_2$
4 $p \leftarrow |\mathbb{G}|$ ; $y \leftarrow\!\$\ \mathbb{Z}_p$ ; $Y \leftarrow g^y$ ; $x \leftarrow\!\$\ \mathbb{Z}_p^*$ ; $X \leftarrow g^x$ ; Return $(Y, X)$

$\textsc{Ch}(R_*)$: // Games $\text{Gm}_0, \text{Gm}_1$
5 $c_* \leftarrow\!\$\ \mathbb{Z}_p$ ; Return $c_*$

$\textsc{Ch}(R_*)$: // Game $\text{Gm}_2$
6 $W \leftarrow R_*^{-1} \cdot Y$ ; $c_* \leftarrow \mathsf{DL}_{\mathbb{G},X}(W)$ ; Return $c_*$

$\text{Tr}(W)$: // Games $\text{Gm}_0, \text{Gm}_1, \text{Gm}_2$
7 $z \leftarrow\!\$\ \mathbb{Z}_p$ ; $c \leftarrow\!\$\ \mathbb{Z}_p$ ; $R \leftarrow g^z \cdot X^{-c}$ ; Return $(R, c, z)$

$\textsc{Fin}(z_*)$: // Games $\text{Gm}_0, \text{Gm}_1$
8 Return $(g^{z_*} = X^{c_*} R_*)$

$\textsc{Fin}(z_*)$: // Games $\text{Gm}_2$
9 Return ( $z_* = \mathsf{DL}_{\mathbb{G},g}(X^{c_*} R_*)$ )
```

Figure 4: Top: MBDL adverary $\mathcal{B}$ for Theorem 1, based on IMP-PA adversary $\mathcal{A}$. Bottom: Games for proof of Theorem 1.

---

must be properly distributed. The analysis will address these.

For the formal proof, consider the games of Figure 4. Procedures indicate (via comments) in which games they are present. Game Gm_1 includes the boxed code at line 2 while Gm_0 does not. The games implement the transcript oracle via the zero-knowledge simulation rather than using the secret key, but otherwise Gm_0 is the same as game $\text{Gm}_{\mathsf{ID}}^{\text{imp-pa}}$ so we have

$$\begin{aligned}\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A}) &= \Pr[\text{Gm}_0(\mathcal{A})] \\ &= \Pr[\text{Gm}_1(\mathcal{A})] + (\Pr[\text{Gm}_0(\mathcal{A})] - \Pr[\text{Gm}_1(\mathcal{A})]) \ .\end{aligned}$$

Games Gm_0, Gm_1 are identical-until-bad, so by the Fundamental Lemma of Game Playing [9] we have

$$\Pr[\text{Gm}_0(\mathcal{A})] - \Pr[\text{Gm}_1(\mathcal{A})] \le \Pr[\text{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \ .$$

Clearly $\Pr[\text{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \le 1/p$. Now we can work with Gm_1, where the public key X is a

Schnorr Identification				
t	ϵ	$\log(p_1)$	$\log(p_2)$	Speedup $s = (\log(p_1)/\log(p_2))^3$
2^{80}	2^{-48}	256	208	1.9
2^{64}	2^{-64}	256	192	2.4
2^{100}	2^{-156}	512	356	3

Schnorr Signatures					
t	q_h	ϵ	$\log(p_1)$	$\log(p_2)$	Speedup $s = (\log(p_1)/\log(p_2))^3$
2^{80}	2^{60}	2^{-48}	316	268	1.6
2^{64}	2^{50}	2^{-64}	306	242	2.0
2^{100}	2^{80}	2^{-156}	592	436	2.5

Figure 5: **Speedups yielded by our results for the Schnorr identification scheme $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ (top) and signature scheme $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$ (bottom).** The target for the first is that IMP-PA adversaries with running time t should have advantage at most ϵ. We show the log of the group size p_i required for this under prior results ($i = 1$), and our results ($i = 2$). Assuming exponentiation in $\mathbb{G}$ is cubic-time, we then show the speedup ratio of scheme algorithms. The target for the second is that UF adversaries with running time t, making q_h queries to H, should have advantage at most ϵ, and the table entries are analogous.

---

random element of $\mathbb{G}^*$ rather than of $\mathbb{G}$. We claim that

$$\Pr[\mathrm{Gm}_1(\mathcal{A}) = \Pr[\mathrm{Gm}_2(\mathcal{A})] \; . \tag{7}$$

We now justify this. At line 4, game Gm_2 picks x directly from $\mathbb{Z}_p^*$, just like Gm_1, and also rewrites FIN in a different but equivalent way. The main thing to check is that CH in Gm_2 is equivalent to that in Gm_1, meaning line 6 results in c_* being uniformly distributed in $\mathbb{Z}_p$. For this regard R_*, X as fixed and define the function $f_{R_*,X} : \mathbb{G} \to \mathbb{Z}_p$ by $f_{R_*,X}(Y) = \mathsf{DL}_{\mathbb{G},X}(R_*^{-1}Y)$. The adversary has no information about Y prior to receiving c_* at line 6, so the claim is established if we show that $f_{R_*,X}$ is a bijection. This is true because $X \in \mathbb{G}^*$ is a generator, which means that the function $h_{R_*,X} : \mathbb{Z}_p \to \mathbb{G}$ defined by $h_{R_*,X}(c_*) = R_* X^{c_*}$ is the inverse of $f_{R_*,X}$. This establishes Eq. (7).

We now claim that adversary $\mathcal{B}$, shown in Fig. 4, satisfies

$$\Pr[\mathrm{Gm}_2(\mathcal{A})] \leq \mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) \; . \tag{8}$$

Putting this together with the above completes the proof, so it remains to justify Eq. (8). Adversary $\mathcal{B}$ has access to oracle DLO as per game $\mathbf{G}^{\mathrm{mbdl}}_{\mathbb{G},g,1}$. In the code, CH and Tr are subroutines defined by $\mathcal{B}$ and used to simulate the oracles of the same names for $\mathcal{A}$. Adversary $\mathcal{B}$ has input the challenge Y whose discrete logarithm in base g it needs to compute, as well as the base X relative to which it may perform one discrete log operation. It runs $\mathcal{A}$ on input X, so that the latter functions as the public key, which is consistent with Gm_2. The subroutine CH uses DLO to produce c_* the same way as line 6 of Gm_2. It simulates Tr as per line 7 of Gm_2. If Gm_2 returns true at line 9 then we have $g^{z_*} = X^{c_*} R_* = W R_* = R_*^{-1} Y R_* = Y$, so $\mathcal{B}$ wins. ▌

QUANTITATIVE COMPARISON. Concrete security improvements are in the end efficiency improvements, because, for a given security level, we can use smaller parameters, and thus the scheme algorithms are faster. Here we quantify this, seeing what Eq. (6) buys us over Eq. (5) in terms of improved efficiency for the identification scheme.

We take as goal to ensure that any adversary $\mathcal{A}$ with running time t has advantage $\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \epsilon$ in violating IMP-PA security of $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$. Here t, ϵ are parameters for which many choices are possible. For example, $t = 2^{90}$ and $\epsilon = 2^{-32}$ is one choice, reflecting a 128-bit security level, where we define the bit-security level as $\log_2(t/\epsilon)$. The cost of scheme algorithms is the cost of exponentiation in the group, which is cubic in the representation size $k = \log p$ of group elements. So we ask what k must be to provably ensure the desired security. Equations (5) and (6) will yield different choices of k, denoted k_1 and k_2, with $k_2 < k_1$. We will conclude that Eq. (6) allows a $s = (k_1/k_2)^3$-fold speedup for the scheme.

Let $\mathcal{B}_1$ denote the DL adversary referred to in Eq. (5), and $\mathcal{B}_2$ the 1-MBDL adversary referred to in (6). To use the equations, we now need estimates on their respective advantages. For this, we assume $\mathbb{G}$ is a group in which the security of discrete-log-related problems is captured by the bounds proven in the generic group model (GGM), as seems to be true, to best of our current understanding, for certain elliptic curve groups. We will ignore the simulation overhead in running time since the number of transcript queries of $\mathcal{A}$ reflects online executions of the identification protocol and should be considerably less than the running time of $\mathcal{A}$, so that we take the running times of both $\mathcal{B}_1$ and $\mathcal{B}_2$ to be about t, the running time of our IMP-PA adversary $\mathcal{A}$. Now the classical result of Shoup [46] says that $\mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{B}_1) \approx t^2/p$, and our Theorem 5 says that also $\mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}_2) \approx t^2/p$.

Here we pause to highlight that these two bounds being the same is a central attribute of the 1-MBDL assumption. That Theorem 1 (as per Figure 5) provides efficiency improvements stems not just from the reduction of Eq. (6) being tight, but also from that fact that the 1-MBDL problem is just as hard to solve as the DL problem, meaning $\mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g}(\mathcal{B}_2) \approx \mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{B}_1) \approx t^2/p$.

Continuing, putting together what we have so far gives two bounds on the IMP-PA advantage of $\mathcal{A}$, the first via Equations (5) and the second via Eq. (6), namely, dropping the $1/p$ terms,

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \epsilon_1(t) = \sqrt{\frac{t^2}{p}} = \frac{t}{\sqrt{p}} \tag{9}$$

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \epsilon_2(t) = \frac{t^2}{p} \ . \tag{10}$$

Recall our goal was to ensure that $\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{SchID}[\mathbb{G},g]}(\mathcal{A}) \leq \epsilon$. We ask, what value of p, in either case, ensures this? Solving for p in the equations $\epsilon = \epsilon_1(t)$ and $\epsilon = \epsilon_2(t)$, we get two corresponding values, namely $p_1 \approx t^2/\epsilon^2$ and $p_2 \approx t^2/\epsilon$. We see that $p_1 > p_2$, meaning Theorem 1 guarantees the same security as Eq. (5) in groups of a smaller size. Finally, the ratio of representation sizes for group elements is

$$r \approx \frac{\log(p_1)}{\log(p_2)} \approx \frac{\log(t^2/\epsilon) + \log(1/\epsilon)}{\log(t^2/\epsilon)} = 1 + \frac{\log(1/\epsilon)}{\log(t^2/\epsilon)} \ .$$

Scheme algorithms employ exponentiation in the group and are thus cubic time, so the ratio of speeds is $s = r^3$, which we call the speedup factor, and we can now estimate it numerically. For a few values of t, ϵ, Figure 5 shows the log of the group size p_i needed to ensure the desired security under prior results ($i = 1$) and ours ($i = 2$). Then it shows the speedup s. For example if we want attacks of time $t = 2^{64}$ to achieve advantage at most $\epsilon = 2^{-64}$, prior results would require a group of size p_1 satisfying $\log(p_1) \approx 256$, while our results allow it with a group of size $\log(p_2) \approx 192$, which yields a 2.4x speedup. Of course many more examples are possible.

<u>Signature schemes.</u> Towards results on the Schnorr signature scheme, we start by recalling definitions. A signature scheme DS specifies key generation algorithm $\mathsf{DS.Kg}$, signing algorithm

```
Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}$

INIT:
1 $\mathsf{h} \leftarrow\!\!\$\ \mathsf{DS.HF}$ ; $(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}$
2 Return $vk$

SIGN($m$):
3 $\sigma \leftarrow\!\!\$\ \mathsf{DS.Sign}^{\mathrm{H}}(sk, m)$ ; $S \leftarrow S \cup \{m\}$
4 Return $\sigma$

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

FIN($m_*, \sigma_*$):
6 Return ( $(m_* \notin S)$ and $\mathsf{DS.Vf}^{\mathrm{H}}(vk, m_*, \sigma_*)$ )
```

Figure 6: Game defining UF security of signature scheme DS.

$\mathsf{DS.Sign}$, deterministic verification algorithm $\mathsf{DS.Vf}$ and a set $\mathsf{DS.HF}$ of functions called the hash function space. Via $(vk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}$ the signer generates a public verification key vk and secret signing key sk. Via $\sigma \leftarrow\!\!\$\ \mathsf{DS.Sign}^{\mathsf{h}}(sk, m)$ the signing algorithm takes sk and a message $m \in \{0,1\}^*$, and, with access to an oracle $\mathsf{h} \in \mathsf{DS.HF}$, returns a signature σ. Via $b \leftarrow \mathsf{DS.Vf}^{\mathsf{h}}(vk, m, \sigma)$, the verifier obtains a boolean decision $b \in \{\mathsf{true}, \mathsf{false}\}$ about the validity of the signature. The correctness requirement is that for all $\mathsf{h} \in \mathsf{DS.HF}$, all $(vk, sk) \in [\mathsf{DS.Kg}]$, all $m \in \{0,1\}^*$ and all $\sigma \in [\mathsf{DS.Sign}^{\mathsf{h}}(sk, m)]$ we have $\mathsf{DS.Vf}^{\mathsf{h}}(vk, m, \sigma) = \mathsf{true}$.

Game $\mathbf{G}^{\mathrm{uf}}$ in Fig. 6 captures UF (unforgeability under chosen-message attack) [30]. Procedure H is the random oracle [8], implemented as a function h chosen at random from $\mathsf{DS.HF}$. We define the UF advantage of adversary $\mathcal{A}$ as $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A})]$.

<u>SCHNORR SIGNATURES.</u> The Schnorr signature scheme $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$ is derived by applying the Fiat-Shamir transform [25] to the Schnorr identification scheme. Its algorithms are shown at the bottom right of Fig. 3. The set $\mathsf{DS.HF}$ consists of all functions $\mathsf{h} : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$.

<u>OUR AND PRIOR RESULTS.</u> We give a reduction, of the UF security of the Schnorr signature scheme to the 1-MBDL problem, that loses only a factor of the number of hash-oracle queries of the adversary. We start by recalling the following lemma from [1]. It derives the UF security of $\mathsf{SchSig}[G, g]$ from the IMP-PA security of $\mathsf{SchID}[G, g]$:

Lemma 2 [1] *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* $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ *and* $\mathsf{DS} = \mathsf{SchID}[\mathbb{G}, g]$ *be the Schnorr identification and signature schemes, respectively. Let $\mathcal{A}_{\mathrm{ds}}$ be an adversary attacking the uf-security of* DS*. Let $\alpha = (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}_{\mathrm{ds}}} + \mathrm{Q}^{\mathrm{SIGN}}_{\mathcal{A}_{\mathrm{ds}}})\mathrm{Q}^{\mathrm{SIGN}}_{\mathcal{A}_{\mathrm{ds}}}$. Then we can construct an adversary $\mathcal{A}_{\mathrm{id}}$ such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}_{\mathrm{ds}}) \leq (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}_{\mathrm{ds}}}) \cdot \mathbf{Adv}^{\mathrm{imp\text{-}pa}}_{\mathsf{ID}}(\mathcal{A}_{\mathrm{id}}) + \frac{\alpha}{p} .$$

Additionally, $\mathrm{T}_{\mathcal{A}_{\mathrm{id}}} \approx \mathrm{T}_{\mathcal{A}_{\mathrm{ds}}}$ and $\mathrm{Q}^{\mathrm{Tr}}_{\mathcal{A}_{\mathrm{id}}} = \mathrm{Q}^{\mathrm{SIGN}}_{\mathcal{A}_{\mathrm{ds}}}$.

Combining this with Theorem 1, we have:

Theorem 3 *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* $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$ *be the Schnorr signature scheme. Let $\mathcal{A}$ be an adversary attacking the uf security* *of* ID*. Let* $\beta = (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}} + \mathrm{Q}^{\textsc{Sign}}_{\mathcal{A}})\mathrm{Q}^{\textsc{Sign}}_{\mathcal{A}} + (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}})$*. Then we can construct an adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}) \cdot \mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) + \frac{\beta}{p} \ . \tag{11}$$

Additionally, $\mathrm{T}_{\mathcal{B}}$ *is roughly* $\mathrm{T}_{\mathcal{A}}$ *plus simulation overhead of the form* $\mathcal{O}(\mathrm{Q}^{\textsc{Sign}}_{\mathcal{A}} \cdot T^{\mathrm{exp}}_{\mathbb{G}})$*.*

Let's compare this to prior results. A simple proof of UF-security of DS from DL can be obtained by combining Lemma 2 with the classical DL-based security of ID as given by Eq. (5). For $\mathcal{A}$ an adversary attacking the UF security of DS, this would yield a discrete log adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}) \cdot \sqrt{\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B})} + \frac{\beta}{p} \ , \tag{12}$$

where β is as in Theorem 3 and $\mathrm{T}_{\mathcal{B}}$ is about $2\,\mathrm{T}_{\mathcal{A}}$ plus the same simulation overhead as above. This is however *not* the best prior bound. One can do better with a direct application of the general Forking Lemma of [4] as per [43]. For $\mathcal{A}$ an adversary attacking the UF security of DS, this would yield a discrete log adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le \sqrt{(1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}) \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B})} + \frac{\beta}{p} \ , \tag{13}$$

where β and $\mathrm{T}_{\mathcal{B}}$ are as above. The reason Eq. (13) is a better bound than Eq. (12) is that the $1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}$ term has moved under the square root. Still we see that Eq. (11) is even better; roughly (neglecting the additive term), the bound in Eq. (11) is the square of the one in Eq. (13), and thus (always) smaller.

<ins>QUANTITATIVE COMPARISONS.</ins> Our numerical comparisons will be with the best prior bound, meaning that of Eq. (13). For a few values of t, q_h, ϵ with $t \ge q_h = \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}$, Figure 5 shows the speedup s from Eq. (11) over Eq. (13). The table shows that the speedup is a bit less than for Schnorr identification shown in the same Figure, but still significant. For example if we want attacks of time $t = 2^{64}$ to achieve advantage at most $\epsilon = 2^{-64}$, Theorem 3 is allowing group sizes to go down enough to yield a 5.4-fold speedup.

To derive these estimates, we use the same framework and setup as we did for identification. Let $\mathbb{G}$ be a group of prime order p with generator g. We take as goal to ensure that any adversary $\mathcal{A}$ with running time t, making q_h queries to H and q_s queries to $\textsc{Sign}$, has advantage $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le \epsilon$ in violating UF security of $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$, where t, ϵ, q_h, q_s are parameters. We assume $q_s <$ $< q_h \le t$, as one expects in practice. Let $\mathcal{B}_1, \mathcal{B}_2$ be the adversaries of Equations (13) and (11), respectively. As before, assume $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}_1) \approx t^2/p$ from [46], and also $\mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}_2) \approx t^2/p$ from Theorem 5. Then

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le \epsilon_1(t, q_h) \approx \sqrt{\frac{q_h t^2}{p}}$$

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le \epsilon_2(t, q_h) \approx q_h \cdot \frac{t^2}{p} = \frac{q_h t^2}{p} \approx \epsilon_1(t, q_h)^2 \ .$$

In the estimates above, we have dropped the additive term, which has order $q_h q_s/p$, because this is negligible compared to the other term for reasonable parameter values, including the ones we consider. This leaves ϵ_1, ϵ_2 not depending on q_s, but recall the latter is expected to be (much) smaller than q_h. Then our bound ϵ_2 is about the square of the prior one, and thus always smaller.

We now ask what value of p ensures $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \le \epsilon$, in each case. Solving $\epsilon_1(t, q_h) \le \epsilon$ yields $p_1 \approx t^2 q_h/\epsilon^2$, and solving $\epsilon_2(t, q_h) \le \epsilon$ yields $p_2 \approx t^2 q_h/\epsilon$. As before we see that $p_2 < p_1$, meaning Theorem 1 guarantees security in groups of smaller size. The ratio of the representation-size of group elements is

$$r \approx \frac{\log(p_1)}{\log(p_2)} \approx \frac{\log(t^2 q_h/\epsilon) + \log(1/\epsilon)}{\log(t^2 q_h/\epsilon)} = 1 + \frac{\log(1/\epsilon)}{\log(t^2 q_h/\epsilon)} \ .$$

As before the ratio of speeds (speedup factor) is $s = r^3$, and we can now estimate it numerically. For a few values of t, ϵ, Figure 5 shows the log of the group size p_i needed to ensure the desired security under prior results ($i = 1$) and ours ($i = 2$). Then it shows the speedup s.

5 Improved security for multi-signatures

The security of the Schnorr-based BN multi-signature scheme was established under DL [4]. We give a tighter reduction from MBDL that allows smaller group sizes and improves efficiency. This is of current interest given the usage of multi-signatures in crypto-currencies and blockchains [37, 15, 21, 22].

Gaps found by [21] in some prior proofs of security for multi-signatures motivate some care. We feel that the reasons for such gaps go back to the definitions, and in particular to the very syntax of multi-signatures not being detailed enough. 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. We address these issues here by starting with a detailed syntax and a security definition written via a code-based game. The syntax views the signing protocol as described by a stateful algorithm, run separately by each player, the state in our definitions maintained by the overlying game.

<u>Multi-signature schemes.</u> A multi-signature scheme 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 $(vk, sk) \leftarrow_{\$} \mathsf{MS.Kg}$, the key generation algorithm generates public signature-verification key vk 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 $d \leftarrow \mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{vk}, m, \sigma)$, the verification algorithm deterministically outputs a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ indicating whether or not σ is a valid signature on message m under a vector $\boldsymbol{vk}$ 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 st and the vector $\mathbf{in}$ of received messages from the other parties, to compute an outgoing message σ (viewed as broadcast to the other parties) and an updated state st', written $(\sigma, \mathsf{st}') \leftarrow \mathsf{MS.Sign}^{\mathrm{H}}(\mathbf{in}, \mathsf{st})$. In the last round, σ is the signature that this party outputs. (See Figure 7.) *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.)

Some conventions will aid further definitions and scheme descriptions. A party's state 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; $\mathsf{st.vk}$ 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 false and may at some point be set to true. The state can, beyond these, have other components that vary from protocol to protocol.

```
Algorithm $\mathsf{Exec}^{\mathsf{h}}_{\mathsf{MS}}(\boldsymbol{sk}, \boldsymbol{vk}, m)$:
1 $n \leftarrow |\boldsymbol{vk}|$
2 For $j = 1, \ldots, n$ do
3     $\mathsf{st}_j \leftarrow \mathsf{StInit}(j, \boldsymbol{sk}[j], \boldsymbol{vk}, m)$
4 $\boldsymbol{b} \leftarrow (\varepsilon, \ldots, \varepsilon)$ // $n$-vector
5 For $i = 1, \ldots, \mathsf{MS.nr}$ do
6     For $j = 1, \ldots, n$ do
7         $(\sigma_j, \mathsf{st}_j) \leftarrow\!\$\ \mathsf{MS.Sign}^{\mathsf{h}}(\boldsymbol{b}, \mathsf{st}_j)$
8     $\boldsymbol{b} \leftarrow (\sigma_1, \ldots, \sigma_n)$
9 Return $\sigma_1$
```

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

FIN:
1 $\mathsf{h} \leftarrow\!\$\ \mathsf{MS.HF}$
2 For $i = 1, \ldots, n$ do
3     $(\boldsymbol{vk}[i], \boldsymbol{sk}[i]) \leftarrow\!\$\ \mathsf{MS.Kg}$
4 $\sigma \leftarrow\!\$\ \mathsf{Exec}^{\mathsf{h}}_{\mathsf{MS}}(\boldsymbol{sk}, \boldsymbol{vk}, m)$
5 $d \leftarrow \mathsf{MS.Vf}^{\mathsf{h}}(\boldsymbol{vk}, m, \sigma)$
6 Return $d$
```

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

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

NS($\boldsymbol{vk}, m$):
2 $\boldsymbol{vk}[1] \leftarrow vk$ ; $u \leftarrow u + 1$ ; $\boldsymbol{vk}_u \leftarrow \boldsymbol{vk}$ ; $m_u \leftarrow m$ ; $\mathsf{st}_u \leftarrow \mathsf{StInit}(1, sk, \boldsymbol{vk}, m)$
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$_j$($s, \boldsymbol{b}$): // $1 \leq j \leq \mathsf{MS.nr1}$
4 $(\sigma, \mathsf{st}_s) \leftarrow\!\$\ \mathsf{MS.Sign}^{\mathrm{H}}(\boldsymbol{b}, \mathsf{st}_s)$ ; Return $(out, \sigma)$

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

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

Figure 7: **Top left:** Procedure specifying an honest execution of the signing protocol associated with multi-signature scheme MS. **Top right:** Correctness game. **Bottom:** Unforgeability game.

(For example, Figure 8 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 $\mathsf{st} \leftarrow \mathsf{StInit}(j, sk, \boldsymbol{vk}, m)$ to initialize st by setting $\mathsf{st.n} \leftarrow |\boldsymbol{vk}|$; $\mathsf{st.me} \leftarrow j$; $\mathsf{st.rnd} \leftarrow 0$; $\mathsf{st.sk} \leftarrow sk$; $\mathsf{st.vk} \leftarrow \boldsymbol{vk}$; $\mathsf{st.msg} \leftarrow m$; $\mathsf{st.rej} \leftarrow \mathsf{false}$. 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 st' all return $\perp$ as the output message.

CORRECTNESS. Algorithm $\mathsf{Exec}_{\mathsf{MS}}$, shown in the left column of Fig. 7, executes the signing protocol of MS on input a vector $\boldsymbol{sk}$ of signing keys, a vector $\boldsymbol{vk}$ of matching verification keys with $|\boldsymbol{sk}| = |\boldsymbol{vk}|$, 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 $\boldsymbol{vk}$. The state st_j of party j at line 3 is initialized using the function StInit defined above. The loop at line 5 executes 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, $\boldsymbol{b}$ 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. 7 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. 7 captures the notion of unforgeability for multi-signatures from [4]. There is one honest player, namely player 1, 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 a vector $\boldsymbol{vk}$ of verification keys that the adversary can choose, possibly dishonestly, subject only to $\boldsymbol{vk}[1]$ being the verification key vk of player 1, as enforced by line 2. The first message of player 1 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 player 1. It is convenient for (later) proofs to have a separate signing oracle SIGN_j for each round $j \in [1..\mathsf{MS.nr}]$. It is required that any $\text{SIGN}_j(s, \cdot)$ satisfy $s \in [1..u]$, 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. Oracle H is the random oracle, simply calling h. Eventually the adversary calls FIN with a vector of verification keys, message and claimed signature. It wins if verification succeeds and the forgery was non-trivial. (At line 7, recall that if $\boldsymbol{x}$ is a vector, then $[\boldsymbol{x}] = \{\, \boldsymbol{x}[i] \;:\; 1 \le i \le |\boldsymbol{x}| \,\}$ is the set containing the components of $\vec{x}$.) The ms-uf-advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A})]$.

BN SCHEME AND PRIOR WORK. Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$ and let $\ell \ge 1$ be an integer. The associated BN [4] multi-signature scheme $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ is shown in detail, in our syntax, in Fig. 8. The set $\mathsf{MS.HF}$ consists of all functions 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 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 15, it returns $\bot$ if one of the received values fails to match its commitment. As per our conventions, when this happens, this player will always broadcast $\bot$ in the future, so for round 3 we assume lines 16,17 of round 2. These lines create the second component $\mathsf{st}.\boldsymbol{z}[j]$ of a Schnorr signature relative to the Schnorr-commitment $\mathsf{st}.R$ defined at line 16, 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 20 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 [4], 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.

We recall the prior result of [4]. 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_{ns} queries to NS. Suppose the number of parties (length of verification-key vector) in queries to NS and FIN is at most n. Let $a = 8q_{\text{ns}} + 1$ and $b = 2q + 16n^2 q_{\text{ns}}$. Let $p = |\mathbb{G}|$. Then BN [4] give a DL-adversary $\mathcal{A}_{\text{dl}}$ such that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\text{ms}}) \le \sqrt{(q + q_{\text{ns}}) \cdot \left(\mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{A}_{\text{dl}}) + \frac{a}{p} + \frac{b}{2^\ell} \right)}\,. \tag{14}$$

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

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

$\underline{\mathsf{MS.Vf}^{\mathrm{H}_0,\mathrm{H}_1}(\boldsymbol{vk}, m, \sigma)}:$
  $(R, z) \leftarrow \sigma$
  For $i = 1, \ldots, n$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{vk}, m))$
  Return ( $g^z = R \cdot \prod_{i=1}^{n} \boldsymbol{vk}[i]^{c_i}$ )
```

```
$\underline{\mathsf{MS.Sign}^{\mathrm{H}_0,\mathrm{H}_1}(\boldsymbol{b}, \mathsf{st})}:$
$j \leftarrow \mathsf{st.me}$ ; $n \leftarrow \mathsf{st.n}$ ; $m \leftarrow \mathsf{st.msg}$ ; $sk \leftarrow \mathsf{st.sk}$ ; $\boldsymbol{vk} \leftarrow \mathsf{st.vk}$
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 $\mathsf{st}.\boldsymbol{z}[j] \leftarrow \bot$
    Else $\mathsf{st}.R \leftarrow \prod_{i=1}^{n} \mathsf{st}.R[i]$ ; $c_j \leftarrow \mathrm{H}_1((j, R, \boldsymbol{vk}, m))$ ; $\mathsf{st}.\boldsymbol{z}[j] \leftarrow sk \cdot c_j + \mathsf{st}.r$
    $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$ ; 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 8: Algorithms of the BN Multi-signature scheme $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$, where $\mathbb{G}$ is a group of prime order p and g is a generator of $\mathbb{G}$.

the bound. This square-root results in large losses in security that must then be compensated for by picking larger groups, decreasing efficiency.

BN FROM MBDL. We instead give the following proof from 1-MBDL that avoids rewinding and the forking lemma, and thus the square-root.

Theorem 4 *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 $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_0, q_1, q_{\mathrm{ns}}$ 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 $\textsc{Fin}$ is at most n. Let $\alpha = q_{\mathrm{ns}}(2q_{\mathrm{ns}}q_0 + 2q_1 + q_{\mathrm{ns}})$ and $\beta = q_0(q_0 + n)$. Then we construct an adversary $\mathcal{A}_{\mathrm{mbdl}}$ for game $\mathbf{G}^{\mathrm{mbdl}}_{\mathbb{G},g,1}$ (shown explicitly in Figure 13) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \leq q_1 \cdot \mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{A}_{\mathrm{mbdl}}) + \frac{\alpha}{2p} + \frac{\beta}{2^{\ell}}\ . \qquad (15)$$

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

Above, q_0 is the number of distinct queries to 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. Also the running time of $\mathcal{A}_{\mathrm{mbdl}}$, as per convention excluding the time taken by its DLO oracle, is not the running time of $\mathcal{A}_{\mathrm{mbdl}}$,

```
INIT: // Games $\mathrm{Gm}_0$–$\mathrm{Gm}_7$
  $(vk, sk) \leftarrow\!\!\$\ \mathsf{MS.Kg}$ ; Return $vk$

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

SIGN$_0(s, \boldsymbol{t})$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1$
  $\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $\mathsf{cs} \leftarrow \mathsf{false}$ ; $\mathrm{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$ ; Return $R_{s,1}$

SIGN$_1(s, \boldsymbol{R})$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$
  $\boldsymbol{R}[1] \leftarrow R_{s,1}$
  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 $\perp$
  $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$ ; $c_{s,1} \leftarrow \mathrm{H}_1((1, R_s, \boldsymbol{vk}_s, m_s))$ ; $z_{s,1} \leftarrow sk \cdot c_{s,1} + r_{s,1}$
  Return $z_{s,1}$

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 ( $\mathsf{cs}$ and $\exists\, u' : x = (1, R_{u',1})$ ) then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{\mathrm{HF}_0[x] \leftarrow t_{u',1}}$
  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 $\mathrm{HF}_1[x]$

FIN$(\boldsymbol{vk}, m, (R, z))$: // Games $\mathrm{Gm}_0$–$\mathrm{Gm}_7$
  $n \leftarrow |\boldsymbol{vk}|$
  For $i = 1, \ldots, n$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{vk}, m))$
  $X \leftarrow \prod_{i=1}^{n} \boldsymbol{vk}[i]^{c_i}$ ; Return ($g^z = RX$)
```

Figure 9: Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ for proof of Lemma **??**. 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.

but the time for the execution of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\text{mbdl}}$, meaning the time taken by oracles is included. A lower bound on q_1 is the length of $\boldsymbol{vk}$ in $\mathcal{A}_{\text{ms}}$'s FIN query, so we can assume it is positive.

Numerical estimates of the improvement provided by Eq. (15) over Eq. (14) in terms of speedup can again be made in the same way as we have done for Schnorr identification and signatures. The calculations show that the speedup ratios for BN would be similar to those for Schnorr signatures shown in Figure 5, under reasonable assumptions on the various parameters involved.

Proof of Theorem 4: 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 SIGN$_1$, SIGN$_2$ for the first two rounds, but omit SIGN$_3$, since this round returns to the adversary only a quantity it could itself compute already. In FIN (for example Figure 9) we assume the query is non-trivial, meaning lines 6,7 of Figure 7 return true, and these lines are thus omitted. We start with games $\mathrm{Gm}_0, \mathrm{Gm}_1$ in Figure 9. Game Gm_0 includes the boxed code, and we claim that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \; . \tag{16}$$

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 $t_{u,1} = \mathrm{H}_0((1, R_{u,1}))$ may leak information about $R_{u,1}$. Rather than define $t_{u,1}$ 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 $t_{u,1} = \mathrm{H}_0((1, R_{u,1}))$, having $R_{u,1} = R_{u',1}$ would imply $t_{u,1} = t_{u',1}$. At line 6, now that the BN-commitments $\boldsymbol{t}$ of all players are known, the games ensure that $t_{u,1}$ indeed equals $\mathrm{H}_0((1, R_{u,1}))$. 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 14. The boolean cs ensures that bad is only set prior to the release of $R_{s,1}$, since the adversary can set it with probability one if it knows $R_{s,1}$. This justifies Eq. (16).

Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [9]

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

The probability of setting bad at line 4 is at most $(0 + 1 + \cdots + q_{\mathrm{ns}} - 1)/p$, while the probability of setting it at line 14 is at most $q_{\mathrm{ns}} q_0 / p$ so

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

Game Gm_2 changes the $\mathrm{NS}, \textsc{Sign}_0, \mathrm{H}_0$ oracles as shown in Figure 10, maintaining the other oracles of Gm_1 from Figure 9. It drops redundant code, which allows it to move the choice of $R_{s,1}$ to line 23. At line 37, 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 Gm_3 (oracles shown across Figures 10 and 9) aims to figure out the $R_{s,j}$-values of parties $j \neq 1$ before having to supply $R_{s,1}$, because we will later need these to program H_1 values. It does this by "inverting" the BN-commitments, meaning at line 27 it seeks inputs to H_0 that result in the BN-commitments in $\boldsymbol{t}$. If these cannot be found, then random values are chosen instead at line 28. (Not finding the inverses is not yet a bad event. It can happen with high probability. It becomes a bad event only at line 33 when the BN-commitments are verified.) The computation of t at that line is only to ensure that H_0 has been called; this variable will not be used. These steps do not change what the oracles return compared to Gm_2, so we have

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

Moving to game 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 27,28 is correct with high probability. The boxed code ensures that in Gm_4, it is always correct. Since $\mathrm{Gm}_3, \mathrm{Gm}_4$ are identical-until-bad we have

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

Line 33 can only set bad if $y_i = \boldsymbol{t}_s[i]$ for all i, due to line 32. So it is set only if there is a collision in 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

$$\Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_0^2 + n q_0}{2^\ell} \; . \tag{17}$$

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

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

$\mathrm{SIGN}_0(s, \boldsymbol{t})$: // Games $\mathrm{Gm}_3, \mathrm{Gm}_4$
$\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,1} \leftarrow\$ \mathbb{Z}_p$ ; $R_{s,1} \leftarrow g^{r_{s,1}}$ ; $\mathrm{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$
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 $\boldsymbol{R}^*_s[i] \leftarrow\$ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}^*_s[i]))$
Return $R_{s,1}$

$\mathrm{SIGN}_1(s, \boldsymbol{R})$: // Games $\mathrm{Gm}_3$, $\boxed{\mathrm{Gm}_4}$
$\boldsymbol{R}[1] \leftarrow R_{s,1}$
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 $\perp$
If $(\boldsymbol{R} \neq \boldsymbol{R}^*_s)$ then bad $\leftarrow$ true ; $\boxed{\boldsymbol{R} \leftarrow \boldsymbol{R}^*_s}$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$ ; $c_{s,1} \leftarrow \mathrm{H}_1((1, R_s, \boldsymbol{vk}_s, m_s))$ ; $z_{s,1} \leftarrow sk \cdot c_{s,1} + r_{s,1}$
Return $z_{s,1}$

$\mathrm{H}_0(x)$: // Games $\mathrm{Gm}_2$–$\mathrm{Gm}_7$
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 $\mathrm{HF}_0[x]$
```

Figure 10: Games for proof of Lemma ??.

In game Gm_4, the $\boldsymbol{R}$ queried to SIGN_1 is the same as the $\boldsymbol{R}^*$ determined in SIGN_0, allowing game Gm_5 (Figure 11) to move line 34 into SIGN_0 as line 42 and to simplify 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 $R_{s,1}$, it becomes possible to successfully program H_1 via the zero-knowledge simulation. Game Gm_6 of Figure 11 does this, setting bad at line 53 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 11) are identical-until-bad, so

$$\Pr[\mathrm{Gm}_6(\mathcal{A})] \leq \Pr[\mathrm{Gm}_7(\mathcal{A})] + \Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets bad}] . \tag{18}$$

When line 53 is executed, the adversary has as yet no information about R_s, which means

$$\Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets bad}] \leq \frac{q_{\mathrm{ns}} q_1}{p} . \tag{19}$$

Towards building adversary $\mathcal{A}_{\mathrm{mbdl}}$, we want to consider which H_1 query corresponds to the forgery submitted by $\mathcal{A}_{\mathrm{ms}}$ to FIN. For this, Figure 12 considers games $\mathrm{Gm}_{8,j}$ for $j \in [1..q_1]$. Game $\mathrm{Gm}_{8,j}$

```
$\text{SIGN}_0(s, \boldsymbol{t})$: // Game $\text{Gm}_5$
38 $\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$ ; $r_{s,1} \leftarrow\!\!\$\; \mathbb{Z}_p$ ; $R_{s,1} \leftarrow g^{r_{s,1}}$ ; $\text{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$
39 For $i = 1, \ldots, n_s$ do
40     If $(\text{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \text{HI}_0[i, \boldsymbol{t}_s[i]]$
41     Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\; \mathbb{G}$ ; $t \leftarrow \text{H}_0((i, \boldsymbol{R}_s^*[i]))$
42 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$ ; $c_{s,1} \leftarrow \text{H}_1((1, R_s, \boldsymbol{vk}_s, m_s))$ ; $z_{s,1} \leftarrow sk \cdot c_{s,1} + r_{s,1}$
43 Return $R_{s,1}$

$\text{SIGN}_1(s, \boldsymbol{R})$: // Game $\text{Gm}_5, \text{Gm}_6, \text{Gm}_7$
44 $\boldsymbol{R}[1] \leftarrow R_{s,1}$
45 For $i = 1, \ldots, n_s$ do $y_i \leftarrow \text{H}_0((i, \boldsymbol{R}[i]))$
46 If $(\, \exists\, i : y_i \neq \boldsymbol{t}_s[i]\, )$ then Return $\perp$ else Return $z_{s,1}$

$\text{SIGN}_0(s, \boldsymbol{t})$: // Game $\boxed{\text{Gm}_6}$, $\text{Gm}_7$
47 $\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
48 $c_{s,1} \leftarrow\!\!\$\; \mathbb{Z}_p$ ; $z_{s,1} \leftarrow\!\!\$\; \mathbb{Z}_p$ ; $R_{s,1} \leftarrow g^{z_{s,1}} vk^{-c_{s,1}}$ ; $\text{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$
49 For $i = 1, \ldots, n_s$ do
50     If $(\text{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \text{HI}_0[i, \boldsymbol{t}_s[i]]$
51     Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\; \mathbb{G}$ ; $t \leftarrow \text{H}_0((i, \boldsymbol{R}_s^*[i]))$
52 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$
53 If $(\text{HF}_1((1, R_s, \boldsymbol{vk}_s, m_s)) \neq \perp)$ then bad $\leftarrow$ true ; $\boxed{c_{s,1} \leftarrow \text{HF}_1[(1, R_s, \boldsymbol{vk}_s, m_s)]}$
54 $\text{HF}_1[(1, R_s, \boldsymbol{vk}_s, m_s)] \leftarrow c_{s,1}$ ; Return $R_{s,1}$
```

Figure 11: Games for proof of Lemma **??**.

records H_1 queries and returns true if the forgery corresponded to the j-th query, so

$$\Pr[\text{Gm}_7(\mathcal{A})] \leq \sum_{j=1}^{q_1} \Pr[\text{Gm}_{8,j}(\mathcal{A})] \ . \tag{20}$$

The second step is to build adversary $\mathcal{A}_{\text{mbdl}}$ so that

$$\mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,1}(\mathcal{A}_{\text{mbdl}}) \geq \frac{1}{q_1} \sum_{j=1}^{q_1} \Pr[\text{Gm}_{8,j}(\mathcal{A})] \ . \tag{21}$$

We specify $\mathcal{A}_{\text{mbdl}}$ in Figure 13. It sets the public key for $\mathcal{A}_{\text{ms}}$ to its input X_1 and runs $\mathcal{A}_{\text{ms}}$. It makes a guess j and hopes $\mathcal{A}_{\text{ms}}$ would win in game $\text{Gm}_{8,j}$. Simulating signatures without knowing the secret key, as $\mathcal{A}_{\text{mbdl}}$ needs to do, is now easy because the oracles of games $\text{Gm}_{8,\cdot}$ already did this, and $\mathcal{A}_{\text{mbdl}}$ can just use the same code. Line 19 considers the j-th query. Line 21 defines W and queries the DLO oracle to get its discrete log in base X_1, which is set as the player-1 challenge c_1 and then as the reply to the H_1 query. We have $X_1^{c_1} = W$, so if the forgery is successful we have

$$g^z = R \cdot \prod_{i=1}^{n} \boldsymbol{vk}[i]^{c_i} = R \cdot X_1^{c_1} \cdot \prod_{i=2}^{n} \boldsymbol{vk}[i]^{c_i} = R \cdot W \cdot \prod_{i=2}^{n} \boldsymbol{vk}[i]^{c_i} = Y \ .$$

So $\mathcal{A}_{\text{mbdl}}$ wins game $\mathbf{G}^{\text{mbdl}}_{\mathbb{G},g,1}$. Eq. (15) is obtained by putting the above together. ▌

```
INIT: // Games $\mathrm{Gm}_{8,j}$
$(vk, sk) \leftarrow\!\!\$\ \mathsf{MS.Kg}$ ; Return $vk$

NS($\boldsymbol{vk}, m$): // Games $\mathrm{Gm}_{8,j}$
$u \leftarrow u+1$ ; $\boldsymbol{vk}[1] \leftarrow \mathrm{vk}$ ; $\boldsymbol{vk}_u \leftarrow \boldsymbol{vk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{vk}|$
$t_{u,1} \leftarrow\!\!\$\ \{0,1\}^{\ell}$ ; Return $t_{u,1}$

SIGN$_0$($s, \boldsymbol{t}$): // Games $\mathrm{Gm}_{8,j}$
$\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
$c_{s,1} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $z_{s,1} \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $R_{s,1} \leftarrow g^{z_{s,1}} \mathrm{vk}^{-c_{s,1}}$ ; $\mathrm{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$
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 $\boldsymbol{R}^*_s[i] \leftarrow\!\!\$\ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}^*_s[i]))$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}^*_s[i]$ ; $Q \leftarrow Q \cup \{(R_s, \boldsymbol{vk}_s, m_s)\}$
$\mathrm{HF}_1[(1, R_s, \boldsymbol{vk}_s, m_s)] \leftarrow c_{s,1}$ ; Return $R_{s,1}$

SIGN$_1$($s, \boldsymbol{R}$): // Games $\mathrm{Gm}_{8,j}$
$\boldsymbol{R}[1] \leftarrow R_{s,1}$
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 $z_{s,1}$

H$_0$($x$): // Games $\mathrm{Gm}_{8,j}$
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 $\mathrm{HF}_0[x]$

H$_1$($x$): // Games $\mathrm{Gm}_{8,j}$
If ($\mathrm{HF}_1[x] \neq \bot$) then Return $\mathrm{HF}_1[x]$
$(i, R, \boldsymbol{vk}, m) \leftarrow x$
If ($i = 1$) then $v \leftarrow v+1$ ; $x_v \leftarrow (R, \boldsymbol{vk}, m)$
$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$ ; Return $\mathrm{HF}_1[x]$

FIN($\boldsymbol{vk}, m, (R, z)$): // Games $\mathrm{Gm}_{8,j}$
For $i = 1, \ldots, |\boldsymbol{vk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{vk}, m))$
$X \leftarrow \prod_{i=1}^{|\boldsymbol{vk}|} \boldsymbol{vk}[i]^{c_i}$
Return ( $(R, \boldsymbol{vk}, m) = x_j$ and $(R, \boldsymbol{vk}, m) \notin Q$ and $(g^z = RX)$ )
```

Figure 12: Games $\mathrm{Gm}_{8,j}$ ($j \in [1..q_1]$) for proof of Theorem 4.

6 MBDL hardness in the Generic Group Model

With a new problem like MBDL it is important to give evidence of hardness. Here we provide this in the most common and accepted form, namely a proof of hardness in the generic group model (GGM).

The quantitative aspect of the result is just as important as the qualitative. Theorem 5 below says that the advantage of a GGM adversary $\mathcal{A}$ in breaking n-MBDL is $\mathcal{O}(q^2/p)$ where q is n plus the number of group operations (time) invested by $\mathcal{A}$, namely about the same as the ggm-dl-advantage of an adversary of the same resources. Reductions (to some problem) from MBDL that are tighter than ones from DL now bear fruit in justifying the secure use of smaller groups, which lowers costs.

```
Adversary $\mathcal{A}_{\mathrm{mbdl}}^{\mathrm{DLO}}(Y, X_1)$:
$vk \leftarrow X_1$ ; $j \leftarrow\$ [1..q_1]$ ; $(\boldsymbol{vk}, m, (R, z)) \leftarrow\$ \mathcal{A}^{\mathrm{NS},\mathrm{SIGN}_0,\mathrm{SIGN}_1,\mathrm{H}_0,\mathrm{H}_1}(vk)$ ; Return $z$
NS($\boldsymbol{vk}, m$):
$u \leftarrow u+1$ ; $\boldsymbol{vk}[1] \leftarrow vk$ ; $\boldsymbol{vk}_u \leftarrow \boldsymbol{vk}$ ; $m_u \leftarrow m$ ; $n_u \leftarrow |\boldsymbol{vk}|$
$t_{u,1} \leftarrow\$ \{0,1\}^{\ell}$ ; Return $t_{u,1}$
SIGN$_0(s, \boldsymbol{t})$:
$\boldsymbol{t}[1] \leftarrow t_{s,1}$ ; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$
$c_{s,1} \leftarrow\$ \mathbb{Z}_p$ ; $z_{s,1} \leftarrow\$ \mathbb{Z}_p$ ; $R_{s,1} \leftarrow g^{z_{s,1}} vk^{-c_{s,1}}$ ; $\mathrm{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$
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 $\boldsymbol{R}_s^*[i] \leftarrow\$ \mathbb{G}$ ; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$
$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$ ; $\mathrm{HF}_1[(1, R_s, \boldsymbol{vk}_s, m_s)] \leftarrow c_{s,1}$ ; Return $R_{s,1}$
SIGN$_1(s, \boldsymbol{R})$:
$\boldsymbol{R}[1] \leftarrow R_{s,1}$
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 $z_{s,1}$
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 $\mathrm{HF}_0[x]$
H$_1(x)$:
If ($\mathrm{HF}_1[x] \neq \bot$) then Return $\mathrm{HF}_1[x]$
$(k, R, \boldsymbol{vk}, m) \leftarrow x$ ; $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$
If ($k = 1$) then
  $v \leftarrow v + 1$ ; $x_v \leftarrow (R, \boldsymbol{vk}, m)$
  If ($v = j$) then
    For $i = 2, \ldots, |\boldsymbol{vk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{vk}, m))$
    $W \leftarrow Y \cdot R^{-1} \cdot \prod_{i=2}^{|\boldsymbol{vk}|} \boldsymbol{vk}[i]^{-c_i}$ ; $c_1 \leftarrow \mathrm{DLO}(1, W)$ ; $\mathrm{HF}_1[x] \leftarrow c_1$
Return $\mathrm{HF}_1[x]$
```

Figure 13: Adversary $\mathcal{A}_{\mathrm{mbdl}}$ for Theorem 4.

The proof of Theorem 5 begins with a Lemma that characterizes the distribution of replies to the DLO query. A game sequence is then used to reduce bounding the adversary advantage to some static problems in linear algebra.

Some prior proofs in the GGM have been found to be wrong (that of [12] as pointed out by [31]) and we, at least, have often found GGM proofs imprecise and hard to verify. This has motivated us to try to be precise with definitions and to attend to details. Starting with definitions, we associate to any encoding function E an explicit binary operation op_{E} that turns the range-set of E into a group. A random choice of E then results in the GGM, with the "generic group" being now explicitly defined as the group associated to E. The proof uses a game sequence and has been done at a level of detail that is perhaps unusual in this domain.

<u>MBDL IN THE GGM.</u> We start with definitions. Suppose G is a set whose size $p = |G|$ is a prime,

```
Game $\mathbf{G}^{\text{gg-mbdl}}_{G,n}$   // Set $G$ is the range of the encoding, $|G|$ is prime.

INIT():
  $p \leftarrow |G|$ ; $\mathrm{E} \leftarrow\!\!\$\ \mathrm{Bijections}(\mathbb{Z}_p, G)$   // Think " $\mathrm{E}(x) = g^x$ "
  $\mathbb{1} \leftarrow \mathrm{E}(0)$ ; $g \leftarrow \mathrm{E}(1)$   // Think $\mathbb{1}$ the identity and $g$ generator
  $y \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $Y \leftarrow \mathrm{E}(y)$
  For $i = 1, \ldots, n$ do $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$ ; $X_i \leftarrow \mathrm{E}(x_i)$
  $\mathrm{GL} \leftarrow \{\mathbb{1}, g, Y, X_1, \ldots, X_n\}$
  Return $\mathbb{1}, g, Y, X_1, \ldots, X_n$

OP($A, B, \mathsf{sgn}$):  // $A, B \in G$ and $\mathsf{sgn} \in \{+, -\}$
  If ($A \notin \mathrm{GL}$ or $B \notin \mathrm{GL}$) then return $\perp$
  $c \leftarrow (\mathrm{E}^{-1}(A)\ \mathsf{sgn}\ \mathrm{E}^{-1}(B)) \bmod p$ ; $C \leftarrow \mathrm{E}(c)$ ; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{C\}$
  Return $C$

DLO($i, W$):  // $i \in [1..n]$ and $W \in G$
  If ($W \notin \mathrm{GL}$) then return $\perp$
  $z \leftarrow x_i^{-1} \cdot \mathrm{E}^{-1}(W) \bmod p$   // $x_i^{-1}$ is inverse of $x_i$ mod $p$
  Return $z$

FIN($y'$):
  Return ($y = y'$)
```

Figure 14: Game defining n-MBDL problem in the generic group model.

and $\mathrm{E}: \mathbb{Z}_p \to G$ is a bijection, called the encoding function. For $A, B \in G$, define $A \ \mathsf{op}_{\mathrm{E}}\ B = \mathrm{E}(\mathrm{E}^{-1}(A) + \mathrm{E}^{-1}(B))$. Then G is a group under the operation op_{E} [50], with identity element $\mathrm{E}(0)$, and the encoding function becomes a group homomorphism: $\mathrm{E}(a+b) = \mathrm{E}(a)\ \mathsf{op}_{\mathrm{E}}\ \mathrm{E}(b)$ for all $a, b \in \mathbb{Z}_p$. The element $g = \mathrm{E}(1) \in G$ is a generator of this group, and $\mathrm{E}^{-1}(A)$ is then the discrete logarithm of $A \in G$ relative to g. We call op_{E} the group operation on G induced by E.

In the GGM, the encoding function E is picked at random and the adversary is given an oracle for the group operation op_{E} induced on G by E. Game $\mathbf{G}^{\text{gg-mbdl}}_{G,n}$ in Fig. 14 defines, in this way, the n-MBDL problem. The set G parameterizes the game, and the random choice of encoding function $\mathrm{E}: \mathbb{Z}_p \to G$ is shown at line 1. Procedure OP then implements either the group operation op_{E} on G induced by E (when sgn is $+$) or its inverse (when sgn is $-$). Lines 3,4 pick $y, x_1, \ldots, x_n$ and define the corresponding group elements $Y, X_1, \ldots, X_n$. Set GL holds all group elements generated so far. The new element here is the oracle DLO that takes $i \in [1..n]$ and $W \in G$ to return the discrete logarithm of W in base X_i. This being x_i^{-1} times the discrete logarithm of W in base g, the procedure returns $z \leftarrow x_i^{-1} \cdot \mathrm{E}^{-1}(W)$. The inverse and the operations here are modulo p. Only one query to this oracle is allowed, and the adversary wins if it halts with output y' that equals y. We let $\mathbf{Adv}^{\text{gg-mbdl}}_{G,n}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{gg-mbdl}}_{G,n}(\mathcal{A})]$ be its ggm-mbdl-advantage.

RESULT. The following upper bounds the ggm-mbdl-advantage of an adversary $\mathcal{A}$ as a function of the number of its OP queries and n.

Theorem 5 *Let G be a set whose size $p = |G|$ is a prime. Let $n \geq 1$ be an integer. Let $\mathcal{A}$ be an adversary making $\mathrm{Q}^{\mathrm{OP}}_{\mathcal{A}}$ queries to its* OP *oracle and one query to its* DLO *oracle. Let*

$q = \mathrm{Q}_{\mathcal{A}}^{\mathrm{Op}} + n + 3$. *Then*

$$\mathbf{Adv}_{G,n}^{\text{gg-mbdl}}(\mathcal{A}) \leq \frac{2 + q(q-1)}{p-1} \ . \tag{22}$$

<u>Proof framework and lemma.</u> Much of our work in the proof is over $\mathbb{Z}_p^{n+2}$ regarded as a vector space over $\mathbb{Z}_p$. We let $\vec{0} \in \mathbb{Z}_p^{n+2}$ be the all-zero vector, and $\vec{e}_i \in \mathbb{Z}_p^{n+2}$ the i-th basis vector, meaning it has a 1 in position i and zeros elsewhere. We let $\langle \vec{a}, \vec{b} \rangle = (\vec{a}[1]\vec{b}[1] + \cdots + \vec{a}[n+2]\vec{b}[n+2])$ denote the inner product of vectors $\vec{a}, \vec{b} \in \mathbb{Z}_p^{n+2}$, where the operations are modulo p.

In the GGM, the encoding function takes as input a point in $\mathbb{Z}_p$. The proof of GGM hardness of the DL problem [46] moved to a modified encoding function that took input a univariate polynomial, the variable representing the target discrete logarithm y. We extend this to have the modified encoding function take input a degree one polynomial in $n+1$ variables, these representing $x_1, \ldots, x_n, y$. The polynomial will be represented by the vector of its coefficients, so that representations, formally, are vectors in $\mathbb{Z}_p^{n+2}$. At some point, games in our proof will need to simulate the reply to a $\mathrm{DLO}(i, W)$ query, meaning provide a reply z without knowing x_i. At this point, $W \in G$ will be represented by a vector $\vec{w} \in \mathbb{Z}_p^{n+2}$ that is known to the game and adversary. The natural simulation approach is to return a random $z \leftarrow\$ \mathbb{Z}_p$ or $z \leftarrow\$ \mathbb{Z}_p^*$, but these turn out to not perfectly mimic the true distribution of replies, because this distribution depends on $\vec{w}$. We start with a lemma that describes how to do a perfect simulation.

While the above serves as motivation for the Lemma, the Lemma itself is self-contained, making no reference to the DLO oracle. We consider the games of Figure 15. They are played with an adversary making a single Init query whose arguments are an integer $i \in [1..n]$ and a vector $\vec{w} \in \mathbb{Z}_p^{n+2}$. The operations in the games, including inverses of elements in $\mathbb{Z}_p^*$, are in the field $\mathbb{Z}_p$. Game $\mathbf{G}_{p,n}^{\mathrm{smp1}}$ captures what, in our vector-representation, will be the "real" game, with z at line 3 computed correctly as a function of x_i. Game $\mathbf{G}_{p,n}^{\mathrm{smp0}}$ represents the simulation, first picking z and then defining x_i. Lines 3,4 show that there are two cases for how z, x_i are chosen in the simulation, depending on the value of $w = \vec{w}[i]$ and the inner product of $\vec{w}$ with $\vec{x}_i$. The games return all variables involved. The claim is that the outputs of the games are identically distributed, captured formally, in the statement of Lemma 6 below, as the condition that any adversary returns `true` with the same probability in the two games.

Lemma 6 *Let p be a prime and $n \geq 1$ an integer. Then for any adversary $\mathcal{A}$ we have*

$$\Pr[\mathbf{G}_{p,n}^{\mathrm{smp1}}(\mathcal{A})] = \Pr[\mathbf{G}_{p,n}^{\mathrm{smp0}}(\mathcal{A})] \ , \tag{23}$$

where the games are in Figure 15.

Proof of Lemma 6: With $i, \vec{w}$ being $\mathcal{A}$'s query to Init, we can regard vector $\vec{x}_i = (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n, 1, y)$ as fixed, since its constituents are chosen identically in the two games. Let $\alpha = \langle \vec{w}, \vec{x}_i \rangle$. Now consider two cases. The first is that $\alpha = 0$. Then, in both games, $z = w$, and x_i is chosen randomly from $\mathbb{Z}_p^*$. The second case is that $\alpha \neq 0$. For $x \in \mathbb{Z}_p^*$ let $Z_{w,\alpha}(x) = w + x^{-1} \cdot \alpha$, so that $z = Z_{w,\alpha}(x_i)$ at line 3 of game $\mathbf{G}_{p,n}^{\mathrm{smp1}}$. That $\alpha \neq 0$ implies $Z_{w,\alpha}(x) \neq w$, meaning the function $Z_{w,\alpha}$ maps as $Z_{w,\alpha} : \mathbb{Z}_p^* \to \mathbb{Z}_p \setminus \{w\}$. For $z \in \mathbb{Z}_p \setminus \{w\}$, let $X_{w,\alpha}(z) = \alpha \cdot (z - w)^{-1}$, so that $x_i = X_{w,\alpha}(z)$ at line 3 of game $\mathbf{G}_{p,n}^{\mathrm{smp0}}$. That $z \neq w$ and $\alpha \neq 0$ means $X_{w,\alpha}(z) \neq 0$, meaning the function $X_{w,\alpha}$ maps as $X_{w,\alpha} : \mathbb{Z}_p \setminus \{w\} \to \mathbb{Z}_p^*$. The proof is complete if we show that these functions are inverses of each other, in particular showing that both are bijections. Indeed, for any $x \in \mathbb{Z}_p^*$ we have $X_{w,\alpha}(Z_{w,\alpha}(x)) = X_{w,\alpha}(w + x^{-1} \cdot \alpha) = \alpha \cdot (w + x^{-1} \cdot \alpha - w)^{-1} = \alpha \cdot x \cdot \alpha^{-1} = x$. ▌

Equipped with this lemma, we give the proof of Theorem 5.

```
Game $\mathbf{G}^{\mathrm{smp1}}_{p,n}$

INIT$(i, \vec{w})$: // $i \in [1..n]$ and $\vec{w} \in \mathbb{Z}_p^{n+2}$
    $x_1, \ldots, x_n \leftarrow\!\!\$\ \mathbb{Z}_p^*$ ; $y \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $w \leftarrow \vec{w}[i]$
    $\vec{x}_i \leftarrow (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n, 1, y)$
    $z \leftarrow w + x_i^{-1} \cdot \langle \vec{w}, \vec{x}_i \rangle$ ; Return $(x_1, \ldots, x_n, y, z)$

Game $\mathbf{G}^{\mathrm{smp0}}_{p,n}$

INIT$(i, \vec{w})$: // $i \in [1..n]$ and $\vec{w} \in \mathbb{Z}_p^{n+2}$
    $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \leftarrow\!\!\$\ \mathbb{Z}_p^*$ ; $y \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $w \leftarrow \vec{w}[i]$
    $\vec{x}_i \leftarrow (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n, 1, y)$
    If ( $\langle \vec{w}, \vec{x}_i \rangle \neq 0$ ) then $z \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{w\}$ ; $x_i \leftarrow (z-w)^{-1} \cdot \langle \vec{w}, \vec{x}_i \rangle$
    Else $z \leftarrow w$ ; $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$
    Return $(x_1, \ldots, x_n, y, z)$
```

Figure 15: Games for Lemma 6.

```
INIT(): // $\mathrm{Gm}_0$–$\mathrm{Gm}_3$
1   $p \leftarrow |G|$ ; E $\leftarrow\!\!\$$ Bijections$(\mathbb{Z}_p, G)$ ; $y \leftarrow\!\!\$\ \mathbb{Z}_p$
2   For $i = 1, \ldots, n$ do $x_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$
3   $\vec{x} \leftarrow (x_1, \ldots, x_n, 1, y)$ ; $\vec{v} \leftarrow \vec{0}$
4   $\mathbb{1} \leftarrow \mathrm{VE}(\vec{0})$ ; $g \leftarrow \mathrm{VE}(\vec{e}_{n+1})$ ; $Y \leftarrow \mathrm{VE}(\vec{e}_{n+2})$
5   For $i = 1, \ldots, n$ do $X_i \leftarrow \mathrm{VE}(\vec{e}_i)$
6   Return $\mathbb{1}, g, Y, X_1, \ldots, X_n$

VE$(\vec{t})$: // $\mathrm{Gm}_0$. Here $\vec{t} \in \mathbb{Z}_p^{n+2}$.
7   If ( $\mathrm{TV}[\vec{t}] \neq \bot$ ) then return $\mathrm{TV}[\vec{t}]$
8   $v \leftarrow \langle \vec{t}, \vec{x} \rangle$ ; $C \leftarrow \mathrm{E}(v)$ ; $\mathrm{TV}[\vec{t}] \leftarrow C$ ; $\mathrm{TI}[C] \leftarrow \vec{t}$ ; Return $\mathrm{TV}[\vec{t}]$

VE$^{-1}(C)$: // $\mathrm{Gm}_0$–$\mathrm{Gm}_3$. Here $\mathrm{TI}[C] \neq \bot$.
9   Return $\mathrm{TI}[C]$

OP$(A, B, \mathsf{sgn})$: // $\mathrm{Gm}_0$–$\mathrm{Gm}_3$. Here $\mathrm{TI}[A], \mathrm{TI}[B] \neq \bot$ and $\mathsf{sgn} \in \{+, -\}$
10  $\vec{c} \leftarrow \mathrm{VE}^{-1}(A)\ \mathsf{sgn}\ \mathrm{VE}^{-1}(B)$ ; $C \leftarrow \mathrm{VE}(\vec{c})$ ; Return $C$

DLO$(i, W)$: // $\mathrm{Gm}_0$. Here $i \in [n]$ and $\mathrm{TI}[W] \neq \bot$.
11  $\vec{w} \leftarrow \mathrm{VE}^{-1}(W)$ ; $z \leftarrow (x_i)^{-1} \cdot \langle \vec{w}, \vec{x} \rangle$ ; Return $z$

FIN$(y')$: // $\mathrm{Gm}_0$–$\mathrm{Gm}_3$
12  Return $(y = y')$
```

Figure 16: Game Gm_0 for the proof of Theorem 5. Some procedures will also be in later games, as marked.

Proof of Theorem 5: By span$(\vec{v})$ we denote the span of a vector $\vec{v} \in \mathbb{Z}_p^{n+2}$, which simply means the set of all $a \cdot \vec{v}$ as a ranges over $\mathbb{Z}_p$. Beyond the procedures of game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$, some of our games define procedures VE and VE^{-1}, the vector-encoding and its inverse. These procedures are not exported, meaning can be called only by other game procedures, not by the adversary.

```
VE($\vec{t}$): // $\boxed{\mathrm{Gm}_1}$, $\mathrm{Gm}_2$. Here $\vec{t} \in \mathbb{Z}_p^{n+2}$.
13 If ( $\mathrm{TV}[\vec{t}] \neq \perp$ ) then return $\mathrm{TV}[\vec{t}]$
14 If ( $\exists\, \vec{t}'$ : ( $\mathrm{TE}[\vec{t}'] \neq \perp$ and $\vec{t} - \vec{t}' \in \mathrm{span}(\vec{v})$ ) ) then
15     $C \leftarrow \mathrm{TV}[\vec{t}']$ ; $\mathrm{TV}[\vec{t}] \leftarrow C$ ; $\mathrm{TI}[C] \leftarrow \vec{t}$ ; Return $\mathrm{TV}[\vec{t}]$
16 $C \leftarrow\!\!\$\; G \setminus \mathrm{GL}$
17 If ( $\exists\, \vec{t}'$ : ( $\mathrm{TE}[\vec{t}'] \neq \perp$ and $\langle \vec{t}, \vec{x} \rangle = \langle \vec{t}', \vec{x} \rangle$ ) ) then
18     bad $\leftarrow$ true ; $\boxed{C \leftarrow \mathrm{TV}[\vec{t}']}$
19 $\mathrm{TV}[\vec{t}] \leftarrow C$ ; $\mathrm{TI}[C] \leftarrow \vec{t}$ ; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{C\}$ ; Return $\mathrm{TV}[\vec{t}]$

DLO($i, W$): // $\mathrm{Gm}_1$, $\mathrm{Gm}_2$. Here $i \in [n]$ and $\mathrm{TI}[W] \neq \perp$.
20 $\vec{w} \leftarrow \mathrm{VE}^{-1}(W)$ ; $z \leftarrow (x_i)^{-1} \cdot \langle \vec{w}, \vec{x} \rangle$ ; $\vec{v} \leftarrow \vec{w} - z \cdot \vec{e}_i$
21 Return $z$
```

Figure 17: Procedures for games $\mathrm{Gm}_1, \mathrm{Gm}_2, \mathrm{Gm}_3$ in the proof of Theorem 5, where Gm_1 includes the boxed code.

Throughout, we assume the adversary $\mathcal{A}$ makes no trivial queries. By this we mean that the checks at lines 7 and 10 of game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$ are not triggered. In our games the consequence is that we assume $\mathrm{TI}[A], \mathrm{TI}[B] \neq \perp$ in any $\textsc{Op}(A, B, \mathsf{sgn})$ query and, for a $\mathrm{DLO}(i, W)$ query, that $i \in [n]$, that $\mathrm{TI}[W] \neq \perp$ and that the number of queries to this oracle is exactly $m = 1$. (The table $\mathrm{TI}[\cdot]$ referred to here starts appearing in Game Gm_0 of Figure 16.)

We start with game Gm_0 of Figure 16, claiming that

$$\mathbf{Adv}^{\text{gg-mbdl}}_{G,n,m}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \; . \tag{24}$$

We now explain the game and justify Eq. (24). At line 10, operation sgn is performed modulo p, and at line 11, the inverse and product in computing z are modulo p. The game picks $y, x_1, \ldots, x_n$ in the same way as game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$. At line 1, it also picks encoding function E in the same way as game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$, but does not use this function directly to do the encoding, instead calling VE, which we call the vector-encoding function, on the indicated vector arguments. This procedure maintains tables $\mathrm{TV}\colon \mathbb{Z}_p^{n+2} \to G \cup \{\perp\}$ and $\mathrm{TI}\colon G \to \mathbb{Z}_p^{n+2} \cup \{\perp\}$ (the "I" stands for "inverse") that from the code can be seen to satisfy the following, where vector $\vec{x}$ is defined at line 3:

(1) If $\mathrm{TV}[\vec{t}] \neq \perp$ then $\mathrm{TV}[\vec{t}] = \mathrm{E}(\langle \vec{t}, \vec{x} \rangle)$
(2) If $\mathrm{TI}[C] \neq \perp$ then $\langle \mathrm{TI}[C], \vec{x} \rangle = \mathrm{E}^{-1}(C)$

This ensures Eq. (24) as follows. From line 4 and the above we have $g = \mathrm{TV}[\vec{e}_{n+1}] = \mathrm{E}(\langle \vec{e}_{n+1}, \vec{x} \rangle) = \mathrm{E}(1)$, and, similarly, we have $Y = \mathrm{E}(y)$ and $X_i = \mathrm{E}(x_i)$ for $i \in [1..n]$, meaning these quantities are as in game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$. Turning to $\textsc{Op}$, by linearity of the inner product and item (2) above, we have

$$\begin{aligned} \langle \vec{c}, \vec{x} \rangle = \langle \mathrm{TI}[A] \,\mathsf{sgn}\, \mathrm{TI}[B], \vec{x} \rangle &= \langle \mathrm{TI}[A], \vec{x} \rangle \,\mathsf{sgn}\, \langle \mathrm{TI}[B], \vec{x} \rangle \\ &= \mathrm{E}^{-1}(A) \,\mathsf{sgn}\, \mathrm{E}^{-1}(B) \; , \end{aligned}$$

so by item (1) we have $\mathrm{VE}(\vec{c}) = \mathrm{E}(\mathrm{E}^{-1}(A) \,\mathsf{sgn}\, \mathrm{E}^{-1}(B))$, as in game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$. Finally, for DLO, item (2) says that $\langle \vec{w}, \vec{x} \rangle = \mathrm{E}^{-1}(W)$, again as in game $\mathbf{G}^{\text{gg-mbdl}}_{G,n,m}$.

Games $\mathrm{Gm}_1, \mathrm{Gm}_2$ are formed by taking the indicated procedures of Figure 16 and adding those of Figure 17, with the former game including the boxed code, and the latter not. Procedure VE no longer invokes E, instead sampling it lazily. The vector $\vec{v}$ defined at line 20 satisfies $\langle \vec{v}, \vec{x} \rangle = \langle \vec{w} - z \cdot \vec{e}_i, \vec{x} \rangle = \langle \vec{w}, \vec{x} \rangle - z \cdot \langle \vec{e}_i, \vec{x} \rangle = \langle \vec{w}, \vec{x} \rangle - x_i^{-1} \cdot \langle \vec{w}, \vec{x} \rangle \cdot x_i = 0$. As a result, at any time, any

```
INIT(): // $\mathrm{Gm}_3$–$\mathrm{Gm}_5$, $\mathrm{Gm}_{\alpha,\beta}$.
 1 $p \leftarrow |G|$ ; $\mathbb{1} \leftarrow \mathrm{VE}(\vec{0})$ ; $g \leftarrow \mathrm{VE}(\vec{e}_{n+1})$ ; $Y \leftarrow \mathrm{VE}(\vec{e}_{n+2})$
 2 For $i = 1, \ldots, n$ do $X_i \leftarrow \mathrm{VE}(\vec{e}_i)$
 3 Return $\mathbb{1}, g, Y, X_1, \ldots, X_n$

VE($\vec{t}$): // $\mathrm{Gm}_3$–$\mathrm{Gm}_5$, $\mathrm{Gm}_{\alpha,\beta}$. Here $\vec{t} \in \mathbb{Z}_p^{n+2}$.
 4 If ( $\mathrm{TV}[\vec{t}] \neq \bot$ ) then return $\mathrm{TV}[\vec{t}]$
 5 $C \leftarrow\$ G \setminus \mathrm{GL}$
 6 If ( $\exists\, \vec{t}'$ : ( $\mathrm{TV}[\vec{t}'] \neq \bot$ and $\vec{t} - \vec{t}' \in \mathrm{span}(\vec{v})$ ) ) then $C \leftarrow \mathrm{TV}[\vec{t}']$
 7 Else $k \leftarrow k + 1$ ; $\vec{t}_k \leftarrow \vec{t}$ ; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{C\}$
 8 $\mathrm{TV}[\vec{t}] \leftarrow C$ ; $\mathrm{TI}[C] \leftarrow \vec{t}$ ; Return $\mathrm{TV}[\vec{t}]$

VE$^{-1}$($C$): // $\mathrm{Gm}_3$–$\mathrm{Gm}_5$, $\mathrm{Gm}_{\alpha,\beta}$. Here $\mathrm{TI}[C] \neq \bot$.
 9 Return $\mathrm{TI}[C]$

OP($A, B, \mathsf{sgn}$): // $\mathrm{Gm}_3$–$\mathrm{Gm}_5$, $\mathrm{Gm}_{\alpha,\beta}$. Here $\mathrm{TI}[A], \mathrm{TI}[B] \neq \bot$ and $\mathsf{sgn} \in \{+, -\}$
10 $\vec{c} \leftarrow \mathrm{VE}^{-1}(A)\ \mathsf{sgn}\ \mathrm{VE}^{-1}(B)$ ; $C \leftarrow \mathrm{VE}(\vec{c})$ ; Return $C$

DLO($i, W$): // $\boxed{\mathrm{Gm}_3}$, $\mathrm{Gm}_4$. Here $i \in [n]$ and $\mathrm{TI}[W] \neq \bot$.
11 $\vec{w} \leftarrow \mathrm{VE}^{-1}(W)$ ; $w \leftarrow \vec{w}[i]$
12 If $(\vec{w} - w \cdot \vec{e}_i = \vec{0})$ then return $w$
13 $z \leftarrow\$ \mathbb{Z}_p \setminus \{w\}$ ; $y \leftarrow\$ \mathbb{Z}_p$ ; $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \leftarrow\$ \mathbb{Z}_p^*$
14 $\vec{x}_i \leftarrow (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n, 1, y)$ ; $x_i \leftarrow (z - w)^{-1} \cdot \langle \vec{w}, \vec{x}_i \rangle$
15 If ( $\langle \vec{w}, \vec{x}_i \rangle = 0$ ) then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{z \leftarrow w \ ;\ x_i \leftarrow\$ \mathbb{Z}_p^*}$
16 $\vec{v} \leftarrow \vec{w} - z \cdot \vec{e}_i$ ; Return $z$

FIN($y'$): // $\mathrm{Gm}_3$, $\mathrm{Gm}_4$.
17 $\vec{x} \leftarrow (x_1, \ldots, x_n, 1, y)$
18 Return ( $(y = y')$ or ( $\exists\, \alpha, \beta$ : $1 \le \alpha < \beta \le k$ and $\langle \vec{t}_\alpha - \vec{t}_\beta, \vec{x} \rangle = 0$ ) )
```

Figure 18: Procedures for games $\mathrm{Gm}_3, \mathrm{Gm}_4$ in the proof of Theorem 5. Some procedures, as marked, will be used in later games.

vector $\vec{u} \in \mathrm{span}(\vec{v})$ satisfies $\langle \vec{u}, \vec{x} \rangle = 0$. Now we claim that

$$\Pr[\mathrm{Gm}_1(\mathcal{A})] = \Pr[\mathrm{Gm}_0(\mathcal{A})] \ . \tag{25}$$

Let us justify this. If the "If" statement at line 14 is true, we have, by the above, $\langle \vec{t} - \vec{t}', \vec{x} \rangle = 0$, or $\langle \vec{t}, \vec{x} \rangle = \langle \vec{t}', \vec{x} \rangle$, and so, as per line 8 of Figure 16, ought indeed to set $\mathrm{TV}[\vec{t}] = \mathrm{TV}[\vec{t}']$. The inclusion of the boxed code at line 18 further ensures consistency with line 8 of Figure 16. So VE is returning the same things in games $\mathrm{Gm}_1, \mathrm{Gm}_0$. While DLO defines some new quantities, what it returns does not change compared to game Gm_0. This concludes the justification of Eq. (25).

Games $\mathrm{Gm}_1, \mathrm{Gm}_2$ are identical-until-bad as defined in [9]. Let B_2 be the event that $\mathrm{Gm}_2(\mathcal{A})$ sets bad. Then by the Fundamental Lemma of Game Playing [9],

$$\Pr[\mathrm{Gm}_1(\mathcal{A})] \le \Pr[\mathrm{Gm}_2(\mathcal{A}) \text{ and } \overline{\mathrm{B}}_2] + \Pr[\mathrm{B}_2] \ , \tag{26}$$

where $\overline{\mathrm{B}}_2$ denotes the complement of event B_2. We claim that

$$\Pr[\mathrm{Gm}_2(\mathcal{A}) \text{ and } \overline{\mathrm{B}}_2] + \Pr[\mathrm{B}_2] \le \Pr[\mathrm{Gm}_3(\mathcal{A})] \ , \tag{27}$$

where game Gm_3 is in Figure 18. It includes the boxed code, which game Gm_4 excludes. In these

```
DLO$(i, W)$: // Gm$_5$, Gm$_{\alpha,\beta}$. Here $i \in [n]$ and TI$[W] \neq \bot$.
  $\vec{w} \leftarrow \text{VE}^{-1}(W)$ ; $w \leftarrow \vec{w}[i]$
  If $(\vec{w} - w \cdot \vec{e}_i = \vec{0})$ then return $w$
  $z \leftarrow\!\!\$\ \mathbb{Z}_p \setminus \{w\}$ ; $\vec{v} \leftarrow \vec{w} - z \cdot \vec{e}_i$ ; Return $z$

FIN$(y')$: // Gm$_5$.
  $y \leftarrow\!\!\$\ \mathbb{Z}_p$ ; Return $(y = y')$

FIN$(y')$: // Gm$_{\alpha,\beta}$.
  If (not $(1 \leq \alpha < \beta \leq k)$) then return false
  $y \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \leftarrow\!\!\$\ \mathbb{Z}_p^*$
  $\vec{x}_i \leftarrow (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n, 1, y)$ ; $x_i \leftarrow (z - w)^{-1} \cdot \langle \vec{w}, \vec{x}_i \rangle$
  $\vec{x} \leftarrow (x_1, \ldots, x_n, 1, y)$
  Return ( $\langle \vec{t}_\alpha - \vec{t}_\beta, \vec{x} \rangle = 0$ )
```

Figure 19: Further procedures to define game Gm$_5$ and games Gm$_{\alpha,\beta}$ $(1 \leq \alpha < \beta \leq q)$ in the proof of Theorem 5.

---

games, VE returns the same thing as in game Gm$_2$, but also indexes (keeps track of) vectors $\vec{t}$ that might set bad in Gm$_2$, so that it can refer to them in FIN. The achievement is that this procedure no longer refers to $\vec{x}$. Now we would like the same to be true for DLO. A natural approach would be to have DLO return a random $z \leftarrow\!\!\$\ \mathbb{Z}_p$. However, the true distribution of z is more complex, and instead we will use Lemma 6. Line 11 sets $w \in \mathbb{Z}_p$ to be the i-th coordinate of vector $\vec{w}$. Line 12 checks if $\vec{w}$ is 0 at all but its i-th coordinate, if so correctly returning w as the answer to the oracle query. At lines 13,14, the choices of z and x_i are made in accordance with one case of Lemma 6, with y, and the x_j for $j \neq i$, chosen correctly. Line 15 checks if it is the other case that happened, and, if so, game Gm$_3$ corrects the choices of z, x_i according to the Lemma. The Lemma thus implies that in game Gm$_3$, the returned z is distributed as it is in game Gm$_2$. FIN of game Gm$_3$ returns true if either $y = y'$, or game Gm$_2$ would set bad, justifying Eq. (27).

Games Gm$_3$, Gm$_4$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [9],

$$\Pr[\text{Gm}_3(\mathcal{A})] \leq \Pr[\text{Gm}_4(\mathcal{A})] + \Pr[\text{Gm}_4(\mathcal{A}) \text{ sets bad}] \ . \tag{28}$$

We claim

$$\Pr[\text{Gm}_4(\mathcal{A}) \text{ sets bad}] \leq \frac{1}{p-1} \ . \tag{29}$$

That is, the probability that $\langle \vec{w}, \vec{x}_i \rangle = 0$ at line 15 is at most $1/(p-1)$. We now justify this. Line 12 tells us that, at line 15, there is some $j \in [1..n+2] \setminus \{i\}$ such that $\vec{w}[j] \neq 0$. Consider two cases. The first is that there is such a j satisfying $j \neq n+1$. If $j = n+2$, there is exactly one choice of $y \in \mathbb{Z}_p$ making $\langle \vec{w}, \vec{x}_i \rangle = 0$, while if $j \in [1..n] \setminus \{i\}$, there is at most one choice of $x_j \in \mathbb{Z}_p^*$ making $\langle \vec{w}, \vec{x}_i \rangle = 0$, so overall the probability that $\langle \vec{w}, \vec{x}_i \rangle = 0$ is at most $1/(p-1)$. The second case is that $\vec{w}[j] = 0$ for all $j \neq n+1$. But then the probability that $\langle \vec{w}, \vec{x}_i \rangle = 0$ is zero. This completes the justification of Eq. (29).

We now define a game Gm$_5$, and also a game Gm$_{\alpha,\beta}$ for each $1 \leq \alpha < \beta \leq q$, where $q = \text{Q}_{\mathcal{A}}^{\text{OP}} + n + 3$. The DLO, FIN procedures of these games are shown in Figure 19, and the other procedures remain as in Figure 18. Since the boxed code is absent in DLO of game Gm$_4$, the only random choice it needs to make is z, yielding the simplified DLO procedure of Figure 19. The other random choices are delayed to FIN. The event resulting in game Gm_4 returning true is broken up in the new games so that, by the union bound,

$$\Pr[\mathrm{Gm}_4(\mathcal{A})] \leq \Pr[\mathrm{Gm}_5(\mathcal{A})] + \sum_{1\leq\alpha<\beta\leq q} \Pr[\mathrm{Gm}_{\alpha,\beta}(\mathcal{A})] \; . \tag{30}$$

Clearly

$$\Pr[\mathrm{Gm}_5(\mathcal{A})] \leq \frac{1}{p} \; . \tag{31}$$

Now, fix any $1 \leq \alpha < \beta \leq q$. We assume wlog that k always equals q. In game $\mathrm{Gm}_{\alpha,\beta}$, let $\vec{d} = \vec{t}_\alpha - \vec{t}_\beta$, let $a = (z-w)^{-1}$ and let $\vec{u} = a \cdot \vec{d}[i] \cdot \vec{w} + \vec{d}$. Let Z be the event that $\langle \vec{d}, \vec{x} \rangle = 0$, and let S be the event that $\vec{d} \in \mathrm{span}(\vec{v})$. Then

$$\begin{aligned}\Pr[\mathrm{Gm}_{\alpha,\beta}(\mathcal{A})] &= \Pr[\mathrm{Z}] = \Pr[\mathrm{Z} \text{ and } \overline{\mathrm{S}}] + \Pr[\mathrm{Z} \text{ and } \mathrm{S}] \\ &\leq \Pr[\,\mathrm{Z} \,|\, \overline{\mathrm{S}}\,] + \Pr[\mathrm{S}] \; . \end{aligned} \tag{32}$$

We will show that

$$\Pr[\,\mathrm{Z} \,|\, \overline{\mathrm{S}}\,] \leq \frac{1}{p-1} \tag{33}$$

$$\Pr[\mathrm{S}] \leq \frac{1}{p-1} \; . \tag{34}$$

We now justify Eq. (33). We have

$$\begin{aligned}\langle \vec{d}, \vec{x} \rangle &= x_i \cdot \vec{d}[i] + \langle \vec{d}, \vec{x}_i \rangle = a \cdot \langle \vec{w}, \vec{x}_i \rangle \cdot \vec{d}[i] + \langle \vec{d}, \vec{x}_i \rangle \\ &= \langle a \cdot \vec{d}[i] \cdot \vec{w} + \vec{d}, \vec{x}_i \rangle = \langle \vec{u}, \vec{x}_i \rangle\end{aligned}$$

Assume $\vec{d} \notin \mathrm{span}(\vec{v})$, meaning event $\overline{\mathrm{S}}$ happens. Then we claim (we will justify this in a bit) that there exists a $j \in [1..n+2] \setminus \{i, n+1\}$ such that $\vec{u}[j] \neq 0$. This means that the random choice of either x_j (if $j \in [1..n] \setminus \{i\}$) or y (if $j = n+2$) has probability at most $1/(p-1)$ of making $\langle \vec{u}, \vec{x}_i \rangle = 0$. To justify the claim, suppose to the contrary that for all $j \in [1..n+2] \setminus \{i, n+1\}$ we have $\vec{u}[j] = 0$. Since $\langle \vec{u}, \vec{x}_i \rangle = 0$, it must be that $\vec{u}[n+1] = 0$ as well. Let $b = -a \cdot \vec{d}[i]$, so that $\vec{d}[i] = -b \cdot a^{-1} = -b \cdot (z-w) = b \cdot (w-z)$. For $j \in [1..n+2] \setminus \{i\}$ we have $a \cdot \vec{d}[i] \cdot \vec{w}[j] + \vec{d}[j] = 0$, or $\vec{d}[j] = -a \cdot \vec{d}[i] \cdot \vec{w}[j] = b \cdot \vec{w}[j]$. Recalling that $\vec{v} = \vec{w} - z \cdot \vec{e}_i$ and $w = \vec{w}[i]$, we see that $\vec{d} = b \cdot \vec{v}$, which puts $\vec{d}$ in $\mathrm{span}(\vec{v})$, contradicting our assumption that $\vec{d} \notin \mathrm{span}(\vec{v})$. This concludes the justification of Eq. (33).

We turn to Eq. (34). Suppose $\vec{d} \in \mathrm{span}(\vec{v})$, meaning $\vec{d} = b \cdot \vec{v} = b \cdot \vec{w} - bz \cdot \vec{e}_i$ for some $b \in \mathbb{Z}_p^*$. By line 4 of Figure 18, $\vec{t}_\alpha \neq \vec{t}_\beta$, so $\vec{d} \neq \vec{0}$ so $b \neq 0$. So there is at most one $z \in \mathbb{Z}_p$ such that $\vec{d}[i] = bw - bz$, and our z chosen at random from $\mathbb{Z}_p \setminus \{w\}$ has probability at most $1/(p-1)$ of being this one.

Putting the above together we have

$$\begin{aligned}\mathbf{Adv}^{\text{gg-mbdl}}_{G,n,m}(\mathcal{A}) &\leq \frac{1}{p-1} + \frac{1}{p} + \frac{q(q-1)}{2}\frac{2}{p-1} \\ &= \frac{1+q(q-1)}{p-1} + \frac{1}{p} \; .\end{aligned}$$

This concludes the proof. ▌

References

[1] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, Apr. / May 2002. 5, 9, 15

[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. 5, 6

[3] 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. 3, 7, 8

[4] 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. 5, 6, 16, 17, 19

[5] 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. 3, 4, 5, 10

[6] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, Heidelberg, Aug. 2004. 7

[7] M. Bellare, B. Poettering, and D. Stebila. From identification to signatures, tightly: A framework and generic transforms. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 435–464. Springer, Heidelberg, Dec. 2016. 10

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 15

[9] 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. 7, 12, 22, 31, 32

[10] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012. 4, 5

[11] 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. 5

[12] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, editors, *ACM CCS 2007*, pages 276–285. ACM Press, Oct. 2007. 26

[13] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, Aug. 2004. 7

[14] 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. 5

[15] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. Cryptology ePrint Archive, Report 2018/483, 2018. https://eprint.iacr.org/2018/483. 17

[16] D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. 7

[17] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999. 7

[18] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. 7

[19] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In R. Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, Heidelberg, Jan. 2010. 3

[20] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 3

[21] 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. 3, 5, 6, 17

[22] M. Drijvers, S. Gorbunov, G. Neven, and H. Wee. Pixel: Multi-signatures for consensus. Cryptology ePrint Archive, Report 2019/514, 2019. https://eprint.iacr.org/2019/514. 17

[23] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. 3

[24] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, June 1988. 4, 10

[25] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, Aug. 1987. 4, 15

[26] M. Fischlin and N. Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 444–460. Springer, Heidelberg, May 2013. 3

[27] 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. 6

[28] G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind schnorr signatures in the algebraic group model. Cryptology ePrint Archive, Report 2019/877, 2019. https://eprint.iacr.org/2019/877. 6

[29] S. D. Galbraith and P. Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 78(1):51–72, 2016. 9

[30] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988. 15

[31] J. Y. Hwang, D. H. Lee, and M. Yung. Universal forgery of the identity-based sequential aggregate signature scheme. In W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, editors, *ASIACCS 09*, pages 157–160. ACM Press, Mar. 2009. 26

[32] IANIX. Things that use Ed25519. https://ianix.com/pub/ed25519-deployment.html. 4

[33] M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein, and E. Teske. Analysis of the xedni calculus attack. *Designs, Codes and Cryptography*, 20(1):41–64, 2000. 9

[34] 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, 36, 37

[35] 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. 5

[36] 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. 6

[37] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. https://eprint.iacr.org/2018/068. 5, 6, 17

[38] 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. 5

[39] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004. 7

[40] 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. 5

[41] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Heidelberg, Aug. 1998. 5

[42] 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. 3

[43] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. 5, 16

[44] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, Jan. 1991. 4, 10

[45] I. Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004. http://eprint.iacr.org/2004/031. 9

[46] 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. 3, 6, 14, 16, 28

[47] J. H. Silverman. The xedni calculus and the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 20(1):5–40, 2000. 9

[48] J. H. Silverman and J. Suzuki. Elliptic curve discrete logarithms and the index calculus. In K. Ohta and D. Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 110–125. Springer, Heidelberg, Oct. 1998. 9

[49] 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. 5, 6

[50] A. Yun. Generic hardness of the multiple discrete logarithm problem. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 817–836. Springer, Heidelberg, Apr. 2015. 27

A Ratio-based tightness

Let $\mathcal{A}$ be an adversary against the IMP-PA security of ID. For any given parameter $N \geq 1$, KMP [34] construct a DL adversary $\mathcal{D}_N$ such that

$$\sqrt{\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{D}_N)} \geq 1 - \left[1 - \left(\mathbf{Adv}^{\mathrm{imp\text{-}pa}}_{\mathsf{ID}}(\mathcal{A}) - \frac{1}{p}\right)\right]^N , \tag{35}$$

and $\mathrm{T}_{\mathcal{D}_N} = 2N \cdot \mathrm{T}_{\mathcal{A}}$. Notice that when $N = 1$, this is identical to Eq. (5), meaning there is no improvement in that case. Next, KMP [34] pick a *specific* value of N that we call N^*. This value is $N^* = (\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A}) - 1/p)^{-1}$. So the term on the right hand side of Eq. (35) becomes

$$1 - \left[1 - \left(\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A}) - \frac{1}{p}\right)\right]^{N^*} \approx 1 - \frac{1}{e} \approx 0.63 \ , \tag{36}$$

a constant close to 1. Let $\mathcal{B}^* = \mathcal{D}_{N^*}$ be the DL adversary for this parameter choice. Then, neglecting $1/p$ as being essentially 0, one has

$$\mathbf{Adv}_{\mathbb{G},g}^{\text{dl}}(\mathcal{B}^*) \geq \left(1 - \frac{1}{e}\right)^2 \approx 0.4 \tag{37}$$

$$\mathrm{T}_{\mathcal{B}^*} = 2N^* \cdot \mathrm{T}_{\mathcal{A}} \approx \frac{\mathrm{T}_{\mathcal{A}}}{\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A})} \ . \tag{38}$$

Dividing, they obtain the ratio tightness

$$\frac{\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A})}{\mathrm{T}_{\mathcal{A}}} \leq \frac{\mathbf{Adv}_{\mathbb{G},g}^{\text{dl}}(\mathcal{B}^*)}{\mathrm{T}_{\mathcal{B}^*}} \ . \tag{39}$$

The running time $\mathrm{T}_{\mathcal{B}^*}$ from Eq. (38) is however in general much larger than $\mathrm{T}_{\mathcal{A}}$ and the ratio tightness only holds when the running time of the DL adversary is increased in this way to make its advantage a constant as per Eq. (37). If we assume $\mathbf{Adv}_{\mathbb{G},g}^{\text{dl}}(\mathcal{B}^*) \approx \mathrm{T}_{\mathcal{B}^*}^2/p$ then from Eq. (38) one has $\mathrm{T}_{\mathcal{B}^*} \approx \sqrt{0.4 \cdot p}$, so

$$\frac{\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A})}{\mathrm{T}_{\mathcal{A}}} \leq \frac{0.4}{\sqrt{0.4 \cdot p}} \ . \tag{40}$$

Overall we are not sure, from the above, how to draw numerical improvements likes ours for given and arbitrary values of t, δ.