UNIVERSITY OF CALIFORNIA SAN DIEGO

On the Concrete Security of Identification and Signature Schemes

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Wei Dai

Committee in charge:

Professor Mihir Bellare, Chair
Professor Nadia Heninger
Professor Russell Impagliazzo
Professor Farinaz Koushanfar
Professor Daniele Micciancio

2021

The dissertation of Wei Dai is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank all those who have shaped my PhD journey.

First and foremost, I would like to thank my advisor, Mihir Bellare, for his patient guidance and insightful advice, not only regarding technical topics but also regarding problem selection in the broader, societal context.

I would like thank my mentors during my internships that have broaden my academic perspective: Atul Luykx, Stefano Tessaro, Tatsuaki Okamoto, and Go Yamamoto.

I would like to thank all of my co-authors and collaborators (in alphabetical order): Shashank Agrawal, Mihir Bellare, Viet Tung Hoang, Lucy Li, Pratyay Mukherjee, Tatsuaki Okamoto, Peter Rindal, Phillip Rogaway, Stefano Tessaro, Go Yamamoto, and Xihu Zhang.

I would like to thank members of my thesis committee: Mihir Bellare, Nadia Heninger, Russell Impagliazzo, Farinaz Koushanfar, and Daniele Micciancio.

Laslty, I would like to thank students and postdocs of the crypto community at UCSD (in alphabetical order): Vivek Arte, Hannah Davis, Nicholas Genise, Felix Gunther, Joseph Jaeger, Matilda Lundin, Baiyu Li, Ruth Ng, Keegan Ryan, Mark Schultz, Laura Shea, Jessica Sorrell, Igors Stepanovs, Adam Suhl, George Sulllivan, Jiahao Sun, Psi Vesely, and Michael Walter.

Chapter 1, in full, is a repreint of the material as it appears in ACM Conference on Computer and Communications Security, 2017. Bellare, Mihir; Dai, Wei. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in full, is a repreint of the material as it appears in International Conference on Cryptology in India, 2020. Bellare, Mihir; Dai, Wei. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a repreint of the material as it appears in International Conference on the Theory and Application of Cryptology and Information Security, Asiacrypt 2021. Bellare, Mihir; Dai, Wei. Springer, 2021. The dissertation author was the primary investigator and author of this paper.

VITA

2015	Bachelor of Science, in Mathematics and Computer Science, *summa cum laude*, University of California Santa Barbara
2016	Master of Science, in Computer Science, University of California Santa Barbara
2021	Doctor of Philosophy, in Computer Science, University of California San Diego

ABSTRACT OF THE DISSERTATION

On the Concrete Security of Identification and Signature Schemes

by

Wei Dai

Doctor of Philosophy in Computer Science

University of California San Diego, 2021

Professor Mihir Bellare, Chair

Digital signature schemes are ubiquitous in real-world applications of cryptography. They are the core cryptographic building block for public-key infrastructures and distributed ledgers. Yet, the exact security of signature and signature-related schemes are often unknown, due to gaps in their security analyses.

A security proof for a cryptographic scheme S rules out attacks on the scheme assuming hardness of some underlying problem P, for example the discrete-logarithm on elliptic curves. Often, there are gaps between the quantitative security evidenced by cryptanalysis and the quantitative security given by security proofs. For many deployed schemes, quantitative security proofs do not give any meaningful security guarantees. The study of concrete security aims to

eliminate this gap.

In this work, we study the concrete security of (1) a "big-key" identification scheme by Alwen, Dodis, and Wichs, (2) Schnorr identification and signature schemes, and (3) discrete-logarithm-based multi-signature schemes. We identify and tighten the gaps between theoretical guarantees, practical expectations, and best-known cryptanalysis.

Introduction

Digital signature schemes are ubiquitous in real-world applications of cryptography. They are the core cryptographic building block for public-key infrastructures and distributed ledgers. Yet, the exact security of signature and signature-related schemes are often unknown, due to gaps in their security analyses.

A security proof for a cryptographic scheme S rules out attacks on the scheme assuming hardness of some underlying problem P, for example the discrete-logarithm on elliptic curves. Often, there are gaps between the quantitative security evidenced by cryptanalysis and the quantitative security promised by security proofs. For many deployed schemes, quantitative security proofs do not give any meaningful security guarantees. The study of concrete security aims to eliminate this gap.

In this work, we study the concrete security of (1) a "big-key" identification scheme by Alwen, Dodis, and Wichs, (2) Schnorr identification and signature schemes, and (3) discrete-logarithm-based multi-signature schemes. We identify and tighten the gaps between theoretical guarantees, practical expectations, and best-known cryptanalysis.

Efficiency Improvements for Big-Key Cryptography

The first chapter is concerned with the security threat of key exfiltration and the efficiency of schemes achieving the goals of symmetric encryption and public-key identification. Key exfiltration happens when attacker-planted malware on the key-storing system uses the system's network connection to convey the key to a remote accomplice. A line of theoretical work has suggested a mitigation, called the Bounded Retrieval Model (BRM) [41, 38, 30, 6, 5], which involves using big keys. BKR [11] initiated an effort to take the BRM (they call it big-key cryptography) to practicality. We continue this effort.

We first identify probe complexity (the number of scheme accesses to the slow storage medium storing the big key) as the dominant cost for BRM schemes. Our *large-alphabet subkey prediction lemma* allows us to minimize the probe complexity required to reach a given level of security, thereby optimizing storage usage. We use this to obtain efficiency improvements for big-key symmetric encryption [11]. We then provide an additional lemma on polynomial-evaluation entropy preservation, and use the two lemmas in conjunction to obtain efficiency improvements for the ADW big-key identification scheme [6]. We note that the big-key identification scheme [6] leads to an entropically unforgeable big-key signature signature scheme via the Fiat-Shamir transform, and our efficiency improvements carries over to the signature setting.

Tight and Non-rewinding Proofs for Schnorr Identification and Signature

The second chapter is concerned with the concrete security of Schnorr identification and signature schemes [79]. For these widely-deployed schemes, all known standard-model proofs [76, 1, 58] exhibit a gap: the proven bound on adversary advantage (success probability) is much inferior to (larger than) the one that cryptanalysis says is "true." (The former is roughly the square-root of the latter. Accordingly we will refer to this as the square-root gap.) The square-root gap is well known and acknowledged in the literature. Filling this long-standing and notorious gap between theory and practice is the subject of this paper.

We introduce the Multi-Base Discrete Logarithm (MBDL) problem. We use this to give reductions, for Schnorr and Okamoto [74] identification and signatures, that are non-rewinding and, by avoiding the notorious square-root loss, tighter than the classical ones from the Discrete Logarithm (DL) problem. This fills a well-known theoretical and practical gap regarding the security of these schemes. 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 justify the security of these primitives for group sizes in actual use.

Chain Reductions for Multi-signatures and the HBMS Scheme

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

We ask, is it possible to bridge this gap to give some valuable support, in the form of tight reductions, for in-practice schemes? As long as we stay in the current paradigm, namely standard-model proofs from DL, the answer is likely NO. To make progress, we need to be willing to change either the model or the assumption. We show that in fact changing either suffices. Our approach is to give, for any scheme, many different paths to security. In particular we give (1) tight reductions from DL in the Algebraic Group Model (AGM) [47], and (2) tight, standard-model reductions from well-founded assumptions other than DL. We obtain these results via a framework in which a reduction is "factored" into a chain of sub-reductions involving intermediate problems.

We implement this approach first with classical 3-round schemes, giving chain reductions yielding (1) and (2) above for the BN [14] and MuSig [64] schemes. Then, in the space of 2-round schemes, we give a new, efficient scheme, called HBMS, for which we do the same.

Chapter 1

Efficiency Improvements for Big-Key Cryptography

1.1 Introduction

This paper is concerned with the threat of key exfiltration. This means attacker-planted malware on the key-storing system uses the system's network connection to convey the key to a remote accomplice. A line of theoretical work has suggested a mitigation, called the Bounded Retrieval Model (BRM) [41, 38, 30, 6, 5], which involves using big keys. BKR [11] initiated an effort to take the BRM (they call it big-key cryptography) to practicality. We continue this effort. We identify probe complexity (the number of scheme accesses to the slow storage medium storing the big key) as the dominant cost. Our *large-alphabet subkey prediction lemma* allows us to minimize the probe complexity required to reach a given level of security, thereby optimizing storage usage. We use this to obtain efficiency improvements for big-key symmetric encryption [11]. We then provide an additional lemma on polynomial-evaluation entropy preservation, and use the two lemmas in conjunction to obtain efficiency improvements for the ADW big-key identification scheme [6].

<u>LARGE-ALPHABET SUBKEY PREDICTION.</u> Let $b \geq 2$ be an integer representing the *block size* in a storage system, for example $b = 32$ or $b = 64$ for words in memory, or $b = 8 \cdot 512$ (512 bytes) for a typical hard-disk drive. Let $q = 2^b$ be the *alphabet size*, and $[q] = \{0, 1, \ldots, q-1\}$ the corresponding alphabet. Let $\boldsymbol{K} = (\boldsymbol{K}[0], \ldots, \boldsymbol{K}[k-1]) \in [q]^k$ be a string over $[q]$ of length k, randomly chosen. It represents a (big) key stored in our storage system as a sequence of k blocks. We imagine that an adversary-chosen function $\mathsf{Lk}\colon [q]^k \to [q]^\ell$ (representing adversary-implanted malware, and here called the leakage function) is applied to $\boldsymbol{K}$, and the result L (representing exfiltrated information, here called the leakage), is provided back to the adversary. Think of ℓ as somewhat smaller than k, for example $\ell \leq k/10$, so that the leakage, although not total, is certainly non-trivial. Despite this, we wish to make secure use of the big key, specifically to (repeatedly) extract "small" keys ($\tau \geq 1$ blocks, for a parameter τ) for use with conventional cryptography. In any such extraction, we make τ random but distinct probes $i_1, \ldots, i_\tau \in [k] = \{0, 1, \ldots, k-1\}$ into $\boldsymbol{K}$ to determine $J = \boldsymbol{K}[i_1] \ldots \boldsymbol{K}[i_\tau]$ as the τ-block short key. Given the leakage L and the probe positions $i_1, \ldots, i_\tau$, the adversary aims to predict (compute in its entirety) J. Two metrics (see Section 1.3 for precise definitions of what we discuss next) are of interest. First is the *subkey prediction advantage*

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) , \tag{1.1}$$

defined as the maximum probability that an adversary can compute J, the maximum being over all leakage functions Lk returning ℓ blocks and over all adversary strategies. It is useful to let $k^* = kb$ denote the amount of storage occupied by the big key in bits, and, correspondingly, $\ell^* = \ell b$ the amount of allowed leakage in bits. (We will want to fix these and vary b, thereby defining k and ℓ.) Now, in usage, we would ask for a certain number s of bits of security (for example $s = 128$), meaning we want the subkey prediction advantage to be at most 2^{-s}, and then want to know the number τ of probes it takes to get there. This is the *probe complexity*,

$$\mathbf{Probes}_{k^*,\ell^*,s}(b) = \min \left\{ \tau : \mathbf{Adv}^{\mathrm{skp}}_{2^b,k^*/b,\tau}(\ell^*/b) \leq 2^{-s} \right\} . \tag{1.2}$$

The probe complexity will be our cost in accesses to a potentially slow storage system, like a disk, and for effiency of the overlying big-key scheme, we want to minimize it. To this end, Theorem 1.3.1 gives a good upper bound on the subkey prediction advantage, whence we obtain a good upper bound on the probe complexity. Next, we compare our bounds to prior ones, and discuss history and applications (to big-key cryptography and thus key exfiltration resistance).

PRIOR WORK AND COMPARISONS. ADW [7, Lemma A.3] is an elegant and general result that, as a special case, gives an upper bound on the subkey prediction advantage (and thus probe complexity) for all values of parameters we consider. The bounds, however are quite poor, so that, to get a desired level of security, one needs a very large number of probes (we will see numbers in a bit), resulting in a significant loss of efficiency for the overlying big-key cryptography schemes. This lead BKR [11] (in their quest for practical big-key symmetric encryption) to formulate subkey prediction, and seek better bounds by direct analysis. They however *only considered the case* $b = 1$ of a binary alphabet. They gave an example to show that there are non-obvious leakage functions that lead to better subkey prediction advantage than one might expect, making the problem of giving a (good) upper bound challenging. Via a combinatorial analysis, they showed that the worst case is when the pre-images of the outputs of the leakage function are approximate Hamming balls in the space of big keys, thereby deriving estimates (not quite upper bounds, something we rectify) on the subkey prediction advantage and probe complexity, for the case $b = 1$ ($q = 2$), that are much better than those obtained via ADW [6, Lemma A.3]. They posed the large alphabet ($b > 1$) case as an open question, asking, specifically, to give bounds on subkey prediction advantage and probe complexity, in the $b > 1$ case, that are better than the ones obtained via ADW [7, Lemma A.3]. (The motivation, as we will see later, was to improve efficiency of big-key symmetric encryption.) Our work answers this question, giving (good) upper bounds as a function of the block size b.

In usage, we would typically first decide on the amount of storage k^* (measured in bits) we allocate to the big key, for example $k^* = 8 \cdot 10^{11}$ bits = 100 GBytes. Next we would fix the amount

Table 1.1: Fix the amount of storage we allocate to the big key at $k^* = 8 \cdot 10^{11}$ bits = 100 GBytes. Fix the amount of leakage at 10% of the length of the big key, $\ell^* = k^*/10 = 10$ GBytes. The first table considers security level $s = 128$, while the second considers $s = 512$. Each table then considers different block sizes b. (Once b is chosen, the length of the big key in blocks is $k = k^*/b$ and the length of the leakage in blocks is $\ell = \ell^*/b$.) The table entries show upper bounds on the probe complexity $\mathbf{Probes}_{k^*,\ell^*,s}(b)$. The "Us" column is our bound via Theorem 1.3.1, and "ADW" is what is obtained via [7, Lemma A.3]. The block sizes are chosen to represent common word or disk sector sizes in storage systems.

b	$s = 128$	
	Us	ADW
1	271	11532
8	61	1584
32	47	592
64	45	434
$8 \cdot 512$	43	287
$8 \cdot 4096$	43	285

b	$s = 512$	
	Us	ADW
1	971	46127
8	219	6335
32	171	2366
64	165	1735
$8 \cdot 512$	159	1146
$8 \cdot 4096$	158	1139

of leakage ℓ^* (also measured in bits), for example $\ell^* = k^*/10 = 10$ GBytes, corresponding to 10% of the length of the big key. The block size b may be determined by the storage system (for example 512 bytes or 4096 bytes) or chosen to optimize security and efficiency as per our bounds. Once it is chosen, the length in blocks $k = k^*/b$ of the big key and $\ell = \ell^*/b$ of the leakage are determined. Now, for a given level s of security, we want to know the probe complexity $\mathbf{Probes}_{k^*,\ell^*,s}(b)$. Smaller (fewer probes into the likely slow storage system) is better. We tabulate results in Fig. 1.1. Our bounds emerge as substantially better than those obtained via ADW [7, Lemma A.3]. For example, for $s = 128$, the improvement ranges from a factor of 26 ($b = 8$) to a factor of 6.6 ($b = 8 \cdot 4096$). Below, we will see how this translates to efficiency improvements for big-key cryptography.

<u>THE BRM.</u> Assume (for concreteness of this discussion) that the primitive is symmetric encryption [11] (we will discuss other primitives later), and let $\boldsymbol{K}$ denote the encryption key, k^* bits long. In the Bounded Retrieval Model (BRM) [41, 38, 30, 6, 5, 11], an adversary-chosen function Lk (representing adversary-implanted malware) is applied to $\boldsymbol{K}$, and the ℓ^*-bit result L (representing the exfiltrated information), is provided back to the adversary. Security would

appear impossible, since Lk could be the identity function, so that $L = \boldsymbol{K}$, but the idea is that $\boldsymbol{K}$ is big (for example $k^* = 100$ GBytes), while L is assumed to be somewhat smaller (like $\ell^* = k^*/10 = 10$ GBytes). In other words, the model assumes that the amount of data exfiltrated can be limited, say via network or system monitoring. Indeed, security product vendors such as McAfee [65] provide tools for this type of monitoring and detection.

If the scheme is poorly designed, the fact that the exfiltrated information is somewhat shorter than the key won't guarantee security. For example if the scheme applies $\mathsf{SHA256}$ to $\boldsymbol{K}$ to get a 256 bit key K and then uses $\mathsf{AES256}$ to encrypt the data, then $\mathsf{Lk}(\boldsymbol{K})$ can just return the 256 bit string $K = \mathsf{SHA256}(\boldsymbol{K})$ and security is entirely compromised no matter how big is $\boldsymbol{K}$. The first requirement for a BRM (also called big-key) scheme is thus *leakage resilience*, meaning an adversary, given $L = \mathsf{Lk}(\boldsymbol{K})$, still cannot violate security, and this must be true for *any* (adversary-chosen) function Lk that returns ℓ^* bits.

<u>PROBE COMPLEXITY.</u> Big keys may help for security, but it would be prohibitively costly to process a 100 GByte key for every encryption. The BRM addresses this via a condition that says that each encryption (or decryption) operation should only make a "small" number of probes into the big key $\boldsymbol{K}$, meaning have low probe complexity. Security in the presence of leakage is a difficult goal under any circumstances, but made even more so here by this requirement.

<u>FROM BITS TO BLOCKS.</u> Viewing the big key $\boldsymbol{K} = (\boldsymbol{K}[0], \ldots, \boldsymbol{K}[k^*-1])$ as a sequence of bits, BKR encryption [11] begins by making some τ^* random probes $i_1, \ldots, i_{\tau^*} \in [k^*]$ into $\boldsymbol{K}$ to extract a τ^*-bit subkey $J = \boldsymbol{K}[i_1] \ldots \boldsymbol{K}[i_{\tau^*}]$. It then applies a (randomized) hash function to J to get a key K for a conventional (AES-based) symmetric encryption scheme, and uses K to encrypt the data. Once J has been obtained, the computation, being symmetric cryptography operations, is quite efficient, but $\boldsymbol{K}$, being big, is likely stored on a slow medium like a hard drive, and so the encryption cost is dominated by the storage accesses needed to get J. For a subkey prediction advantage of $s = 128$ (based on which BKR show ind-cpa style security of their encryption scheme at the same security level), BKR will need $\tau^* = 271$ probes into the storage.

(This is as per the $b = 1$ row of the first table in Fig. 1.1. BKR's subkey prediction lemma gives an estimate, not a bound, so we use our number, but numerically the two are almost the same.)

But (as BKR themselves point out), their scheme is making very poor use of storage by drawing only a bit of the big key per probe. Letting b be some appropriate block size determined by the storage system (for example $b = 8 \cdot 512$ bits = 512 Bytes), $\boldsymbol{K}$ would actually be stored as a sequence of blocks, and a single probe into the storage can retrieve an entire block at about the same cost as retrieving a single bit. Indeed, a typical storage API does not even provide a way to directly access a bit, so an implementation of BKR would, for a probe for bit-position j, have to draw the block containing this bit position, take the corresponding bit, and throw the other bits away. A natural improvement (suggested by BKR) is to draw (and use) an entire block per probe. Thus, we now view the big key $\boldsymbol{K} = (\boldsymbol{K}[0], \ldots, \boldsymbol{K}[k-1]) \in [2^b]^k$ as a sequence of blocks, corresponding to the way it is actually stored, where $k = k^*/b$ is the number of blocks. Now, making some τ probes $i_1, \ldots, i_\tau \in [k]$ into $\boldsymbol{K}$, one obtains the subkey $J = \boldsymbol{K}[i_1] \ldots \boldsymbol{K}[i_\tau]$. The rest of the encryption process is as before, and as we have already noted, is efficient, even though J could be a bit longer. Continuing to require a subkey prediction advantage of $s = 128$, the question is, what value of τ guarantees this? This is the question that BKR could not answer. It is answered by our large-alphabet subkey prediction lemma. Specifically, the first table of Fig. 1.1 gives values of τ for different choices of b. For $b = 512$ Bytes, we see that $\tau = 43$. Recalling that BKR needed $\tau^* = 271$ probes, we have reduced the number of probes (storage accesses) by a factor of $271/43 \approx 6$, meaning offer a 6x speedup.

The price we pay (as alluded to above) is that J is longer, specifically, 271 bits for BKR and $43 \cdot 512 \approx 22$ KBytes for us. This means the hashing of J to obtain the AES key K takes longer, but modern hash functions are fast, and the time saved in storage accesses is more than the time lost in extra hashing [31, 34]. This is especially true since the hashing can be pipelined, taking advantage of the iterated structure of hash functions to process blocks incrementally as soon as they are retrieved rather than delaying hashing until after all blocks are retrieved.

BIG-KEY IDENTIFICATION. In a (public-key) identification scheme, a user (called the prover) has a secret key sk whose associated public key pk is held by the server (called the verifier). Access control is enforced by having the prover identify itself as the owner of sk via an interactive identification protocol. The Schnorr [79] and Okamoto [74] schemes are well-known examples, but they are of course conventional (small-key) schemes. Identification is an interesting target for a BRM scheme. Here it is the secret key sk that would be big (100 GBytes)— we want the public key pk to remain of conventional size. The usage we envision is hardware-assisted access control, where sk is on an auxiliary device like a USB stick that the user plugs into a possibly infected machine to identify herself (login) to the server across the network. The key being large, and reading from a USB being slow, the malware will have difficulty obtaining enough information about the key (10 GBytes) to violate BRM security, even after a significant number of usages (logins) by the user.

Identification in the BRM was first treated by ADW [6], who gave (asymptotic) security definitions and a clever scheme that involves combining multiple instances of the Okamoto scheme [74] in a compact way. We target making this scheme practical. The quest is meaningless in the absence of concrete security, for practicality is fundamentally about maximizing efficiency for a given level (eg. 128 bits) of security. A first and central step is thus a *concrete-security treatment*. We render the definitions of big-key identification (the goal is security against impersonation under active attack) concretely, then revisit the asymptotically-stated result of ADW [6] to render it, too, in concrete form. We note that for the ADW scheme, probe complexity dictates the computational cost of the two most costly phases of the protocol, the response phase and verification phase (as we will demonstrate in Fig. 1.2). Hence, improvements in probe complexity directly translate into improvements in efficiency. Towards lowering probe complexity for a given level of security, we first *improve the concrete security* of the reduction via a lemma on the entropy preservation of polynomial evaluation that improves bounds from ADW [6]. We then obtain further reductions in probe complexity, by *using our large-alphabet subkey prediction* *lemma* in place of ADW's own [7, Lemma A.3]. The large-alphabet aspect here is crucial, for the scheme draws, from the big key, a value in $\mathbb{Z}_p^m$, where p is a prime of 512 bits long (for 128-bit security of the identification scheme), and $m \geq 2$ is an integer parameter, so probes need to return blocks of the (large) size $b = m \cdot \lceil \log_2(p) \rceil$. Putting it all together gives a reasonable-cost big-key identification scheme, and the first concrete rendition of the ADW big-key identification scheme. A preliminary implementation shows that with a pairing-friendly group of 512 bits, the execution of the protocol takes a few seconds.

1.2 Preliminaries

For n a positive integer, we let $[n] = \{0, 1, \ldots, n-1\}$, and $[1..n] = \{1, \ldots, n\}$. We also use the notation $\mathbb{Z}_n$ to denote the set $[n]$ in contexts where we use the underlying algebraic structure modulo n. If $\boldsymbol{x}$ is a vector, then $|\boldsymbol{x}|$ denotes its length and $\boldsymbol{x}[i]$ denotes its i-th coordinate. We call $\boldsymbol{x}$ an n-vector if $|\boldsymbol{x}| = n$. We number coordinates starting from 0. For example if $\boldsymbol{x} = (10, 0, 11)$ then $|\boldsymbol{x}| = 3$ and $\boldsymbol{x}[2] = 11$. We let ε denote the empty vector, which has length 0. If $0 \leq i \leq |\boldsymbol{x}| - 1$ then we let $\boldsymbol{x}[0..i] = (\boldsymbol{x}[0], \ldots, \boldsymbol{x}[i])$, this being ε when $i = 0$. We say that $\boldsymbol{x}$ is a vector over set S if all its coordinates belong to S. We let S^n denote the set of all n-vectors over S and we let S^* denote the set of all finite-length vectors over the set S. If S is a set then $|S|$ denotes its size. If $\tau \leq |S|$ is a positive integer, we let $S^{(\tau)}$ be the set of τ-vectors over S with distinct entries. Strings are treated as the special case of vectors over $\{0, 1\}$. Thus, if x is a string then $|x|$ is its length, $x[i]$ is its i-th bit, $x[0..i] = x[0]...x[i]$, ε is the empty string, $\{0, 1\}^n$ is the set of n-bit strings and $\{0, 1\}^*$ the set of all strings. For $\mathbf{K} \in [q]^k$ and $\mathsf{pp} \in [k]^*$, we let $\mathbf{K}[\mathsf{pp}] = (\mathbf{K}[\mathsf{pp}[0]], \mathbf{K}[\mathsf{pp}[1]], \ldots, \mathbf{K}[\mathsf{pp}[|\mathsf{pp}| - 1]])$, this being ε when $\mathsf{pp} = \varepsilon$.

If X is a finite set, we let $x \leftarrow\$\, X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \cdots; r)$ denote running A with random coins r

on inputs $x_1, \cdots$ and assigning the output to y. We let $y \leftarrow\!\$\, A(x_1, \cdots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \cdots; r)$. We let $[A(x_1, \cdots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \cdots$.

We use the code-based game-playing framework [19] (see Fig. 1.1 for an example). By $\Pr[\mathrm{Gm}]$ we denote the probability that game Gm returns true. Uninitialized boolean variables, sets and integers are assume initialized to false, the empty set and 0, respectively.

Following [17], our random oracle H is variable range. This means it takes two inputs, x and Img, where the latter is a (description of) an efficiently sampleable set, and returns as output a random point in Img. In schemes, we (implicitly or explicitly) fix a unique description for each range set that is used. For example, $x \leftarrow\!\$\, \mathsf{H}(x, [k])$ will return a random element in $[k]$, and $\mathsf{pp} \leftarrow\!\$\, \mathsf{H}(x, [k]^{(\tau)})$ will return a random τ-dimensional vector over $[k]$ *with distinct entries.*

<u>HAMMING BALLS.</u> Let $q \geq 2$ and $n \geq 1$ be integers. We define the weight of a n-vector v over $[q]$ to be

$$w(v) = \left| \{ i \in [n] \mid v[i] \neq 0 \} \right| ,$$

the number of coordinates of v that are non-zero. Let $\mathcal{K} \subseteq [q]^n$ for some integer n, we define the weight of $\mathcal{K}$ to be

$$w(\mathcal{K}) = \sum_{x \in \mathcal{K}} w(x) ,$$

the sum of weights of vectors in $\mathcal{K}$. For $0 \leq r \leq k$, the q-ary hamming ball of radius r over $[q]^k$ is the set

$$\mathbf{B}_{q,k}(r) = \left\{ v \in [q]^k \,:\, w(v) \leq r \right\}$$

of k-vectors over $[q]$ that have more at most r non-zero coordinates. We let $B_{q,k}(r)$ denote the size of the set $\mathbf{B}_{q,k}(r)$ and note that

$$B_{q,k}(r) = \sum_{i=0}^{r} (q-1)^i \binom{k}{i} .$$

Game $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A},\mathsf{Lk})$

$\boldsymbol{K} \leftarrow\!\!\$\, [q]^k; L \leftarrow \mathsf{Lk}(\boldsymbol{K})$
$\mathbf{p} \leftarrow\!\!\$\, [k]^{(\tau)}$
$J \leftarrow\!\!\$\, \mathcal{A}(L,\mathbf{p})$
Return $(J = \boldsymbol{K}[\mathbf{p}])$

Game $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A},\mathcal{K})$

$\boldsymbol{K} \leftarrow\!\!\$\, \mathcal{K}$
$\mathbf{p} \leftarrow\!\!\$\, [k]^{(\tau)}$
$J \leftarrow\!\!\$\, \mathcal{A}(\mathbf{p})$
Return $(J = \boldsymbol{K}[\mathbf{p}])$

$q \geq 2$	:	the alphabet size. A *block* is an element of $[q]$.
k	:	the length in blocks of the big key
$\tau \leq k$	:	the number probes into the big key $\boldsymbol{K}$
$\mathcal{A}$	:	the adversary
Lk	:	the leakage function, $\mathsf{Lk}\colon [q]^k \to [q]^\ell$
ℓ	:	the length of the output of the leakage function, called the leakage length, in blocks
$b \geq 1$	:	the block length, meaning $q = 2^b$. Theorem 1.3.1 does not assume q is a power of two, but it is in some applications.
L	:	the leakage, an ℓ-vector over $[q]$ returned by Lk
$\boldsymbol{K}$	:	the big key, a vector of length k over $[q]$
$\mathbf{p}$	:	the probe vector, a τ-vector over $[k]$ all of whose coordinates are distinct
k^*	:	the length of the big key in bits, $k^* = kb$
ℓ^*	:	the length of the leakage in bits, $\ell^* = \ell b$
ρ	:	the leakage rate, $\rho = \ell^*/k^* = \ell/k$

Figure 1.1: **Top Left:** Subkey prediction game $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}$. **Bottom Left:** Restricted subkey prediction game $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}$ used in Section 1.3.3. **Right:** Summary of quantities involved.

For convenience of stating our results, we establish the following conventions: if $r > k$ then we let $B_{q,k}(r) = B_{q,k}(k) = q^k$, and if $k = 0$ then for all $r \geq 0$ we let $B_{q,k}(r) = 1$. We also define the function

$$\mathsf{rd}_{q,k}(N) = \max\left\{ r \in [k+1] \,:\, B_{q,k}(r) \leq N \right\}$$

to return the largest radius r in the range $0 \leq r \leq k$ such that the ball $\mathbf{B}_{q,k}(r)$ has size at most N.

1.3 Large-Alphabet Subkey Prediction

Here we define the subkey prediction problem parameterized by alphabet size and give our results about it.

1.3.1 The Problem

We consider the subkey-prediction game, $\mathbf{G}^{\text{skp}}_{q,k,\tau}(\mathcal{A},\mathsf{Lk})$, shown on the top left of Fig. 1.1. (Ignore the game below it for now.) The quantities involved in the game, as well as associated ones, are summarized on the right of the same Figure. In the game, a k-vector $\boldsymbol{K}$ over $[q]$, called the big key, is randomly chosen from $[q]^k$. Then, a random τ-vector $\mathbf{p}$ is chosen from $[k]^{(\tau)}$, so that its coordinates are all distinct. (Recall that $[k]^{(\tau)}$, the set from which $\mathbf{p}$ is selected in the game in Fig. 1.1, denotes the set of all τ-vectors over $[k]$ all of whose coordinates are distinct.) We refer to $\mathbf{p}$ as the probe vector. Each of its coordinates is a probe, pointing to a location in the big key. Adversary $\mathcal{A}$ is given the leakage $L = \mathsf{Lk}(\boldsymbol{K})$ and the probe vector $\mathbf{p}$. Its goal is to predict (compute, output) $\boldsymbol{K}[\mathbf{p}] = (\boldsymbol{K}[\mathbf{p}[1]],\ldots,\boldsymbol{K}[\mathbf{p}[\tau]])$, the τ-vector consisting of the coordinates of $\boldsymbol{K}$ selected by the coordinates of the probe vector. The adversary returns J as its guess, and the game returns true if $\mathcal{A}$ succeeds, meaning $J = \boldsymbol{K}[\mathbf{p}]$. We define the following advantage metrics:

$$\mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\mathcal{A},\mathsf{Lk}) = \Pr\left[\mathbf{G}^{\text{skp}}_{q,k,\tau}(\mathcal{A},\mathsf{Lk})\right],$$

$$\mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\mathsf{Lk}) = \max_{\mathcal{A}} \mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\mathcal{A},\mathsf{Lk}),$$

$$\mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\ell) = \max_{\mathsf{Lk}:[q]^k\to[q]^\ell} \mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\mathsf{Lk}).$$

The first advantage is the probability that the game outputs true, meaning the probability that the adversary successfully returns $\boldsymbol{K}[\mathbf{p}]$. The second advantage is obtained by maximizing the first one over all adversaries $\mathcal{A}$. Note that this is well-defined since here we consider all computationally unbounded adversaries. The third advantage is obtained by maximizing the second advantage over all leakage functions Lk that output ℓ blocks.

Now fix some big-key length k^* (in bits) and leakage length ℓ^* (in bits). Also fix an integer s representing the desired security. For any block length $b \geq 1$ such that b divides k^* and ℓ^*, we let

$$\mathbf{Probes}_{k^*,\ell^*,s}(b) = \min\left\{\tau : \mathbf{Adv}^{\mathrm{skp}}_{2^b,k^*/b,\tau}(\ell^*/b) \le 2^{-s}\right\} . \tag{1.3}$$

Here, we have set the alphabet size to $q = 2^b$. The length k of the big key and ℓ of the leakage in blocks are determined, respectively, by $k = k^*/b$ and $\ell = \ell^*/b$. Then, $\mathbf{Probes}_{k^*,\ell^*,s}(b)$ is the smallest number of probes τ that will guarantee that $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ is at most 2^{-s}.

The subkey prediction game and problem formulated by BKR [11] differs in two ways. First, they had only considered the $q = 2$ case (that is, $b = 1$) of a binary alphabet. The large alphabet aspect of our treatment refers to the fact that our alphabet size is a parameter q that we view as quite large. In some applications, $q = 2^b$ where b is the block size of our storage medium, but Theorem 1.3.1 does not assume q is a power of two. The second difference with BKR [11] is that their probes $\mathbf{p}[1],\ldots,\mathbf{p}[\tau]$ were random and independent, so in particular two of them might be the same, but ours are random subject to being distinct. This is important towards our being able to get a provable upper bound on the subkey prediction advantage, whereas BKR were only able to get (for their setting) an estimate or approximate upper bound.

Now our goal is to upper bound, as well as possible, the subkey prediction advantage $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ as a function of q,k,τ,ℓ. Thence we will obtain upper bounds on $\mathbf{Probes}_{k^*,\ell^*,s}(b)$.

1.3.2 Subkey Prediction Theorem

The bound in our subkey prediction theorem is the ratio of the sizes of two q-ary hamming balls. We refer to Section 1.2 for definitions.

Theorem 1.3.1 (Subkey-prediction bound) *Let q,k,ℓ,τ be integers with $q \ge 2$ and $\ell,\tau \le k$. Let r be any integer in the range $0 \le r \le \mathsf{rd}_{q,k}(q^{k-\ell})$. Then*

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)} . \tag{1.4}$$

The theorem allows us to pick the parameter r arbitrarily in the given range, so for the best estimates we would pick a r that minimizes the ratio. We postpone the proof to first discuss how this compares to prior work and how to use it to get numerical bounds.

<u>COMPARISON.</u> BKR [11] give an upper bound we denote $G^{\mathrm{bkr}}_{k,\tau}(2^{k-\ell})$ on the subkey prediction advantage in their setting. Recall that their setting differs from ours in two ways. First, $q = 2$ in their case. Second, in their game, the τ probes are random and independent, while in our game they are random but distinct. Their function $G^{\mathrm{bkr}}_{k,\tau}(N)$ is a sum of $\mathsf{rd}_{2,k}(N)$ terms. It is quite complex and it is hard to estimate numerically. BKR gave a simpler expression, that approximates $G^{\mathrm{bkr}}_{k,\tau}(N)$, and that they use for numerical estimates, but this expression is not an upper bound, and thus it is not clear their numerical estimates are upper bounds either. Our bound, the ratio of the sizes of two q-ary Hamming balls, is simpler than the bound of BKR (this makes crucial use of the probes being distinct), and, we will see, more analytically tractable, even when $q = 2$. In particular, we are able to upper bound $\min_r B_{q,k-\tau}(r)/B_{q,k}(r)$, subjected to $0 \leq r \leq \mathsf{rd}_{q,k}(q^{k-\ell})$, quite nicely for numerical estimates, as discussed next.

<u>TOOLS FOR DERIVING NUMERICAL BOUNDS.</u> Theorem 1.3.1 upper bounds the subkey prediction advantage as the ratio of the sizes of two hamming balls. Below, we present tools to bound this ratio. First, we need some definitions. Let H_2 be the binary entropy function, defined for $x \in [0,1]$ by $H_2(x) = -x\log_2(x) - (1-x)\log_2(1-x)$. We note that the value of $x\log_q(x)$ is taken to be 0 when $x = 0$. This ensures that H_2 is continuous over $[0,1]$. More generally, for an integer $q \geq 2$ the *q-ary entropy function* is defined for $x \in [0,1]$ by

$$\begin{aligned} H_q(x) &= x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x) \\ &= \frac{H_2(x)}{\log_2(q)} + x\log_q(q-1)\ . \end{aligned}$$

We note that H_q attains its maximum at $x = 1-1/q$. We define its inverse function, $H_q^{-1} : [0,1] \to [0, 1-1/q]$ to be such that $H_q^{-1}(H_q(x)) = x$ for any $x \in [0, 1-1/q]$. We define the following error function for $q \geq 2$ and $0 \leq r \leq k$,

$$\varepsilon(q,k,r) = \log_q(e)\left(\frac{1}{12r} + \frac{1}{12(k-r)} - \frac{1}{12k+1}\right) + \frac{1}{2}\log_q\left(\frac{2\pi r(k-r)}{k}\right). \qquad (1.5)$$

The following lemmas, the proofs of which are given in Section 1.6, are key to deriving numerical bounds. The first gives both upper and lower bounds on the size of a Hamming ball.

Lemma 1.3.2 *Let k,q,r be integers with $q \geq 2$ and $0 \leq r \leq k$. Then,*

$$q^{kH_q(r/k)-\varepsilon(q,k,r)} \leq B_{q,k}(r)\ . \qquad (1.6)$$

Additionally, if $0 \leq r \leq k(1-1/q)$,

$$B_{q,k}(r) \leq q^{kH_q(r/k)}\ . \qquad (1.7)$$

The second lemma lower bounds the value of $\mathsf{rd}_{q,k}(N)$.

Lemma 1.3.3 *Let N,q,k be positive integers such that $q \geq 2$ and $N \leq q^k$. Then,*

$$\left\lfloor H_q^{-1}\left(\frac{\log_q(N)}{k}\right)\cdot k\right\rfloor \leq \mathsf{rd}_{q,k}(N).$$

The following provides a two-sided bound on H_q^{-1}:

Lemma 1.3.4 *Let $q \geq 2$ be an integer, and $x \in [0,1]$ a real number. Then,*

$$\min(x, 1-\frac{1}{q}) - \frac{1}{\log_2(q)} \leq H_q^{-1}(x) \leq x(1-\frac{1}{q})\ .$$

These bounds are good when q is large.

<u>Deriving numerical bounds.</u> We now use the above to derive upper bounds for example parameter values. Let $b \geq 1$ be a block size, so that the alphabet has size $q = 2^b$. Fix

some big-key length k^* (in bits) and leakage length ℓ^* (in bits) that are multiples of b, and let $k = k^*/b$ and $\ell = \ell^*/b$ be the big-key and leakage lengths, respectively, in blocks. We assume that τ and ℓ satisfy that $\ell \geq \tau$, as the below method only apply when this condition is met. We note that this is a reasonable assumption for practical applications, as leakage length ℓ is usually large, and we are attempting to keep the probe complexity, τ, small. Now, suppose we have obtained some integer value r such that: (1) $r \leq \mathsf{rd}_{q,k}(q^{k-\ell})$ and (2) $0 \leq r \leq (k-\tau)(1-1/q)$. Then, we use Equation (1.6) to lower bound $B_{q,k}(r)$. Given condition (2), we can use Equation (1.7) to upper bound the quantity $B_{q,k-\tau}(r)$. This results in an upper bound, denoted $\text{RatioBound}_{q,k,\ell,\tau}(r)$, for the ratio $B_{q,k-\tau}(r)/B_{q,k}(r)$:

$$\text{RatioBound}_{q,k,\tau}(r) = \frac{q^{(k-\tau)H_q(r/(k-\tau))}}{q^{kH_q(r/k)-\varepsilon(q,k,r)}} \ .$$

Note that in the above expression, the terms $H_q(r/(k-\tau))$, $H_q(r/k)$ and $\varepsilon(q,k,r)$ can be computed numerically for any given value of q, k, τ and r. Hence, deriving numerical upper bound for the ratio $B_{q,k-\tau}(r)/B_{q,k}(r)$ amounts to obtaining a value r satisfying the two conditions given above. We take r to be $r_{q,k,\ell}$, defined as

$$r_{q,k,\ell} = \left\lfloor H_q^{-1}\left(\frac{k-\ell}{k}\right) \cdot k \right\rfloor \ .$$

Here, we assume that a method of obtaining numerical lower bounds for $H_q^{-1}(x)$ is available[1]. We now check the two conditions required. For condition (1), we know that $r_{q,k,\ell} \leq \mathsf{rd}_{q,k}(q^{k-\ell})$ by Lemma 1.3.3 (taking $N = q^{k-\ell}$). For condition (2), note that by Lemma 1.3.4 and the assumption that $\ell \geq \tau$,

$$H_q^{-1}\left(\frac{k-\ell}{k}\right) \leq \frac{k-\ell}{k}(1-1/q) \leq \frac{k-\tau}{k}(1-1/q) \ .$$

[1]For example, this is available in mathematical software `Sage`. Also, when q is large, Lemma 1.3.4 provides a good lower bound for H_q^{-1} that is easily computed numerically.

Hence,

$$r_{q,k,\ell} = \left\lfloor H_q^{-1}(\frac{k-\ell}{k}) \cdot k \right\rfloor \le (k-\tau)(1-1/q) \ .$$

We consider the quantity

$$\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) = \mathrm{RatioBound}_{q,k,\ell,\tau}(r_{q,k,\ell}) \ . \tag{1.8}$$

We note that since $r = r_{q,k,\ell}$ satisfies condition (1) and (2), by Theorem 1.3.1 and above analysis,

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le \frac{B_{q,k-\tau}(r_{q,k,\ell})}{B_{q,k}(r_{q,k,\ell})} \le \overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) \ .$$

Hence, $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ is an upper bound for $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$. Now, given a particular desired security level, s, we want to find the smallest τ such that $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le 2^{-s}$. We let

$$\overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s) = \min\left\{\tau \in [k+1] \, : \, \overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le 2^{-s}\right\} .$$

Note that this is similar to the definition of $\mathbf{Probes}_{k^*,\ell^*,b}(s)$ (Equation (1.3)), only that $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ is replaced with $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell)$. Thence,

$$\mathbf{Probes}_{k^*,\ell^*,b}(s) \le \overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s) \ . \tag{1.9}$$

We note that $\overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s)$ can be computed numerically by iteratively incrementing τ and computing $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell)$. Fig. 1.1 shows values of $\overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s)$ for various practical values of k^*, ℓ^*, b and s.

PLOTS. For the left plot, we fix the following:

– Blocksize $b = 32$ bits, so that $q = 2^{32}$.

– Leakage length $\ell^* = 8 \cdot 10^{10}$ bits = 10 GBytes, so that $\ell = \ell^*/32$.

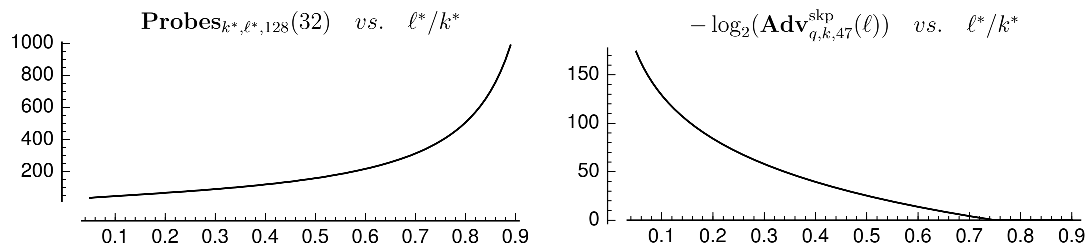

Figure 1.2: Fix the big key length k^* to be 100 GBytes. The left graph plots (an upper bound on) $\mathbf{Probes}_{k^*,\rho k^*,128}(32)$ as a function of the leakage rate ρ. The right graph plots (a lower bound on) $-\log_2(\mathbf{Adv}^{\mathrm{skp}}_{2^{32},k,47}(\rho k))$ as a function of ρ, where $k = k^*/32$.

$-$ Desired security level $s = 128$ bits.

The left graph in Fig. 1.2 plots $\overline{\mathbf{Probes}}_{\ell^*/\rho,\ell^*,b}(s)$, upper bound for $\mathbf{Probes}_{\ell^*/\rho,\ell^*,s}(b)$, as a function of the leakage rate ρ. The left plot shows that the number of probes needed to maintain s bits of security increases faster once the leakage rate goes over 50%. Hence, for applications, it may be beneficial to use big keys that are big enough so that the leakage rate can be assumed to be less than 50%. For example, if 10 GBytes is the leakage bound, one might, for efficiency, target big key of size at least 20 GBytes.

For the right plot, we fix the following

$-$ Blocksize $b = 32$ bits, so that $q = 2^{32}$.

$-$ Big key length $k^* = 8 \cdot 10^{11}$ bits = 100 GBytes, so that $k = k^*/32$.

$-$ Number of probes $\tau = 47$.

The number 47 has been chosen because, as per Fig. 1.1, it ensures $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(k/10) \leq 2^{-128}$. Now with b, k^*, τ (and thus also q, k) fixed, the right graph plots $-\log_2(\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}\rho \cdot k)$, lower bound for $-\log_2(\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\rho \cdot k))$, as a function of leakage rate ρ. The right plot in Fig. 1.2 demonstrates that, even though a scheme is designed for 10% leakage, security degrades gradually as the leakage rate goes over 10%.

1.3.3 Proof of Theorem 1.3.1

We follow the framework of the proof of BKR [11].

<u>RESTRICTED SUBKEY PREDICTION.</u> The proof involves consideration of a simpler game, called the restricted subkey prediction game, denoted $\mathbf{G}^{\mathrm{rskp}}$ and shown on the right in Fig. 1.1. Game $\mathbf{G}^{\mathrm{rskp}}$ is similar to game $\mathbf{G}^{\mathrm{skp}}$, except that there is no leakage function Lk and leakage L. Instead, the big key $\boldsymbol{K}$ is drawn from a restricted subset $\mathcal{K} \subseteq [q]^k$ of big keys. We define the following advantage metrics:

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K}) = \Pr\left[\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K})\right],$$

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K}),$$

$$\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) = \max_{\mathcal{K}\subseteq[q]^k, |\mathcal{K}|=N} \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}).$$

The first advantage is the probability that the game outputs $\mathtt{true}$, meaning the probability that the adversary successfully returns $\boldsymbol{K}[\mathbf{p}]$. The second advantage is obtained by maximizing the first one over all adversaries $\mathcal{A}$. The third advantage is obtained by maximizing the second advantage over all sets $\mathcal{K} \subseteq [q]^k$ that have size N. We note that the first two advantages do not have q in the subscript, which is due to the fact that $\mathcal{K}$ encodes the value of q.

<u>MONOTONE SETS.</u> Let x, x' be vectors in $[q]^k$. We say that x dominates x', or x' is dominated by x, written $x' \leq x$, if x' can be obtained by changing non-zero coordinates of x to 0. We let

$$\mathrm{DS}_{q,k}(x) = \{x' \in [q]^k : x' \leq x\}$$

be the set of all x' dominated by x. A set $\mathcal{K} \subseteq [q]^k$ is *monotone* if

$$\bigcup_{x\in\mathcal{K}} \mathrm{DS}_{q,k}(x) \subseteq \mathcal{K}\,.$$

That is, if $x \in \mathcal{K}$, and x' is dominated by x, then $x' \in \mathcal{K}$. For example, a Hamming ball in $[q]^k$, of any radius, is a monotone set.

<u>SOME NOTATION.</u> For integers $x, \tau \geq 0$, we let

$$x_{(\tau)} = \prod_{i=0}^{\tau-1}(x-i) = \prod_{j=x-\tau+1}^{x} j \ . \tag{1.10}$$

Notice that $x_{(\tau)} = 0$ if $\tau > x$. This can be seen because, if $\tau > x$, then, in the second product above, the starting value for j is ≤ 0, and since $x \geq 0$, this means the term $j = 0$ is included in the product. Also when $\tau = 0$, the product has zero terms, and hence by convention takes value 1, meaning $x_{(0)} = 1$ for all $x \geq 0$. We use below the notation from Equation (1.10).

For a nonempty $\mathcal{K} \subseteq [q]^k$, we define the function

$$g_{k,\tau}(\mathcal{K}) = \frac{1}{|\mathcal{K}|} \sum_{x \in \mathcal{K}} \frac{(k - w(x))_{(\tau)}}{k_{(\tau)}} \ . \tag{1.11}$$

The following lemma says that if $\mathcal{K}$ is monotone, then the restricted subkey prediction advantage for big keys drawn from $\mathcal{K}$ can be expressed *exactly*, and in particular by the function of Equation (1.11).

Lemma 1.3.5 *Let q, τ, k be positive integers such that $\tau \leq k$ and $q \geq 2$. Let $\mathcal{K} \subseteq [q]^k$ be a non-empty monotone set. Then,*

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = g_{k,\tau}(\mathcal{K}).$$

Proof of Lemma 1.3.5: Let $\mathcal{A}_0$ be the adversary that, on input $\mathbf{p}$, always returns the all-0 τ-vector. We claim that this adversary maximizes the advantage, meaning

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}, \mathcal{A}_0) \ .$$

This follows from the assumption that $\mathcal{K}$ is monotone. Now, we compute the advantage of $\mathcal{A}_0$.

For $\boldsymbol{K} \in [q]^k$, let $\mathrm{Z}(\boldsymbol{K})$ denote the set of all $\mathbf{p} \in [k]^{(\tau)}$ such that $\mathbf{K}[\mathsf{pp}] = (0, \ldots, 0)$. We have

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}, \mathcal{A}_0) = \frac{1}{|\mathcal{K}|} \sum_{\boldsymbol{K} \in \mathcal{K}} \frac{|\mathrm{Z}(\boldsymbol{K})|}{|[k]^{(\tau)}|} = \frac{1}{|\mathcal{K}|} \sum_{\boldsymbol{K} \in \mathcal{K}} \frac{(k - w(\mathbf{K}))_{(\tau)}}{k_{(\tau)}} = g_{k,\tau}(\mathcal{K}) \; . \quad \blacksquare$$

We say that a set $\mathcal{K} \subseteq [q]^k$ is *sandwiched between hamming balls* if

$$B_{q,k}(r) \subseteq \mathcal{K} \subset B_{q,k}(r+1)$$

for $r = \mathsf{rd}_{q,k}(|\mathcal{K}|)$. For N an integer such that $1 \leq N \leq q^k$, we define

$$G_{q,k,\tau}(N) = \frac{1}{N} \sum_{i=0}^{\mathsf{rd}_{q,k}(N)} (q-1)^i \binom{k}{i} \frac{(k-i)_{(\tau)}}{k_{(\tau)}} + \left(1 - \frac{B_{q,k}(\mathsf{rd}_{q,k}(N))}{N}\right) \frac{(k - (\mathsf{rd}_{q,k}(N)+1))_{(\tau)}}{k_{(\tau)}} \; . \tag{1.12}$$

The following says that if $\mathcal{K}$ is monotone and sandwiched between Hamming balls, then the restricted subkey prediction advantage for big keys drawn from $\mathcal{K}$ can be expressed *exactly*, and in particular by the function of Equation (1.12).

Lemma 1.3.6 *Let q, τ, k be positive integers such that $\tau \leq k$ and $q \geq 2$. Let $\mathcal{K} \subseteq [q]^k$ be a non-empty monotone set that is also sandwiched between hamming balls, i.e.* $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K} \subset \mathbf{B}_{q,k}(r+1)$ *for* $r = \mathsf{rd}_{q,k}(|\mathcal{K}|)$. *Then*

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = G_{q,k,\tau}(|\mathcal{K}|) \; .$$

Proof of Lemma 1.3.6: Let $N = |\mathcal{K}|$. By Lemma 1.3.5, we have

$$\mathbf{Adv}^{\mathrm{skp}}_{k,\tau}(\mathcal{K}) = \frac{1}{N} \sum_{x \in \mathcal{K}} \frac{(k - w(x))_{(\tau)}}{k_{(\tau)}} \; .$$

Since $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K} \subset \mathbf{B}_{q,k}(r+1)$. This means $\mathbf{B}_{q,k}(i) \subseteq \mathcal{K}$ for $i = 0, \ldots, r$, and $\mathcal{K}$ contains

$N - B_{q,k}(r)$ vectors of weight $r+1$. Thus, the above equals

$$\frac{N - B_{q,k}(r)}{N} \frac{(k-r-1)_{(\tau)}}{k_{(\tau)}} + \frac{1}{N} \sum_{i=0}^{r} (q-1)^i \binom{k}{i} \frac{(k-i)_{(\tau)}}{k_{(\tau)}} = G_{q,k,\tau}(N)$$

as claimed. ▮

Next, we show that monotone sets sandwiched between Hamming balls are the extremal cases for the restricted subkey prediction game, meaning that they maximize the restricted subkey prediction advantage. The following is analogous to [11, Lemmas 6,8]. We streamline their analysis and extend it to large alphabets.

Lemma 1.3.7 *Let q,k,N be positive integers. Suppose $q \geq 2$, $N \leq q^k$ and $\tau \leq k$. Then, there is a non-empty monotone set $\mathcal{K} \subseteq [q]^k$ of size N such that*

$$\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) = \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) \ .$$

Additionally, $\mathcal{K}$ is also sandwiched between hamming balls, i.e. for $r = \mathsf{rd}_{q,k}(N)$,

$$\mathbf{B}_{q,k}(r) \subseteq \mathcal{K} \subset \mathbf{B}_{q,k}(r+1) \ .$$

The proof of Lemma 1.3.7 is deferred to Section 1.3.3. As a direct corollary of Lemma 1.3.6 and Lemma 1.3.7, we get the following result.

Corollary 1.3.8 Let q, τ, k be positive integers such that $\tau \leq k$ and $q \geq 2$. Then,

$$\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) = G_{q,k,\tau}(N) \ . \tag{1.13}$$

Hence, from this point on, we identify the two functions $\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(\cdot)$ and $G_{q,k,\tau}(\cdot)$. Next, we observe a useful property of $G_{q,k,\tau}(N)$. In particular, it is decreasing in the domain $[1..q^k]$.

Lemma 1.3.9 *Let q, τ, k be positive integers such that $\tau \leq k$ and $q \geq 2$. Let i, j be integers such that $1 \leq i \leq j \leq q^k$. Then,*

$$G_{q,k,\tau}(i) \geq G_{q,k,\tau}(j) .$$

We proceed to relate the restricted subkey-prediction game to the subkey-prediction game via the lemma below.

Lemma 1.3.10 *Let ℓ, q, k, τ be integers such that $0 \leq \ell \leq k$, $q \geq 2$, and $1 \leq \tau \leq k$. Then,*

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \leq \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-\ell}) .$$

The proofs of Lemma 1.3.9 and Lemma 1.3.10 are deferred to Section 1.3.3. Finally, we give a way to bound the expression $G_{q,k,\tau}(N)$. In particular, we show that it is at most the ratio of two hamming balls of the same radius $\mathsf{rd}_{q,k}(N)$; one with dimension $k-\tau$ and one with dimension k. Recall that BKR did not give concrete numerical upper bounds for their subkey-prediction advantage, only *estimates*. Due to assuming the uniqueness of probes, we are able to simplify our expression $G_{q,k,\tau}(N)$. In particular, we note that for non-negative integers k, i, τ such that $i, \tau \leq k$,

$$\binom{k}{i}\frac{(k-i)_{(\tau)}}{k_{(\tau)}} = \frac{k_{(i)}}{i_{(i)}} \cdot \frac{(k-i)_{(\tau)}}{k_{(\tau)}} = \frac{(k-\tau)_{(i)}}{i_{(i)}} = \binom{k-\tau}{i}. \tag{1.14}$$

This property allows us to prove the following lemma.

Lemma 1.3.11 *Let N, q, k, τ, r be positive integers such that $q \geq 2$, $N \leq q^k$, $\tau \leq k$ and $r \leq \mathsf{rd}_{q,k}(N)$. Then*

$$G_{q,k,\tau}(N) \leq \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)}.$$

Proof of Lemma 1.3.11: By Lemma 1.3.9,

$$G_{q,k,\tau}(N) \leq G_{q,k,\tau}(B_{q,k}(r)).$$

By Equation (1.12) and Equation (1.14),

$$\begin{aligned} G_{q,k,\tau}(B_{q,k}(r)) &\leq \frac{1}{B_{q,k}(r)} \sum_{i=0}^{\mathsf{rd}_{q,k}(B_{q,k}(r))} (q-1)^i \binom{k}{i} \frac{(k-i)_{(\tau)}}{k_{(\tau)}} \\ &= \frac{1}{B_{q,k}(r)} \sum_{i=0}^{r} (q-1)^i \binom{k-\tau}{i} \\ &= \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)} . \end{aligned}$$

∎

The proof of Theorem 1.3.1 follows directly.

Proof of Theorem 1.3.1: Note that when $r = 0$, Equation (1.4) is trivially true. Hence, we let $r \leq \mathsf{rd}_{q,k}(N)$ be a positive integer. Then,

$$\begin{aligned} \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(l) &\leq \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-l}) && \text{(Lemma 1.3.10)} \\ &= G_{q,k,\tau}(q^{k-l}) && \text{(Corollary 1.3.8)} \\ &\leq \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)} . && \text{(Lemma 1.3.11)} \end{aligned}$$

∎

Proof of Lemma 1.3.7

Let

$$\mathcal{T} = \left\{ \mathcal{K} \subseteq [q]^k \,:\, |\mathcal{K}| = N \text{ and } \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) \right\}.$$

Let $\mathcal{K} \in \mathcal{T}$ be the minimal weight element, i.e. the element $\mathcal{K} \in \mathcal{T}$ that minimizes the value $w(\mathcal{K}) = \sum_{x \in \mathcal{K}} w(x)$. We will show that $\mathcal{K}$ is a set satisfying the properties claimed in the lemma. We will prove the two properties separately, namely that $\mathcal{K}$ is monotone and $B_{q,k}(r) \subseteq \mathcal{K} \subset B_{q,k}(r+1)$. We first claim that $\mathcal{K}$ is monotone. The idea is to define a “shifting” operation for any set $\mathcal{K}' \subseteq [q]^k$ at a coordinate to increase $\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}')$ while decreasing $w(\mathcal{K}')$. Seeking a contradiction, suppose $\mathcal{K}$ is not monotone. Without loss of generality, suppose that for all pairs of $x \in \mathcal{K}$ and $y \notin \mathcal{K}$ such that $y \leq x$, we have that x and y differ only in the first component. We build another set $\mathcal{K}'$ with the following properties.

1. $|\mathcal{K}'| = |\mathcal{K}|$

2. $w(\mathcal{K}') \leq w(\mathcal{K})$

3. $\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}') \geq \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K})$

We first explain briefly explain the construction of $\mathcal{K}'$ on the high level before giving the formal construction. Let $z \in [q]^{k-1}$. We will attempt to "swap" vectors of the form $\alpha \| z$, for $\alpha \in [q]$, in and out of $\mathcal{K}$. The swapping is done in two cases. We define D_z to contain the α's such that $\alpha \| z \in \mathcal{K}$. First, if $0 \in D_z$ or $D_z = \emptyset$, no swapping will be done. Second, if $0 \notin D_z$ and $D_z \neq \emptyset$, then we will do the following. Let $\beta = \max D_z$. We will remove the element $\beta \| z$ from $\mathcal{K}$ and add the element $0 \| z$ to $\mathcal{K}$. After such operations are done for all $z \in [q]^{k-1}$, the resulting set will be $\mathcal{K}'$. Formally, the construction of $\mathcal{K}'$ is given below. $\mathcal{K}'$ is constructed from $\mathcal{K}$ via the function $\phi : [q]^k \to [q]^k$, which is defined relative to the set B (set A is used in the later analysis). Sets A and B partition the set of strings of length $k-1$. Set A consists of z's such that no swapping will be done. Set B consists of z's such that swapping will be done. The formal definition for A, B, ϕ, and $\mathcal{K}'$ is as follows:

$$A = \left\{ z \in [q]^{k-1} : 0 \in D_z \text{ or } D_z = \emptyset \right\},$$

$$B = \left\{ z \in [q]^{k-1} : 0 \notin D_z \text{ and } D_z \neq \emptyset \right\},$$

$$\phi(\alpha \| z) = \begin{cases} 0\|z & \text{if } z \in B \text{ and } \alpha = \max D_z \\ (\max D_z)\|z & \text{if } z \in B \text{ and } \alpha = 0 \\ \alpha\|z & \text{otherwise} \end{cases},$$

$$\mathcal{K}' = \left\{ \phi(x) : x \in \mathcal{K} \right\}.$$

By construction, we note that the swapping operation preserves the size of the set and only decreases its overall weight. Hence, $|\mathcal{K}'| = |\mathcal{K}|$ and $w(\mathcal{K}') \leq w(\mathcal{K})$. It remains to show property (3). Let $\mathcal{A}$ be an adversary such that $\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K}) = \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K})$. Consider the adversary $\mathcal{A}'$ that behaves exactly as $\mathcal{A}$ with the exception that it always guess 0 for the first position. More precisely, $\mathcal{A}'$ does the following.

<u>Adversary $\mathcal{A}'((s_1, \ldots, s_\tau))$</u>

$J' \leftarrow \mathcal{A}((s_1, \ldots, s_\tau))$

For $i \leftarrow 1, \ldots, \tau$ do

  If $s_i = 1$ then $J'[i] \leftarrow 0$

Return J'

Let $P(\cdot)$ denote the probability function in game $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K})$ and $P'(\cdot)$ the probability function in game $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}', \mathcal{K}')$. We now define three events for both games $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K})$, $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}', \mathcal{K}')$, where $z \in [q]^{k-1}$.

$$\begin{aligned}
\text{WIN} &: \text{ The game returns true} \\
\text{ONE} &: 1 \in \{s_1, \ldots, s_\tau\} \\
\text{S}_z &: (\mathbf{K}[1..k] = z), \text{ ONE and } (\forall i, s_i \neq 1 : J'[i] = \mathbf{K}[s_i])
\end{aligned}$$

Note that $P(\text{ONE}) = P'(\text{ONE})$, and $P(\text{WIN} \mid \neg\text{ONE}) = P'(\text{WIN} \mid \neg\text{ONE})$. We claim that $P(\text{WIN} \mid \text{ONE}) \leq P'(\text{WIN} \mid \text{ONE})$. If so we have

$$\begin{aligned}
\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K}) &= P(\text{WIN}) \\
&= P(\text{WIN} \mid \text{ONE}) \cdot P(\text{ONE}) + P(\text{WIN} \mid \neg\text{ONE}) \cdot P(\neg\text{ONE}) \\
&= P(\text{WIN} \mid \text{ONE}) \cdot P'(\text{ONE}) + P'(\text{WIN} \mid \neg\text{ONE}) \cdot P'(\neg\text{ONE}) \\
&\leq P'(\text{WIN} \mid \text{ONE}) \cdot P'(\text{ONE}) + P'(\text{WIN} \mid \neg\text{ONE}) \cdot P'(\neg\text{ONE}) \\
&= P'(\text{WIN}) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{A}', \mathcal{K}') \,.
\end{aligned}$$

So now we need to show that $P(\text{WIN} \mid \text{ONE}) \leq P'(\text{WIN} \mid \text{ONE})$. We have

$$\begin{aligned}
P(\text{WIN} \mid \text{ONE}) &= \sum_{z \in [q]^{k-1}} P(\text{WIN} \mid \text{S}_z) \cdot P(\text{S}_z) \\
&= \sum_{z \in [q]^{k-1}} P(\text{WIN} \mid S_z) \cdot P'(S_z) \qquad (1.15) \\
&\leq \sum_{z \in [q]^{k-1}} P'(\text{WIN} \mid S_z) \cdot P'(S_z) \qquad (1.16) \\
&= P'(\text{WIN} \mid \text{ONE}) \,.
\end{aligned}$$

Equation (1.15) is true because $P(\text{S}_z) = P'(\text{S}_z)$ for all $z \in [q]^{k-1}$, since the swapping operation do not change the last $k-1$ component of any vector. Next, we argue the validity of Equation (1.16). Let $z \in [q]^{k-1}$ such that $P(S_z) \neq 0$ (and hence $P'(S_z) \neq 0$), which means that there is some $\alpha \in [q]$ such that $\alpha \| z \in \mathcal{K}$. For any $z \in [q]^{k-1}$, consider the sets $U_z = \{\alpha \in [q] : \alpha \| z \in \mathcal{K}\}$ and

$V_z = \{\alpha \in [q] : \alpha \| z \in \mathcal{K}'\}$. Note that $P(\text{WIN} \mid \text{S}_z) \leq 1/|U_z|$ and $P'(\text{WIN} \mid \text{S}_z) \leq 1/|V_z|$. Additionally, we note that $|U_z| = |V_z|$, and V_z always contains 0. Since $\mathcal{A}'$ always guess 0 for the first component, we have $P'(\text{WIN} \mid \text{S}_z) = 1/|V_z|$. Therefore,

$$P(\text{WIN} \mid \text{S}_z) \leq \frac{1}{|U_z|} = \frac{1}{|V_z|} = P'(\text{WIN} \mid \text{S}_z) \,.$$

Next, we show that $\mathcal{K}$ must be sandwiched between two hamming balls. We first claim that $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Seeking a contradiction, suppose that $\mathbf{B}_{q,k}(r) \not\subseteq \mathcal{K}$. Let x' be a point in $\mathbf{B}_{q,k}(r) \setminus \mathcal{K}$ of minimal Hamming weight. Let x be a point in $\mathcal{K} \setminus \mathbf{B}_{q,k}(r)$ of maximal Hamming weight. We claim that $w(x) > w(x')$, otherwise $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Let $\mathcal{K}'$ be obtained by removing x from $\mathcal{K}$ and then adding x', i.e. $\mathcal{K}' = (\mathcal{K} \setminus \{x\}) \cup \{x'\}$. Because x' was minimal in Hamming weight and x was maximal in Hamming weight, the set $\mathcal{K}'$ continues to be monotone, and it has size N. Also $g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}')$ because $w(x) > w(x')$. Hence, by Lemma 1.3.5

$$\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}) = g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}') = \mathbf{Adv}^{\text{rskp}}_{q,k,\tau}(\mathcal{K}') \,.$$

This contradicts the assumption that $\mathbf{Adv}^{\text{rskp}}_{q,k,\tau}(N) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K})$. Hence, it must be that $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Now suppose $\mathcal{K} \not\subseteq \mathbf{B}_{q,k}(r+1)$. Let x' be a point in $\mathbf{B}_{q,k}(r+1) \setminus \mathcal{K}$. Such a point exists because we know that $N < B_{q,k}(r+1)$. It must be that $w(x') = r+1$ since $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Let x be a point in $\mathcal{K} \setminus \mathbf{B}_{q,k}(r+1)$ of maximal Hamming weight. Note that $w(x) > r+1 = w(x')$. Let $\mathcal{K}'$ be obtained by removing x from $\mathcal{K}$ and then adding x', meaning $\mathcal{K}' = (\mathcal{K} \setminus \{x\}) \cup \{x'\}$. The set $\mathcal{K}'$ continues to be monotone, and it has size N. Also $g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}')$ because $w(x) > w(x')$. Hence, by Lemma 1.3.5,

$$\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}) = g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}') = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}').$$

This contradicts the assumption that $\mathbf{Adv}^{\text{rskp}}_{q,k,\tau}(N) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K})$. Hence, it must be that $\mathcal{K} \subset$

$\mathbf{B}_{q,k}(r+1)$. ∎

Proof of Lemma 1.3.9 and Lemma 1.3.10

To prove Lemma 1.3.9 and Lemma 1.3.10, we recall the notion of *discrete concavity*. Suppose $F:[1..M] \to \mathbb{R}$. We say that F is concave if $F(a+1)-F(a) \le F(b+1)-F(b)$ for all $a,b \in [1..M]$ satisfying $a \ge b$. Now suppose t,m are integers with $1 \le m \le t$. Then we let

$$S(M,m,t) = \left\{ (x_1,\ldots,x_m) \in [1..M]^m \,:\, x_1+\cdots+x_m = t \right\} .$$

Define $F^m:[1..M]^m \to \mathbb{R}$ by $F^m(x_1,\ldots,x_m) = F(x_1)+\cdots+F(x_m)$. We use the following lemma proved by [11].

Lemma 1.3.12 ([11]) *Suppose $F:[1..M] \to \mathbb{R}$ is concave. Suppose $1 \le m \le t$ are integers such that m divides t and $t/m \in [1..M]$. Then*

$$\max_{(x_1,\ldots,x_m)\in S(M,m,t)} F^m(x_1,\ldots,x_m) = m \cdot F(t/m) \ .$$

Lemma 1.3.13 *The function, $F_{q,k,\tau}:[1..q^k] \to \mathbb{R}$, defined below, is concave.*

$$F_{q,k,\tau}(N) = \frac{N}{q^k} \cdot \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N).$$

Proof: Let N_0, N_1 be two integers such that $q^k \ge N_0 \ge N_1 \ge 1$. Consider, for $i = 0,1$,

$$\Delta_i = F_{q,k,\tau}(N_i+1) - F_{q,k,\tau}(N_i),$$

For $i = 0,1$, we let r_i be defined as follows. If $N_i = B_{q,k}(r)$ for some r, then we take r_i to be the value such that $B_{q,k}(r_i) = N_i$. Otherwise, we let $r_i = \mathsf{rd}_{q,k}(N_i)+1$. Note that we can now express

Δ_i in terms of r_i as follow (via Equation (1.12)),

$$\Delta_i = q^k \cdot \frac{(k-r_i)_{(\tau)}}{k_{(\tau)}} \ .$$

Since $N_0 \geq N_1$, we note that $r_0 \geq r_1$. Therefore, we have $\Delta_0 \leq \Delta_1$ and that $F_{q,k,\tau}$ is concave. ▌

We first prove Lemma 1.3.9 using Lemma 1.3.13.

Proof of Lemma 1.3.9: Note that,

$$G_{q,k,\tau}(N) = \frac{q^k \cdot F_{q,k,\tau}(N)}{N} \ .$$

We let $\Delta_i = q^k \cdot F_{q,k,\tau}(i+1) - q^k \cdot F_{q,k,\tau}(i)$ for all $i = 0, \ldots, q^k - 1$. We define $\Delta_0 = q^k \cdot F_{q,k,\tau}(1) = q^k \cdot G_{q,k,\tau}(1)$. Hence, by construction $G_{q,k,\tau}(i) = (\sum_{j=0}^{i-1} \Delta_i)/i$. Note that since $F_{q,k,\tau}(\cdot)$ is concave in the domain $[1..q^k]$, the sequence $\Delta_1, \ldots, \Delta_{q^k-1}$ is non-increasing, meaning that $\Delta_i \geq \Delta_j$ whenever $1 \leq i \leq j \leq q^k - 1$. Additionally, we check that $\Delta_0 = q^k$ and $\Delta_1 \leq q^k$, hence $\Delta_0 \geq \Delta_1$. Therefore, the partial averages of the sequence $\Delta_0, \ldots, \Delta_{q^k-1}$,

$$\big(\sum_{j=0}^{i-1} \Delta_j\big) \,/\, i = G_{q,k,\tau}(i) \ ,$$

is non-increasing as claimed. ▌

Lastly, we prove Lemma 1.3.10 using Lemma 1.3.12 and 1.3.13.

Proof of Lemma 1.3.10: Let $M = q^k$, $m = q^\ell$ and $t = q^k$. We note that the leakage function $\mathsf{Lk} : [q]^k \to [q]^\ell$ defines a partition of $[q]^k$ into q^ℓ sets, with each set being $\mathsf{Lk}^{-1}(L)$ for some $L \in [q]^\ell$. Hence, we can expand $\Pr[\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk})]$ by conditioning on the value of L. Suppose

$[q]^\ell = \{L_1, \ldots, L_m\}$. We let $N_i = |\mathsf{Lk}^{-1}(L_i)|$. We derive

$$
\begin{aligned}
&\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \\
&= \max_{\mathsf{Lk}} \left(\sum_{L} \frac{|\mathsf{Lk}^{-1}(L)|}{q^k} \cdot \max_{\mathcal{A}} \Pr[\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk}) \,|\, \mathsf{Lk}(\mathbf{K}) = L] \right) \\
&= \max_{\mathsf{Lk}} \left(\sum_{L} \frac{|\mathsf{Lk}^{-1}(L)|}{q^k} \cdot \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathsf{Lk}^{-1}(L)) \right) \\
&\leq \max_{(N_1,\ldots,N_m)\in S(M,m,t)} \sum_{i=1}^{m} F_{q,k,\tau}(N_i) \\
&= \max_{(N_1,\ldots,N_m)\in S(M,m,t)} F^m_{q,k,\tau}(N_1, \ldots, N_m) \\
&= m \cdot F_{q,k,\tau}(2^{k-\ell}) && (1.17) \\
&= m \cdot \frac{q^{k-\ell}}{q^k} \cdot \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-\ell}) = \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-\ell}) \,. && (1.18)
\end{aligned}
$$

Equation (1.17) is justified since $F_{q,k,\tau}$ is concave and $t/m = 2^{k-\ell}$. Equation (1.18) is by definition of F and because $m = q^\ell$. ▮

1.3.4 Multi-challenge Subkey Prediction

Here, we present an extension of $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}$ with multiple challenges, $\mathbf{G}^{\mathrm{mcskp}}_{q,k,\tau,t}$ (Fig. 1.3), the multi-challenge subkey prediction game. Note that [11] considers the multi-challenge version directly. However, we only need this extension in the proof of Theorem 1.4.1.

Let q, k, τ, t, ℓ be positive integers such that $q \geq 2$, $k \geq \tau$, $k \geq \ell$, $t \geq 1$. We define the following advantages associated with the game $\mathbf{G}^{\mathrm{mcskp}}_{q,k,\tau,t}$, analogously to the advantages associated with $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}$.

$$\boxed{\begin{array}{l} \underline{\textbf{Game } \mathbf{G}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk})} \\ \boldsymbol{K} \leftarrow\!\!\$\, [q]^k; L \leftarrow \mathsf{Lk}(\boldsymbol{K}) \\ \text{For } i \in [t] \text{ do } \mathbf{p}_i \leftarrow\!\!\$\, [k]^{(\tau)} \\ J \leftarrow\!\!\$\, \mathcal{A}(L, \mathbf{p}_0, \ldots, \mathbf{p}_{t-1}) \\ \text{Return } (\exists i \in [t] : J = \boldsymbol{K}[\mathbf{p}_i]) \end{array}}$$

Figure 1.3: Multi-challenge subkey prediction game $\mathbf{G}^{\mathrm{mcskp}}_{q,k,\tau,t}$.

$$\mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk}) = \Pr\left[\mathbf{G}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk})\right],$$

$$\mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathsf{Lk}) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk}),$$

$$\mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\ell) = \max_{\mathsf{Lk}:[q]^k \to [q]^\ell} \mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathsf{Lk}).$$

Lemma 1.3.14 *Let q, k, τ, t be positive integers such that $q \geq 2$, $k \geq \tau$, $k \geq \ell$, $t \geq 1$. Then,*

$$\mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\ell) \leq t \cdot \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \ .$$

Proof: Let $\mathsf{Lk} : [q]^k \to [q]^\ell$ be any leakage function. Let $\mathcal{A}$ be a $\mathbf{G}^{\mathrm{mcskp}}_{q,k,\tau,t}$ adversary. We construct $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}$ adversary $\mathcal{A}'$ such that

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}', \mathsf{Lk}) \geq \frac{1}{t} \cdot \mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk}) \ , \tag{1.19}$$

which implies the lemma by maximizing over all $\mathcal{A}$ and Lk. $\mathcal{A}'$ is defined as follows.

$\underline{\text{Adversary } \mathcal{A}'(L, \mathbf{p}')}$

$j \leftarrow\!\!\$\, [t]$; $\mathbf{p}_j \leftarrow \mathbf{p}'$

For $i \in [t] - \{j\}$ do $\mathbf{p}_i \leftarrow [k]^{(\tau)}$

$J' \leftarrow \mathcal{A}(L, \mathbf{p}_0, \ldots, \mathbf{p}_{t-1})$

Game $\mathbf{G}^{\text{key}}_{\mathsf{KEY}}(\mathcal{A})$	$\mathsf{ROR}()$
$b \leftarrow\$ \{0,1\}$;	$R \leftarrow\$ \{0,1\}^r$
$\mathbf{K} \leftarrow\$ [q]^k$	If $(b = 0)$ then $K \leftarrow\$ \{0,1\}^\kappa$
$(\mathsf{Lk}, \sigma) \leftarrow\$ \mathcal{A}^{\mathsf{H}}()$	Else $K \leftarrow \mathsf{KEY}^{\mathsf{H}}(\mathbf{K}, R)$
$L \leftarrow \mathsf{Lk}^{\mathsf{H}}(\mathbf{K})$	Return (R, K)
$b' \leftarrow\$ \mathcal{A}^{\mathsf{ROR},\mathsf{H}}(L, \sigma)$	$\mathsf{H}(x, \mathrm{Img})$
Return $(b' = b)$	If not $T[x, \mathrm{Img}]$ then $T[x, \mathrm{Img}] \leftarrow\$ \mathrm{Img}$
	Return $T[x, n]$

Figure 1.4: Game for defining the security of a big-key key encapsulation algorithm $\mathsf{KEY}: \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^\kappa$.

Algorithm $\mathsf{XKEY}^{\mathsf{H}}_{q,k,\kappa,\tau,r}(\mathbf{K}, R)$ // $\mathbf{K} \in [q]^k$, $|R| = r$

$\mathsf{pp} \leftarrow \mathsf{H}(R,\ [k]^{(\tau)})$; $J \leftarrow \mathbf{K}[\mathsf{pp}]$; $K \leftarrow \mathsf{H}(R \| J,\ \kappa)$; Return K

Figure 1.5: Encapsulation algorithm XKEY. Given a length-k big-key $\mathbf{K}$ and a length-r selector R, the algorithm returns a length-κ subkey K. The value τ specifies the number of unique probes used.

Return J'

Let E_1 be the event that $\mathcal{A}$ succeed in the game $\mathbf{G}^{\text{skp}}_{q,k,\tau}(\mathcal{A}', \mathsf{Lk})$, i.e. $J' = \mathbf{K}[\mathbf{p}_\alpha]$ for some $\alpha \in [t]$. Note that α is random variable that is well-defined given E_1 (in case $J' = \mathbf{K}[\mathbf{p}_\alpha]$ for multiple $\alpha \in [t]$, we can take the smallest one). We note that since $\mathcal{A}'$ simulates the multi-challenge game for $\mathcal{A}$ perfectly, $\Pr[E_1] = \mathbf{Adv}^{\text{mcskp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk})$. Let $E_2 \subseteq E_1$ be the event that $\mathcal{A}'$ also guesses the correct α, i.e. $j = \alpha$ in the game $\mathbf{G}^{\text{mcskp}}_{q,k,\tau,t}(\mathcal{A}', \mathsf{Lk})$. We note that $\Pr[E_2] = \frac{1}{t} \cdot \Pr[E_1]$, since j is independently uniform in $[t]$ and the distribution of $(\mathbf{p}_0, \ldots, \mathbf{p}_{t-1})$ does not depend on the value of j. Notice that $\mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\mathcal{A}', \mathsf{Lk}) \geq \Pr[E_2] = \frac{1}{t} \cdot \Pr[E_1] = \frac{1}{t} \cdot \mathbf{Adv}^{\text{mcskp}}_{q,k,\tau,t}(\mathcal{A}, \mathsf{Lk})$. ▮

1.4 Big-Key Symmetric Encryption

In [11], Big-Key symmetric encryption schemes are constructed modularly from Big-Key encapsulation schemes. In this section, we present a block-based big key encapsulation scheme that is more efficient than achieved previously.

KEY ENCAPSULATION SCHEMES. A (symmetric, Big-Key) encapsulation schemes, on input a big key $\mathbf{K}$ and a random string R, returns a (short) key K. The string R encapsulates the short key K in the sense that any party holding the big key $\mathbf{K}$ can derive K from R. The security of a key encapsulation schemes is captured by $\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})$ (Fig. 1.4). In this game, a big key $\mathbf{K}$ is randomly sampled. The goal of the two-stage adversary $\mathcal{A}$ is to guess whether the real-or-random oracle, ROR, is returning real keys, derived using key encapsulation scheme KEY from randomly sampled R, or randomly sampled keys that is independent of R. In its first stage, $\mathcal{A}$ gets access to H and chooses a leakage function Lk and state σ. Next, the game computes $L \leftarrow \mathsf{Lk}^{\mathsf{H}}(\mathbf{K})$ and run the second stage of $\mathcal{A}$ with inputs L, σ and oracles ROR and H. $\mathcal{A}$ wins the game if it successfully guesses the bit b. We define the following advantage of $\mathcal{A}$ against key encapsulation scheme KEY

$$\mathbf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) = 2 \cdot \Pr\left[\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})\right] - 1 \; .$$

OUR CONSTRUCTION. Our random oracle model construction is given in Fig. 1.5.

Theorem 1.4.1 *Let $k, b, \kappa, \tau, r \geq 1$ be integers. Let $q = 2^b$. Let $\mathsf{KEY} = \mathsf{XKEY}_{q,k,\kappa,\tau,r}$ be the big-key encapsulation scheme associated to them as per Fig. 1.5. Let $\mathcal{A}$ be an adversary making at most t queries to its* ROR *oracle and leaking $\ell \cdot b$ bits. Assume the number of* H *queries made by $\mathcal{A}$ in its first stage, plus the number made by the oracle leakage function* Lk *that it outputs in this stage, is at most q_1, and the number of* H *queries made by $\mathcal{A}$ in its second stage is at most q_2. Then*

$$\mathbf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) \;\leq\; q_2 \cdot t \cdot \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) + \frac{t \cdot (2q_1 + t - 1)}{2^{r+1}} \; . \tag{1.20}$$

The proof of Theorem 1.4.1 is deferred to Section 1.4.1.

SAMPLING UNIQUE PROBES. In XKEY, we have outsourced the sampling of the unique probes to the variable-range random oracle. We note that sampling from $[k]^{(\tau)}$ can be done via

Algorithm $\textsf{SE}.\mathsf{Enc}^{\mathsf{H}}(\mathbf{K},M)$	Algorithm $\textsf{SE}.\mathsf{Dec}^{\mathsf{H}}(\mathbf{K},M)$
$R \leftarrow\!\!\$ \{0,1\}^r$; $K \leftarrow \mathsf{KEY}^{\mathsf{H}}(\mathbf{K},R)$	$(R,C) \leftarrow \overline{C}$
$C \leftarrow \mathsf{AE.Enc}(K,M)$; $\overline{C} \leftarrow (R,C)$	$K \leftarrow \mathsf{KEY}^{\mathsf{H}}(\mathbf{K},R)$
Return $\overline{C}$	$M \leftarrow \mathsf{AE.Dec}(K,C)$
	Return M

Figure 1.6: Big-Key Symmetric Encryption Scheme [11, Section 5], **SE**, using a standard symmetric key encryption scheme AE and a key encapsulation mechanism KEY.

rejection sampling efficiently. For example, per Lemma 1.7.1 in Section 1.7, it holds with all but $2^{-3\tau}$ probability that 4τ samples from $[k]$ contains τ unique probes (hence for parameters involved in Fig. 1.1, the failure probability is less than 2^{-129} since $\tau \geq 43$).

SYMMETRIC ENCRYPTION SCHEMES. To obtain a (big-key) symmetric encryption scheme, one can plug our XKEY construction directly into the (big-key) symmetric encryption scheme (in Fig. 1.6) by BKR. For security, we omit the details here and appeal to [11, Theorem 13].

EFFICIENCY. Let $k^* = 8 \cdot 10^{11}$ = 100 GBytes, and $\ell^* = 10$ GBytes. Using $b = 8 \cdot 512 =$ 512 Bytes, our XKEY makes roughly the same number of H queries compared to [11] but makes significantly less access into the big key $\mathbf{K}$ (43 vs. 271, Fig. 1.1). In practical instantiations where $\mathbf{K}$ is stored on slow storage medium (e.g. hard disk), this translate to 6x improvement in efficiency.

1.4.1 Proof of Theorem 1.4.1

Proof of Proof (of Theorem 1.4.1): Consider the games, $\mathrm{Gm}_0, \ldots, \mathrm{Gm}_3$ defined in Fig. 1.7. Let $\mathsf{KEY} = \mathsf{XKEY}_{q,k,\kappa,\tau,r}$. We note that game Gm_0, with the boxed code included, simulates the game $\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})$ exactly for $b = 1$ case and outputs true when $\mathcal{A}$ outputs 1. Similarly, we note that Gm_3, without the boxed code, simulates the game $\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})$ exactly for $b = 0$ case and outputs true

Game $\boxed{\mathrm{Gm}_0}$ Gm_1

$\mathbf{K} \leftarrow\!\!\$\, [q]^k$
For $j \leftarrow 1,\ldots,t$ do
  $R[j] \leftarrow\!\!\$\, \{0,1\}^r$; $K[j] \leftarrow\!\!\$\, \{0,1\}^\kappa$
  $P[j] \leftarrow\!\!\$\, [k]^{(\tau)}$
  For $i \leftarrow 1,\ldots,t$ do
    If $R[i] = R[j]$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{K[j] \leftarrow K[i]}$
stage $\leftarrow 1$
$(\mathsf{Lk},\sigma) \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{H}_0}()$; $L \leftarrow\!\!\$\, \mathsf{Lk}^{\mathsf{H}_0}(\mathbf{K})$
stage $\leftarrow 2$; $b' \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{ROR},\mathsf{H}_0}(\sigma, L)$
Return $(b' = 1)$

$\mathsf{H}_0(x, \mathrm{Img})$
If not $T[x,\mathrm{Img}]$ then
  $T[x,\mathrm{Img}] \leftarrow\!\!\$\, \mathrm{Img}$
  If stage $= 1$ then For $j \in [t]$ do
    If $x = R[j]$ and $\mathrm{Img} = [k]^{(\tau)}$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{T[x,\mathrm{Img}] \leftarrow P[j]}$
    If $x = R[j] \| J[j]$ and $\mathrm{Img} = \{0,1\}^\kappa$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{T[x,\mathrm{Img}] \leftarrow K[j]}$
  If stage $= 2$ then For $j \in [t]$ do
    If $x = R[j]$ and $\mathrm{Img} = [k]^{(\tau)}$ then
      $T[x,\mathrm{Img}] \leftarrow P[j]$
    If $x = R[j] \| J[j]$ and $\mathrm{Img} = \{0,1\}^\kappa$ then
      $T[x,\mathrm{Img}] \leftarrow K[j]$
Return $T[x,\mathrm{Img}]$

Game $\boxed{\mathrm{Gm}_2}$ Gm_3

$\mathbf{K} \leftarrow\!\!\$\, [q]^k$
For $j \leftarrow 1,\ldots,t$ do
  $R[j] \leftarrow\!\!\$\, \{0,1\}^r$; $K[j] \leftarrow\!\!\$\, \{0,1\}^\kappa$
  $P[j] \leftarrow\!\!\$\, [k]^{(\tau)}$
stage $\leftarrow 1$
$(\mathsf{Lk},\sigma) \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{H}_1}()$; $L \leftarrow\!\!\$\, \mathsf{Lk}^{\mathsf{H}_1}(\mathbf{K})$
stage $\leftarrow 2$; $b' \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{ROR},\mathsf{H}_1}(\sigma, L)$
Return $(b' = 1)$

$\mathsf{H}_1(x, \mathrm{Img})$
If not $T[x,\mathrm{Img}]$ then
  $T[x,\mathrm{Img}] \leftarrow\!\!\$\, \mathrm{Img}$
  If stage $= 2$ then For $j \in [t]$ do
    If $x = R[j]$ and $\mathrm{Img} = [k]^{(\tau)}$ then
      $T[x,\mathrm{Img}] \leftarrow P[j]$
    If $x = R[j] \| J[j]$ and $\mathrm{Img} = \{0,1\}^\kappa$ then
      $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{T[x,\mathrm{Img}] \leftarrow K[j]}$
Return $T[x,\mathrm{Img}]$

$\mathsf{ROR}()$
$j \leftarrow j+1$; Return $(R[j], K[j])$

Figure 1.7: Games $\mathrm{Gm}_0,\ldots,\mathrm{Gm}_3$. All games share the same procedure ROR shown on the bottom of the middle column.

when $\mathcal{A}$ outputs 1. Hence,

$$\mathbf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0] - \Pr[\mathrm{Gm}_3]\ . \tag{1.21}$$

Game Gm_4	$\mathsf{H}_2(x, \mathrm{Img})$

$\mathbf{K} \leftarrow\!\!\$\, [q]^k$
For $j \leftarrow 1, \ldots, t$ do
  $R[j] \leftarrow\!\!\$\, \{0,1\}^r$; $K[j] \leftarrow\!\!\$\, \{0,1\}^\kappa$
  $P[j] \leftarrow\!\!\$\, [k]^{(\tau)}$
stage $\leftarrow 1$; $(\mathsf{Lk}, \sigma) \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{H}_2}()$; $L \leftarrow\!\!\$\, \mathsf{Lk}^{\mathsf{H}_2}(\mathbf{K})$
stage $\leftarrow 2$; $b' \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{ROR}, \mathsf{H}_2}(\sigma, L)$
Return $(b' = 1)$
 	
$\mathsf{ROR}()$
$j \leftarrow j + 1$; Return $(R[j], K[j])$
	If not $T_0[x, \mathrm{Img}]$ then
	  $T_0[x, \mathrm{Img}] \leftarrow\!\!\$\, \mathrm{Img}$
	  If stage $= 2$ then For $j \in [t]$ do
	    If $x = R[j]$ and $\mathrm{Img} = [k]^{(\tau)}$ then
	      $T_0[x, \mathrm{Img}] \leftarrow P[j]$
	    If $x = R[j] \| J[j]$ and $\mathrm{Img} = \{0,1\}^\kappa$ then
	      $\mathsf{bad} \leftarrow \mathsf{true}$
	Return $T_0[x, \mathrm{Img}]$

Figure 1.8: Game Gm_4. Note that T_0 is a table obtained via coin-fixing.

Adversary $\mathcal{B}(L, \mathsf{pp}_0, \ldots, \mathsf{pp}_{t-1})$	$\mathsf{H}_3(x, \mathrm{Img})$

$i \leftarrow 0$; For $j \in [t]$ do
  $R[j] \leftarrow\!\!\$\, \{0,1\}^r$; $P[j] \leftarrow\!\!\$\, \mathsf{pp}_j$
$b' \leftarrow\!\!\$\, \mathcal{A}'^{\mathsf{ROR}, \mathsf{H}_3}(L)$
$\alpha \leftarrow\!\!\$\, [i]$
Return J_α
 	
$\mathsf{ROR}()$
$j \leftarrow j + 1$; Return $(R[j], K[j])$
	If not $T_0[x, \mathrm{Img}]$ then
	  $T_0[x, \mathrm{Img}] \leftarrow\!\!\$\, \mathrm{Img}$
	  If stage $= 2$ then For $j \in [t]$ do
	    If $x = R[j]$ and $\mathrm{Img} = [k]^{(\tau)}$ then
	      $T_0[x, \mathrm{Img}] \leftarrow P[j]$
	    If $\mathrm{Img} = \{0,1\}^\kappa$ then
	      $R_i \| J_i \leftarrow x$; $i \leftarrow i + 1$
	Return $T_0[x, \mathrm{Img}]$

Figure 1.9: Subkey prediction adversary $\mathcal{B}$.

We will proceed to bound $\Pr[\mathrm{Gm}_0]$. Note that Gm_0 and Gm_1 are identical-until-bad. Hence, via the Fundamental Lemma of Game Playing [19]

$$\begin{aligned} \Pr[\mathrm{Gm}_0] &= \Pr[\mathrm{Gm}_1] + (\Pr[\mathrm{Gm}_0] - \Pr[\mathrm{Gm}_1]) \\ &\leq \Pr[\mathrm{Gm}_1] + \Pr\left[\mathrm{Gm}_1 \text{ sets } \mathsf{bad}\right] . \end{aligned} \tag{1.22}$$

Next, we claim that

$$\Pr[\mathrm{Gm}_1 \text{ sets } \mathsf{bad}] \leq \frac{t(t-1)}{2^{r+1}} + \frac{t \cdot q_1}{2^r} . \tag{1.23}$$

First, there is at most $t(t-1)/2^{r+1}$ probability of collision in the r-bit values $R[1],\ldots,R[t]$ by the birthday bound. Next, H_0 sets bad only when $\mathsf{stage} = 1$, and there are exactly q_1 H_0-queries when $\mathsf{stage} = 1$. We note that each H_0-query when $\mathsf{stage} = 1$ has at most $t/2^r$ probability of setting bad since there are at most t distinct values for $R[1],\ldots,R[t]$.

We proceed to bound $\Pr[\mathrm{Gm}_1]$. We note that Gm_2, with the boxed code included, is equivalent to Gm_1. Furthermore, Gm_3, without the boxed code, is identical to Gm_2 until bad is set. Hence,

$$\begin{aligned}\Pr[\mathrm{Gm}_1] = \Pr[\mathrm{Gm}_2] &= \Pr[\mathrm{Gm}_3] - (\Pr[\mathrm{Gm}_2] - \Pr[\mathrm{Gm}_3]) \\ &\leq \Pr[\mathrm{Gm}_3] + \Pr[\mathrm{Gm}_3 \text{ sets } \mathsf{bad}] \ .\end{aligned} \tag{1.24}$$

Lastly, we claim that

$$\Pr[\mathrm{Gm}_3 \text{ sets } \mathsf{bad}] \leq q_2 \cdot t \cdot \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \ , \tag{1.25}$$

Notice that the theorem follows from Equations (1.21), (1.22), (1.23), (1.24), and (1.25). It remains to show Equation (1.25). The justification of Equation (1.25) involves two step. First, we argue that there is some fixing of the coins of $\mathcal{A}, \mathsf{H}_1, \mathsf{Lk}$, which results in a deterministic leakage function Lk', an adversary $\mathcal{A}'$, and partial H table T_0 such that

$$\Pr[\mathrm{Gm}_3 \text{ sets } \mathsf{bad}] \leq \Pr[\mathrm{Gm}_4 \text{ sets } \mathsf{bad}] \ , \tag{1.26}$$

where Gm_4 is given in Fig. 1.8. Next, we show that

$$\Pr[\mathrm{Gm}_4 \text{ sets } \mathsf{bad}] \leq q_2 \cdot \mathbf{Adv}^{\mathrm{mcskp}}_{q,k,\tau,t}(\mathcal{B}, \mathsf{Lk}') \leq q_2 \cdot t \cdot \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \ , \tag{1.27}$$

by constructing a multi-challenge subkey prediction adversary $\mathcal{B}$, which is given in Fig. 1.9. $\mathcal{B}$ will embed the probes given, $\mathsf{pp}_0,\ldots,\mathsf{pp}_{t-1}$ into the H response and run $\mathcal{A}'$. It will guess, at random, one of the H queries of $\mathcal{A}'$ of the form $(R\|J,\{0,1\}^\kappa)$. Hence, if Gm_4 sets bad, then with at least $\frac{1}{q_2}$ probability, $\mathcal{B}$ succeeds. The second part of Equation (1.27) follows from Lemma 1.3.14. This justifies Equation (1.25) and concludes the proof of the theorem. ▮

1.5 Big-Key Identification

<u>IDENTIFICATION SCHEMES.</u> An identification scheme ID specifies the following:

– Via $\mathsf{prm} \leftarrow\$ \mathsf{ID.ParamGen}$, parameter generation algorithm $\mathsf{ID.ParamGen}$ generates parameter prm, which is a common input to all other algorithms.

– Via $(sk, \mathsf{vk}, \mathsf{hlp}) \leftarrow\$ \mathsf{ID.KeyGen}(\mathsf{prm})$, key generation algorithm $\mathsf{ID.KeyGen}$ is run by the prover to generate secret key sk, corresponding verification key vk and a string hlp called the help string. The last is information that, conceptually, can be viewed as part of the public verification key vk, meaning public and available to the adversary, but to keep the verification key small, hlp is stored by the prover along with sk.

– Via $(\mathsf{com}, \mathsf{st}) \leftarrow\$ \mathsf{ID.Com}(\mathsf{prm})$, commitment algorithm $\mathsf{ID.Com}$ is run by the prover to generate its first message com, called the commitment, along with state information st that it saves.

– Via $c \leftarrow\$ \{0,1\}^{\mathsf{ID.Chl}}$, the verifier generates a random challenge c to return to the prover.

– Via $z \leftarrow \mathsf{ID.Rsp}(\mathsf{prm}, \mathsf{hlp}, sk, \mathsf{st}, c)$, deterministic response algorithm $\mathsf{ID.Rsp}$ is run by the prover to generate its response z.

– Via $d \leftarrow \mathsf{ID.Vrf}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, c, z)$, deterministic decision algorithm $\mathsf{ID.Vrf}$ returns a boolean decision d for the verifier to accept or reject.

In the ROM, algorithms may have oracle access to the random oracle H. This syntax is non-asymptotic, in that there is no explicit security parameter. Correctness requires that

$$\Pr[\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, \mathsf{vk}, sk, \mathsf{hlp})] = 1$$

for all $\mathsf{prm} \in [\mathsf{ID.ParamGen}]$ and $(sk, \mathsf{vk}, \mathsf{hlp}) \in [\mathsf{ID.KeyGen}(\mathsf{prm})]$, where

Game $\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})$

$\mathsf{prm} \leftarrow\!\!\$\ \mathsf{ID.ParamGen}$; $s \leftarrow 0$
$(sk, \mathsf{vk}, \mathsf{hlp}) \leftarrow\!\!\$\ \mathsf{ID.KeyGen}(\mathsf{prm})$
$\mathsf{st} \leftarrow\!\!\$\ \mathcal{A}.\mathsf{Setup}^{\mathsf{Leak}_\ell, \mathsf{Prover}, \mathsf{H}}(\mathsf{prm}, \mathsf{vk}, \mathsf{hlp})$
$(\mathsf{com}, \mathsf{st}') \leftarrow\!\!\$\ \mathcal{A}.\mathsf{Com}^{\mathsf{H}}(\mathsf{st})$;
$c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.Chl}}$
$z \leftarrow\!\!\$\ \mathcal{A}.\mathsf{Rsp}^{\mathsf{H}}(\mathsf{prm}, \mathsf{hlp}, sk, \mathsf{st}', c)$
$d \leftarrow\!\!\$\ \mathsf{ID.Vrf}^{\mathsf{H}}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, c, z)$
Return d

$\mathsf{Leak}_\ell(f)$

$L \leftarrow\!\!\$\ f(sk)$; $s \leftarrow s + |L|$
If $s \leq \ell$ then return L else return $\perp$

$\mathsf{Prover}(i, \mathsf{args})$

If $\mathsf{pst}[i] = \perp$ then // Commit
  $(\mathsf{pcom}[i], \mathsf{pst}[i]) \leftarrow\!\!\$\ \mathsf{ID.Com}(\mathsf{prm})$
  Return $\mathsf{pcom}[i]$
If $\mathsf{prsp}[i] = \perp$ then // Response
  $\mathsf{prsp}[i] \leftarrow\!\!\$\ \mathsf{ID.Rsp}^{\mathsf{H}}(\mathsf{prm}, \mathsf{hlp}, sk, \mathsf{pst}[i], \mathsf{args})$
  Return $\mathsf{prsp}[i]$
Return $\perp$

$\mathsf{H}(x, \mathrm{Img})$

If $T[x, \mathrm{Img}] = \perp$ then $T[x, \mathrm{Img}] \leftarrow\!\!\$\ \mathrm{Img}$
Return $T[x, \mathrm{Img}]$

Figure 1.10: Game defining security of identification scheme ID under pre-impersonation leakage.

Game $\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, \mathsf{vk}, sk, \mathsf{hlp})$

$(\mathsf{com}, \mathsf{st}) \leftarrow\!\!\$\ \mathsf{ID.Com}(\mathsf{prm})$

$c \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ID.Chl}}$

$z \leftarrow \mathsf{ID.Rsp}(\mathsf{prm}, \mathsf{hlp}, sk, \mathsf{st}, c)$

$d \leftarrow \mathsf{ID.Vrf}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, c, z)$

Return d

SECURITY OF IDENTIFICATION SCHEMES. We give definitions allowing concrete-security assessments. The core definition is that of adversary advantage. The notion captured is security against impersonation under active attack [45, 15] in the further presence of leakage on the secret key [6].

Let ID be an identification scheme. Let ℓ be an integer representing a bound (in bits) on the leakage. Let $\mathcal{A}$ be an *impersonation adversary*, made up of component algorithms $\mathcal{A}.\mathsf{Setup}$, $\mathcal{A}.\mathsf{Com}$, and $\mathcal{A}.\mathsf{Rsp}$. We associate to these the game of Fig. 1.10. First, the parameters and keys are generated. Next, $\mathcal{A}$.Setup is run with access to a leakage oracle Leak_ℓ a prover oracle Prover and the random oracle H. The leakage oracle takes input a function Lk from the adversary and returns leakage $L = \mathsf{Lk}(sk)$. This oracle can be called adaptively and any number of times, its code ensuring that the total number of bits returned to the adversary does not exceed ℓ. The prover oracle allows an active attack in which the adversary, playing the role of a dishonest verifier, can generate prover instances and interact with them. The commitment and state of instance i are produced by the game and stored as $\mathsf{pcom}[i]$ and $\mathsf{pst}[i]$, respectively. If instance i has been activated, meaning $\mathsf{pst}[i] \neq \perp$, then the adversary can submit, via args, a challenge of its choice, and obtain response $\mathsf{prsp}[i]$. After exiting this setup phase, the adversary turns into a dishonest prover, aiming to convince the honest verifier to accept. It produces its commitment via $\mathcal{A}$.Com, receives a random challenge c, and produces its response via $\mathcal{A}$.Rsp. The game returns the boolean decision d of the verifier's decision function. We define the leakage impersonation advantage of $\mathcal{A}$ against ID to be

$$\mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A}) = \Pr\left[\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})\right].$$

<u>GROUPS.</u> We fix a *bilinear group description* $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$, where

- $p \geq 3$ is a prime number that will be the order of the groups
- G, G_T are (cyclic) groups of order p
- $g \in G$ is a generator of G
- $\mathbf{e} : G \times G \to G_T$ is an efficiently computable, non-degenerate bilinear map. This means that (1) $e(g^a, g^b) = \mathbf{e}(g, g)^{ab}$ for all $a, b \in [p]$, and (2) $\mathbf{e}(g, g)$ is not the identity element of G_T.

We will base security on the assumed hardness of the CDH (Computational Diffie-Hellman) and DL (Discrete Logarithm) problems in G. The definitions are based on games $\mathbf{G}^{\mathrm{cdh}}$ and DL in

Game $\mathbf{G}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})$

$(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$
$x, y \leftarrow\!\!\$\, [p]\ ;\ h \leftarrow\!\!\$\, \mathcal{A}(\mathcal{G}, g^x, g^y)$
Return $(h = g^{xy})$

Game $\mathsf{DL}_{\mathcal{G}}(\mathcal{A})$

$(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$
$x \leftarrow\!\!\$\, [p]\ ;\ x' \leftarrow\!\!\$\, \mathcal{A}(\mathcal{G}, g^x)$
Return $(x = x')$

Game $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$

For $i \in [k]$ do $sk[i] \leftarrow\!\!\$\, \mathbb{Z}_p^m$
$\mathsf{pp} \leftarrow\!\!\$\, [k]^{(\tau)}$; $e \leftarrow\!\!\$\, \mathbb{Z}_p$
For $j \in [m]$ do
  $sk^*[j] = \sum_{i=0}^{\tau-1} (sk[\mathsf{pp}[i]][j]) e^i$
$L \leftarrow \mathsf{Lk}(sk)$; $\overline{sk} \leftarrow\!\!\$\, \mathcal{A}(\mathsf{pp}, e, L)$
Return $(sk^* = \overline{sk})$.

Figure 1.11: **Left:** Games $\mathbf{G}^{\mathrm{cdh}}_{\mathcal{G}}$ and $\mathsf{DL}_{\mathcal{G}}$ defining the security of CDH and DL problems in $\mathcal{G}$. **Right:** Game $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$. Where $\mathsf{Lk} : [q]^k \to [q]^\ell$ is a leakage function. $[k]^{(\tau)}$ contains the set of τ-dimensional vectors over $[k]$ with distinct entries.

Fig. 1.11, associated to $\mathcal{G}$ and an adversary $\mathcal{A}$. We define the following CDH and DL advantages:

$$\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})]$$
$$\mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}) = \Pr[\mathsf{DL}_{\mathcal{G}}(\mathcal{A})]\ .$$

Hardness of CDH of course implies hardness of DL. Quantitatively, given $\mathcal{A}$, one can construct $\mathcal{A}'$ with similar running time such that

$$\mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}').$$

ADW IDENTIFICATION SCHEME. We present a variant of ADW's identification scheme [6], which uses a random oracle to derive the challenges (as considered in [6] without analysis). The scheme $\mathsf{ID} = \mathsf{ADW}[\mathcal{G}, k, m, \tau, r]$ is parameterized by a bilinear group description $\mathcal{G}$ and positive integers k, m, τ, r. We require that $m \geq 2$ and $k \geq \tau \geq 1$. Here k is the number of blocks of the secret key, where each block is an m-dimensional vector over $\mathbb{Z}_p$, and τ is the number of probes that algorithms make into the secret key. The parameter r determines the challenge length, meaning we set $\mathsf{ID}.c = r$. The algorithms $\mathsf{ID.ParamGen}, \mathsf{ID.KeyGen}, \mathsf{ID.Com}, \mathsf{ID.Rsp}, \mathsf{ID.Vrf}$ are

$\underline{\mathsf{ID.ParamGen}()}$ For $i \in [m]$ do $g_i \leftarrow\!\!\$\, G$ Return $(g_0, \ldots, g_{m-1})$	$\underline{\mathsf{ID.Com}(\mathsf{prm})}$ $y \leftarrow\!\!\$\, (\mathbb{Z}_p)^m$ $a \leftarrow\!\!\$\, \prod_{j=0}^{m-1} g_i^{y[j]}$ Return (a, y)	$\underline{\mathsf{Derive}^{\mathsf{H}}(R)}$ $\mathsf{pp} \leftarrow \mathsf{H}(R, [k]^{(\tau)})$ $e \leftarrow\!\!\$\, \mathsf{H}(0\|R, [p])$; $c^* \leftarrow\!\!\$\, \mathsf{H}(1\|R, [p])$ Return (pp, e, c^*)
$\underline{\mathsf{ID.KeyGen}^{\mathsf{H}}(\mathsf{prm})}$ $s \leftarrow\!\!\$\, \mathbb{Z}_p$; $\mathsf{vk} \leftarrow g^s$ For $i \in [k]$ do $\quad sk[i] \leftarrow\!\!\$\, (\mathbb{Z}_p)^m$ $\quad pk[i] \leftarrow \prod_{j=0}^{m-1} g_i^{sk[i][j]}$ $\quad \sigma[i] \leftarrow\!\!\$\, (\mathsf{H}(i, G) pk[i])^s$ $\mathsf{hlp} \leftarrow (pk, \sigma)$ Return $(sk, \mathsf{vk}, \mathsf{hlp})$	$\underline{\mathsf{ID.Rsp}^{\mathsf{H}}(\mathsf{prm}, \mathsf{hlp}, sk, \mathsf{st}, c)}$ $(\mathsf{pp}, e, c^*) \leftarrow \mathsf{Derive}^{\mathsf{H}}(c)$ For $j \in [m]$ do $\quad sk^*[j] \leftarrow \sum_{i=0}^{\tau-1} sk[\mathsf{pp}[i]][j] \cdot e^i$ $pk^* \leftarrow \prod_{i=0}^{\tau-1} pk[\mathsf{pp}[i]]^{e^i}$ $\sigma^* \leftarrow \prod_{i=0}^{\tau-1} \sigma[\mathsf{pp}[i]]^{e^i}$ For $j \in [m]$ do $\quad z \leftarrow y[j] + c^* \cdot sk^*[j]$ Return (pk^*, σ^*, z)	$\underline{\mathsf{ID.Vrf}^{\mathsf{H}}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, c, z)}$ $a \leftarrow \mathsf{com}$ $(\mathsf{pp}, e, c^*) \leftarrow \mathsf{Derive}^{\mathsf{H}}(c)$ $(pk^*, \sigma^*, z) \leftarrow z$ $A \leftarrow (\prod_{i=0}^{m-1} g_i^{z[i]} = a(pk^*)^{c^*})$ $h_1 \leftarrow \mathbf{e}(pk^* \prod_{i=0}^{\tau-1} \mathsf{H}(\mathsf{pp}[i], G)^{e^i}, \mathsf{vk})$ $h_2 \leftarrow \mathbf{e}(\sigma^*, g)$ $B \leftarrow (h_1 = h_2)$ Return $(A \wedge B)$

Figure 1.12: Algorithms of identification scheme $\mathsf{ID} = \mathsf{ADW}[\mathcal{G}, k, m, \tau, r]$ associated to bilinear group description $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$ and parameters k, m, τ, r satisfying $m \geq 2$ and $k \geq \tau \geq 1$. Here H is a variable range function, meaning $\mathsf{H}(\cdot, \mathrm{Img})$ returns outputs in the set (described by) Img. In addition, algorithms $\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Rsp}, \mathsf{Vrf}$ also takes prm as argument.

given in Fig. 1.12.

Intuitively, the scheme consists of k generalized Okamoto identification scheme [74, 6], and one instance of BLS signature scheme [28]. Each block of the secret key (in $\mathbb{Z}_p^m$) is a secret key for a generalized Okamoto identification scheme of dimension m. The public keys, $pk[0], \ldots, pk[k-1]$, of the k Okamoto's identification schemes, are signed using the BLS signature scheme under signing key s, yielding signatures $\sigma[0], \ldots, \sigma[k-1]$. The public verification key of the identification scheme, consists only of the verification key, vk, of the BLS signature scheme. During identification, a random τ instances out of k instances is chosen (via H by the verifier) and compressed via polynomial evaluation to sk^*, pk^*, and σ^* by the prover. During response phase, the prover, in addition to answering the challenge from the Okamoto identification scheme, needs to transmit pk^* and σ^* to the verifier. We note that the signing key, s, of the underlying signature scheme *must not* be visible to the attacker. This signing key is simply be discarded after KeyGen. (However, we note that, as ADW has pointed out, there are advanced uses of this key such as updating the big secret key.) The correctness of $\mathsf{ID} = \mathsf{ADW}[\mathcal{G}, k, m, \tau, r]$ is checked as follows. Let $\mathsf{prm} \in [\mathsf{ID.ParamGen}]$ and $(sk, \mathsf{vk}, \mathsf{hlp}) \in [\mathsf{ID.KeyGen}(\mathsf{prm})]$. We claim that, during a honest execution of the protocol ($\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, sk, \mathsf{vk}, \mathsf{hlp})$), the flags A, B in $\mathsf{ID.Vrf}$ will both be set to true. A is set to true because

$$\begin{aligned}\prod_{i=0}^{m-1} g_i^{z[i]} &= \prod_{i=0}^{m-1} g^{y[i]+c^* \cdot sk^*[i]} \\ &= \prod_{i=0}^{m-1} g^{y[i]} \cdot (\prod_{i=0}^{m-1} g^{sk^*[i]})^c = a \cdot pk^{*c^*} \quad .\end{aligned}$$

B is set to true because

$$\begin{aligned}\mathbf{e}(pk^* \prod_{i=0}^{\tau-1} \mathsf{H}(\mathsf{pp}[i], G)^{e^i}, \mathsf{vk}) &= \mathbf{e}(\prod_{i=0}^{\tau-1} pk[\mathsf{pp}[i]]^{e^i} \prod_{i=0}^{\tau-1} \mathsf{H}(\mathsf{pp}[i], G)^{e^i}, g^s) \\ &= \mathbf{e}(\prod_{i=0}^{\tau-1} ((pk[\mathsf{pp}[i]] \mathsf{H}(\mathsf{pp}[i], G))^s)^{e^i}, g) \\ &= \mathbf{e}(\prod_{i=0}^{\tau-1} (\sigma[\mathsf{pp}[i]])^{e^i}, g) = \mathbf{e}(\sigma^*, g) \quad .\end{aligned}$$

Hence, $\Pr[\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, sk, \mathsf{vk}, \mathsf{hlp})] = 1$, and ID satisfies correctness.

<u>EFFICIENCY.</u> As pointed out in [6], the identification scheme has nice efficiency properties. First, the public key (verification key) is very short (one group element). Second, the communication costs of all phases are very small. The bulk of communication happens in the response phase, which outputs 2 group elements and m elements from $\mathbb{Z}_p$. Third, the scheme has probe complexity depending τ, which can be made small while preserving security. In particular, during each run of the protocol, only τ locations of the secret-key will be accessed (each location consist of m elements of $\mathbb{Z}_p$). Fig. 1.2 demonstrates the computation and communication costs of different operations. Note that very small values of τ makes the scheme insecure. The crux of the

Table 1.2: **Left:** Table illustrating computation and communication cost of different operations of the identification scheme $\mathsf{ADW}_{G,k,m,\tau,r}$. Chl here represents the challenge phase of the protocol. **Right:** Example parameters for ADW scheme to achieve 128-bit security. The schemes uses group of size p such that $2^{511} < p < 2^{512}$, and we impose a bound on the leakage of 10% on a big-key of size 100 GB = 8×10^{11} bits. For each value of m on the left column, we look the value of τ needed to achieve 128-bit security for the identification scheme, both using our bound and using ADW's bound.

Computation cost					
	KeyGen	Com	Chl	Rsp	Vrf
Mult G	$k \cdot m$	$m-1$	0	2τ	$m+\tau$
Exp G	$k(m+1)+1$	m	0	$2\tau-2$	$\tau+m+1$
Mult $\mathbb{Z}_p$	0	0	0	m	1
Exp $\mathbb{Z}_p$	m	0	0	2τ	τ
e eval	0	0	0	0	1
Communication cost					
G	-	1	0	2	0
$\mathbb{Z}_p$	-	0	0	m	0
$\{0,1\}^r$	-	0	1	0	0

Example parameters		
m	τ (Us)	τ (ADW)
2	718	3951
4	349	2397
8	245	1996
16	201	1840
32	180	1771
64	169	1739

security analysis amounts to giving a lower bound of τ for a desired security level. Here is where we make significant *concrete security* improvements over ADW.

CONCRETE-SECURITY ANALYSIS. Before we present the theorem stating the concrete security of the ADW identification scheme, we first need to define the following special subkey prediction game. The game $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathsf{Lk},\mathcal{A})$ (Fig. 1.11) captures a particular type of subkey prediction game in which the subkey is interpreted as a tuple of polynomials. In this game, the adversary $\mathcal{A}$ needs to predict the value of these polynomials at a random point e, which is given to $\mathcal{A}$. We define the following prediction advantage

$$\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell) = \max_{\mathcal{A},\,\mathsf{Lk}:(\mathbb{Z}_p^m)^k \to (\mathbb{Z}_p^m)^\ell} \Pr\left[\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A},\mathsf{Lk})\right].$$

We state a theorem which captures the *concrete security* of the ADW identification scheme. The theorem streamlines the original analysis of ADW to a precise relation of advantages, which allows us to instantiate parameters of practical sizes.

Theorem 1.5.1 *Let $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$ be a group with efficient pairngs. Let* $\mathsf{ID} = \mathsf{ADW}_{\mathcal{G},k,m,\tau,r}$ *be the ADW identification scheme shown in Fig. 1.12. Let $\mathcal{A} = (\mathcal{A}.\mathsf{Setup}, \mathcal{A}.\mathsf{Com}, \mathcal{A}.\mathsf{Rsp})$ be a leakage impersonation adversary. Let q denote the number of* H *queries plus the number of* Prover *queries that* $\mathcal{A}.\mathsf{Setup}$ *and* $\mathcal{A}.\mathsf{Com}$ *makes. Fig. 1.14 and Fig. 1.14 gives two adversaries $\mathcal{A}_{\mathrm{cdh}}$ and $\mathcal{A}_{\mathrm{dl}}$ such that*

$$\mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})^2 \leq \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}_{\mathrm{cdh}}) + m \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}_{\mathrm{dl}}) + \mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell + k/m) + \frac{q}{2^r} + \frac{1}{p} \,. \tag{1.28}$$

Additionally, let t_1 be the running time of $\mathcal{A}.\mathsf{Setup}$*, t_2 be the running time of* $\mathcal{A}.\mathsf{Com}$*, t_3 be the running time of* $\mathcal{A}.\mathsf{Rsp}$*, and let t_4 be the running time of* $\mathsf{ID.KeyGen}$*. We have that the running time of $\mathcal{A}_{\mathrm{cdh}}$ and $\mathcal{A}_{\mathrm{dl}}$ is approximately $t_1 + t_2 + 2 \cdot t_3 + t_4$.*

The proof of Theorem 1.5.1 is given in Section 1.5.1. The following lemma relates $\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell + k/m)$ to the large-alphabet subkey prediction advantage (as bounded in Section 1.3.3).

Lemma 1.5.2 *Let p, m, k, τ, ℓ be positive integers, then*

$$\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell) \leq \sqrt{\mathbf{Adv}^{\mathrm{skp}}_{pm,k,\tau}(\ell) + \frac{\tau}{p}} \,.$$

We note that with Lemma 1.5.2, we can bound the term $\mathbf{Adv}^{\mathrm{pskp}}_{p,k,k,\tau}(\ell)$ for any value p, m, k, τ, ℓ. Hence, the only term that is not explicitly bounded on the right-hand side of Equation (1.28) are $\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})$ and $m \cdot \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})$, which can be assumed to be small when the CDH and DL problems are suspected to be hard in group G.

COMPARISON WITH ADW'S ANALYSIS. Our analysis of ADW's identification scheme improves upon the original analysis in the following ways. First, we analyze the scheme in which the challenge is generated using a random oracle directly. (The construction that uses a random oracle to derive the challenge is mentioned to be secure in [6] with no proof.) Second, while ADW's analysis is offered in the asymptotic case, we state and prove a reduction that gives *concrete security*, which lead to practical instantiation of parameters. The reduction gives a bound of the impersonation advantage in terms of three dominating quantities: CDH and DL advantages in $\mathcal{G}$, and a special form of subkey-prediction advantage under polynomial compression, $\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathsf{Lk},\mathcal{A})$. Hence, giving a good numerical bound of the impersonation advantage amounts to bounding $\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathsf{Lk},\mathcal{A})$. Here is where we make significant improvements: we use the large-alphabet subkey prediction lemma (Theorem 1.3.1) as well as a tighter polynomial-evaluation entropy preservation lemma (Lemma 1.5.2) to give significantly better concrete bounds. The comparison of parameters can be found in Fig. 1.2.

<u>PARAMETER INSTANTIATION.</u> We give an example instantiation of the ADW identification scheme with 128-bits security. First, we find a pairing friendly group G with symmetric pairing $\mathbf{e} : G \times G \to G_T$. Because of the square-root loss of security, we need 256-bit of security for CDH and DL in G. Hence, G needs to be of size roughly 512 bits. We consider $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$, where p is a prime of roughly 512 bits ($2^{511} < p < 2^{512}$). We represent elements in $\mathbb{Z}_p$ using exactly 512 bits. We pick a big key size of 100 GB, i.e. $k^* = 8 \cdot 10^{11}$. For a choice of $m \geq 2$, we have that the block size in bits is $b = m \cdot 512$. We let $k = k^*/b$ be the size of the big key in blocks. We fix a leakage rate of 10%. By Theorem 1.5.1 and Lemma 1.5.2, to achieve 128-bit security for the identification scheme, we need 512 bits of security from $\mathbf{Adv}^{\mathrm{skp}}_{pm,k,\tau}(\ell + \frac{k}{m})$. Hence, we need

$$\tau = \mathbf{Probes}_{k^*,\ell^*+k^*/m,s}(m \cdot 512)$$

probes. Values of $\mathbf{Probes}_{k^*,\ell^*+k^*/m,s}(m \cdot 512)$ versus various values of m is shown in Fig. 1.2 using both our bound and ADW's bound.

<u>ENTROPY PRESERVATION UNDER POLYNOMIAL EVALUATION.</u> Lemma 1.5.2 relates the prediction advantage to the large-alphabet subkey prediction advantage. Note that our bound is

quantitatively better than [6, Corollary A.1]. In particular, we prove $\frac{1}{2}$ rate entropy preservation while ADW proves a rate of $\frac{1}{3}$. Before proving the lemma, we define the following quantities for jointly distributed random variables (X,Y). Let X be a random variable, the *prediction* and *collision* probability of X is defined, respectively, to be

$$\mathsf{Pred}(X) = \max_x \Pr[X = x]\ , \quad \mathsf{CP}(X) = \Pr[X = X']\ ,$$

where X' is an independent random variable that is identically distributed to X. Additionally, suppose that (X,Y) are jointly distributed, we define the *conditional* prediction and collision probability of X *given* Y, respectively, to be

$$\widetilde{\mathsf{Pred}}(X \mid Y) = \mathsf{E}_Y[\mathsf{Pred}(X \mid Y)]\ , \quad \widetilde{\mathsf{CP}}(X \mid Y) = \mathsf{E}_Y[\mathsf{CP}(X \mid Y)]\ .$$

We note that $\mathsf{Pred}(X \mid Y)$ and $\mathsf{CP}(X \mid Y)$ are random variables in Y. We need the following well-known lemma,

Lemma 1.5.3 *Let (X,Y) be jointly distributed random variables, then*

$$\widetilde{\mathsf{CP}}(X \mid Y) \le \widetilde{\mathsf{Pred}}(X \mid Y) \le \sqrt{\widetilde{\mathsf{CP}}(X \mid Y)}\ .$$

Proof: For each value y of the random variable Y, we consider the probability mass function of the random variable $X \mid Y = y$, $P_{X|Y=y}(\cdot)$. We note that

$$\mathsf{Pred}(X \mid Y = y) = \max_x P_{X|Y=y}(x)\ ,$$

$$\mathsf{CP}(X \mid Y = y) = \sum_x P_{X|Y=y}(x)^2\ .$$

First, we derive that

$$\mathsf{CP}(X \mid Y = y) \geq \left(\max_x P_{X|Y=y}(x)\right) \cdot \sum_x P_{X|Y=y}(x) = \mathsf{Pred}(X \mid Y = y) \ .$$

Taking expectation over y sampled from Y on both sides of the above equation, we obtain that $\widetilde{\mathsf{Pred}}(X \mid Y) \leq \widetilde{\mathsf{CP}}(X \mid Y)$. Next, we note that viewing $\mathsf{Pred}(X \mid Y = y), \sqrt{\mathsf{CP}(X \mid Y = y)}$ as 1- and 2-norms of $P_{X|Y=y}(\cdot)$ respectively, we have $\mathsf{Pred}(X \mid Y = y) \leq \sqrt{\mathsf{CP}(X \mid Y = y)}$. Hence, by the above property and Jensen's inequality

$$\begin{aligned}
\widetilde{\mathsf{Pred}}(X \mid Y) &= \mathsf{E}_Y[\mathsf{Pred}(X \mid Y)] \\
&\leq \mathsf{E}_Y[\sqrt{\mathsf{CP}[X \mid Y]}] \\
&\leq \sqrt{\mathsf{E}_Y[\mathsf{CP}[X \mid Y]]} \\
&= \widetilde{\mathsf{CP}}(X \mid Y) \ .
\end{aligned}$$

∎

Proof of Lemma 1.5.2: Let $\mathcal{A}$ be any adversary and $\mathsf{Lk} : [q]^k \to [q]^\ell$ be a leakage function.

Consider the sample space defined by the experiment $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$ (all the coins used by the experiment and adversary $\mathcal{A}$). We consider all the variables used inside $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$ as random variables (e.g. sk^* and $L = \mathsf{Lk}(\boldsymbol{K})$). We note that

$$\Pr\left[\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})\right] \leq \widetilde{\mathsf{Pred}}(sk^* \mid \mathsf{pp}, L, e) \ .$$

Furthermore, by Lemma 1.5.3,

$$\widetilde{\mathsf{Pred}}(sk^* \mid \mathsf{pp}, L, e) \leq \sqrt{\widetilde{\mathsf{CP}}(sk^* \mid \mathsf{pp}, L, e)} \ .$$

We now need to bound $\widetilde{\mathsf{CP}}(sk^* \mid \mathsf{pp}, L, e)$. To compute this quantity. We consider another independent execution of $\mathbf{G}^{\text{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$, where the variables in the second execution is denoted with $'$, e.g. sk'. We restrict to the event that $\mathsf{Lk}(sk) = \mathsf{Lk}(sk')$ and $\mathsf{pp} = \mathsf{pp}'$. We define polynomials $p_1, \ldots, p_j$, which are functions of sk, sk', pp, $p_j(x) = \sum_{i=0}^{\tau-1}(sk[\mathsf{pp}[i]][j] - sk'[\mathsf{pp}[i]][j])x^i$. Notice that these polynomials are of degree at most τ. If $sk \neq sk'$, then at least one of p_j is a non-zero polynomial, and has at most τ roots. Hence, if $sk \neq sk'$, over a independently uniform e, the probability that $p_j(e) = 0$ is at most $\frac{\tau}{p}$ when p_j is not the zero polynomial. Finally, we derive that

$$\begin{aligned}
\widetilde{\mathsf{CP}}(sk^* \mid \mathsf{pp}, L, e) &= \mathsf{E}_{\mathsf{pp},L,e}\Big[\mathsf{CP}(sk^* \mid \mathsf{pp}, L, e)\Big] \\
&\leq \mathsf{E}_{\mathsf{pp},L,e}\Big[\Pr\left[sk[\mathsf{pp}] = sk'[\mathsf{pp}] \mid \mathsf{pp}, L, e\right] \\
&\qquad + \Pr\left[\forall j \in [m] : p_j(e) = 0 \mid sk[\mathsf{pp}] \neq sk'[\mathsf{pp}], \mathsf{pp}, L, e\right]\Big] \\
&= \widetilde{\mathsf{CP}}(sk[\mathsf{pp}] \mid \mathsf{pp}, L) + \mathsf{E}_e\Big[\Pr\left[\forall j \in [m] : p_j(e) = 0\right]\Big] \\
&\leq \widetilde{\mathsf{Pred}}(sk[\mathsf{pp}] \mid \mathsf{pp}, L) + \frac{\tau}{p} \\
&\leq \mathbf{Adv}^{\text{skp}}_{pm,k,\tau}(\ell) + \frac{\tau}{p}\,.
\end{aligned}$$

∎

1.5.1 Proof of Theorem 1.5.1

We follow the proof technique used by [6]. Let $\mathsf{ID} = \mathsf{ADW}_{\mathcal{G},k,m,\tau,r}$ be the ADW identification scheme. The reduction is very similar to the reduction from [6, Appendix B.5]. Rewind attemps to run a given leakage impersonation adverasry $\mathcal{A}$ twice with two different programmed challenges that only differ in the element c^* (R and e stay the same). Rewind takes an algorithm Gen that generates $(\mathsf{prm}, \mathsf{vk}, sk, \mathsf{hlp}, T)$, where T is the table used by H. Rewind simulates H for $\mathcal{A}$ using $\mathsf{H_r}$ as decribed by the code. Rewind returns the success status of the rewinding process, along with the two responses of the two executions (z_1, z_2), plus the honest response (z^*) and the

$\underline{\text{Game Rewind}^1(\text{Gen}, \mathcal{A})\ \boxed{\text{Rewind}^2(\text{Gen}, \mathcal{A})}}$

$(\mathsf{prm}, \mathsf{vk}, sk, \mathsf{hlp}, T) \leftarrow \mathsf{Gen}();\ s \leftarrow 0$

$\mathsf{st} \leftarrow\!\!\$\ \mathcal{A}.\mathsf{Setup}^{\mathsf{Leak}, \mathsf{Prover}, \mathsf{H_r}}(\mathsf{prm}, \mathsf{vk}, \mathsf{hlp})$

$(\mathsf{com}, \mathsf{st}') \leftarrow \mathcal{A}.\mathsf{Com}^{\mathsf{H_r}}(\mathsf{st});\ c \leftarrow \{0,1\}^{\mathsf{ID.Chl}}$

$T_1[0\|c, [p]] \leftarrow\!\!\$\ [p];\ T_1[1\|c, [p]] \leftarrow \perp$

$T_1[c, [k]^{(\tau)}] \leftarrow\!\!\$\ [k]^{(\tau)};\ T_2 \leftarrow T_1$

If $C[c]$ then

  $\mathsf{bad} \leftarrow \mathsf{true}$

  $\boxed{T[0\|c, [p]] \leftarrow \perp;\ T[1\|c, [p]] \leftarrow \perp;\ T[c, [k]^{(\tau)}] \leftarrow \perp}$

$z_1 \leftarrow \mathcal{A}.\mathsf{Rsp}^{\mathsf{H_r}[T_1]}(\mathsf{st}', c)$

$z_2 \leftarrow \mathcal{A}.\mathsf{Rsp}^{\mathsf{H_r}[T_2]}(\mathsf{st}', c)$

For $j \in [m]$ do $y[j] \leftarrow 0$

$(pk^*, \sigma^*, z^*) \leftarrow \mathsf{ADW.Rsp}^{\mathsf{H_r}[T_2]}(\mathsf{prm}, \mathsf{hlp}, \mathsf{vk}, sk, y, c)$

$(\mathsf{pp}, c, e) \leftarrow \mathsf{Derive}^{\mathsf{H}}(c)$

For $j \in [m]$ do $sk^*[j] \leftarrow \sum_{i=0}^{\tau-1}(sk[\mathsf{pp}[i]][j])e^i$

$A \leftarrow \mathsf{Vrf}^{\mathsf{H_r}[T_1]}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, c, z_1)$

$B \leftarrow \mathsf{Vrf}^{\mathsf{H_r}[T_2]}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, c, z_2)$

$C \leftarrow (T_1[1\|c, [p]] \neq T_2[1\|c, [p]])$

Return $(A \wedge B \wedge C, z_1, z_2, z^*, sk^*)$

$\underline{\mathsf{Gen}()}$

$\mathsf{prm} \leftarrow\!\!\$\ \mathsf{ParamGen}();\ (\mathsf{vk}, sk, \mathsf{hlp}) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{prm})$

Return $(\mathsf{prm}, \mathsf{vk}, sk, \mathsf{hlp}, \perp)$

$\underline{\mathsf{H_r}[T'](x, \mathrm{Img})}$

If $\mathrm{Img} = [p]$ then $b\|x \leftarrow x$

$C[x] \leftarrow \mathsf{true}$

If $T[x, \mathrm{Img}]$ then return $T[x, \mathrm{Img}]$

If T' then

  If not $T'[x, \mathrm{Img}]$ then $T'[x, \mathrm{Img}] \leftarrow\!\!\$\ \mathrm{Img}$

  Return $T'[x, \mathrm{Img}]$

Else if not $T[x, \mathrm{Img}]$ then $T[x, \mathrm{Img}] \leftarrow\!\!\$\ \mathrm{Img}$

Return $T[x, \mathrm{Img}]$

$\underline{\mathsf{Prover}(i, \mathsf{args})}$

If $\mathsf{pst}[i] = \perp$ then // Commit

  $(\mathsf{pcom}[i], \mathsf{pst}[i]) \leftarrow\!\!\$\ \mathsf{ID.Com}(\mathsf{prm})$

  Return $\mathsf{pcom}[i]$

Else If $\mathsf{prsp}[i] = \perp$ then // Response

  $\mathsf{prsp}[i] \leftarrow\!\!\$\ \mathsf{ID.Rsp}^{\mathsf{H_r}}(\mathsf{prm}, \mathsf{hlp}, sk, \mathsf{pst}[i], \mathsf{args})$

  Return $\mathsf{prsp}[i]$

Return $\perp$

Figure 1.13: Game Rewind^1 and Rewind^2 (boxed). The oracle Leak is the same as the one given in Fig. 1.10.

honestly generated and compressed secrete key (sk^*). Let $x \in \{1,2\}$, we use $\Pr[\mathsf{Rewind}^x(\mathsf{Gen}, \mathcal{A})]$ to denote the probablity that the first component of the output of Rewind^x is true. First, using the well-known rewind technique [15], we will argue that

$$\Pr[\mathsf{Rewind}^1(\mathsf{Gen}, \mathcal{A})] \geq \mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})^2 - \frac{1}{p}\,. \tag{1.29}$$

We now justify Equation (1.29). We consider the event that the flags A, B, C are all set to true. Notice that the marginal probability that A is true and the marginal probability that B is ture are both exactly $\Pr[\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})]$. We partition the random tape for $\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})$ into two parts: the random

Adversary $\mathcal{A}_{\mathrm{cdh}}(\mathcal{G}, v, h)$

$(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$
$(t, z_1, z_2, z^*, sk^*) \leftarrow \mathsf{Rewind}^2(\mathsf{Gen}_{\mathrm{cdh}}, \mathcal{A})$
$(pk_1^*, \sigma_1^*, z^{(1)}) \leftarrow z_1$
$(pk_2^*, \sigma_2^*, z^{(2)}) \leftarrow z_2$
$(pk^*, \sigma^*, z^*) \leftarrow z^*$
$\hat{\sigma} \leftarrow ((\sigma_1^*)^{c_1^*}/(\sigma_2^*)^{c_2^*}) \cdot \sigma^{*c_2^* - c_1^*}$
$\omega = \sum_{j=1}^{m} \gamma_j (z_j^{(1)} - z_j^{(2)} - x_j^*(c_1^* - c_2^*))$
$s' \leftarrow (\hat{\sigma})^{1/\omega}$
Return s'

$\mathsf{Gen}_{\mathrm{cdh}}()$

$(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$
For $j \in [m]$ do
  $\gamma_j \leftarrow \mathbb{Z}_p$; $g_j \leftarrow h^{\gamma_j}$
$\mathsf{prm} = (g_0, \ldots, g_{m-1}, g)$; $\mathsf{vk} \leftarrow v$
For $i \in [m]$ do
  $sk[i] \leftarrow [p]^m$; $pk[i] \leftarrow \prod_{j=0}^{m-1} g_j^{sk[i][j]}$
  $\beta_i \leftarrow [p]$; $\sigma[i] \leftarrow \mathsf{vk}^{\beta_i}$
  $T[i, G] \leftarrow g^{\beta_i}/pk[i]$
$\mathsf{hlp} \leftarrow (pk, \sigma)$
Return $(\mathsf{prm}, \mathsf{vk}, sk, \mathsf{hlp}, T)$

Adversary $\mathcal{A}_{\mathrm{dl}}(\mathcal{G}, X)$

$(t, z_1, z_2, z^*, sk^*) \leftarrow \mathsf{Rewind}^2(\mathsf{Gen}_{\mathrm{dl}}, \mathcal{A})$
$(pk_1^*, \sigma_1^*, z^{(1)}) \leftarrow z_1$
$(pk_2^*, \sigma_2^*, z^{(2)}) \leftarrow z_2$
$(pk^*, \sigma^*, z^*) \leftarrow z^*$
For $j = 1, \ldots, m$ do
  $\hat{sk}^*[j] \leftarrow (z^{(1)}[j] - z^{(2)}[j])/(c_1^* - c_2^*)$
$x \leftarrow (\sum_{i \in [m]-\{\rho\}} x_i(\hat{sk}^*[i] - sk^*[i]))/(sk^*[\rho] - \hat{sk}^*[\rho])$
Return x

$\mathsf{Gen}_{\mathrm{dl}}()$

$\rho \leftarrow_{\$} [m]$; $g_\rho \leftarrow X$
For $j \in [m] - \{\rho\}$ do
  $x_j \leftarrow_{\$} \mathbb{Z}_p$; $g_j \leftarrow g^{x_j}$
$\mathsf{prm} = (g_0, \ldots, g_{m-1}, g)$
$(pk, sk, \mathsf{hlp}) \leftarrow \mathsf{ADW.KeyGen}(\mathsf{prm})$
Return $(\mathsf{prm}, pk, sk, \mathsf{hlp}, \perp)$

Figure 1.14: **Left:** Adversary $\mathcal{A}_{\mathrm{cdh}}$. **Right:** Adversary $\mathcal{A}_{\mathrm{dl}}$.

tape that is used upto right before $\mathcal{A}.\mathsf{Rsp}$ is run, and the rest of the tape that is used after $\mathcal{A}.\mathsf{Rsp}$ starts its execution. Let T be a random variable denoting the first part of the random tape. For any value of T, say t, we let $\mathbf{G}(t)$ be the game $\mathbf{G}_{\mathsf{ID},\ell}^{\mathrm{imp}}(\mathcal{A})$ with the first part of random tape fixed to t.

We have that

$$\begin{aligned}
\Pr\left[\mathsf{Rewind}^1(\mathsf{Gen},\mathcal{A})\right] &= \Pr[A \wedge B \wedge C] \\
&= \mathsf{E}_T\left[\Pr[A \wedge B \wedge C \mid T]\right] \\
&\geq \mathsf{E}_T\left[\Pr[A \wedge B \mid T] - \Pr[\neg C \mid T]\right] \\
&= \mathsf{E}_T[\Pr[\mathbf{G}(T)]^2] - \frac{1}{p} \\
&\geq \Pr\left[\mathbf{G}^{\mathsf{imp}}_{\mathsf{ID},\ell}(\mathcal{A})\right]^2 - \frac{1}{p} \quad ,
\end{aligned}$$

where at the last step we used Jensen's inequality and the convexity of squaring. This justifies Equation (1.29). Second, we argue that

$$\Pr[\mathsf{Rewind}^1(\mathsf{Gen},\mathcal{A})] - \Pr[\mathsf{Rewind}^2(\mathsf{Gen},\mathcal{A})] \leq \Pr[\mathsf{Rewind}^1(\mathsf{Gen},\mathcal{A}) \text{ sets bad}] = \frac{q}{2^r} \,. \tag{1.30}$$

This is because the size of the table C is upper-bounded by the number of queries that $\mathcal{A}.\mathsf{Setup}$ and $\mathcal{A}.\mathsf{Com}$ makes to H and Derive, which is q. Next, we attempt to bound $\Pr[\mathsf{Rewind}^2(\mathsf{Gen},\mathcal{A})]$. We define the following events in the game $\mathsf{Rewind}^2(\mathsf{Gen},\mathcal{A})$.

$$E: \quad A \wedge B \wedge C \wedge \left(\frac{(pk_1^*)^{(c_1^*)}}{(pk_2^*)^{(c_2^*)}} = (pk^*)^{c_1^*-c_2^*}\right) ,$$

$$\overline{E}: \quad A \wedge B \wedge C \wedge \left(\frac{(pk_1^*)^{(c_1^*)}}{(pk_2^*)^{(c_2^*)}} \neq (pk^*)^{c_1^*-c_2^*}\right) .$$

Notice that per construction of the events,

$$\Pr[\mathsf{Rewind}^2(\mathsf{Gen},\mathcal{A})] = \Pr[E] + \Pr[\overline{E}] \,. \tag{1.31}$$

Consider $\mathcal{A}_{\mathsf{cdh}}$ (Fig. 1.14) and $\mathcal{A}_{\mathsf{dl}}$ (Fig. 1.14), which attemps to break CDH and DL problems, respectively, using Rewind^2. We will show the following (in)equalities

$$\Pr[\overline{E}] = \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{cdh}}) \;, \tag{1.32}$$

and

$$\Pr[E] \le m \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}_{\mathsf{dl}}) + \mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell + \frac{k}{m}) \;. \tag{1.33}$$

This part of the analysis follows from [6, Appendix B.5] and we restate their derivation here. Assume E or $\overline{E}$, since the signatures verifies, for $w = \prod_{i=0}^{\tau} \mathsf{H}(\mathsf{pp}[i], G)^{e^i}$, we have

$$\sigma^* = (pk^* w)^s \;,\; \sigma_1^* = (pk_1^* w)^s \;,\; \sigma_2^* = (pk_2^* w)^s \;.$$

If $\overline{E}$ happens, the following two values are distinct

$$\frac{(\sigma_1)^{c_1^*}}{(\sigma_2)^{c_2^*}} = \left(w^{(c_1^* - c_2^*)} \frac{(pk_1^*)^{c_1^*}}{(pk_2^*)^{c_2^*}} \right)^s \;, \quad (\sigma^*)^{c_1^* - c_2^*} = \left(w^{c_1^* - c_2^*} (pk)^{c_1^* - c_2^*} \right)^s \;.$$

Hence, the value $\hat{\sigma}$ computed by $\mathcal{A}_{\mathsf{cdh}}$ is

$$\hat{\sigma} = \left(\frac{(pk_1^*)^{c_1^*}}{(pk_2^*)^{c_2^*}} \cdot \frac{1}{(pk^*)^{c_1^* - c_2^*}} \right)^s = (g^{\omega})^s \;.$$

Therefore, $\mathcal{A}_{\mathsf{cdh}}$ can compute g^s and solve the CDH problem that it was given. This concludes the proof for Equation (1.32). If E happens, then we claim that $sk^* = \hat{sk}$ with probability at most $\mathbf{Adv}^{\mathrm{pskp}}_{p,k,k,\tau}(\ell + k/m)$. This is true per definition of $\mathbf{Adv}^{\mathrm{pskp}}_{p,k,k,\tau}(\ell + k/m)$. Notice that if $sk^* \ne \hat{sk}$, then with probability $1/m$, $\mathcal{A}_{\mathsf{dl}}$ can solved the DL problem that it embedded into the parameters. This is because $\mathcal{A}_{\mathsf{dl}}$ has two representation of pk^* in the basis $g_0, \ldots, g_{m-1}$, namely sk^* and $\hat{sk}$. This concludes the proof of Equation (1.33). Notice that Equations (1.29), (1.31), (1.30), (1.32), and (1.33) together implies the theorem. Finally, notice that $\mathcal{A}_{\mathsf{cdh}}$ and $\mathcal{A}_{\mathsf{dl}}$ has roughly the running time of Rewind^2 and ADW.KeyGen, which is about $t_1 + t_2 + 2t_3 + t_4$. ∎

1.6 Proofs of Lemmas 1.3.2, 1.3.3, and 1.3.4

We need the following version of Stirling's approximation of $n!$.

Lemma 1.6.1 *[77] For any $n \in \mathbb{Z}^+$,*

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} \leq n! \leq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}} .$$

We first prove Lemma 1.3.2.

Proof of Lemma 1.3.2: We first show the lower bound Equation (1.6) . Notice that by definition of $H_q(r/k)$,

$$q^{kH_q(r/k)} = (q-1)^r (r/k)^{-r} (1 - r/k)^{r-k} .$$

Hence, by Lemma 1.6.1,

$$\begin{aligned}
B_{q,k}(r) &= \sum_{i=0}^{r} (q-1)^i \binom{k}{i} \\
&\geq (q-1)^r \frac{k!}{r!(k-r)!} \\
&\geq (q-1)^r \frac{\sqrt{2\pi k}(\frac{k}{e})^k e^{\frac{1}{12k+1}}}{\sqrt{2\pi r}(\frac{r}{e})^r e^{\frac{1}{12r}} \sqrt{2\pi(k-r)}(\frac{k-r}{e})^{k-r} e^{\frac{1}{12(k-r)}}} \\
&= q^{kH_q(r/k)} \frac{\sqrt{k} e^{\frac{1}{12k+1}}}{\sqrt{2\pi r(k-r)} e^{\frac{1}{12r}} e^{\frac{1}{12(k-r)}}} \\
&= q^{kH_q(r/k) - \varepsilon(k,r)} .
\end{aligned}$$

Now, we assume that $r \leq k - k/q$ and derive the upper bound, Equation (1.7).

$$\begin{aligned}
\frac{B_{q,k}(r)}{q^{kH_q(r/k)}} &= \frac{\sum_{i=0}^{r}(q-1)^i\binom{k}{i}}{(q-1)^r(r/k)^{-r}(1-r/k)^{r-k}} \\
&= \sum_{i=0}^{r}\binom{k}{i}(q-1)^{i-r}(r/k)^r(1-r/k)^{k-r} \\
&= \sum_{i=0}^{r}\binom{k}{i}(q-1)^i(1-r/k)^k\Big(\frac{r/k}{(q-1)(1-r/k)}\Big)^r \\
&\leq \sum_{i=0}^{r}\binom{k}{i}(q-1)^i(1-r/k)^k\Big(\frac{r/k}{(q-1)(1-r/k)}\Big)^i \\
&= \sum_{i=0}^{r}\binom{k}{i}(r/k)^i(1-r/k)^{k-i} \\
&\leq \sum_{i=0}^{k}\binom{k}{i}(r/k)^i(1-r/k)^{k-i} \\
&= 1\ ,
\end{aligned}$$

where the first inequality is by the fact that $r/k \leq (q-1)(1-r/k)$ if $r \leq k - k/q$. ▮ Lemma 1.3.3 follows from Lemma 1.3.2. **Proof of Lemma 1.3.3:** Per definition of $\mathsf{rd}_{q,k}(N)$, it suffices to show that

$$B_{q,k}(r) \leq N\ ,$$

for $r = \lfloor H_q^{-1}(\log_q(N)/k) \cdot k \rfloor$. Per definition of H_q^{-1}, $r \leq (1-1/q) \cdot k$. Hence, we can apply Equation (1.7) and obtain

$$B_{q,k}(r) \leq q^{kH_q(r/k)} \leq q^{kH_q(H_q^{-1}(\log_q(N)/k))} = N\ . \qquad ▮$$

Lastly, we prove Lemma 1.3.4.

Proof of Lemma 1.3.4: We first show the lower bound that

$$\min(x, 1-\frac{1}{q}) - \frac{1}{\log_2(q)} \leq H_q^{-1}(x) \,. \tag{1.34}$$

Note that this is trivially true if the left-hand side of Equation (1.34) is negative. Hence, we suppose that the left-hand side of Equation (1.34) is non-negative. As noted before, H_q is increasing in the domain $[0, 1-1/q]$. Additionally, note that $\min(x, 1-1/q) - 1/\log_2(q) \leq 1-1/q$. Hence, it suffices to show

$$H_q\Big(\min(x, 1-\frac{1}{q}) - \frac{1}{\log_2(q)}\Big) \leq x \,. \tag{1.35}$$

We consider two cases. Case 1, $x \leq (1-1/q)$. Case 2, $(1-1/q) \leq x \leq 1$. We claim that both cases follow from the equation below, which holds for $x \in [\log_2(q), 1]$.

$$H_q(x - \frac{1}{\log_2(q)}) \leq x \,. \tag{1.36}$$

Case 1 is directly implied by Equation (1.36). For case 2, note that the left-hand side of Equation (1.34) always evaluate to $1-1/q-1/\log_2(q)$. Hence, by Equation (1.36), $H_q(1-1/q-1/\log_2(q)) \leq 1-1/q \leq x$. Finally, we justify Equation (1.36). Recall that $H_q(x) =$

$H_2(x)/\log_2(q) + x\log_q(q-1)$. We compute

$$\begin{aligned}
H_q(x - \frac{1}{\log_2(q)}) &= \frac{H_2(x - \frac{1}{\log_2(q)})}{\log_2(q)} + (x - \frac{1}{\log_2(q)})\log_q(q-1) \\
&\leq \frac{1}{\log_2(q)} + x\log_q(q \cdot \frac{q-1}{q}) - \frac{\log_q(q-1)}{\log_2(q)} \\
&= \frac{1}{\log_2(q)} + x - x\log_q(\frac{q}{q-1}) - \frac{\log_q(q-1)}{\log_2(q)} \\
&= x + \frac{1}{\log_2(q)}(1 - \log_q(\frac{q^{(x\log_2(q))}}{(q-1)^{(x\log_2(q)-1)}})) \\
&\leq x + \frac{1}{\log_2(q)}(1 - \log_q(q)) \\
&= x.
\end{aligned}$$

Next, we show the upper bound that

$$H_q^{-1}(x) \leq x(1 - \frac{1}{q}) \; . \qquad (1.37)$$

Similar to the lower bound we just obtained, we note that it suffices to show $H_q(x(1-\frac{1}{q})) \geq x$. Let us define, for $x \in [0,1]$:

$$f(x) = \frac{x}{q}\log_q(\frac{x}{q}) - x\log_q(x) - (1 - x + \frac{x}{q})\log_q(1 - x + \frac{x}{q}) \; .$$

We will show that $H_q(1(1-1/q)) = x + f(x)$. The derivation is as follows.

$$
\begin{aligned}
& H_q(x(1-1/q)) \\
& = x(1-1/q)\log_q(q-1) - x(1-1/q)\log_q(x(1-1/q)) \\
& \quad - (1-x+x/q)\log_q(1-x+x/q) \\
& \geq x\log_q(q-1) - x/q\log_q(q-1) \\
& \quad - x(1-1/q)(\log_q(x) + \log_q(1-1/q)) \\
& \quad - (1-x+x/q)\log_q(1-x+x/q) \\
& = x\log_q(q-1) - x/q\log_q(q-1) \\
& \quad - x\log_q(x) - x\log_q(1-1/q) + x/q\log_q(x) + x/q\log_q(1-1/q) \\
& \quad - (1-x+x/q)\log_q(1-x+x/q) \\
& = x\Big(\log_q(q-1) + \log_q(q/(q-1))\Big) - x\log_q(x) \\
& \quad - x/q\Big(\log_q(q-1) + \log_q(1/x) + \log_q(q/(q-1))\Big) \\
& \quad - (1-x+x/q)\log_q(1-x+x/q) \\
& = x + x/q\log_q(x/q) - x\log_q(x) - (1-x+x/q)\log_q(1-x+x/q) \\
& = x + f(x)\ .
\end{aligned}
$$

Lastly, we show that $f(x) \geq 0$ for any $x \in [0,1]$. First, we check that $f(0) = f(1) = 0$. Next, check that the second derivative of f,

$$f''(x) = \frac{q-1}{x(qx-q-x)} \leq 0\ ,$$

is at most 0 for any $x \in [0,1]$. We omit the details of the derivative computation here. Hence, f is concave over the domain $[0,1]$, with $f(0) = f(1) = 0$. Thence, $f(x) \geq 0$ for all $x \in [0,1]$. ▮

1.7 A Rejection Sampling Lemma

We prove the following lemma, which allows one to efficiently sample from $[k]^{(\tau)}$, for appropriately constrained integers k, τ, via rejection sampling.

Lemma 1.7.1 *Let τ, k, c be positive integers. Suppose $k \geq 2(c+1) \cdot \tau^2$. Let $x_1, \ldots, x_t \leftarrow_{\$} [k]$ be i.i.d samples, where*

$$t = \tau + \left\lceil \frac{c\tau}{\log_2(k) - \log((c+1)\tau^2)} \right\rceil \leq \tau + c\tau\,.$$

Let $S = \{x_1, \ldots, x_t\}$. Then

$$\Pr[|S| < \tau] \leq 2^{-c \cdot \tau}\,.$$

Proof: Let $\delta = t - \tau$. Since $k \geq 2(c+1) \cdot \tau^2$, we have $\log_2(k) - \log_2((c+1)\tau^2) \geq 1$. Hence, $\delta \leq c \cdot \tau$. Define $S_i = \{x_1, \ldots, x_i\}$. Hence, $S_0 = \emptyset$ and $S_t = S$. Suppose that $|S| < \tau$, then there exists at least δ positions i such that $x_i \in S_{i-1}$. Since x_i is a independent uniform sample from $[k]$, the probability that $x_i \in S_{i-1}$ is $|S_{i-1}|/k$, which is at most τ/k. Hence,

$$\begin{aligned}
\Pr[|S| < \tau] &\leq \binom{\tau+\delta}{\delta} (\frac{\tau}{k})^{\delta} \\
&\leq \binom{(c+1)\tau}{\delta} (\frac{\tau}{k})^{\delta} \\
&\leq \left(\frac{(c+1)\tau^2}{k}\right)^{\delta}.
\end{aligned}$$

Hence,

$$\log_2(\Pr[|S| < \tau]) \leq \delta \log_2(\frac{(c+1)\tau^2}{k}) \leq -c \cdot \tau\,.$$ ▮

1.8 Acknowledgements

We thank the anonymous CCS 2018 reviewers for their helpful comments. We thank Yevgeniy Dodis and Daniel Wichs for helpful discussions.

Bellare was supported in part by NSF grants CNS-1526801 and CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft corporation. Dai was supported in part by a Powell Fellowship and grants of Bellare.

Chapter 1, in full, is a repreint of the material as it appears in ACM Conference on Computer and Communications Security, 2017. Bellare, Mihir; Dai, Wei. The dissertation author was the primary investigator and author of this paper.

Chapter 2

Tight and Non-rewinding Proofs for Schnorr Identification and Signatures

2.1 Introduction

It would not be an exaggeration to say that Schnorr identification and signatures [79] are amongst the best-known and most influential schemes in cryptography. With regard to practical impact, consider that Ed25519, a Schnorr-derived signature scheme over twisted Edwards curves [20], is used, according to IANIX [55], in over 200 different things. (OpenSSL, OpenSSH and GnuPG to name a tiny fraction.) Meanwhile the algebraic structure of the Schnorr schemes has resulted in their being the basis for many advanced primitives including multi- [66, 14, 8, 63], ring- [2, 53] and threshold- [84, 59] signatures.

Proving security of these schemes has accordingly attracted much work. Yet, all known standard-model proofs [76, 1, 58] exhibit a gap: the proven bound on adversary advantage (success probability) is much inferior to (larger than) the one that cryptanalysis says is "true." (The former is roughly the square-root of the latter. Accordingly we will refer to this as the square-root gap.)

The square-root gap is well known and acknowledged in the literature. Filling this long-standing and notorious gap between theory and practice is the subject of this paper. We start with some background.

<u>SCHNORR SCHEMES.</u> Let $\mathbb{G}$ be a group of prime order p, and $g \in \mathbb{G}$ a generator of $\mathbb{G}$. We let $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ denote the Schnorr identification scheme [79] (shown in Figure 2.3). The security goal for it is IMP-PA (impersonation under passive attack [44]). The Schnorr signature scheme $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$ [79] is derived from ID via the Fiat-Shamir transform [45] (also shown in Figure 2.3). The security goal for it is UF (unforgeability under chosen-message attack [50]) in the ROM (random oracle model [18]).

Recall that, $\mathbb{G}, g$ being public, the DL problem is for an adversary, given $X = g^x$, to recover x. Since we will introduce variants, we may, for emphasis, refer to DL itself as the "basic" version of the discrete-logarithm problem. Existing standard-model proofs for both ID and DS [76, 1, 58] are based on the assumed hardness of DL. The heart of the proof for DS, and the cause of the square-root gap, is the rewinding reduction in the proof for ID. This makes ID the first and most basic concern.

<u>THE SITUATION WITH ID.</u> The simplest proof of IMP-PA for $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ uses the Reset Lemma of [15]. It shows that, roughly:

$$\varepsilon^{\text{imp-pa}}(t) \leq \sqrt{\varepsilon^{\text{dl}}(t)} + \frac{1}{p} , \tag{2.1}$$

where $\varepsilon^{\text{imp-pa}}(t)$ is the probability of breaking IMP-PA security of ID in time t and $\varepsilon^{\text{dl}}(t)$ is the probability of breaking DL in time t. To draw quantitative conclusions about $\varepsilon^{\text{imp-pa}}(t)$ as required in practice, however, we now also need to estimate $\varepsilon^{\text{dl}}(t)$. The accepted way to do this is via the Generic Group Model (GGM) bound [81], believed to be accurate for elliptic curve groups. It says that

$$\varepsilon^{\mathrm{dl}}(t) \approx \frac{t^2}{p} . \tag{2.2}$$

Putting together the two equations above, we get, roughly:

$$\varepsilon^{\text{imp-pa}}(t) \leq \frac{t}{\sqrt{p}} . \tag{2.3}$$

There is, however, no known attack matching the bound of Eq. (2.3). Indeed, the best known time t attack on ID is via discrete-log computation and thus has the considerably lower success probability of t^2/p. For example if $p \approx 2^{256}$ the best known attack against ID gives a time $t = 2^{80}$ attacker a success probability of $t^2/p = 2^{-96}$, but Eq. (2.3) only rules out a success probability of $t/\sqrt{p} = 2^{-48}$. The proof is thus under-estimating security by a fairly large margin.

Accordingly in practice 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. (2.1) to bring the theory more in line with the indications of cryptanalysis has been a long-standing open question.

TIERS AND KNOBS. Before continuing with how we address this question, we draw attention to the two-tiered framework of a security proof for a scheme S (above, $\mathsf{S} = \mathsf{ID}$) based on the assumed hardness of some problem P (above, P=DL). The first tier is the reduction from P. It is represented above by Eq. (2.1). The second tier is the estimate of the security of P itself, made (usually) in an idealized model such as the GGM [81] or AGM (Algebraic Group Model) [47]. It is represented above by Eq. (2.2). Both tiers are necessary to draw quantitative conclusions. This two-tier structure is an accepted one for security proofs, and widely, even if not always explicitly, used.

In this structure, we have the flexibility of choice of P, making this a "knob" that we can tune. Creative and new choices of P have historically been made, and been very valuable in cryptography, yielding proofs for existing schemes and then going on to be useful beyond. Historically, a classical example of such a (at the time, new) P is the Diffie-Hellman problem; the schemes S whose proof this allows include the Diffie-Hellman secret-key exchange [39] and the El Gamal public-key encryption scheme [43]. An example P closer to our work is the One-More Discrete Logarithm (OMDL) problem [13], which has by now been used to prove many schemes S [15, 37, 75, 46, 40]. But this knob-tuning approach is perhaps most visible in the area of bilinear maps, where many choices of problem P have been made, justified in the GGM, and then used to prove security of many schemes S. In the same tradition, we ask, how can we tune the knob to fill the square-root gap? Our answer is a choice of P we call MBDL.

MBDL. Our Multi-Base Discrete Logarithm (MBDL) problem is a variant of the One-More Discrete Logarithm (OMDL) problem of [13]. Continue to fix a cyclic group $\mathbb{G}$ and generator g of $\mathbb{G}$. In MBDL, 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 $p = |\mathbb{G}|$.

CORE RESULTS. We suggest that the square-root gap of Eq. (2.1) is a manifestation of an unformalized strength of the discrete logarithm problem. We show that this strength is captured by the MBDL problem. We do this by giving a proof of IMP-PA security of the Schnorr identification scheme $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ with a *tight* reduction from 1-MBDL: letting $\varepsilon^{\text{1-mbdl}}(t)$

Table 2.1: **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.

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

be the probability of breaking the 1-MBDL problem in time t, Theorem 2.4.1 says that, roughly:

$$\varepsilon^{\text{imp-pa}}(t) \le \varepsilon^{\text{1-mbdl}}(t) + \frac{1}{p} . \tag{2.4}$$

Eq. (2.4) does not suffer from the square-root gap of Eq. (2.1). Progress. But this is in the first of the two tiers discussed above. Turning to the second, we ask, how hard is MBDL? Theorem 2.5.1 shows that, in the GGM, roughly:

$$\varepsilon^{\text{1-mbdl}}(t) \approx \frac{t^2}{p} . \tag{2.5}$$

That is, 1-MBDL problem has essentially the same GGM quantitative hardness as DL. Putting

Eqs. (2.4) and (2.5) together, we get (roughly) the following improvement over Eq. (2.3):

$$\varepsilon^{\text{imp-pa}}(t) \leq \frac{t^2}{p} \,. \tag{2.6}$$

This bound is tight in the sense that it matches the indications of cryptanalysis.

A direct indication of the practical value of this improvement is that, for a given target level of provable security, we increase efficiency. Thus suppose that, for some chosen values of ε, t, we want to pick the group $\mathbb{G}$ to ensure $\varepsilon^{\text{imp-pa}}(t) \leq \varepsilon$. Eq. (2.6) allows us to use smaller groups than Eq. (2.3). Since scheme algorithms have running time cubic in the log of $p = |\mathbb{G}|$, this results in a performance improvement. Figure 2.1 says that this improvement can range from 1.9x to 3x.

<u>WHAT HAS BEEN GAINED?</u> A natural question is that our results rely on a new assumption (MBDL), so what has been gained? Indeed, MBDL, as with any new assumption, should be treated with caution. However, it seems that improving Eq. (2.1) to something like Eq. (2.4) under the basic DL assumption is out of reach and likely not 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.

Of course it would not be hard to introduce *some* problem which allows this. But we believe MBDL, and our results, are "interesting" in this regard, for the following reasons. First, MBDL is not a trivial reformulation of the IMP-PA security of ID, meaning we are not just assuming the square-root problem out of existence. Second, and an indication of the first, is that the proof of the IMP-PA security of ID from MBDL (see "Reduction approach" below) is correspondingly not trivial. Third, the use of MBDL is not confined to Schnorr identification; as we also discuss below under "MBDL as a hub," it already has many further applications and uses, and we imagine even more will arise in the future.

REDUCTION APPROACH. The proof of Eq. (2.1) uses a 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. The Reset Lemma [15] says it is successful with probability roughly the square of the IMP-PA advantage of $\mathcal{A}$, leading to the square-root in Eq. (2.1).

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. (2.4) 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 2.4.

MBDL AS A HUB. Having identified MBDL, we find that its applicability extends well beyond what is discussed above, making it a hub. Here we briefly discuss further results from MBDL.

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 [76, 73, 1, 58]. The bound —recalled in Eq. (2.15)— continues to exhibit the square-root gap. Theorem 2.4.3 gives a square-root avoiding reduction from 1-MBDL to fill this gap. Figure 2.1 shows resulting speedup factors of 1.6x to 2.5x for Schnorr signatures.

Security above refers to the single-user setting. Our results extend to tightly reduce the multi-user IMP-PA security of $\mathsf{SchID}[\mathbb{G}, g]$ to 1-MBDL, and analogously for signatures. This can be shown directly, but is also a consequence of general results of [58].

The situation for the Okamoto identification and signature schemes [74] is analogous to that for Schnorr, meaning the reductions in the current security proofs, from DL, use rewinding and has the square-root loss. In Section 2.6 we give results for Okamoto that are analogous to our results for Schnorr, meaning reductions from 1-MBDL that avoid the square root.

There's more. In a follow-up work, we also give reductions from MBDL that improve security of the following: (1) Bellare-Neven multi-signatures [14] (2) Abe, Ohkubo, Suzuki 1-out-of-n (ring/group) signatures [2] and (3) Schnorr-based threshold signatures [84].

RELATED WORK. One prior approach to resolving the square-root gap has been to use *only* an idealized model like the GGM [81] or AGM [47]. Thus, Shoup [81] directly showed that $\varepsilon^{\text{imp-pa}}(t) \leq t^2/p$ in the GGM. Fuchsbauer, Plouviez and Seurin [48] give, in the AGM, a tight reduction from DL to the UF security of $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$. These results correspond to a setting of the knob, in the above-discussed two-tier framework, that is maximal: P is the target scheme itself (here Schnorr identification or signatures), so that the first tier is trivial and the second tier directly proves the scheme secure in the idealized model.

But it is well understood that idealized models have limitations. Proofs in the GGM assume the adversary does not exploit the representation of group elements. In the AGM, it is assumed that, whenever an adversary provides a group element Z, it is possible to extract its representation as a product of known powers of prior group elements. This is analogous to a "knowledge of exponent assumption" [35, 51, 16]. However, even in a typical elliptic curve group, an adversary can quite easily create group elements without "knowing" such a representation. The maximal setting of knob (working purely in an idealized model) means the security guarantee on the scheme is fully subject to the limitations of the idealized model.

With MBDL, we, instead make a non-trivial, moderate setting of the knob. Our tight reductions from MBDL, such as Eq. (2.4), are in the standard model, and make no GGM or AGM-like assumptions on adversaries. It is of course true that we justify MBDL in the GGM (Theorem 2.5.1), but we are limiting the use of the idealized model to show security for a purely number-theoretic problem, namely MBDL. The first direct benefit is better security guarantees for the schemes. The second is that MBDL is a hub. As discussed above, we can prove security of many schemes from it, which reduces work compared to proving them all from scratch in idealized models, and also increases understanding by identifying a problem that is at the core of many things.

Another prior approach to improving reduction tightness has been to change metrics, measuring tightness, not via success probability and running time taken individually, but via their ratio [58]. This however does not translate to actual, numeric improvements. To discuss this further, let IMP-KOA denote impersonation under key-only attack. (That is, IMP-PA for adversaries making zero queries to their transcript oracle.) Kiltz, Masny and Pan (KMP) [58] define a problem they call 1-IDLOG that is a restatement of ("precisely models," in their language) the IMP-KOA security of $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$. Due to the zero knowledge of ID, its IMP-PA security reduces tightly to its IMP-KOA security and thus to 1-IDLOG. Now, KMP [58] give a reduction of 1-IDLOG to DL that is ratio-tight, meaning preserves ratios of advantage to running time. This, however, uses rewinding, and is not tight in our sense, incurring the usual square-root loss when one considers running time and advantage separately. In particular the results of KMP do not seem to allow group sizes any smaller than allowed by the classical Eq. (2.1). Our reductions, in contrast, are tight for advantage and time taken individually, and across the full range for these values, and numerical estimates (Figure 2.1) show clear improvements over what one gets from Eq. (2.1). Also our results establish 1-IDLOG tightly (not merely ratio-tightly) under 1-MBDL. We discuss ratio-tightness further in Section 2.7.

<u>DISCUSSION.</u> Measuring quality of a reduction in terms of bit security effectively only reflects the resources required to attain an advantage close to 1. Under this metric, whether one

starts from Eq. (2.1) or Eq. (2.4), one concludes that $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ has $\log_2(|\mathbb{G}|)/2$-bits of security. This reflects bit security being a coarse metric. The improvement offered by Eq. (2.4) over Eq. (2.1) becomes visible when one considers the full curve of advantage as a function of runtime, and is visible in Figure 2.1.

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 [67, 33]. Pairing-based cryptography has seen a host of assumptions that have had many further applications, including the bilinear Diffie-Hellman (BDH) assumption of [26] and the DLIN assumption of [24]. The RSA Φ-Hiding assumption of [29] 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. There are interactive assumptions that are unbroken and successful, like OMDL [13], while many non-interactive ones 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 [16]. (The latter result assumes DL is hard.) MBDL is similarly amenable to cryptanalytic evaluation.

2.2 Preliminaries

<u>NOTATION.</u> 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 $\mathrm{Func}(D,R)$ 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 [19]. (See Fig. 2.2 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 $\Pr[\mathrm{Gm}(\mathcal{A})]$ we denote the event that the execution of game Gm with adversary $\mathcal{A}$ results in output 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, the running 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 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 $\mathbb{Z}_{|\mathbb{G}|}$.

Game $\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}$	Game $\mathbf{G}^{\mathrm{mbdl}}_{\mathbb{G},g,n}$
INIT:	INIT:
$p \leftarrow \lvert\mathbb{G}\rvert$; $y \leftarrow\$ Z_p$; $Y \leftarrow g^y$	$p \leftarrow \lvert\mathbb{G}\rvert$; $y \leftarrow\$ Z_p$; $Y \leftarrow g^y$
Return Y	For $i = 1, \ldots, n$ do
FIN(y'):	$\quad x_i \leftarrow\$ \mathbb{Z}_p^*$; $X_i \leftarrow g^{x_i}$
Return $(y = y')$	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 2.1: Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. 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 $X \in \mathbb{G}^*$.

2.3 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 [13], 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 $\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}$ on the left of Figure 2.1. 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 $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A})]$ be the discrete-log advantage of $\mathcal{A}$.

In the OMDL problem [13], 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. 2.1, where $n \geq 1$ is an integer parameter called the number of bases. The adversary's input, as provided by $\textsc{Init}$, 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 $y' = \mathsf{DL}_{\mathbb{G},g}(Y)$ to $\textsc{Fin}$. We define the mbdl-advantage of $\mathcal{A}$ by

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

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 $a = \mathrm{DLO}(1,Y) = \mathsf{DL}_{\mathbb{G},X_1}(Y)$ and $b = \mathrm{DLO}(1,g) = \mathsf{DL}_{\mathbb{G},X_1}(g)$, 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.

As evidence for the hardness of MBDL, Theorem 2.5.1 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 [82, 80, 83, 57, 49], but, to the best of our assessment, these

$\mathsf{Exec}_{\mathsf{ID}}(pk, sk)$:	Game $\mathrm{Gm}^{\text{imp-pa}}_{\mathsf{ID}}$
$(R, \mathsf{st}) \leftarrow\!\!\$\; \mathsf{ID.Cmt}(pk)$ $c \leftarrow\!\!\$\; \mathsf{ID.Chl}$ $z \leftarrow \mathsf{ID.Rsp}(sk, c, \mathsf{st})$ $b \leftarrow \mathsf{ID.Vf}(pk, R, c, z)$ $\mathsf{tr} \leftarrow (R, c, z)$ Return (b, tr)	$\textsc{Init}$: $(pk, sk) \leftarrow\!\!\$\; \mathsf{ID.Kg}$; Return pk Tr: $(b, \mathsf{tr}) \leftarrow\!\!\$\; \mathsf{Exec}_{\mathsf{ID}}(pk, sk)$; Return tr $\textsc{Ch}(R^*)$: // One query $c^* \leftarrow\!\!\$\; \mathsf{ID.Chl}$; Return c^* $\textsc{Fin}(z^*)$: Return $\mathsf{ID.Vf}(pk, R^*, c^*, z^*)$

Figure 2.2: Left: Algorithm defining an honest execution of the canonical identification scheme ID given key pair (sk, pk). Right: Game defining IMP-PA security of ID.

do not yield attacks on MBDL that beat the GGM bound of Theorem 2.5.1.

The MBDL problem as we have defined it 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 $\mathrm{DLO}(i, \cdot)$ query. In this paper, we do not need or use this extension. We have found applications based on it, but not pursued them because we have been unable to prove security of this extended version of MBDL in the GGM. We consider providing such a GGM proof an intriguing open question, resolving which would open the door to several new applications.

Our formalizations of DL and MBDL fix the generator g. See [9] for a discussion of fixed versus random generators.

2.4 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.

<u>IDENTIFICATION SCHEMES.</u> We recall that a (canonical) identification scheme [1] ID

(see Figure 2.3 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 $\mathsf{ID.Kg}$, $\mathsf{ID.Cmt}$, $\mathsf{ID.Rsp}$, and $\mathsf{ID.Vf}$, as well as a set $\mathsf{ID.Chl}$ of challenges Via $(pk, sk) \leftarrow\!\!\$\, \mathsf{ID.Kg}$, the key generation algorithm generates public verification key pk and associated secret key sk. Algorithms $\mathsf{ID.Cmt}$ and $\mathsf{ID.Rsp}$ are the prover algorithms. The commitment algorithm $\mathsf{ID.Cmt}$ takes input the public key pk 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 $c \in \mathsf{ID.Chl}$ 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. 2.2. It takes input a key pair $(pk, sk) \in [\mathsf{ID.Kg}]$ 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 $(pk, sk) \in [\mathsf{ID.Kg}]$ and any $(b, \mathsf{tr}) \in [\mathsf{Exec}_{\mathsf{ID}}(sk, pk)]$ we have $b = \mathsf{true}$.

Impersonation under passive attack (IMP-PA) [44] 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 $\mathrm{Gm}^{\mathrm{imp\text{-}pa}}_{\mathsf{ID}}$ given in the right column of Fig. 2.2. An adversary has input the public key pk 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

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

$\underline{\mathsf{ID.Kg}:}$	$\underline{\mathsf{DS.Kg}:}$
$x \leftarrow\$ \mathbb{Z}_{\vert\mathbb{G}\vert}$; $X \leftarrow g^x$; Return (X, x)	$x \leftarrow\$ \mathbb{Z}_{\vert\mathbb{G}\vert}$; $X \leftarrow g^x$
$\underline{\mathsf{ID.Cmt}(X):}$	Return (X, x)
$r \leftarrow\$ \mathbb{Z}_{\vert\mathbb{G}\vert}$; $R \leftarrow g^r$; Return (R, r)	$\underline{\mathsf{DS.Sign}^{\mathrm{H}}(x, m):}$
$\underline{\mathsf{ID.Rsp}(x, c, r):}$	$r \leftarrow\$ \mathbb{Z}_{\vert\mathbb{G}\vert}$; $R \leftarrow g^r$
$z \leftarrow (xc + r) \mod \vert\mathbb{G}\vert$	$c \leftarrow \mathrm{H}(R, m)$
Return z	$z \leftarrow (xc + r) \mod \vert\mathbb{G}\vert$
$\underline{\mathsf{ID.Vf}(X, R, c, z):}$	Return (R, z)
$b \leftarrow (g^z = X^c R)$; Return b	$\underline{\mathsf{DS.Vf}^{\mathrm{H}}(X, m, \sigma):}$
	$(R, z) \leftarrow \sigma$
	$c \leftarrow \mathrm{H}(R, m)$
	Return $(g^z = X^c R)$

Figure 2.3: 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 $\mathrm{H} : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$.

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 [79] $\mathsf{ID} = \mathsf{SchID}[\mathbb{G}, g]$ in Fig. 2.3. The public key $pk = X = g^x \in \mathbb{G}$ 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 $\mathsf{ID.Chl} = \mathbb{Z}_p$, 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 [15]. It leads to the following (cf. [15, Theorem 2], [17, 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

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \sqrt{\mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{B})} + \frac{1}{p} \,. \tag{2.7}$$

Additionally, the running time $\mathrm{T}_{\mathcal{B}}$ of $\mathcal{B}$ is roughly $2\,\mathrm{T}_{\mathcal{A}}$ plus simulation overhead $O(\mathrm{Q}^{\mathrm{Tr}}_{\mathcal{A}} \cdot T^{\mathrm{e}}_{\mathbb{G}})$, where $T^{\mathrm{e}}_{\mathbb{G}}$ 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. Refer to Section 3.2 for notation like $\mathrm{T}_{\mathcal{A}}, \mathrm{Q}^{\mathrm{Tr}}_{\mathcal{A}}$.

Theorem 2.4.1 *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 2.4) such that*

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) + \frac{1}{p} \,. \tag{2.8}$$

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

Proof of Theorem 2.4.1: 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

Adversary $\mathcal{B}^{\mathrm{DLO}}$:

$(Y,X) \leftarrow\!\!\$\ \textsc{Init}()$; $z^* \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Ch},\mathrm{Tr}}(X)$; Return z^*

$\textsc{Ch}(R^*)$:

$W \leftarrow R^{*-1} \cdot Y$; $c^* \leftarrow \mathrm{DLO}(1,W)$; 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 Gm_0 / Gm_1 / Gm_2

$\textsc{Init}$: // Games Gm_0, $\boxed{\mathrm{Gm}_1}$

$p \leftarrow |\mathbb{G}|$; $y \leftarrow\!\!\$\ \mathbb{Z}_p$; $Y \leftarrow g^y$; $x \leftarrow\!\!\$\ \mathbb{Z}_p$

If $(x = 0)$ then bad $\leftarrow$ true ; $\boxed{x \leftarrow\!\!\$\ \mathbb{Z}_p^*}$

$X \leftarrow g^x$; Return (Y,X)

$\textsc{Init}$: // Game Gm_2

$p \leftarrow |\mathbb{G}|$; $y \leftarrow\!\!\$\ \mathbb{Z}_p$; $Y \leftarrow g^y$; $x \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $X \leftarrow g^x$; Return (Y,X)

$\textsc{Ch}(R^*)$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1$

$c^* \leftarrow\!\!\$\ \mathbb{Z}_p$; Return c^*

$\textsc{Ch}(R^*)$: // Game Gm_2

$W \leftarrow R^{*-1} \cdot Y$; $c^* \leftarrow \mathsf{DL}_{\mathbb{G},X}(W)$; Return c^*

$\mathrm{Tr}(W)$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$

$z \leftarrow\!\!\$\ \mathbb{Z}_p$; $c \leftarrow\!\!\$\ \mathbb{Z}_p$; $R \leftarrow g^z \cdot X^{-c}$; Return (R,c,z)

$\textsc{Fin}(z^*)$: // Games $\mathrm{Gm}_0, \mathrm{Gm}_1$

Return $(g^{z^*} = X^{c^*} R^*)$

$\textsc{Fin}(z^*)$: // Games Gm_2

Return $(z^* = \mathsf{DL}_{\mathbb{G},g}(X^{c^*} R^*))$

Figure 2.4: Top: MBDL adverary $\mathcal{B}$ for Theorem 2.4.1, based on IMP-PA adversary $\mathcal{A}$. Bottom: Games for proof of Theorem 2.4.1.

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 $c^* = \mathsf{DL}_{\mathbb{G},X}(YR^{*-1})$, 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 must be properly distributed. The analysis will address these.

For the formal proof, consider the games of Figure 2.4. 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 $\mathrm{Gm}_{\mathsf{ID}}^{\text{imp-pa}}$ so we have

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

Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [19] we have

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

Clearly $\Pr[\mathrm{Gm}_1(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq 1/p$. Now we can work with Gm_1, where the public key X is a random element of $\mathbb{G}^*$ rather than of $\mathbb{G}$. We claim that

$$\Pr[\mathrm{Gm}_1(\mathcal{A}) = \Pr[\mathrm{Gm}_2(\mathcal{A})]\ . \tag{2.9}$$

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. (2.9).

We now claim that adversary $\mathcal{B}$, shown in Fig. 2.4, satisfies

$$\Pr[\text{Gm}_2(\mathcal{A})] \leq \mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) . \tag{2.10}$$

Putting this together with the above completes the proof, so it remains to justify Eq. (2.10). Adversary $\mathcal{B}$ has access to oracle DLO as per game $\mathbf{G}^{\text{mbdl}}_{\mathbb{G},g,1}$. 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 $g^{z^*} = X^{c^*} R^* = WR^* = R^{*-1} Y R^* = Y$, so $\mathcal{B}$ wins. ▌

<u>QUANTITATIVE COMPARISON.</u> 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. (2.8) buys us over Eq. (2.7) 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 $\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \varepsilon$ 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 $\varepsilon = 2^{-32}$ is one choice, reflecting a

128-bit security level, where we define the bit-security level as $\log_2(t/\varepsilon)$. 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 (2.7) and (2.8) will yield different choices of k, denoted k_1 and k_2, with $k_2 < k_1$. We will conclude that Eq. (2.8) 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. (2.7), and $\mathcal{B}_2$ the 1-MBDL adversary referred to in (2.8). 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 [81] says that $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}_1) \approx t^2/p$, and our Theorem 2.5.1 says that also $\mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}_2) \approx t^2/p$.

Here we pause to highlight that these two bounds being the same is a central attribute of the 1-MBDL assumption. That Theorem 2.4.1 (as per Figure 2.1) provides efficiency improvements stems not just from the reduction of Eq. (2.8) being tight, but also from that fact that the 1-MBDL problem is just as hard to solve as the DL problem, meaning $\mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g}(\mathcal{B}_2) \approx \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}_1) \approx t^2/p$.

Continuing, putting together what we have so far gives two bounds on the IMP-PA advantage of $\mathcal{A}$, the first via Equations (2.7) and the second via Eq. (2.8), namely, dropping the $1/p$ terms,

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \varepsilon_1(t) = \sqrt{\frac{t^2}{p}} = \frac{t}{\sqrt{p}} \tag{2.11}$$

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \varepsilon_2(t) = \frac{t^2}{p} \,. \tag{2.12}$$

Recall our goal was to ensure that $\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{SchID}[\mathbb{G},g]}(\mathcal{A}) \leq \varepsilon$. We ask, what value of p, in either case, ensures this? Solving for p in the equations $\varepsilon = \varepsilon_1(t)$ and $\varepsilon = \varepsilon_2(t)$, we get two corresponding values, namely $p_1 \approx t^2/\varepsilon^2$ and $p_2 \approx t^2/\varepsilon$. We see that $p_1 > p_2$, meaning Theorem 2.4.1 guarantees the same security as Eq. (2.7) in groups of a smaller size. Finally, the ratio of representation sizes for group elements is

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

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 2.1 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 $\varepsilon = 2^{-64}$, 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 DS.Kg, signing algorithm DS.Sign, deterministic verification algorithm DS.Vf and a set DS.HF of functions called the hash function space. Via $(pk, sk) \leftarrow\!\!\$\, \mathsf{DS.Kg}$ the signer generates a public verification key pk 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 $b \leftarrow \mathsf{DS.Vf}^{\mathsf{h}}(pk, m, \sigma)$, 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 $(pk, sk) \in [\mathsf{DS.Kg}]$, all $m \in \{0,1\}^*$ and all $\sigma \in [\mathsf{DS.Sign}^{\mathsf{h}}(sk, m)]$ we have $\mathsf{DS.Vf}^{\mathsf{h}}(pk, m, \sigma) = \mathsf{true}$.

Game $\mathbf{G}^{\text{uf}}$ in Fig. 2.5 captures UF (unforgeability under chosen-message attack) [50]. Procedure H is the random oracle [18], implemented as a function h chosen at random from

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

$\textsc{Init}$:

$\mathsf{h} \leftarrow\!\!\$\ \mathsf{DS.HF}$; $(pk, sk) \leftarrow\!\!\$\ \mathsf{DS.Kg}$

Return pk

$\textsc{Sign}(m)$:

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

Return σ

$\mathrm{H}(x)$:

Return $\mathsf{h}(x)$

$\textsc{Fin}(m_*, \sigma_*)$:

Return $((m_* \notin S)$ and $\mathsf{DS.Vf}^{\mathrm{H}}(pk, m_*, \sigma_*))$

Figure 2.5: Game defining UF security of signature scheme DS.

$\mathsf{DS.HF}$. 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 [45] to the Schnorr identification scheme. Its algorithms are shown at the bottom right of Fig. 2.3. 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 2.4.2 [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 $\alpha = (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}_{\mathrm{ds}}} + \mathrm{Q}^{\textsc{Sign}}_{\mathcal{A}_{\mathrm{ds}}})\mathrm{Q}^{\textsc{Sign}}_{\mathcal{A}_{\mathrm{ds}}}$. 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* $\mathrm{Q}^{\mathrm{Tr}}_{\mathcal{A}_{\mathrm{id}}} = \mathrm{Q}^{\textsc{Sign}}_{\mathcal{A}_{\mathrm{ds}}}$.

Combining this with Theorem 2.4.1, we have:

Theorem 2.4.3 *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 $\beta = (1 + \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}} + \mathrm{Q}_{\mathcal{A}}^{\mathrm{SIGN}})\mathrm{Q}_{\mathcal{A}}^{\mathrm{SIGN}} + (1 + \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}})$. Then we can construct an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}_{\mathsf{DS}}^{\mathrm{uf}}(\mathcal{A}) \leq (1 + \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}}) \cdot \mathbf{Adv}_{\mathbb{G},g,1}^{\mathrm{mbdl}}(\mathcal{B}) + \frac{\beta}{p} \,. \tag{2.13}$$

Additionally, $\mathrm{T}_{\mathcal{B}}$ is roughly $\mathrm{T}_{\mathcal{A}}$ plus simulation overhead $O(\mathrm{Q}_{\mathcal{A}}^{\mathrm{SIGN}} \cdot T_{\mathbb{G}}^{\mathrm{e}})$.

Let's compare this to prior results. A simple proof of UF-security of DS from DL can be obtained by combining Lemma 2.4.2 with the classical DL-based security of ID as given by Eq. (2.7). For $\mathcal{A}$ an adversary attacking the UF security of DS, this would yield a discrete log adversary $\mathcal{B}$ such that

$$\mathbf{Adv}_{\mathsf{DS}}^{\mathrm{uf}}(\mathcal{A}) \leq (1 + \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}}) \cdot \sqrt{\mathbf{Adv}_{\mathbb{G},g}^{\mathrm{dl}}(\mathcal{B})} + \frac{\beta}{p} \,, \tag{2.14}$$

where β is as in Theorem 2.4.3 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 [14] as per [76]. For $\mathcal{A}$ an adversary attacking the UF security of DS, this would yield a discrete log adversary $\mathcal{B}$ such that

$$\mathbf{Adv}_{\mathsf{DS}}^{\mathrm{uf}}(\mathcal{A}) \leq \sqrt{(1 + \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}}) \cdot \mathbf{Adv}_{\mathbb{G},g}^{\mathrm{dl}}(\mathcal{B})} + \frac{\beta}{p} \,, \tag{2.15}$$

where β and $\mathrm{T}_{\mathcal{B}}$ are as above. The reason Eq. (2.15) is a better bound than Eq. (2.14) is that the $1 + \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}}$ term has moved under the square root. Still we see that Eq. (2.13) is even better; roughly (neglecting the additive term), the bound in Eq. (2.13) is the square of the one in Eq. (2.15), and thus (always) smaller.

QUANTITATIVE COMPARISONS. Our numerical comparisons will be with the best prior bound, meaning that of Eq. (2.15). For a few values of t, q_h, ε with $t \geq q_h = \mathrm{Q}_{\mathcal{A}}^{\mathrm{H}}$, Figure 2.1 shows the speedup s from Eq. (2.13) over Eq. (2.15). 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 $\varepsilon = 2^{-64}$, Theorem 2.4.3 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 SIGN, has advantage $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \varepsilon$ in violating UF security of $\mathsf{DS} = \mathsf{SchSig}[\mathbb{G}, g]$, where t, ε, q_h, q_s are parameters. We assume $q_s < < q_h \leq t$, as one expects in practice. Let $\mathcal{B}_1, \mathcal{B}_2$ be the adversaries of Equations (2.15) and (2.13), respectively. As before, assume $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}_1) \approx t^2/p$ from [81], and also $\mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}_2) \approx t^2/p$ from Theorem 2.5.1. Then

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \varepsilon_1(t, q_h) \approx \sqrt{\frac{q_h t^2}{p}}$$
$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \varepsilon_2(t, q_h) \approx q_h \cdot \frac{t^2}{p} = \frac{q_h t^2}{p} \approx \varepsilon_1(t, q_h)^2 \,.$$

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 $\varepsilon_1, \varepsilon_2$ 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 $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \varepsilon$, in each case. Solving $\varepsilon_1(t, q_h) \leq \varepsilon$ yields $p_1 \approx t^2 q_h/\varepsilon^2$, and solving $\varepsilon_2(t, q_h) \leq \varepsilon$ yields $p_2 \approx t^2 q_h/\varepsilon$. As before we see that $p_2 < p_1$, meaning Theorem 2.4.1 guarantees security in groups of smaller size. The ratio of the representation-size of group elements is

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

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, sgn): / $A, B \in G$ and $\mathrm{sgn} \in \{+, -\}$
If ($A \notin \mathrm{GL}$ or $B \notin \mathrm{GL}$) then return $\perp$
$c \leftarrow (\mathrm{E}^{-1}(A) \text{ sgn } \mathrm{E}^{-1}(B)) \bmod p$; $C \leftarrow \mathrm{E}(c)$; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{C\}$
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 2.6: Game defining n-MBDL problem in the generic group model.

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 2.1 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.

2.5 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 2.5.1 below says that the advantage of a GGM adversary $\mathcal{A}$ in breaking n-MBDL is $O(q^2/p)$ 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.

The proof of Theorem 2.5.1 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. (An example is that of [22] as pointed out by [54]. We note that the assumption was changed to fill the gap in [23].) Also 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.

MBDL IN THE GGM. We start with definitions. Suppose G is a set whose size $p = |G|$ is a prime, and $\mathrm{E}\colon \mathbb{Z}_p \to G$ 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} [86], 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. 2.6 defines, in this way, the n-MBDL problem. The set G parameterizes the game, and the random choice of encoding function $\mathrm{E}\colon \mathbb{Z}_p \to G$ 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 2.5.1 *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 $q = \mathrm{Q}^{\mathrm{OP}}_{\mathcal{A}} + n + 3$. Then*

$$\mathbf{Adv}^{\text{gg-mbdl}}_{G,n}(\mathcal{A}) \leq \frac{2 + q(q-1)}{p-1} \ . \tag{2.16}$$

PROOF FRAMEWORK AND LEMMA. 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 [81] 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.

Game $\mathbf{G}^{\text{smp-1}}_{p,n}$

INIT$(i,\vec{w})$: // $i \in [1..n]$ and $\vec{w} \in \mathbb{Z}_p^{n+2}$

1 $x_1,\ldots,x_n \leftarrow\$\, \mathbb{Z}_p^*$; $y \leftarrow\$\, \mathbb{Z}_p$; $w \leftarrow \vec{w}[i]$

2 $\vec{x}_i \leftarrow (x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n,1,y)$

3 $z \leftarrow w + x_i^{-1} \cdot \langle \vec{w},\vec{x}_i \rangle$; Return $(x_1,\ldots,x_n,y,z)$

Game $\mathbf{G}^{\text{smp-0}}_{p,n}$

INIT$(i,\vec{w})$: // $i \in [1..n]$ and $\vec{w} \in \mathbb{Z}_p^{n+2}$

1 $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n \leftarrow\$\, \mathbb{Z}_p^*$; $y \leftarrow\$\, \mathbb{Z}_p$; $w \leftarrow \vec{w}[i]$

2 $\vec{x}_i \leftarrow (x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n,1,y)$

3 If ($\langle \vec{w},\vec{x}_i \rangle \neq 0$) then $z \leftarrow\$\, \mathbb{Z}_p \setminus \{w\}$; $x_i \leftarrow (z-w)^{-1} \cdot \langle \vec{w},\vec{x}_i \rangle$

4 Else $z \leftarrow w$; $x_i \leftarrow\$\, \mathbb{Z}_p^*$

5 Return $(x_1,\ldots,x_n,y,z)$

Figure 2.7: Games for Lemma 2.5.2.

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 2.7. 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 $\mathbf{G}^{\text{smp-1}}_{p,n}$ 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 $\mathbf{G}^{\text{smp-0}}_{p,n}$ 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 2.5.2 below, as the condition that any adversary returns true with the same probability in the two games.

Lemma 2.5.2 *Let p be a prime and $n \geq 1$ an integer. Then for any adversary $\mathcal{A}$ we have*

$$\Pr[\mathbf{G}_{p,n}^{\text{smp-1}}(\mathcal{A})] = \Pr[\mathbf{G}_{p,n}^{\text{smp-0}}(\mathcal{A})] \,, \tag{2.17}$$

where the games are in Figure 2.7.

Proof of Lemma 2.5.2: 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 $\mathbf{G}_{p,n}^{\text{smp-1}}$. 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 $\mathbf{G}_{p,n}^{\text{smp-0}}$. 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 2.5.1.

Proof of Theorem 2.5.1: By $\text{span}(\vec{v})$ 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 $\mathbf{G}_{G,n,m}^{\text{gg-mbdl}}$, 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. 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 $\mathbf{G}_{G,n,m}^{\text{gg-mbdl}}$ are not triggered. In our games the consequence is that we assume $\text{TI}[A], \text{TI}[B] \neq \bot$ in any $\text{OP}(A, B, \mathsf{sgn})$ query and, for a $\text{DLO}(i, W)$ query, that $i \in [n]$, that $\text{TI}[W] \neq \bot$ and that the number of queries to this oracle is exactly $m = 1$. (The table $\text{TI}[\cdot]$ referred to here starts appearing in Game Gm_0 of Figure 2.8.)

INIT(): **/** Gm_0–Gm_3

1 $p \leftarrow |G|$; $\mathrm{E} \leftarrow\!\!\$\ \mathrm{Bijections}(\mathbb{Z}_p, G)$; $y \leftarrow\!\!\$\ \mathbb{Z}_p$

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 $\mathbb{1}, g, Y, X_1, \ldots, X_n$

$\mathrm{VE}(\vec{t})$: **/** 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 $\mathrm{TV}[\vec{t}]$

$\mathrm{VE}^{-1}(C)$: **/** Gm_0–Gm_3. Here $\mathrm{TI}[C] \neq \bot$.

9 Return $\mathrm{TI}[C]$

OP(A, B, sgn): **/** Gm_0–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

$\mathrm{DLO}(i, W)$: **/** 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'): **/** Gm_0–Gm_3

12 Return $(y = y')$

Figure 2.8: Game Gm_0 for the proof of Theorem 2.5.1. Some procedures will also be in later games, as marked.

We start with game Gm_0 of Figure 2.8, claiming that

$$\mathbf{Adv}^{\text{gg-mbdl}}_{G,n,m}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \,. \tag{2.18}$$

We now explain the game and justify Eq. (2.18). 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 $\mathrm{TV}\colon \mathbb{Z}_p^{n+2} \to G \cup \{\bot\}$ and $\mathrm{TI}\colon G \to \mathbb{Z}_p^{n+2} \cup \{\bot\}$ (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:

$\mathrm{VE}(\vec{t})$: **/** $\boxed{\mathrm{Gm}_1}$, 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 $(\exists \vec{t}' : (\mathrm{TV}[\vec{t}'] \neq \perp \text{ and } \vec{t} - \vec{t}' \in \mathsf{span}(\vec{v})))$ then
15 $\quad C \leftarrow \mathrm{TV}[\vec{t}']$; $\mathrm{TV}[\vec{t}] \leftarrow C$; $\mathrm{TI}[C] \leftarrow \vec{t}$; Return $\mathrm{TV}[\vec{t}]$
16 $C \leftarrow\!\!\$\; G \setminus \mathrm{GL}$
17 If $(\exists \vec{t}' : (\mathrm{TV}[\vec{t}'] \neq \perp \text{ and } \langle \vec{t}, \vec{x} \rangle = \langle \vec{t}', \vec{x} \rangle))$ then
18 $\quad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{C \leftarrow \mathrm{TV}[\vec{t}']}$
19 $\mathrm{TV}[\vec{t}] \leftarrow C$; $\mathrm{TI}[C] \leftarrow \vec{t}$; $\mathrm{GL} \leftarrow \mathrm{GL} \cup \{C\}$; Return $\mathrm{TV}[\vec{t}]$

$\mathrm{DLO}(i, W)$: **/** Gm_1, 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 2.9: Procedures for games $\mathrm{Gm}_1, \mathrm{Gm}_2, \mathrm{Gm}_3$ in the proof of Theorem 2.5.1, where Gm_1 includes the boxed code.

(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. (2.18) 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 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 2.8 and adding those of Figure 2.9, 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 vector $\vec{u} \in \text{span}(\vec{v})$ satisfies $\langle \vec{u}, \vec{x} \rangle = 0$. Now we claim that

$$\Pr[\text{Gm}_1(\mathcal{A})] = \Pr[\text{Gm}_0(\mathcal{A})] \; . \tag{2.19}$$

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 2.8, ought indeed to set $\text{TV}[\vec{t}] = \text{TV}[\vec{t}']$. The inclusion of the boxed code at line 18 further ensures consistency with line 8 of Figure 2.8. So VE is returning the same things in games Gm_1, 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. (2.19).

Games Gm_1, Gm_2 are identical-until-bad as defined in [19]. Let B_2 be the event that $\text{Gm}_2(\mathcal{A})$ sets bad. Then by the Fundamental Lemma of Game Playing [19],

$$\Pr[\text{Gm}_1(\mathcal{A})] \leq \Pr[\text{Gm}_2(\mathcal{A}) \text{ and } \overline{\mathrm{B}}_2] + \Pr[\mathrm{B}_2] \; , \tag{2.20}$$

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

$$\Pr[\text{Gm}_2(\mathcal{A}) \text{ and } \overline{\mathrm{B}}_2] + \Pr[\mathrm{B}_2] \leq \Pr[\text{Gm}_3(\mathcal{A})] \; , \tag{2.21}$$

where game Gm_3 is in Figure 2.10. It includes the boxed code, which game Gm_4 excludes. In these 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 2.5.2. 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

INIT(): // Gm_3–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 $\mathbb{1}, g, Y, X_1, \ldots, X_n$

$\mathrm{VE}(\vec{t})$: // Gm_3–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 $\mathrm{TV}[\vec{t}]$

$\mathrm{VE}^{-1}(C)$: // Gm_3–Gm_5, $\mathrm{Gm}_{\alpha,\beta}$. Here $\mathrm{TI}[C] \neq \bot$.

9 Return $\mathrm{TI}[C]$

OP(A, B, sgn): // Gm_3–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): // $\boxed{\mathrm{Gm}_3}$, 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 $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{z \leftarrow w \text{ ; } x_i \leftarrow\$ \mathbb{Z}_p^*}$

16 $\vec{v} \leftarrow \vec{w} - z \cdot \vec{e}_i$; Return z

FIN(y'): // Gm_3, Gm_4.

17 $\vec{x} \leftarrow (x_1, \ldots, x_n, 1, y)$

18 Return ($(y = y')$ or ($\exists \alpha, \beta : 1 \leq \alpha < \beta \leq k$ and $\langle \vec{t}_\alpha - \vec{t}_\beta, \vec{x} \rangle = 0$))

Figure 2.10: Procedures for games $\mathrm{Gm}_3, \mathrm{Gm}_4$ in the proof of Theorem 2.5.1. Some procedures, as marked, will be used in later games.

accordance with one case of Lemma 2.5.2, 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. (2.21).

Games $\mathrm{Gm}_3, \mathrm{Gm}_4$ are identical-until-bad, so by the Fundamental Lemma of Game Play-

DLO(i,W): // Gm_5, $\mathrm{Gm}_{\alpha,\beta}$. Here $i \in [n]$ and $\mathrm{TI}[W] \neq \perp$.

$\vec{w} \leftarrow \mathrm{VE}^{-1}(W)$; $w \leftarrow \vec{w}[i]$

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'): // $\mathrm{Gm}_{\alpha,\beta}$.

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 2.11: Further procedures to define game Gm_5 and games $\mathrm{Gm}_{\alpha,\beta}$ $(1 \leq \alpha < \beta \leq q)$ in the proof of Theorem 2.5.1.

ing [19],

$$\Pr[\mathrm{Gm}_3(\mathcal{A})] \leq \Pr[\mathrm{Gm}_4(\mathcal{A})] + \Pr[\mathrm{Gm}_4(\mathcal{A}) \text{ sets } \mathsf{bad}] \ . \tag{2.22}$$

We claim

$$\Pr[\mathrm{Gm}_4(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{1}{p-1} \ . \tag{2.23}$$

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. (2.23).

We now define a game Gm_5, and also a game $\mathrm{Gm}_{\alpha,\beta}$ for each $1 \leq \alpha < \beta \leq q$, where

$q = \mathrm{Q}_{\mathcal{A}}^{\mathrm{OP}} + n + 3$. The $\mathrm{DLO}, \mathrm{FIN}$ procedures of these games are shown in Figure 2.11, and the other procedures remain as in Figure 2.10. 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 2.11. 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,

$$\Pr[\mathrm{Gm}_4(\mathcal{A})] \leq \Pr[\mathrm{Gm}_5(\mathcal{A})] + \sum_{1\leq\alpha<\beta\leq q} \Pr[\mathrm{Gm}_{\alpha,\beta}(\mathcal{A})] \; . \tag{2.24}$$

Clearly

$$\Pr[\mathrm{Gm}_5(\mathcal{A})] \leq \frac{1}{p} \; . \tag{2.25}$$

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

$$\begin{aligned} \Pr[\mathrm{Gm}_{\alpha,\beta}(\mathcal{A})] &= \Pr[\mathrm{Z}] = \Pr[\mathrm{Z} \text{ and } \overline{\mathrm{S}}] + \Pr[\mathrm{Z} \text{ and } \mathrm{S}] \\ &\leq \Pr[\,\mathrm{Z} \,|\, \overline{\mathrm{S}}\,] + \Pr[\mathrm{S}] \; . \end{aligned} \tag{2.26}$$

We will show that

$$\Pr[\,\mathrm{Z} \,|\, \overline{\mathrm{S}}\,] \leq \frac{1}{p-1} \tag{2.27}$$

$$\Pr[\mathrm{S}] \leq \frac{1}{p-1} \; . \tag{2.28}$$

We now justify Eq. (2.27). We have

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

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. (2.27).

We turn to Eq. (2.28). 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 2.10, $\vec{t}_\alpha \neq \vec{t}_\beta$, 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

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

This concludes the proof. ∎

Prover		**Verifier**
Input: $(g_2, X), (g_2, x_1, x_2)$		Input: (g_2, X)
$r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p$		
$R \leftarrow g^{r_1} g_2^{r_2}$	$\xrightarrow{R}$	$c \leftarrow\!\!\$\ \mathbb{Z}_p$
	$\xleftarrow{c}$	
$z_1 \leftarrow (x_1 c + r_1) \bmod p$		
$z_2 \leftarrow (x_2 c + r_2) \bmod p$	$\xrightarrow{(z_1, z_2)}$	$b \leftarrow (g^{z_1} g_2^{z_2} = RX^c)$

ID	DS
$\underline{\mathsf{ID.Kg}}$:	$\underline{\mathsf{DS.Kg}}$:
$g_2 \leftarrow\!\!\$\ \mathbb{G}^*$	$g_2 \leftarrow\!\!\$\ \mathbb{G}^*$
$x_1, x_2 \leftarrow\!\!\$\ \mathbb{Z}_{\lvert\mathbb{G}\rvert}$; $X \leftarrow g^{x_1} g_2^{x_2}$	$x_1, x_2 \leftarrow\!\!\$\ \mathbb{Z}_{\lvert\mathbb{G}\rvert}$; $X \leftarrow g^{x_1} g_2^{x_2}$
Return $((g_2, X), (g_2, x_1, x_2))$	Return $((g_2, X), (g_2, x_1, x_2))$
$\underline{\mathsf{ID.Cmt}((g_2, X))}$:	$\underline{\mathsf{DS.Sign}^{\mathrm{H}}((g_2, x_1, x_2), m)}$:
$r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_{\lvert\mathbb{G}\rvert}$; $R \leftarrow g^{r_1} g_2^{r_2}$	$r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_{\lvert\mathbb{G}\rvert}$; $R \leftarrow g^{r_1} g_2^{r_2}$
Return $(R, (r_1, r_2))$	$c \leftarrow \mathrm{H}(R, m)$
$\underline{\mathsf{ID.Rsp}((g_2, x_1, x_2), c, (r_1, r_2))}$:	$z_1 \leftarrow (x_1 c + r_1) \mod \lvert\mathbb{G}\rvert$
$z_1 \leftarrow (x_1 c + r_1) \mod \lvert\mathbb{G}\rvert$	$z_2 \leftarrow (x_2 c + r_2) \mod \lvert\mathbb{G}\rvert$
$z_2 \leftarrow (x_2 c + r_2) \mod \lvert\mathbb{G}\rvert$	Return $(R, (z_1, z_2))$
Return (z_1, z_2)	$\underline{\mathsf{DS.Vf}^{\mathrm{H}}((g_2, X), m, \sigma)}$:
$\underline{\mathsf{ID.Vf}(X, R, c, (z_1, z_2))}$:	$(R, (z_1, z_2)) \leftarrow \sigma$
$b \leftarrow (g^{z_1} g_2^{z_2} = X^c R)$; Return b	$c \leftarrow \mathrm{H}(R, m)$
	Return $(g^{z_1} g_2^{z_2} = X^c R)$

Figure 2.12: Let $\mathbb{G}$ be a group of prime order $p = \lvert\mathbb{G}\rvert$ and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. The Okamoto ID scheme $\mathsf{ID} = \mathsf{OkaID}[\mathbb{G}, g]$ is shown pictorially at the top and algorithmically at the bottom left. At the bottom right is the Okamoto signature scheme $\mathsf{DS} = \mathsf{OkaSig}[\mathbb{G}, g]$, using $\mathrm{H}: \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$.

2.6 Okamoto Identification and Signatures from MBDL

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

<u>Okamoto identification scheme and prior results.</u> Let $\mathbb{G}$ be a group of prime order $p = \lvert\mathbb{G}\rvert$, and $g \in \mathbb{G}^*$ a generator of $\mathbb{G}$. We recall the Okamoto identification scheme [74] $\mathsf{ID} = \mathsf{OkaID}[\mathbb{G}, g]$ in Fig. 2.12. The public key has the form $pk = (g_2, X) \in \mathbb{G}^2$ where g_2 is

a generator and $X = g^{x_1} g_2^{x_2}$, where the secret key is $sk = (g_2, x_1, x_2) \in \mathbb{Z}_p^3$. The commitment is $R = g^{r_1} g_2^{r_2} \in \mathbb{G}$, and (r_1, r_2) is returned as the prover state by the commitment algorithm. Challenges are drawn from $\mathsf{ID.Chl} = \mathbb{Z}_p$, and the response z and decision b are computed as shown.

Given an IMP-PA adversary $\mathcal{A}$ against $\mathsf{ID} = \mathsf{OkaID}[\mathbb{G}, g]$, the classical proof of [74] builds a DL-adversary $\mathcal{B}$, as follows. On input a target point Y whose discrete-log it wants to compute, $\mathcal{B}$ sets $g_2 = Y$. It then itself picks x_1, x_2 and sets $X = g^{x_1} g_2^{x_2}$, so that (x_1, x_2) is what's called a representation of X. Now $\mathcal{B}$ runs $\mathcal{A}$ on public key (g_2, X). Knowing the secret key (g_2, x_1, x_2), it is easy for $\mathcal{B}$ to simulate the Tr oracle. When $\mathcal{A}$ makes its impersonation attempt, rewinding is used, as usual, to obtain two accepting conversation transcripts with the same commitment R^*. From these, $\mathcal{B}$ can compute another representation of X, namely some a_1, a_2 such that $X = g^{a_1} g_2^{a_2}$. The witness indistinguishability property of the protocol says that $(a_1, a_2) \neq (x_1, x_2)$, except with probability $1/p$. Finally, from the two distinct representations of X, adversary $\mathcal{B}$ can compute $\mathsf{DL}_{\mathbb{G},g}(g_2)$. Again the simplest analysis is via the Reset Lemma of [15], which says that

$$\mathbf{Adv}_{\mathsf{ID}}^{\text{imp-pa}}(\mathcal{A}) \leq \sqrt{\mathbf{Adv}_{\mathbb{G},g}^{\mathrm{dl}}(\mathcal{B})} + \frac{2}{p}\,, \tag{2.29}$$

the extra $1/p$ term compared to Equation (2.7) being due to the probability that the two representations are equal. The running time $\mathrm{T}_{\mathcal{B}}$ of $\mathcal{B}$ is roughly $2\,\mathrm{T}_{\mathcal{A}}$ plus simulation overhead $O(\mathrm{Q}_{\mathcal{A}}^{\mathrm{Tr}} \cdot T_{\mathbb{G}}^{\mathrm{e}})$, where $T_{\mathbb{G}}^{\mathrm{e}}$ is the time for an exponentiation in $\mathbb{G}$.

<u>OUR RESULT.</u> We show that the IMP-PA-security of the Okamoto identification scheme reduces *tightly* to the 1-MBDL problem. As with Schnorr, the reduction does not use rewinding.

Theorem 2.6.1 *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 $\mathsf{ID} = \mathsf{OkaID}[\mathbb{G}, g]$ be the Okamoto 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 2.13)*

Adversary $\mathcal{B}^{\mathrm{DLO}}$:

$(Y,X) \leftarrow\!\!\$\ \textsc{Init}()$; $w \leftarrow\!\!\$\ \mathbb{Z}_p^*$; $g_2 \leftarrow g^w$

$(z_1,z_2) \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Ch},\mathrm{Tr}}((g_2,X))$

Return $z_1 + wz_2$

$\textsc{Ch}(R^*)$:

$W \leftarrow R^{*-1} \cdot Y$; $c^* \leftarrow \mathrm{DLO}(1,W)$; Return c^*

Tr:

$z_1,z_2 \leftarrow\!\!\$\ \mathbb{Z}_p$; $c \leftarrow\!\!\$\ \mathbb{Z}_p$; $R \leftarrow g^{z_1}g_2^{z_2} \cdot X^{-c}$; Return $(R,c,(z_1,z_2))$

Figure 2.13: MBDL adverary $\mathcal{B}$ for Theorem 2.6.1, based on IMP-PA adversary $\mathcal{A}$.

such that

$$\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) + \frac{1}{p}\,. \tag{2.30}$$

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

Proof of Theorem 2.6.1: Our reduction from MBDL deviates from the prior one discussed above. It does not set g_2 to the target point Y, instead picking w and setting $g_2 = g^w$. It sets X to a base under which it can take a discrete logarithm. When adversary $\mathcal{A}$ provides R^* in its impersonation attempt, adversary $\mathcal{B}$ picks c^* so that $Y = R^*X^{c^*}$. Then, from $\mathcal{A}$, it gets (z_1,z_2) satisfying $g^{z_1}g_2^{z_2} = R^*X^{c^*} = Y$. Using w, adversary $\mathcal{B}$ then finds $\mathsf{DL}_{\mathbb{G},g}(Y)$. It simulates the Tr oracle using the zero-knowledge simulator. Thus, while in the prior approach the reduction knows the secret key but not $\mathsf{DL}_{\mathbb{G},g}(g_2)$, in ours the reduction does not know the secret key but knows $\mathsf{DL}_{\mathbb{G},g}(g_2)$.

For the formal proof, we claim that the adversary $\mathcal{B}$, shown in Fig. 2.13, satisfies Equation (2.30). Since the analysis is similar to that in the proof of Theorem 2.4.1, we will be brief. The X provided by $\mathcal{B}$ to $\mathcal{A}$ is a generator. In the scheme, $X = g^{x_1+wx_2}$ fails to be generator iff $x_1 + wx_2 = 0$, which happens with probability $1/p$, accounting for this additive term in the bound. Adversary $\mathcal{B}$ simulates the transcript oracle correctly by the usual zero-knowledge method. If $\mathcal{A}$ succeeds, we have $g^{z_1}g_2^{z_2} = R^*X^{c^*}$. But $g^{z_1}g_2^{z_2} = g^{z_1+wz_2}$ and $R^*X^{c^*} = Y$, so $z_1 + wz_2$ can be returned as the discrete log of Y. ▌

<u>OKAMOTO SIGNATURES.</u> The Okamoto signature scheme $\mathsf{DS} = \mathsf{OkaSig}[\mathbb{G}, g]$ is derived by applying the Fiat-Shamir transform [45] to the Okamoto identification scheme. Its algorithms are shown at the bottom right of Fig. 2.12. The set $\mathsf{DS.HF}$ consists of all functions $\mathsf{h}: \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$.

Combining Lemma 2.4.2 with Theorem 2.6.1, we get the following reduction, of the UF security of the Okamoto signature scheme to the 1-MBDL problem, that loses only a factor of the number of hash-oracle queries of the adversary.

Theorem 2.6.2 *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 $\mathsf{DS} = \mathsf{OkaSig}[\mathbb{G}, g]$ be the Okamoto signature scheme. Let $\mathcal{A}$ be an adversary attacking the uf security of ID. Let $\beta = (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}} + \mathrm{Q}^{\mathrm{SIGN}}_{\mathcal{A}})\mathrm{Q}^{\mathrm{SIGN}}_{\mathcal{A}} + (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}})$. Then we can construct an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq (1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}) \cdot \mathbf{Adv}^{\mathrm{mbdl}}_{\mathbb{G},g,1}(\mathcal{B}) + \frac{\beta}{p}\,. \tag{2.31}$$

Additionally, $\mathrm{T}_{\mathcal{B}}$ is roughly $\mathrm{T}_{\mathcal{A}}$ plus simulation overhead $O(\mathrm{Q}^{\mathrm{SIGN}}_{\mathcal{A}} \cdot T^{\mathrm{e}}_{\mathbb{G}})$.

As before, the best prior result, obtained via the general Forking Lemma of [14], said that given an adversary $\mathcal{A}$ attacking the UF security of DS, one can construct a discrete log adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{DS}}(\mathcal{A}) \leq \sqrt{(1 + \mathrm{Q}^{\mathrm{H}}_{\mathcal{A}}) \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B})} + \frac{\beta}{p}\,, \tag{2.32}$$

where β and $\mathrm{T}_{\mathcal{B}}$ are as above. Roughly the bound in Eq. (2.31) is the square of the one in Eq. (2.32), and thus (always) smaller.

2.7 Ratio-based tightness

KMP [58] claims a tight reduction between passive impersonation security of Schnorr identification and discrete log. Their results are claimed to be tight when evaluated under time-to-sucesss ratio. We show here why their result does not give bounds that are as good as ours.

Let ID be the Schnoor identification scheme defined in Section 2.4. Let $\mathcal{A}$ be an adversary against the IMP-PA security of ID with running time $\mathrm{T}_{\mathcal{A}}$. For any given parameter $N \geq 1$, KMP [58][Lemma 3.5] construct a DL adversary $\mathcal{D}_N$ such that

$$\sqrt{\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{D}_N)} \geq 1 - \left[1 - \left(\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) - \frac{1}{p}\right)\right]^N , \tag{2.33}$$

and $\mathrm{T}_{\mathcal{D}_N} = 2N \cdot \mathrm{T}_{\mathcal{A}}$. Notice that when $N = 1$, this is identical to Eq. (2.7), meaning there is no improvement in that case. Next, KMP [58] pick a *specific* value of N that we call N^*. This value is $N^* = (\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) - 1/p)^{-1}$. So the term on the right hand side of Eq. (2.33) becomes

$$1 - \left[1 - \left(\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A}) - \frac{1}{p}\right)\right]^{N^*} \approx 1 - \frac{1}{e} \approx 0.63 , \tag{2.34}$$

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

$$\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}^*) \geq \left(1 - \frac{1}{e}\right)^2 \approx 0.4 \tag{2.35}$$

$$\mathrm{T}_{\mathcal{B}^*} = 2N^* \cdot \mathrm{T}_{\mathcal{A}} \approx \frac{\mathrm{T}_{\mathcal{A}}}{\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A})} . \tag{2.36}$$

Dividing, they obtain the ratio tightness

$$\frac{\mathbf{Adv}^{\text{imp-pa}}_{\mathsf{ID}}(\mathcal{A})}{\mathrm{T}_{\mathcal{A}}} \leq \frac{\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}^*)}{\mathrm{T}_{\mathcal{B}^*}} . \tag{2.37}$$

"Tightness" is claimed because the time-to-success ratio is preserved. However, we will show that one cannot use the above to instantiate parameters that as competitive as the ones guaranteed by our bounds. This is because the running time $\mathrm{T}_{\mathcal{B}^*}$ from Eq. (2.36) is 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. (2.35).

As before, let us the GGM bound for breaking DL, i.e. $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{B}^*) \le \mathrm{T}^2_{\mathcal{B}^*}/p$. Then, from Eq. (2.35) one has $\mathrm{T}_{\mathcal{B}^*} \approx \sqrt{0.4 \cdot p}$, so

$$\frac{\mathbf{Adv}^{\mathrm{imp\text{-}pa}}_{\mathsf{ID}}(\mathcal{A})}{\mathrm{T}_{\mathcal{A}}} \le \frac{0.4}{\sqrt{0.4 \cdot p}}\,, \tag{2.38}$$

which means that one would need a group of size

$$p \approx \left(\frac{\mathrm{T}_{\mathcal{A}}}{\mathbf{Adv}^{\mathrm{imp\text{-}pa}}_{\mathsf{ID}}(\mathcal{A})}\right)^2 . \tag{2.39}$$

This is exactly the same requirement as dictated by the prior results, namely Equation (2.7) and Equation (2.11). Hence, the guarantee by the results of KMP is the same as offered by prior results in Fig. 2.1.

2.8 Acknowledgements

We thank the Indocrypt 2020 reviewers for their insightful comments.

Bellare was supported in part by NSF grant CNS-1717640 and a gift from Microsoft. Dai was supported in part by a Powell Fellowship and grants of the Bellare.

Chapter 2, in full, is a repreint of the material as it appears in International Conference on Cryptology in India, 2020. Bellare, Mihir; Dai, Wei. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Chain Reductions for Multi-signatures and the HBMS Scheme

3.1 Introduction

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

We ask, is it possible to bridge this gap to give some valuable support, in the form of tight reductions, for in-practice schemes? As long as we stay in the current paradigm, namely standard-model proofs from DL, the answer is likely NO. To make progress, we need to be willing to change either the model or the assumption. We show that in fact changing either suffices. Our approach is to give, for any scheme, many different paths to security. In particular we give (1) tight reductions from DL in the Algebraic Group Model (AGM) [47], and (2) tight, standard-model reductions from well-founded assumptions other than DL. We obtain these results via a framework in which a reduction is "factored" into a chain of sub-reductions involving intermediate problems.

We implement this approach first with classical 3-round schemes, giving chain reductions yielding (1) and (2) above for the BN [14] and MuSig [64] schemes. Then, in the space of 2-round schemes, we give a new, efficient scheme, called HBMS, for which we do the same. We now look at all this in more detail.

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

Following its suggestion in the 1980s [56], the primitive has seen much evolution [52, 60, 72, 66, 14]. Early schemes assumed all signers in the signing protocol picked their verification keys honestly. "Rogue-key attacks," in which a malicious signer picked its verification key as function of that of an honest signer, lead to an upgraded target, schemes that retain security even in the presence of adversarially-chosen verification keys. Towards this challenging end we first saw schemes either using interactive key-generation [66] or making the "knowledge of secret key" assumption [21, 61]. Finally, BN [14] gave an efficient, Schnorr-based scheme in the "plain public-key" model, where security was provided even in the face of maliciously-chosen verification keys, yet no more was assumed about these keys than their having certificates as per a standard PKI.

The BN model and definition have become the preferred target; it is the one used in the schemes we discuss next, and in our scheme as well. We denote the security goal as MS-UF. In Section 3.4 we define it via a game, and define the ms-uf advantage of an adversary as its probability of winning this game.

A NEW WAVE. Applications in blockchains and cryptocurrencies —see [25] for details— have fueled a resurgence of interest in multi-signatures. The desire here is MS-UF-secure, DL-based schemes that work over standard elliptic curves such as Secp256k1 or Curve25519. (Pairing-based schemes [25] are thus precluded.) The natural candidate is BN. But the new application arena has lead to a desire for the following further features, not possessed by BN: (1) Key aggregation. There should be a way to aggregate a set of verification keys into a single, short aggregate key, relative to which signatures are verified. (2) Two rounds. A signing protocol using only 2 rounds of interaction, as opposed to the 3 used by BN.

MuSig [64, 25] broke ground by adapting BN to add key aggregation. Now the effort moved to reducing the number of rounds. This proved challenging. Early proposals of two-round schemes —[8, 62, 85] as well as an early, two-round version of MuSig [64]— were broken by DEFKLNS [40]. To fill the gap, DEFKLNS gave a new two-round scheme, mBCJ. Other proposals followed: MuSig2 [69], MuSig-DN [70] and DWMS [4]. All these support key aggregation.

All the schemes discussed here come with proofs of MS-UF security based on the hardness of the DL (Discrete Log) problem in the underlying group $\mathbb{G}$, up to variations in the model (standard or AGM [47]) or the type of DL problem (plain or OMDL [13]).

CURRENT BOUNDS. On being informed that a scheme has a proof of security based on the hardness of the DL problem in an underlying elliptic-curve group $\mathbb{G}$, the expectation of a practitioner is that the probability that a time t attacker can violate MS-UF security is no more than the probability of successfully computing a discrete logarithm in $\mathbb{G}$, which, as per [81], is t^2/p, where p, a prime, is the size of $\mathbb{G}$. Concretely, with the 256-bit curves Secp256k1 or Curve25519 —$p \approx 2^{256}$— they would expect that a time $t \approx 2^{80}$ attacker has ms-uf advantage at

Table 3.1: **Bounds on ms-uf advantage for the 3-round schemes BN and MuSig.** First we show prior bounds, then ours. In each case we first show the upper bound $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p)$ as a formula, where t,q,q_s are, respectively the adversary running time, the number of its RO queries and the number of executions of the signing protocol, while prime p is the size of the underlying group $\mathbb{G}$. We then show the evaluation with $t = q = 2^{80}$, $q_s = 2^{30}$ and $p \approx 2^{256}$, to capture security over 256-bit curves Secp256k1 or Curve25519. Our bounds assume generic hardness of IDL (for BN) and XIDL (for MuSig), which both hold in AGM. Our bounds are also tight and optimal, matching those for the DL problem itself.

Scheme MS	**Previous** $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p)$	**Previous** $p \approx 2^{256}$	**Ours** $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p)$	**Ours** $p \approx 2^{256}$
BN [14]	$\sqrt{(q \cdot t^2)/p}$	2^{-8}	t^2/p	2^{-96}
MuSig [25, 64]	$\sqrt[4]{(q^3 \cdot t^2)/p}$	1	t^2/p	2^{-96}

most $2^{160-256} = 2^{-96}$.

But this expectation is only correct if the reduction in the proof is tight. Current proofs for DL-based multi-signature schemes are loose. With the 256-bit curves Secp256k1 or Curve25519, and for a 2^{80}-time attacker, the proof of [14] for BN can preclude only a 2^{-8} ms-uf advantage, while the proof of [64, 25] for MuSig cannot even preclude a ms-uf advantage of 1, meaning there may be, per the proof, no security at all (cf. Figure 3.1). For 2-round schemes, the advantage precluded by current proofs is 2^{-16} in one case, and again just 1 for the others (cf. Figure 3.1). Overall, the proofs fail, by big margins, to support the parameter choices and expectations of practice.

Before continuing, let us expand on the above estimates. A proof of MS-UF security for a multi-signature scheme MS gives a formula $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p)$ that upper bounds the ms-uf advantage of an adversary as a function of its running time t, the number q of its queries to the random oracle, and the number q_s of executions of the signing protocol in the chosen-message attack in the ms-uf game. They are shown in Figures 3.1 and 3.1. We assume that $t \geq q \geq q_s$. To get these formulas, we first assume that the best attack against the DL problem is generic, so that a time t attacker has success probability at most t^2/p [81]. Next, we use the concrete-security

Figure 3.1: **Bounds on ms-uf advantage for 2-round schemes.** First we show bounds for prior schemes, then the bounds for our new scheme HBMS. As before, we first show the upper bound formula $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p)$, where t,q,q_s are, respectively the adversary running time, the number of its RO queries and the number of executions of the signing protocol, while prime p is the size of the underlying group $\mathbb{G}$. We then show the evaluation with $t = q = 2^{80}$, $q_s = 2^{30}$ and $p \approx 2^{256}$, to capture security over 256-bit curves Secp256k1 or Curve25519. For MuSig2, results differ depending on a parameter ν of the scheme. We also show estimates of signing time (per signer) and verification time. Here T^{me}_n is the time to compute one n-multi-exponentiation in $\mathbb{G}$. The "NIZK" for MuSig-DN indicates that signing requires computation and verification of a NIZKs, which is (much) more expensive then other operations shown.

Scheme	**Security**		**Efficiency**	
	$\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p)$	$p \approx 2^{256}$	Sign	Vf
mBCJ [40]	$(q_s^3 \cdot q^2 \cdot t^2)/p$	1	$T^{\text{me}}_2 + T^{\text{me}}_3$	$3T^{\text{me}}_2$
MuSig-DN [70]	$\sqrt[4]{(q^3 \cdot t^2)/p}$	1	NIZK	T^{me}_2
MuSig2, $\nu \geq 4$ [69]	$\sqrt[4]{(q^3 \cdot t^2)/p}$	1	T^{me}_ν	T^{me}_2
MuSig2, $\nu = 2$ [69]	$(t^2 + q^3)/p$	2^{-16}	T^{me}_2	T^{me}_2
DWMS [4]	$t^2/p + q/\sqrt{p}$	2^{-48}	$T^{\text{me}}_2 + T^{\text{me}}_{2N}$	T^{me}_2
HBMS	t^2/p	2^{-96}	T^{me}_2	T^{me}_3

results, in theorems in the papers, that give reductions from the DL problem to the MS-UF security of their scheme. The square-roots in the formulas arise from uses of forking lemmas [76, 14, 8]; the fourth-roots from nested use. The bounds in our Figures are approximate, dropping negligible additive terms. The proofs on which the bounds of Figures 3.1 and 3.1 are based, are, for BN [14], MuSig [25, 64], mBCJ [40], MuSig-DN [70] and MuSig2 ($\nu \geq 4$) [69], in the standard model; and for MuSig2 ($\nu = 2$) [69], DWMS [4] and HBMS, in the AGM. See Section 3.8 for details.

TOWARDS BETTER BOUNDS. Our thesis is that proofs should provide, not merely a qualitative guarantee, but one whose bounds quantitatively support parameter choices made in practice and the indications of cryptanalysis. Accordingly we want multi-signature schemes for which we can prove tight bounds on ms-uf advantage. How are we to reach this end? Impossibility results for Schnorr signatures [75, 58], on which the multi-signature schemes under consideration are based, indicate that a search for tight reductions that are both (1) in the standard model, and

(2) from DL, is unlikely to succeed. We need to be flexible, and relax either (1) or (2). In fact we show that relaxing either suffices: We give (1) tight reductions from DL in the Algebraic Group Model (AGM) [47], and (2) tight, standard-model reductions from assumptions other than DL. Together, these provide valuable theoretical support for the use of practical multi-signature schemes in 256-bit groups.

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

When considering the merits of the AGM, an important one to keep in mind is that a proof in the AGM immediately implies a proof in the well-accepted Generic Group Model (GGM) of [81]. (So the AGM is only "better" than the GGM.) In more detail, a tight AGM reduction from DL to some problem X immediately yields a GGM bound on adversary advantage, for X, that matches the GGM bound for DL [47]. Thus, overall, tight AGM reductions provide a valuable guarantee. This is recognized by Fuchsbauer, Plouviez and Seurin [48] who use the AGM to give a tight reduction from DL to the UF security of the Schnorr signature scheme. Their result gives hope, realized here, that such reductions are possible for multi-signatures as well.

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

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

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

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

2. When **1.** is not possible, we give BOTH of the following:

2.1 A tight AGM reduction, AND ALSO

2.2 A non-tight standard-model reduction.

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

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

Chain reductions can be seen as a way to implement a modular proof framework in the style of [58], in which steps are reused across proofs for different schemes.

NEW BOUNDS FOR CLASSICAL SCHEMES. We start by revisiting the classical 3-round schemes, namely BN and MuSig. Figure 3.2 illustrates our chains, that we now discuss.

IDL, formulated in [58] —they call it IDLOG, which we have abbreviated— is a purely group-based problem that is equivalent to the security against parallel impersonation under key-only attack (PIMP-KOA) of the Schnorr ID scheme. A tight GGM bound for IDL was shown by [58], but an AGM reduction $\mathrm{DL} \to \mathrm{IDL}$ does not seem to be in the literature; we fill this gap by providing it in Theorem 3.3.1. A (non-tight) standard model $\mathrm{DL} \to \mathrm{IDL}$ reduction is in [58], but we slightly improve it in Theorem 3.3.2.

Now our chain for BN is $\mathrm{DL} \to \mathrm{IDL} \to \mathsf{BN}$. This chain has length 2. Our main result for

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

	Reduction	SM	AGM
1	DL → IDL	Th. 3.3.2	**Th. 3.3.1**
2	IDL → BN	**Th. 3.5.1**	–
3	IDL → XIDL	Th. 3.3.4	**Th. 3.3.3**
4	XIDL → MuSig	**Th. 3.6.1**	–
5	XIDL → HBMS	Th. 3.7.2	**Th. 3.7.1**

Figure 3.2: **Chain reductions for multi-signatures.** SM stands for "Standard Model" and AGM for "Algebraic Group Model." An arrow P → Q means a reduction from P to Q; i.e. a proof that P implies Q. A boldface **Theorem Number** indicates the reduction is **tight**. A blank appears in the AGM column when a (tight) SM reduction to its left makes the AGM reduction unnecessary. Writing a MS scheme like BN, MuSig, HBMS as a point in a chain refers to MS-UF security of the scheme in question.

BN is Theorem 3.5.1, which shows IDL → BN with a *tight, standard model* reduction. Putting this together with our above-mentioned tight DL → IDL AGM-reduction of Theorem 3.3.1, we get a tight DL → BN AGM-reduction. Also our tight, standard-model IDL → BN reduction says that BN is as secure as the Schnorr identification scheme, which is valuable in its own right since the latter has withstood cryptanalysis for many years.

We introduce an intermediate, purely group-based problem we call XIDL. We show IDL → XIDL with a tight AGM reduction (Theorem 3.3.3) and a (non-tight) standard-model reduction (Theorem 3.3.4).

Our chain for MuSig is DL → IDL → XIDL → MuSig. This chain has length 3. Our main result for MuSig is Theorem 3.6.1, which shows XIDL → MuSig with a *tight, standard model* reduction. Putting this together with the rest of the chain, we get a tight DL → MuSig AGM-reduction. If we are willing to view XIDL as an assumption extending IDL, we can also view MuSig as based tightly on that.

This means we show that $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p) \leq t^2/p$ for both schemes, matching the DL bound. This is tight and optimal, since the multi-signature schemes can be broken by taking discrete-logs. Figure 3.1 compares our results with the prior ones.

NEW 2-ROUND SCHEME. Turning to 2-round schemes, we give a new scheme, called HBMS. HBMS supports key aggregation, in line with other 2-round schemes. Our chain for our new 2-round HBMS scheme is DL $\rightarrow$ IDL $\rightarrow$ XIDL $\rightarrow$ HBMS. This chain has length 3. We show XIDL $\rightarrow$ HBMS with a tight AGM reduction (Theorem 3.7.1) and a (non-tight) standard-model reduction (Theorem 3.7.2). Putting this together with the rest of the chain, we get a tight DL $\rightarrow$ HBMS AGM-reduction, in particular showing $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p) \leq t^2/p$, matching the DL bound. We also get a (non-tight) DL $\rightarrow$ HBMS standard-model-reduction.

Figure 3.1 compares HBMS with prior 2-round schemes. It shows that our improvement in security is not at the cost of efficiency. (Signing in HBMS is as efficient, or more so, than in prior schemes. For verification, MuSig-DN [70] is slightly faster, but signing in the latter is prohibitive due to the use of NIZKs.)

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

EQUIVALENCES. As discussed above, Theorem 3.5.1 shows IDL $\rightarrow$ BN with a tight, standard model reduction. We also give, in Theorem 3.5.2, a converse, namely a tight, standard-model reduction showing BN $\rightarrow$ IDL. This shows that IDL and BN are, security-wise, equivalent. Similarly, as discussed above, Theorem 3.6.1 shows MuSig $\rightarrow$ XIDL with a tight, standard model reduction, and we also give, in Theorem 3.6.2, a converse, namely a tight, standard-model reduction showing XIDL $\rightarrow$ MuSig. This shows that XIDL and MuSig are equivalent. Overall, this shows that IDL and XIDL are not arbitrary choices, but characterizations of the schemes whose consideration is necessary.

DEFINITIONAL CONTRIBUTIONS. DEFKLNS [40] found subtle gaps in some prior proofs of security for some two-round multi-signature schemes [8, 62, 85]. This indicates a need for greater care in the domain of multi-signatures. We suggest that this needs to begin with *definitions*. The ones in prior work, stemming mostly from [14], suffer from some lack of detail and precision. In particular, the very *syntax* of a multi-signature scheme is not specified in detail. This results in scheme descriptions that lack in precision, and proofs that stay at a high level in part due to lack of technical language in which to give details. This in turn can lead to bugs.

To address these issues, we revisit the definitions. We start by giving a detailed syntax that formalizes the signing protocol as a stateful algorithm, run separately by each player. Details addressed include that a player knows its position in the signer list, that player identities are separate from public keys, and integration of the ROM through a parameter describing the type of ideal hash functions needed. Then we give a security definition written via a code-based game. See Section 3.4.

RELATED WORK. The interest for blockchains and cryptocurrencies, and thus our focus, is DL-based schemes over elliptic curves. There are many other multi-signature schemes, based on other hard problems. Aggregate signatures [27, 12] yield multi-signatures, but these use pairings (bilinear maps). A pairing-based multi-signature scheme is also given in [25]. Lattice-based multi-signature schemes include [42, 36].

As noted above, IDL [58] captures the security against parallel impersonation under key-only attack (PIMP-KOA) of the Schnorr ID scheme and thus, given the ZK property of the scheme, also its security against parallel impersonation under passive attack (PIMP-PA). "Parallel" means multiple impersonation attempts are allowed. IMP-PA, traditional security against impersonation under passive attack, is the case where just one impersonation attempt is allowed. The Reset Lemma [15] gives a standard model DL $\rightarrow$ IMP-PA reduction. This uses rewinding and is non-tight, with a square-root loss. BD [10] introduce the Multi-Base Discrete Logarithm (MBDL) problem, give a tight standard-model MBDL $\rightarrow$ IMP-PA reduction, and show that, in the GGM, the security of MBDL is the same as that of DL. An interesting open question is whether MBDL can be used as a starting point for tight reductions for multi-signature schemes. Rotem and Segev [78] give a standard model DL $\rightarrow$ IMP-PA reduction that improves the square-root-loss reduction but is still not tight.

3.2 Preliminaries

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

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

<u>GAMES.</u> We use the code-based game playing framework of [19]. (See Fig. 3.3 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 $\emptyset$.

A procedure (oracle) with a certain name O may appear in several games. (For example, CH appears in two games in Figure 3.3.) To disambiguate, we may write Gm.O for the one in game Gm.

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

GROUPS. Throughout, $\mathbb{G}$ is a group whose order, assumed prime, we denote by p. We will use multiplicative notation for the group operation, and we let $1_{\mathbb{G}}$ denote the identity element of $\mathbb{G}$. We let $\mathbb{G}^* = \mathbb{G} \setminus \{1_{\mathbb{G}}\}$ denote the set of non-identity elements, which is the set of generators of $\mathbb{G}$ since the latter has prime order. If $g \in \mathbb{G}^*$ is a generator and $X \in \mathbb{G}$, then $\mathsf{DL}_{\mathbb{G},g}(X) \in \mathbb{Z}_p$ denotes the discrete logarithm of X in base g.

ALGEBRAIC ALGORITHMS. We recall the definition of algebraic algorithms [47]. As above, fix a group $\mathbb{G}$ of prime order p, and let g be a generator. In all of our security games involving $\mathbb{G}$ and g, we assume that any inputs and outputs of game oracles that are group elements (meaning, in $\mathbb{G}$) are distinguished. In particular, it will be clear from the game pseudocode definition which components of inputs and outputs are such group elements. We say that an adversary, against game Gm, is algebraic, if, whenever it submits a group element $Y \in \mathbb{G}$ as an oracle query, it also provides, alongside, a representation of Y in terms of group elements previously returned by the game oracles (the latter including INIT). Specifically, suppose during an execution of adversary $\mathcal{A}$ with game Gm, the adversary submits a group element $Y \in G$ to game oracle O. Then, alongside, it must provide a vector $(v_0, v_1, \ldots, v_m) \in \mathbb{Z}_p^m$, called a representation of Y, such that $Y = g^{v_0} \cdot h_1^{v_1} \cdots h_m^{v_m}$, where $h_1, \ldots, h_m$ are the group elements that have been returned to the adversary by game oracles of Gm, so far. When considering an execution of game Gm with an adversary $\mathcal{A}$ that is not algebraic, we omit the writing of representations in the oracle calls.

<u>HEDGING.</u> Not all attacks are algebraic. The thesis of [47] is that natural ones are, and thus proving security relative to algebraic adversaries gives meaningful guarantees in practice. We adopt this here but add hedging. Recall this means that, for the same scheme, we seek both (1) A tight AGM reduction from DL, and (2) a standard-model (even if non-tight) reduction from DL. The former is used to guide and support parameter choices. The latter is viewed as at least qualitatively ruling out non-algebraic attacks.

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

3.3 Hardness of problems in groups

Our chain reductions exploit three computational problems related to groups: standard discrete log (DL); IDL [58]; and a new problem XIDL that we introduce. Here we give the definitions. We then show the length-2 chain DL $\rightarrow$ IDL $\rightarrow$ XIDL. We give reductions that are tight in the AGM and also give (non-tight) standard-model reductions, a total of four results. Referring to Figure 3.2, we are establishing the four theorems, shown in the table, that correspond to arrows 1 and 3. For the rest of the section, we fix a group $\mathbb{G}$ of prime order p, and a generator $g \in \mathbb{G}$.

<u>DL.</u> We recall the standard discrete logarithm (DL) problem via game $\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}$ in Figure 3.3. INIT provides the adversary, as input, a random challenge group element X, and to win it must output $x' = \mathsf{DL}_{\mathbb{G},g}(X)$ to FIN. We let $\mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A})]$ be the discrete-log advantage of adversary $\mathcal{A}$.

<u>IDL.</u> The identification discrete logarithm (IDL) problem, introduced by KMP [58],

characterizes the hardness of parallel impersonation under key-only attack (PIMP-KOA) security [58] of the Schnorr identification scheme [79]. Formally, consider the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$ given in Fig. 3.3, where parameter q is a positive integer. The IDL-adversary receives a random target point X from INIT. It is additionally given access to a challenge oracle CH that can be called at most q times. The oracle takes as query a group element R (representing the commitment sent by the prover in Schnorr identification), stores it as R_i, and responds with a random challenge $c_i \leftarrow\$ \mathbb{Z}_p$ (representing the one sent by the verifier). The adversary wins if it can produce the discrete log z (representing the final prover response) of the group element $R_i \cdot X^{c_i}$, for a choice of i, denoted I, made by the adversary. We define the IDL-advantage of $\mathcal{A}$ to be $\mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}) = \Pr[\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A})]$.

KMP [58] study IDL in the Generic Group Model (GGM) [81] and prove a bound matching that for DL. Here, we strengthen this to give a tight AGM reduction DL $\rightarrow$ IDL. This could be seen as implicit in part of the AGM proof of security for the Schnorr signature scheme given in [48], although they make no connection to IDL.

Theorem 3.3.1 [DL $\rightarrow$ IDL, AGM] *Let $\mathbb{G}$ be a group of prime order p with generator g. Let q be a positive integer. Let $\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}}$ be an algebraic adversary against $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$. Then, adversary $\mathcal{A}_{\mathrm{dl}}$ can be constructed so that*

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

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

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

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

INIT:

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

FIN(x'):

Return $(x = x')$

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

INIT:

$x \leftarrow \mathbb{Z}_{|\mathbb{G}|}$; $X \leftarrow g^x$

Return X

CH(R): // At most q queries.

$i \leftarrow i+1$; $R_i \leftarrow R$

$c_i \leftarrow\!\!\$\, \mathbb{Z}_{|\mathbb{G}|}$; Return c_i

FIN(I, z):

Return $(g^z = R_I \cdot X^{c_I})$

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

INIT:

$x \leftarrow \mathbb{Z}_{|\mathbb{G}|}$; $X \leftarrow g^x$

Return X

NWTAR(S): // At most q_1 queries.

$j \leftarrow j+1$; $S_j \leftarrow S$

$e_j \leftarrow\!\!\$\, \mathbb{Z}_{|\mathbb{G}|}$; $T_j \leftarrow S_j \cdot X^{e_j}$

Return e_j

CH(j_{sel}, R): // At most q_2 queries.

$i \leftarrow i+1$; $R_i \leftarrow R$; $Y_i \leftarrow T_{j_{\mathrm{sel}}}$

$c_i \leftarrow\!\!\$\, \mathbb{Z}_{|\mathbb{G}|}$; Return c_i

FIN(I, z):

Return $(g^z = R_I \cdot Y_I^{c_I})$

Figure 3.3: 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 q, q_1, q_2 be positive integers. Top: Game defining discrete logarithm (DL) problem. Bottom left: Game defining identification logarithm (IDL) problem. Bottom right: Game defining random-target identification logarithm (XIDL) problem.

of q accounts for the adversary's having a choice of I made after receiving challenges. The full proof is given in Section 3.10.

By q-IDL, we refer to IDL with parameter q. 1-IDL corresponds to IMP-KOA security of the Schnorr identification scheme, and a reduction DL $\rightarrow$ 1-IDL is obtained via the Reset Lemma of [15]. KMP show that 1-IDL $\rightarrow$ q-IDL. Overall this gives a standard model (very non-tight) DL $\rightarrow$ q-IDL reduction. However, a somewhat tighter (but still non-tight) result can be obtained when the forking lemma of [14] (which we recall as Lemma 3.9.1) is applied directly instead. Concretely, we give the following theorem, improving the prior reduction by a $\sqrt{q}$ factor. The proof is in Section 3.11.

Theorem 3.3.2 [DL $\rightarrow$ IDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order $p = |\mathbb{G}|$, and let $g \in \mathbb{G}^*$ be a generator of $\mathbb{G}$. Let q be a positive integer. Let $\mathcal{A}_{\text{idl}}$ be an adversary against the game $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q}$. The proof constructs an adversary $\mathcal{A}_{\text{dl}}$ (explicitly given in Fig. 3.11) such that*

$$\mathbf{Adv}^{\text{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\text{idl}}) \leq \sqrt{q \cdot \mathbf{Adv}^{\text{dl}}_{\mathbb{G},g}(\mathcal{A}_{\text{idl}})} + \frac{q}{p} \,. \tag{3.1}$$

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

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

With $\mathbb{G}, p, g$ fixed as usual, XIDL is parameterized by positive integers q_1, q_2. Formally, consider the game $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_1,q_2}$ given in Fig. 3.3. The adversary receives a randomly chosen group element X from INIT. The game maintains a list $T_1, \ldots, T_{q_1}$ of "targets." The adversary can create a target by querying the New Target oracle NWTAR with a group element S of its choosing, whence $T_j = S \cdot X^{e_j}$ is added to the list of targets, for e_j chosen randomly from $\mathbb{Z}_p$ by the game and returned to the adversary. The adversary can query the challenge oracle $\text{CH}(j_{\text{sel}}, R)$ by supplying an index j_{sel} and a group element R. The oracle records $T_{j_{\text{sel}}}$ as Y_i, and R as R_i, based on the counter i it maintains. Intuitively, CH is similar to the challenge oracle CH in IDL game, besides that our adversary here needs to specify the target $T_{j_{\text{sel}}}$ it is trying to impersonate against. The adversary wins the game if it can produce the discrete log z of $R_I \cdot Y_I^{c_I}$, for an index I of its choice. The oracles NWTAR and CH are allowed to be called at most q_1 and q_2 times, respectively. We define the XIDL advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\text{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}) = \Pr[\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A})]$.

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

Theorem 3.3.3 [IDL $\to$ XIDL, AGM] *Let $\mathbb{G}$ be a group of order p with generator g. Let q_1, q_2 be positive integers. Let $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$ be an algebraic adversary against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}$. Then, adversary $\mathcal{A}_{\mathrm{dl}}$ can be constructed so that*

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

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

The full proof is given in Section 3.12. Here we sketch the intuition. Since $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$ is algebraic, the j-th query to NwTar is of the form $S_j, s_{j,1}, s_{j,2}$ such that $S_j = g^{s_{j,1}} X^{s_{j,2}}$, and the i-th query to Ch is of the form $j_{\mathrm{sel}}, R_i, r_{i,1}, r_{i,2}$ such that $R_i = g^{r_{i,1}} X^{r_{i,2}}$. Let e_j, c_i denote, respectively, the responses to the j-th query to NwTar and the i-th query to Ch. Eventually, $\mathcal{A}_{\mathrm{xidl}}$ outputs I, z. Assuming it succeeds, the equation $g^z = R_I \cdot T_J^{c_I} = R_I \cdot (S_J \cdot X^{e_J})^{c_I}$ must hold, where J was the selected index j_{sel} in the I-th query to Ch. This means that $g^z = g^{r_{I,1}} X^{r_{I,2}} (g^{s_{J,1}} X^{s_{J,2}} X^{e_J})^{c_I}$, whence $g^{z - r_{I,1} - s_{J,1} \cdot c_I} = X^w$, where $w = r_{I,2} + (s_{J,2} + e_J)c_I$. As long as w is non-zero modulo p, one can solve for the value of $\mathsf{DL}_{\mathbb{G},g}(X)$. But e_J and c_I were independently chosen after the adversary supplied $s_{J,2}$ and $r_{I,2}$, respectively. The probability that there exists j such that $(s_{j,2} + e_j) = 0$ mod p is at most q_1/p over q_1 queries to NwTar. Assuming there is no such j, the probability that $w = 0$ is at most q_2/p, due to the q_2 queries to Ch that $\mathcal{A}^{\mathrm{alg}}_{\mathrm{xidl}}$ can make.

In the standard model, techniques in the security proof of MuSig [25, 64] could be used to show DL $\to$ XIDL, which involves two applications of the Forking Lemma, leading to a fourth-root in the bound. Instead, we give a modular result showing IDL $\to$ XIDL, using a single application of the forking lemma. The same quantitive standard model bound (with fourth-root loss) can be obtained by composing Theorem 3.3.2 and Theorem 3.3.4.

Theorem 3.3.4 [IDL $\to$ XIDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order p with generator g. Let q_1, q_2 be positive integers. Let $\mathcal{A}_{\mathrm{xidl}}$ be an adversary against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}$. Then, an* *adversary $\mathcal{A}_{\text{idl}}$ can be constructed so that*

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

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

The full proof is given in Section 3.13. We now sketch the intuition. Adversary $\mathcal{A}_{\text{idl}}$ receives X from game $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}$ and runs adversary $\mathcal{A}_{\text{xidl}}$, forwarding it X as the target point. It answers queries to $\mathcal{A}_{\text{xidl}}$'s NwTar oracle using its own $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}.\text{Ch}$ oracle. Specifically, the j-th query S to NwTar is responded to with $e_j \leftarrow\$\, \text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}.\text{Ch}(S)$, and $\mathcal{A}_{\text{idl}}$ additionally records the group element $T_j \leftarrow S \cdot X^{e_j}$. It simulates adversary $\mathcal{A}_{\text{xidl}}$'s Ch oracle locally, meaning the i-th query $\text{Ch}(j_{\text{sel}}, R)$ is responded to with a fresh challenge $c_i \leftarrow\$\, \mathbb{Z}_p$. Eventually, adversary $\mathcal{A}_{\text{xidl}}$ gives a response I, z. Our $\mathcal{A}_{\text{idl}}$ adversary wins game $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}$ if it can produce the discrete log of T_j for any j of its choice. To do so, $\mathcal{A}_{\text{idl}}$ uses rewinding, the analysis of which uses the Forking Lemma [14] that we recall as Lemma 3.9.1. Rewinding is used to produce another response, (I', z'), from a forked execution of $\mathcal{A}_{\text{xidl}}$. The Forking Lemma applies to an execution of an algorithm making queries to one oracle, but adversary $\mathcal{A}_{\text{xidl}}$ has two oracles NwTar and Ch. We only "fork" $\mathcal{A}_{\text{xidl}}$ on its queries to Ch. Specifically, we program oracle NwTar to behave identically compared to the first run (meaning we use previously recorded values of $e_1, \ldots$ as long as they are defined). In the second run, oracle Ch is replied with $c_1, \ldots, c_{I-1}, c'_I, \ldots$, where $c'_I, \ldots$ are randomly chosen from $\mathbb{Z}_p$. Let us assume that $\mathcal{A}_{\text{idl}}$ has derived two valid responses from $\mathcal{A}_{\text{xidl}}$ using the Forking Lemma. Then it is guaranteed that $I = I'$ and $c_I \neq c'_I$. Moreover, we know the two executions of $\mathcal{A}_{\text{xidl}}$ only differ *after* the response of the I-th query to Ch, so the I-th query to Ch in both runs is some J, R_I. This allows our adversary to solve the equations $g^z = R_I \cdot T_J^{c_I}$ and $g^{z'} = R_I \cdot T_J^{c'_I}$ (which are guaranteed to be true if both runs succeed) to compute $\mathsf{DL}_{\mathbb{G},g}(T_J)$ and thus win the IDL game.

3.4 Definitions for multi-signatures

DEFKLNS [40] found subtle gaps in some prior proofs of security for some two-round multi-signature schemes [8, 62, 85]. Some of the latter schemes had been around for quite a long time before this happened. This suggests that, in the domain of multi-signatures, we need more care and careful analyses. We suggest that this needs to begin with *definitions*. The ones in prior work, stemming mostly from [14], suffer from some lack of detail and precision. In particular, the very *syntax* of a multi-signature scheme is not specified in detail. This results in scheme descriptions that lack somewhat in precision, and to proofs that stay at a high level in part due to lack of technical language in which to give details. This in turn can lead to bugs.

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

<u>SYNTAX.</u> A multi-signature scheme MS specifies algorithms MS.Kg, MS.Vf, MS.Sign, as well as a set MS.HF of functions, and an integer MS.nr, whose intent and operation is as follows. *Key generation.* Via $(pk, sk) \leftarrow\$ \mathsf{MS.Kg}$, the key generation algorithm generates public signature-verification key pk and secret signing key sk for a user. (Each user is expected to run this independently to get its keys.) *Hash functions.* MS.HF is a set of functions, from which, via $\mathsf{h} \leftarrow\$ \mathsf{MS.HF}$, one is drawn and provided to scheme algorithms (except key generation) and the adversary as the random oracle. Specifying this as part of the scheme allows the domain and range of the random oracle to be scheme-dependent. *Verification.* Via $d \leftarrow \mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma)$, the verification algorithm deterministically outputs a decision $d \in \{\mathsf{true}, \mathsf{false}\}$ indicating whether or not σ is a valid signature on message m under a vector $\boldsymbol{pk}$ of verification keys. *Signing.*

The signing protocol is specified by signing algorithm $\mathsf{MS.Sign}$. In each round, each party, applies this algorithm to its current state st and the vector **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 3.4.) *Rounds.* The interaction consists of a fixed number $\mathsf{MS.nr}$ of rounds. (We number the rounds $0, \ldots, \mathsf{MS.nr}$. The final broadcast of the signature is not counted as in practice it is a local output.) *Key Aggregation.* We say that a multi-signature scheme MS supports key aggregation if MS has additional two algorithms $\mathsf{MS.Ag}$ and $\mathsf{MS.VfAg}$ such that: (1) Via $apk \leftarrow\!\$\, \mathsf{MS.Ag}^{\mathrm{H}}(pk_1, \ldots, pk_n)$, $\mathsf{MS.Ag}$ generates an aggregate public key, (2) Via $d \leftarrow \mathsf{MS.VfAg}^{\mathrm{H}}(apk, m, \sigma)$, the aggregate verification algorithm deterministically outputs a decision $d \in \{\mathsf{true}, \mathsf{false}\}$, and (3) the verification algorithm $\mathsf{MS.Vf}$ is defined exactly as $\mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma) := \mathsf{MS.VfAg}^{\mathrm{H}}(\mathsf{MS.Ag}^{\mathrm{H}}(\boldsymbol{pk}), m, \sigma)$.

Some conventions will aid further definitions and scheme descriptions. A party's state st has several parts: $\mathsf{st.n}$ is the number of parties in the current execution of the protocol; $\mathsf{st.me} \in [1..\mathsf{st.n}]$ is the party's own identity; $\mathsf{st.rnd} \in [0..\mathsf{MS.nr}]$ is the current round number; $\mathsf{st.sk}$ is the party's own signing key; $\mathsf{st.pk}$ is the $\mathsf{st.n}$-vector of all verification keys; $\mathsf{st.msg}$ is the message being signed; $\mathsf{st.rej} \in \{\mathsf{true}, \mathsf{false}\}$ is the decision to reject (not produce a signature) or accept. It is assumed and required that each invocation of $\mathsf{MS.Sign}$ leaves all of these unchanged except for $\mathsf{st.rnd}$, which it increments by 1, and $\mathsf{st.rej}$, which is assumed initialized to false and may at some point be set to true. The state can, beyond these, have other components that vary from protocol to protocol. (For example, Figure 3.5 describing the BN scheme has $\mathsf{st}.\boldsymbol{R}[j], \mathsf{st}.\boldsymbol{t}[j], \mathsf{st}.\boldsymbol{z}[j], \mathsf{st}.R, \ldots$.) We write $\mathsf{st} \leftarrow \mathsf{StInit}(j, sk, \boldsymbol{pk}, m)$ to initialize st by setting $\mathsf{st.n} \leftarrow |\boldsymbol{pk}|$; $\mathsf{st.me} \leftarrow j$; $\mathsf{st.rnd} \leftarrow 0$; $\mathsf{st.sk} \leftarrow sk$; $\mathsf{st.pk} \leftarrow \boldsymbol{pk}$; $\mathsf{st.msg} \leftarrow m$; $\mathsf{st.rej} \leftarrow \mathsf{false}$. If an execution $(\sigma, \mathsf{st}') \leftarrow \mathsf{MS.Sign}^{\mathrm{H}}(\mathbf{in}, \mathsf{st})$ returns $\sigma = \perp$ then it is assumed and required that further executions starting from st' all return $\perp$ as the output message.

<u>CORRECTNESS.</u> Algorithm $\mathsf{Exec}_{\mathsf{MS}}$, shown in the left column of Fig. 3.4, executes the

Algorithm $\mathsf{Exec}^{\mathsf{h}}_{\mathsf{MS}}(\boldsymbol{sk}, \boldsymbol{pk}, m)$:

1 $n \leftarrow |\boldsymbol{pk}|$
2 For $j = 1, \ldots, n$ do
3 $\quad \mathsf{st}_j \leftarrow \mathsf{StInit}(j, \boldsymbol{sk}[j], \boldsymbol{pk}, m)$
4 $\boldsymbol{b} \leftarrow (\varepsilon, \ldots, \varepsilon)$ // n-vector
5 For $i = 1, \ldots, \mathsf{MS.nr}$ do
6 $\quad$ For $j = 1, \ldots, n$ do
7 $\quad\quad (\sigma_j, \mathsf{st}_j) \leftarrow\!\!\$\ \mathsf{MS.Sign}^{\mathsf{h}}(\boldsymbol{b}, \mathsf{st}_j)$
8 $\quad \boldsymbol{b} \leftarrow (\sigma_1, \ldots, \sigma_n)$
9 Return σ_1

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

FIN:

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

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

INIT:

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

NS$(k, \boldsymbol{pk}, m)$:

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

SIGN$_j(s, \boldsymbol{b})$: // $1 \leq j \leq \mathsf{MS.nr}$

$(\sigma, \mathsf{st}_s) \leftarrow\!\!\$\ \mathsf{MS.Sign}^{\mathrm{H}}(\boldsymbol{b}, \mathsf{st}_s)$; Return σ

H(x):

Return $\mathsf{h}(x)$

FIN$(k, \boldsymbol{pk}, m, \sigma)$:

If $(\boldsymbol{pk}[k] \neq pk)$ then Return false
If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then Return false
Return $\mathsf{MS.Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma)$

Figure 3.4: **Top left:** Procedure specifying an honest execution of the signing protocol associated with multi-signature scheme MS. **Top right:** Correctness game. **Bottom:** Unforgeability game.

signing protocol of MS on input a vector $\boldsymbol{sk}$ of signing keys, a vector $\boldsymbol{pk}$ of matching verification keys with $|\boldsymbol{sk}| = |\boldsymbol{pk}|$, and a message m to be signed, and with access to random oracle $\mathsf{h} \in \mathsf{MS.HF}$. The number of parties n at line 1 is the number of coordinates (length) of $\boldsymbol{pk}$. The state 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. 3.4 has only one procedure, namely FIN. We say that MS satisfies (perfect) correctness if for all positive integers n we have $\Pr[\mathbf{G}^{\text{ms-cor}}_{\mathsf{MS},n}] = 1$.

<u>UNFORGEABILITY.</u> Game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ in Fig. 3.4 captures a notion of unforgeability for multi-signatures that slightly extends [14]. There is one honest player whose keys are picked at line 1, the adversary controlling all the other players. A new instance of the signing protocol is initialized by calling NS with an index k and a vector $\boldsymbol{pk}$ of verification keys that the adversary can choose, possibly dishonestly, subject only to $\boldsymbol{pk}[k]$ being the verification key pk of the honest player, as enforced by line 2. The first message of the honest player is sent out, and at this point $\mathsf{st}_u.\mathsf{rnd} = 1$. Now the adversary can run multiple concurrent instances of the signing protocol with the honest signer. Oracle H is the random oracle, simply calling h. Eventually the adversary calls FIN with a forgery index k, a vector of verification keys (subjected to $\boldsymbol{pk}[k]$ being the public key of the honest signer), a message and a claimed signature. It wins if verification succeeds and the forgery was non-trivial. The ms-uf-advantage of adversary $\mathcal{A}$ is $\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A})]$.

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

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

3.5 Analysis of the BN scheme

BN SCHEME. Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. The associated BN [14] multi-signature scheme $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ is shown in detail, in our syntax, in Fig. 3.5. The set $\mathsf{MS.HF}$ consists of all functions h such that $\mathsf{h}(0,\cdot)\colon\{0,1\}^* \to \{0,1\}^\ell$ and $\mathsf{h}(1,\cdot)\colon\{0,1\}^* \to \mathbb{Z}_p$. 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 20, it returns $\perp$ if one of the received values fails to match its commitment. As per our conventions, when this happens, this player will always broadcast $\perp$ in the future, so for round 3 we assume lines 21 and 22 are executed. These lines create the second component $\mathsf{st}.\boldsymbol{z}[j]$ of a Schnorr signature relative to the Schnorr-commitment $\mathsf{st}.\boldsymbol{R}[j]$ defined at line 13, and the player's own secret key, the computations being modulo p. This $\mathsf{st}.\boldsymbol{z}[j]$ is broadcast, so that, in round 3, our player receives the corresponding values from the other players. At line 27 it forms their modulo-p sum z and then forms the final signature $(\mathsf{st}.R, z)$.

Our description of the signing protocol differs, from that in [14], in some details that are brought out by our syntax, for example in using explicit party identities rather than seeing these as implicit in public keys.

PRIOR BOUNDS. We recall the prior result of [14]. Let $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ and 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 distinct queries across $\mathrm{H}_0, \mathrm{H}_1$ and at most q_{s} queries to NS. Suppose the number of parties

$\underline{\mathsf{Kg}:}$

$sk \leftarrow\!\!\$\, \mathbb{Z}_p$; $pk \leftarrow g^{sk}$

Return (pk, sk)

$\underline{\mathsf{Vf}^{\mathrm{H}}(\boldsymbol{pk}, m, \sigma):}$

$(R, z) \leftarrow \sigma$; $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$

$\underline{\mathsf{BN}:}$

  For $i = 1, \ldots, n$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$

  Return $(\, g^z = R \cdot \prod_{i=1}^{n} pk_i^{c_i} \,)$

$\underline{\mathsf{MuSig}:}$

  $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$

  $c \leftarrow \mathrm{H}_1((R, apk, m))$

  Return $(\, g^z = R \cdot apk^c \,)$

$\underline{\mathsf{Sign}^{\mathrm{H}}(\boldsymbol{b}, \mathsf{st}):}$

$j \leftarrow \mathsf{st.me}$; $n \leftarrow \mathsf{st.n}$; $m \leftarrow \mathsf{st.msg}$; $sk \leftarrow \mathsf{st.sk}$; $\boldsymbol{pk} \leftarrow \mathsf{st.pk}$

If $(\mathsf{st.rnd} = 0)$ then

  $\mathsf{st.}r \leftarrow\!\!\$\, \mathbb{Z}_p$; $\mathsf{st.}\boldsymbol{R}[j] \leftarrow g^r$; $\mathsf{st.}\boldsymbol{t}[j] \leftarrow \mathrm{H}_0((j, \mathsf{st.}\boldsymbol{R}[j]))$; $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$

  Return $(\mathsf{st.}\boldsymbol{t}[j], \mathsf{st})$

If $(\mathsf{st.rnd} = 1)$ then

  For all $i \neq j$ do $\mathsf{st.}\boldsymbol{t}[i] \leftarrow \boldsymbol{b}[i]$

  $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$; Return $(\mathsf{st.}\boldsymbol{R}[j], \mathsf{st})$

If $(\mathsf{st.rnd} = 2)$ then

  For all $i \neq j$ do $\mathsf{st.}\boldsymbol{R}[i] \leftarrow \boldsymbol{b}[i]$

  If $(\, \exists i : \mathrm{H}_0((i, \mathsf{st.}R[i])) \neq \mathsf{st.}\boldsymbol{t}[i] \,)$ then Return $(\bot, \mathsf{st})$

  $\mathsf{st.}R \leftarrow \prod_{i=1}^{n} \mathsf{st.}R[i]$

  $\underline{\mathsf{BN}:}$ $c_j \leftarrow \mathrm{H}_1((j, R, \boldsymbol{pk}, m))$; $\mathsf{st.}\boldsymbol{z}[j] \leftarrow sk \cdot c_j + \mathsf{st.}r$

  $\underline{\mathsf{MuSig}:}$

    $apk \leftarrow \prod_{i=1}^{n} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$; $c \leftarrow \mathrm{H}_1((R, apk, m))$

    $\mathsf{st.}\boldsymbol{z}[j] \leftarrow sk \cdot \mathrm{H}_2((\mathsf{st.me}, \boldsymbol{pk})) \cdot c + \mathsf{st.}r$

  $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$; Return $(\mathsf{st.}\boldsymbol{z}[j], \mathsf{st})$

If $(\mathsf{st.rnd} = 3)$ then

  For all $i \neq j$ do $\mathsf{st.}\boldsymbol{z}[i] \leftarrow \boldsymbol{b}[i]$

  $z \leftarrow \sum_{i=1}^{n} \mathsf{st.}\boldsymbol{z}[i]$; Return $((\mathsf{st.}R, z), \mathsf{st})$

Figure 3.5: Algorithms of the multi-signature scheme $\mathsf{BN}[\mathbb{G}, g, \ell]$ and $\mathsf{MuSig}[\mathbb{G}, g, \ell]$, where $\mathbb{G}$ is a group of prime order p with generator g. Code that differs between the two schemes is marked explicitly. Oracle $\mathrm{H}_i(\cdot)$ is defined to be $\mathrm{H}(i, \cdot)$ for $i = 0, 1$ (BN) and $i = 0, 1, 2$ (MuSig).

(length of verification-key vector) in queries to NS and FIN is at most n. Let $a = 8q_{\mathrm{s}} + 1$ and

$b = 2q + 16n^2 q_s$. Let $p = |\mathbb{G}|$. Then BN [14] give a DL-adversary $\mathcal{A}_{dl}$ such that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{ms}) \leq \sqrt{(q + q_s) \cdot \left(\mathbf{Adv}^{dl}_{\mathbb{G},g}(\mathcal{A}_{dl}) + \frac{a}{p} + \frac{b}{2^\ell}\right)}\,. \tag{3.2}$$

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

<u>SECURITY OF BN FROM IDL.</u> We give a IDL $\to$ BN reduction that is *tight* and in the *standard model.* Combining this with our tight AGM reduction DL $\to$ IDL of Theorem 3.3.1 we conclude a tight AGM reduction DL $\to$ BN. However, the standard model tight IDL $\to$ BN reduction is also interesting in its own right. It says that BN is just as secure as the Schnorr identification scheme. Since the latter has been around and resisted cryptanalysis for quite some time, this is good support for the security of BN.

Theorem 3.5.1 [IDL $\to$ BN, Standard Model] *Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ be the associated BN multi-signature scheme. Let $\mathcal{A}_{ms}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 3.4. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{ms}$ has at most q_0, q_1, q_s distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{NS}$, respectively, and the number of parties (length of verification-key vector) in queries to NS and FIN is at most n. Let $\alpha = q_s(4q_0 + 2q_1 + q_s)$ and $\beta = q_0(q_0 + n)$. Then we construct an adversary $\mathcal{A}_{id}$ for game $\mathrm{Gm}^{idl}_{\mathbb{G},g,q_1}$ (shown explicitly in Figure 3.17) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{ms}) \leq \mathbf{Adv}^{idl}_{\mathbb{G},g,q_1}(\mathcal{A}_{idl}) + \frac{\alpha}{2p} + \frac{\beta}{2^\ell}\,. \tag{3.3}$$

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

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. A lower bound on q_1 is the length of $\boldsymbol{pk}$ in $\mathcal{A}_{\text{ms}}$'s FIN query, so we can assume it is positive. With the above theorem, we can now derive an upperbound $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_{\text{s}},p)$ of the advantage of any MS adversary with running time t, making q queries to H, and q_{s} signing interactions. We take $\ell \approx \log_2(p)$ and assume that $q_{\text{s}} \leq q \leq t \leq p$. Additionally, we assume that the advantage of any IDL adversary with running time t is at most t^2/p (as justified by Theorem 3.3.2). We obtain $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t,q,q_s,p) \leq t^2/p$ as shown in Fig. 3.1.

The full proof of Theorem 3.5.1 is given in Section 3.14. Here we give a sketch. The reduction adversary $\mathcal{A}_{\text{idl}}$ receives a group element X from $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}$ and forwards it to adversary $\mathcal{A}_{\text{ms}}$ as the target public key. In order to run adversary $\mathcal{A}_{\text{ms}}$, our adversary needs to be able to simulate the signing oracles $\text{NS}, \text{SIGN}_1, \text{SIGN}_2$ as well as random oracles H_0 and H_1 without knowing $\mathsf{DL}_{\mathbb{G},g}(X)$. We first describe how the reduction proceeds if $\mathcal{A}_{\text{ms}}$ makes no queries to NS, SIGN_1 or SIGN_2, as this steps constitutes the main difference between our proof and the original proof of security for BN [14]. Adversary $\mathcal{A}_{\text{idl}}$ uses the challenge oracle $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}.\text{CH}$ to program the random oracle H_1 (hence CH needs to be able to be queried upto the number of times H_1 is evaluated). In particular, for each query $\text{H}_1((k,R,\boldsymbol{pk},m))$ where $\boldsymbol{pk}[k] = X$, our adversary first computes $T \leftarrow R \cdot \prod_{j \neq k} \boldsymbol{pk}[j]^{\text{H}_1((j,R,\boldsymbol{pk},m))}$, then obtains $c \leftarrow\!\!\$\ \text{CH}(T)$ before returning c as the return value for the query $\text{H}_1((k,R,\boldsymbol{pk},m))$. By construction, a valid forgery for $\boldsymbol{pk}, m$ is some signature $\sigma = (R,z)$ such that

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

where the first equality is by the verification equation of BN and the second equality is by the way H_1 is programmed. This means that adversary $\mathcal{A}_{\text{idl}}$ can simply forward the value of z from a valid forgery, along with the index of the CH query corresponding to the H_1 query of the forgery, to break game $\text{Gm}^{\text{idl}}_{\mathbb{G},g,q_1}$. Moreover, adversary $\mathcal{A}_{\text{idl}}$ succeeds as long as the forgery given by $\mathcal{A}_{\text{ms}}$

is valid.

It remains to show that oracles $\mathrm{NS}, \mathrm{SIGN}_1, \mathrm{SIGN}_2$ can be simulated without knowledge of the secret key, $\mathsf{DL}_{\mathbb{G},g}(X)$. Roughly, this is done using the zero-knowledge property of the underlying Schnorr identification scheme as well as by programming the random oracles H_0 and H_1. The original proof by [14] constructs an adversary and argues that it simulates these oracles faithfully if certain bad events do not happen. We take a more careful approach and do this formally via a sequence of seven games and use the code-base game playing framework of [19]. This game sequence incurs the additive loss as indicated in Equation (3.3).

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

Theorem 3.5.2 [$\mathsf{BN} \to$ IDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{BN}[\mathbb{G}, g, \ell]$ be the associated BN multi-signature scheme. Let q be a positive integer and $\mathcal{A}_{\text{idl}}$ be an adversary against $\mathrm{Gm}^{\text{idl}}_{\mathbb{G},g,q}$. Then, we can construct an adversary $\mathcal{A}_{\text{ms}}$ for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$, making no queries to NS, and at most $2q$ queries to H_1, such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\text{ms}}) \geq \mathbf{Adv}^{\text{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\text{idl}}) \,. \tag{3.4}$$

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

Proof of Theorem 3.5.2: Consider the adversary given in Fig. 3.6. The adversary receives the target public key pk from the MS-UF game and samples a key pair $(pk', sk') \leftarrow\$\, \mathsf{MS.Kg}$. The adversary will attempt to forge a signature against the vector of public keys (pk, pk'). Adversary $\mathcal{A}_{\text{ms}}$ forwards $X = pk$ as the target point and runs IDL adversary $\mathcal{A}_{\text{idl}}$. For each query $\mathrm{CH}(R)$ of

$\underline{\mathcal{A}_{\mathrm{ms}}^{\mathrm{H}_1}(pk)}$:

1 $X \leftarrow pk$; $(pk', sk') \leftarrow\!\!\$\ \mathsf{MS.Kg}()$

2 $(I, z) \leftarrow \mathcal{A}_{\mathrm{xidl}}^{\mathrm{CH}}(pk)$ // $g^z = R_I \cdot pk^{c_{I,1}}$

3 $\sigma \leftarrow (R_I, z + sk' \cdot c_{I,2} \mod p)$; Return $((pk, pk'), m_I, \sigma)$

$\underline{\mathrm{CH}(R)}$:

4 $i \leftarrow i+1$; $R_i \leftarrow R$; $m_i \leftarrow \langle i \rangle$

5 $c_{i,1} \leftarrow\!\!\$\ \mathrm{H}_1((1, R_i, (pk, pk'), m_i))$; $c_{i,2} \leftarrow\!\!\$\ \mathrm{H}_1((2, R_i, (pk, pk'), m_i))$

6 Return $c_{i,1}$

Figure 3.6: Adversary $\mathcal{A}_{\mathrm{ms}}$ for Theorem 3.6.1. For an integer i, $\langle i \rangle$ denote the binary representation of i.

$\mathcal{A}_{\mathrm{idl}}$, adversary $\mathcal{A}_{\mathrm{ms}}$ simulates the response as per line 4 to 6. If $\mathcal{A}_{\mathrm{idl}}$ succeeds, it must be that

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

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

3.6 Analysis of the MuSig scheme

The current three-round version of MuSig has been proposed and analyzed by both [64] and [25]. Roughly, it is the BN scheme with added key aggregation.

Let $\mathbb{G}$ be a group of prime order p. And let g be a generator of g and $\ell \geq 1$ be an integer. The formal specification of $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ in our syntax is shown in Fig. 3.5. There are minimal differences between MuSig and BN and we only highlight the differences. The set $\mathsf{MS.HF}$ consists of all functions h such that $h(0, \cdot) : \{0,1\}^* \to \{0,1\}^\ell$ and $h(i, \cdot) : \{0,1\}^* \to \mathbb{Z}_p$ for $i = 1, 2$. Verification is done as follows. First, an aggregate key apk for the list of keys $\boldsymbol{pk} = (pk_1, \ldots, pk_n)$ is computed as $apk \leftarrow pk_1^{\mathrm{H}_2((1, \boldsymbol{pk}))} \cdots pk_n^{\mathrm{H}_2((n, \boldsymbol{pk}))}$ (line 8). Next, a single challenge is derived from the commitment R and aggregate key apk (line 9). The signature (R, z) is valid if $g^z = R \cdot apk^c$. The second round of signing also changes accordingly to generate a valid signature (line 24 and 25).

The following gives a tight, standard-model reduction XIDL $\rightarrow$ MuSig. Combining this with our tight AGM chain DL $\rightarrow$ IDL $\rightarrow$ XIDL from Theorems 3.3.1 and 3.3.3, we get a tight AGM reduction DL $\rightarrow$ MuSig.

Theorem 3.6.1 [XIDL $\rightarrow$ MuSig, Standard Model] *Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$ and $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ be the associated MuSig multi-signature scheme. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 3.4. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_0, q_1, q_2, q_{\mathrm{s}}$ distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$, respectively, and the number of parties (length of verification-key vector) in queries to NS and $\textsc{Fin}$ is at most n. Let $\alpha = q_{\mathrm{s}}(4q_0 + 2q_1 + q_{\mathrm{s}}) + 2q_1q_2$ and $\beta = q_0(q_0 + n)$. Then we construct an adversary $\mathcal{A}_{\mathrm{xidl}}$ for game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$ (shown explicitly in Figure 3.23) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \leq \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}}) + \frac{\alpha}{2p} + \frac{\beta}{2^{\ell}} \,. \tag{3.5}$$

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

We remark that the values of q_1 and q_2 above arise from the number of queries to H_1 and H_2 made in the execution of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}})$. As a result, the appearance of q_1 and q_2 has their orders "switched" compared to in Section 3.3. With the above theorem, we can now derive an upperbound $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_{\mathrm{s}}, p)$ of the advantage of any MS adversary with running time t, making q queries to H, and q_{s} signing interactions. We take $\ell \approx \log_2(p)$ and assume that $q_{\mathrm{s}} \leq q \leq t \leq p$. Additionally, we assume that the advantage of any XIDL adversary with running time t is at most t^2/p (as justified by Theorem 3.3.4). We obtain $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_s, p) \leq t^2/p$ as shown in Fig. 3.1.

We again describe the reduction at a high level and defer the full proof to Section 3.15. First, the reduction adversary $\mathcal{A}_{\mathrm{xidl}}$ receives group element X from game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G}g,q_2,q_1}$ and runs $\mathcal{A}_{\mathrm{ms}}$ with the target public key set to X. Similar to the proof of Theorem 3.5.1, our adversary needs to simulate the signing oracles $\mathrm{NS}, \mathsf{Sign}_1, \mathsf{Sign}_2$ as well as $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ without knowing $\mathsf{DL}_{\mathbb{G},g}(X)$ in order to run $\mathcal{A}_{\mathrm{ms}}$. This again relies on the zero-knowledge property of the underlying Schnorr identification scheme and the programming of $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$. This step is done formally in a game sequence in the full proof and incurs the additive loss in Equation (3.5). To turn a forgery into a break against XIDL, our adversary programs H_1 and H_2 as follows. For the j-th query of $\mathrm{H}_2((k, \boldsymbol{pk}))$ where $\boldsymbol{pk}[k] = X$, the adversary first computes $S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$, then obtains $e_j \leftarrow\$ \, \mathrm{NwTar}(S)$ before returning e_j as the response for the query. We remark that this particular query of H_2 have created an aggregate public key $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))} = S \cdot X^{e_j}$, which is also the value of T_j that is recorded in the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. For each i-th query of $\mathrm{H}_1((R, apk, m))$, the adversary first finds the index j_{sel} of the H_2-query that corresponds to the input apk, then obtains $c_i \leftarrow\$ \, \mathrm{Ch}(j_{\mathrm{sel}}, R)$ before returning c_i as the response for the query. If the eventual forgery is given for these two particular queries to H_1 and H_2, meaning forgery is $\boldsymbol{pk}, m, (R, z)$ for some z, then the verification equation of the signature scheme says that $g^z = R \cdot apk^{\mathrm{H}_1((R,apk,m))}$. But this matches exactly the winning condition of $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$, since $apk = T_{j_{\mathrm{sel}}}$ and $c_i = \mathrm{H}_1((R, apk, m))$. Hence, our adversary $\mathcal{A}_{\mathrm{xidl}}$ can simply return (i, z) to break XIDL, as long as the forgery provided by $\mathcal{A}_{\mathrm{ms}}$ is valid.

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

Theorem 3.6.2 [$\mathsf{MuSig} \rightarrow$ XIDL, Standard Model] *Let $\mathbb{G}$ be a group of prime order p. Let g be a generator of $\mathbb{G}$ and let $\ell \geq 1$ be an integer. Let $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ be the associated MuSig multi-signature scheme. Let q_1, q_2 be a positive integers and $\mathcal{A}_{\mathrm{xidl}}$ be an adversary against* $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. *Then, we can construct an adversary $\mathcal{A}_{\mathrm{ms}}$ for game $\mathbf{G}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}$, making no queries to*

$\underline{\mathcal{A}_{\mathrm{ms}}^{\mathrm{H}_1,\mathrm{H}_2}(pk)\text{:}}$

1 $X \leftarrow pk$; $(I,z) \leftarrow \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{Ch}}(pk)$; $J \leftarrow \mathrm{TI}[I]$

2 $\sigma \leftarrow (R_I, z)$; Return $((pk, S_J), m_I, \sigma)$

$\underline{\mathrm{NwTar}(S)\text{:}}$

3 $j \leftarrow j+1$; $S_j \leftarrow S$

4 $e_{j,1} \leftarrow\!\!\$\; \mathrm{H}_2((1,(pk,S)))$; $e_{j,2} \leftarrow\!\!\$\; \mathrm{H}_2((2,(pk,S)))$; $e_j \leftarrow e_{j,2}/e_{j,1} \mod p$

5 $apk_j \leftarrow pk^{e_{j,1}} S^{e_{j,2}}$; $T_j \leftarrow pk \cdot S^{e_j}$; Return e_j

$\underline{\mathrm{Ch}(j_{\mathrm{sel}}, R)\text{:}}$

6 $i \leftarrow i+1$; $R_i \leftarrow R$; $m_i \leftarrow \langle i \rangle$; $\mathrm{TI}[i] \leftarrow j_{\mathrm{sel}}$

7 $c_i \leftarrow \mathrm{H}_1((apk_{j_{\mathrm{sel}}}, R, m_i)) \cdot e_{j_{\mathrm{sel}},1}$; Return c_i

Figure 3.7: Adversary $\mathcal{A}_{\mathrm{ms}}$ for Theorem 3.6.1. For an integer i, $\langle i \rangle$ denote the binary representation of i.

NS, *and at most* $2q_1$ *and* $2q_2$ *queries to* H_1 *and* H_2 *respectively, such that*

$$\mathbf{Adv}_{\mathsf{MS}}^{\text{ms-uf}}(\mathcal{A}_{\mathrm{ms}}) \geq \mathbf{Adv}_{\mathbb{G},g,q_2,q_1}^{\mathrm{xidl}}(\mathcal{A}_{\mathrm{xidl}}) \; . \tag{3.6}$$

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

Proof of Theorem 3.6.2: Consider the adversary given in Fig. 3.7. The adversary receives the target publick key pk from the MS-UF game. Adversary $\mathcal{A}_{\mathrm{ms}}$ forwards $X = pk$ as the target point and runs XIDL adversary $\mathcal{A}_{\mathrm{idl}}$. For each query $\mathrm{NwTar}(S)$ of $\mathcal{A}_{\mathrm{xidl}}$, adversary $\mathcal{A}_{\mathrm{ms}}$ uses S as a public key to generate the aggregate key apk for the list (pk, S). By construction, the j-th target T_j for the XIDL game is related to apk_j by $apk_j = T_j^{e_{j,1}}$. For each $\mathrm{Ch}(j_{\mathrm{sel}}, R)$ query of $\mathcal{A}_{\mathrm{xidl}}$, adversary $\mathcal{A}_{\mathrm{ms}}$ programs in the H_1 outputs corresponding to a forgery agaisnt the aggregate key $apk_{j_{\mathrm{sel}}}$ (line 6 and 7). By construction, if $\mathcal{A}_{\mathrm{xidl}}$ succeeds, it must be that

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

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

$\underline{\mathsf{MS.Kg}:}$

$sk \leftarrow\!\!\$\; \mathbb{Z}_p \; ; \; pk \leftarrow g^{sk}$

Return (pk, sk)

$\underline{\mathsf{MS.Vf}^{\mathrm{H}_0,\mathrm{H}_1,\mathrm{H}_2}(\boldsymbol{pk}, m, \sigma)}:$

$(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk} \; ; \; apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$

$(T, s, z) \leftarrow \sigma \; ; \; c \leftarrow \mathrm{H}_1((T, apk, m))$

$h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$; Return $(g^z h^s = T \cdot apk^c)$

$\underline{\mathsf{MS.Sign}^{\mathrm{H}_0,\mathrm{H}_1,\mathrm{H}_2}(\boldsymbol{b}, \mathsf{st})}:$

$j \leftarrow \mathsf{st.me} \; ; \; n \leftarrow \mathsf{st.n} \; ; \; m \leftarrow \mathsf{st.msg} \; ; \; sk \leftarrow \mathsf{st.sk} \; ; \; \boldsymbol{pk} \leftarrow \mathsf{st.pk}$

$(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk} \; ; \; apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$

If $(\mathsf{st.rnd} = 0)$ then

    $\mathsf{st}.r[j] \leftarrow\!\!\$\; \mathbb{Z}_p \; ; \; \mathsf{st}.s[j] \leftarrow\!\!\$\; \mathbb{Z}_p$

    $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m)) \; ; \; \mathsf{st}.\boldsymbol{R}[j] \leftarrow g^{\mathsf{st}.r[j]} \; ; \; \mathsf{st}.\boldsymbol{T}[j] \leftarrow \mathsf{st}.\boldsymbol{R}[j] \cdot h^{\mathsf{st}.s[j]}$

    $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$; Return $(\mathsf{st}.\boldsymbol{T}[j], \mathsf{st})$

If $(\mathsf{st.rnd} = 1)$ then

    For all $i \neq j$ do $\mathsf{st}.\boldsymbol{T}[i] \leftarrow \boldsymbol{b}[i]$

    $\mathsf{st}.T \leftarrow \prod_{i=1}^n \mathsf{st}.\boldsymbol{T}[i] \; ; \; \mathsf{st}.c \leftarrow \mathrm{H}_1((\mathsf{st}.T, apk, m)) \; ; \; e_j \leftarrow \mathrm{H}_2((j, \boldsymbol{pk}))$

    $\mathsf{st}.z[j] \leftarrow sk \cdot c \cdot e_j + \mathsf{st}.r[j] \; ; \; \mathsf{st}.\boldsymbol{t}[j] \leftarrow (\mathsf{st}.s[j], \mathsf{st}.z[j])$

    $\mathsf{st.rnd} \leftarrow \mathsf{st.rnd} + 1$; Return $(\mathsf{st}.\boldsymbol{t}[j], \mathsf{st})$

If $(\mathsf{st.rnd} = 2)$ then

    For all $i \neq j$ do $\mathsf{st}.\boldsymbol{t}[i] \leftarrow \boldsymbol{b}[i]$

    $(s, z) \leftarrow \sum_i^n \boldsymbol{t}[i]$; Return $((\mathsf{st}.T, s, z), \mathsf{st})$

Figure 3.8: Two-round multi-signature scheme $\mathsf{MS} = \mathsf{HBMS}[\mathbb{G}, g]$ parameterized by a group $\mathbb{G}$ of prime order p with generator g.

3.7 HBMS: Our new two-round multi-signature scheme

Recall that BN and MuSig are three-round schemes, and two-round schemes are desired due to blockchain applications. In this section, we introduce our new, efficient two-round multi-signature scheme supporting key-aggregation, HBMS. We first demonstrate its tight security against algebraic adversaries (Theorem 3.7.1), before justifying its security in the standard model (Theorem 3.7.2). Referring to Fig. 3.2, these results establish arrow 5. We refer to Fig. 3.1 for comparisons of HBMS against other two-round schemes.

TWO-ROUND MS SCHEME HBMS. The formal definition of our scheme is given in Fig. 3.8. HBMS has the same key generation algorithm Kg and key aggregation Ag algorithm as

MuSig. We describe informally the process involved to sign a message m under a vector of public keys $\boldsymbol{pk}$. In the first round, each signer i samples s_i and r_i uniformly from $\mathbb{Z}_p$ and computes a commitment

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

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

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

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

<u>TIGHT SECURITY AGAINST ALGEBRAIC ADVERSARIES.</u> We first show that HBMS is tightly MS-UF-secure against algebraic adversaries.

Theorem 3.7.1 [DL $\rightarrow$ HBMS, AGM] *Let $\mathbb{G}$ be a group of prime order p with generator g. Let* MS *be the* HBMS$[\mathbb{G}, g]$ *scheme. Let $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$ be an algebraic adversary for game $\mathbf{G}_{\mathsf{MS}}^{\text{ms-uf}}$ of Figure 3.4. Assume the execution of game $\mathbf{G}_{\mathsf{MS}}^{\text{ms-uf}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most q_1, q_2 distinct queries to* $\mathrm{H}_1, \mathrm{H}_2$*, respectively. Then we construct an adversary $\mathcal{A}_{\mathrm{dl}}$ for game* $\mathsf{DL}_{\mathbb{G},g}$ *(shown explicitly in Figure 3.25) such that*

$$\mathbf{Adv}_{\mathsf{MS}}^{\text{ms-uf}}(\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}) \leq \mathbf{Adv}_{\mathbb{G},g}^{\mathrm{dl}}(\mathcal{A}_{\mathrm{dl}}) + \frac{(q_1+1)q_2}{p} . \tag{3.7}$$

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

Above, a reduction is given directly from DL, and there is no multiplicative loss. As before, assuming $q_{\mathrm{s}} \leq q \leq t \leq p$ and the generic hardness of DL (advantage of t-time adversary to be at most t^2/p), we derive that $\mathbf{UB}^{\text{ms-uf}}_{\mathsf{MS}}(t, q, q_{\mathrm{s}}, p) \leq t^2/p$, as shown in Fig. 3.1.

We give the highlevel proof sketch here and defer the full proof to Section 3.16. Let $\mathcal{A}_{\mathrm{ms}}$ be the algebraic adversary against HBMS. Our reduction adversary $\mathcal{A}_{\mathrm{dl}}$ sets its own target point X (which it needs to obtain the discrete log of) as the target public key for $\mathcal{A}_{\mathrm{ms}}$. In order to run $\mathcal{A}_{\mathrm{ms}}$, our adversary $\mathcal{A}_{\mathrm{dl}}$ needs to be able to simulate oracles $\mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2$ (oracles representing the honest signer) as well as random oracles $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$. We first tackle the problem of simulating the honest signer without knowledge of the corresponding secret key. This is done by programming of random oracle H_0. Suppose for $\boldsymbol{pk}, m$, we set $\mathrm{H}_0((\boldsymbol{pk}, m))$ to be $h = g^{\alpha} pk^{\beta}$ for some $\alpha, \beta \neq 0 \in \mathbb{Z}_p$ (whose exact distribution will be specified later). When the adversary interacts with the honest signer, the honest signer must first provide some commitment $T \in G$ (in the output of NS), then later produce $z, s \in \mathbb{Z}_p$ (in the output of $\textsc{Sign}_1$) such that

$$g^z h^s = T \cdot pk^c \;, \tag{3.8}$$

where $c \in \mathbb{Z}_p$ is some challenge value (that is derived using the random oracle and the responses of the adversary). To do this, our adversary set commitment $T = g^a h^b$ for $a, b \leftarrow\!\!\$\, \mathbb{Z}_p$. It shall be convenient to express pk in terms of g and h as well. Note that as long as $\beta \neq 0$, $pk = h^{(\beta^{-1})} g^{-\alpha(\beta^{-1})}$. Since both T and pk are known to be of the form $g^{\star} h^{\star}$ (where $\star$ denotes some element of $\mathbb{Z}_p$), so is the group element $T \cdot pk^c$ (for any known value of c). Hence, the right-hand side of Equation (3.8) is of the form $g^z h^s$ for some values z and s that our adversary can compute, and our adversary can return them as response in the second round. Above, we noted that this works as long as $\beta \neq 0$. To guarantee this, we sample $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p$ and $\beta \leftarrow\!\!\$\, \mathbb{Z}_p^*$ in H_0. It remains to check that such way of simulating the honest signer is indistinguishable from the behavior of an honest signer holding the secrete key and executing the protocol. Roughly, this is because in both cases, the triple (T, z, s) is uniformly distributed over $\mathbb{G} \times \mathbb{Z}_p^2$, subjected to the condition that Equation (3.8) holds.

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

$$g^z h^s = T \cdot apk^c \;, \tag{3.9}$$

where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$ and $c = \mathrm{H}_1((T, apk, m))$. Since $\mathcal{A}_{\mathrm{ms}}$ is algebraic, our adversary $\mathcal{A}_{\mathrm{dl}}$ can rewrite Equation (3.9) to the form $g^{\alpha_g} = X^{\alpha_X}$, which allows us to compute the discrete log of X as $\alpha_g \alpha_x^{-1} \mod p$, as long as α_X is not zero. The full proof upperbounds the probability that $\alpha_X = 0$ to be at most $q_1 q_2 / p$. Outside of this bad event, our adversary $\mathcal{A}_{\mathrm{dl}}$ will successfully compute the value of $\mathsf{DL}_{\mathbb{G},g}(X)$ from a valid forgery.

STANDARD MODEL SECURITY OF HBMS. We reduce the security of HBMS to the hardness of XIDL, with factor q_{s} loss. For applications, the number of signing queries q_{s} is much less than adversarial hash function evaluations. As a result, even though our reduction here is non-tight, the reduction loss is smaller compared to previous results for BN, MuSig or other two round schemes (cf. Figure 3.1 and 3.1), at the expense of assuming the hardness of XIDL. Interestingly, due to Theorem 3.6.2, our results also state that HBMS is secure as long as MuSig is (via the reduction chain MuSig $\rightarrow$ XIDL $\rightarrow$ HBMS), and this reduction again only losses a factor of q_{s} in the advantage.

Theorem 3.7.2 [XIDL $\rightarrow$ HBMS, Standard Model] *Let $\mathbb{G}$ be a group of prime order p with generator g. Let* MS *be the* HBMS$[\mathbb{G}, g]$ *scheme given in Fig. 3.8. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Figure 3.4. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most $q_0, q_1, q_2, q_{\mathrm{s}}$ distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$, respectively. Then we construct an adversary*

$\mathcal{A}_{\text{xidl}}$ *for game* $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$ *(shown explicitly in Figure 3.27) such that*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\text{ms}}) \leq e(q_{\text{s}}+1) \cdot \mathbf{Adv}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\text{xidl}}) + \frac{q_1 q_2}{p} , \tag{3.10}$$

where e is the base of the natural logarithm. Adversary $\mathcal{A}_{\text{xidl}}$ *makes* q_2 *queries to* $\textsc{NwTar}$ *and* q_1 *queries to* $\textsc{Ch}$. *The running time of* $\mathcal{A}_{\text{xidl}}$ *is about that of the execution of game* $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ *with* $\mathcal{A}_{\text{ms}}$.

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

We sketch the highlevel proof here and give the full proof in Section 3.17. Our adversary receives the target point X from the XIDL game and sets it as the target public key for adversary $\mathcal{A}_{\text{ms}}$. As before, in order to run $\mathcal{A}_{\text{ms}}$, we need to simulate oracles $\textsc{NwTar}, \textsc{Sign}_1, \textsc{Sign}_2$ as well as $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$. Recall that in the AGM proof, we can simulate the honest signer for $\boldsymbol{pk}, m$ if we set $\mathrm{H}_0((\boldsymbol{pk},m)) = g^{\alpha}h^{\beta}$. However, this way of programming H_0 does not facilitate in turning a forgery into a break for XIDL. Instead, we would like to program $\mathrm{H}_0((\boldsymbol{pk},m)) = g^{\alpha}$ for the forgery $\boldsymbol{pk}, m$. To do this, we use a technique of Coron [32], which programs $\mathrm{H}_0((\boldsymbol{pk},m))$ randomly in one of these two ways depending on a biased coin flip (with probability ρ of giving 1). The reduction only succeeds if correct "guesses" are made. Specifically, we need that for every $\boldsymbol{pk}, m$ that is queried to the honest signer (in NS) then $\mathrm{H}_0((\boldsymbol{pk},m))$ must have been programmed to be $g^{\alpha}pk^{\beta}$ (for some α and β), and for the forgery $\boldsymbol{pk}, m$, it must be that $\mathrm{H}_0((\boldsymbol{pk},m)) = g^{\alpha}$ (for some α). We can then optimize for the value of ρ, resulting in a multiplicative loss of $e(1+q_{\text{s}})$.

Suppose adversary $\mathcal{A}_{\text{ms}}$ returns a forgery $(\boldsymbol{pk}, m, (T,s,z))$ where we have previously programmed $\mathrm{H}_0((\boldsymbol{pk},m)) = g^{\alpha}$. The verification equation say that $g^z h^s = T \cdot apk^c$. Since h is just a power of g, the left-hand side of the verification equation is also a known power of g (specifically $g^{z+\alpha \cdot s}$). This means that our adversary $\mathcal{A}_{\text{xidl}}$ can proceed exactly as the reduction for MuSig. In particular, for the j-th query of $\mathrm{H}_2((k,\boldsymbol{pk}))$ where $\boldsymbol{pk}[k] = X$, the adversary first computes

$S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$, then obtains $e_j \leftarrow\!\$\, \textsc{NwTar}(S)$ before returning e_j as the response for the query. We remark that this particular query of H_2 have created an aggregate public key $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))} = S \cdot X^{e_j}$, which is also the value of T_j that is recorded in the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. For each i-th query of $\mathrm{H}_1((T, apk, m))$, the adversary first finds the index j_{sel} of the H_2-query that corresponds to the input apk, then obtains $c_i \leftarrow\!\$\, \textsc{Ch}(j_{\mathrm{sel}}, T)$ before returning c_i as the response for the query. If the eventual forgery is given for these two particular queries to H_1 and H_2, meaning forgery is $\boldsymbol{pk}, m, (T, s, z)$, then the verification equation of the signature scheme says that $g^{z+\alpha \cdot s} = T \cdot apk^{\mathrm{H}_1((T,apk,m))}$ (if we programmed $\mathrm{H}_0((\boldsymbol{pk}, m))$ to be g^{α}). Hence, our adversary $\mathcal{A}_{\mathrm{xidl}}$ can simply return $(i, z + \alpha \cdot s)$ to break XIDL, as long as the forgery provided by $\mathcal{A}_{\mathrm{ms}}$ is valid and we have made the right guesses in programming H_0.

3.8 Security bounds of multi-signature schemes

We survey previous results on discrete-log-based multi-signature schemes, with a focus on their reduction loss. We restate these results in the same notation and framework to facilitate comparisons. We have used this to obtain the estimates in Figures 3.1 and 3.1.

For the rest of the section, fix a group $\mathbb{G}$ of prime order p that shall be used by each of the schemes of interest. Additionally, we assume that we fix adversaries $\mathcal{A}_{\mathrm{ms}}$ attacking each multi-signature scheme of interest, with running time t (this is the total execution time of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}})$ and includes the running time of all oracles), making q queries to the random oracle, q_{s} queries to NS involving maximum of N-signers while achieving success advantage of ε. For convenience, we let $q_{\mathrm{T}} = 1 + q + q_{\mathrm{s}}$.

<u>BN.</u> Bellare and Neven [14] gave a 3-round MS scheme that is based on the DL problem. In particular, they showed that given an MS-UF adversary $\mathcal{A}$, there exists DL-adversary with

running time t' achieving success advantage ε':

$$\varepsilon' \geq \frac{\varepsilon^2}{q+q_\mathrm{s}} - \frac{2q+16N^2q_\mathrm{s}}{2^\ell} - \frac{8Nq_\mathrm{s}}{p} , \tag{3.11}$$

$$t' \approx 2t , \tag{3.12}$$

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

MuSig. BDN [25] and MPSW [64] gave a 3-round MS scheme that adds key aggregation on-top of BN. For security, BDN showed [25][Theorem 4] that given an MS-UF adversary $\mathcal{A}$, there exists DL-adversary with running time t' achieving success advantage ε' where

$$\varepsilon' = \frac{\varepsilon - \delta}{64} , \tag{3.13}$$

$$t' = 512 \cdot t \cdot q_\mathrm{T}^2 (\varepsilon - \delta)^{-1} \ln^{-2}(64/(\varepsilon - \delta)) , \tag{3.14}$$

$$\delta = \frac{4Nq_\mathrm{T}}{p} , \tag{3.15}$$

as long as $p > 8q/\varepsilon$. MPSW gave a tighter result by two direct applications of the forking lemma. In particular, they showed that [64][Theorem 1] given an MS-UF adversary $\mathcal{A}_\mathrm{ms}$, there exists DL-adversary with running time t' achieving success advantage ε' where

$$\varepsilon' = \frac{\varepsilon^4}{q_\mathrm{T}^3} - \frac{16q_\mathrm{s}(q+N \cdot q_\mathrm{s})}{p} - \frac{16(q+Nq_\mathrm{s})^2+3}{2^\ell} , \tag{3.16}$$

$$t' \approx 4t . \tag{3.17}$$

mBCJ. DEFKLNS [40] gave a 2-round MS scheme mBCJ. For security, they showed that given an MS-UF adversary $\mathcal{A}$, there exists DL-adversary with running time t' achieving success advantage ε' where

$$\varepsilon' = \frac{\varepsilon}{8e(q_{\mathrm{s}}+1)} , \tag{3.18}$$

$$t' = t \cdot 64(N+1)^2 q_{\mathrm{T}}(q_{\mathrm{s}}+1)\varepsilon^{-1} \ln^{-1}(8e(N+1)(q_{\mathrm{s}}+1)/\varepsilon) , \tag{3.19}$$

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

MuSig-DN. NRSW [71] gave a 2-round MS scheme that has deterministic signing. For security, their result [71][Theorem 1] roughly translates to: given an an adversary attack MuSig-DN, there exists OMDL adversary attacking DL with success advantage approximately

$$\varepsilon' \geq \left(\varepsilon - q_{\mathrm{s}}\delta - \frac{q_{\mathrm{T}}^2}{2^{\lambda-2}} - \frac{2}{2^{\lambda/4}}\right)^4 q_{\mathrm{T}}^{-3} , \tag{3.20}$$

$$t' \approx 4t , \tag{3.21}$$

where λ is a parameter of the scheme and δ is a small constant associated with the group.

MuSig2. NRS [68] gave a 2-round MS scheme, parameterized by ν. For $\nu \geq 4$, they showed that if there exists $\mathcal{A}$ attacking their scheme, they [68][Theorem 1] can build νq_{s}-OMDL adversary with running time t' achieving success advantage ε' where

$$\varepsilon' \geq \frac{\varepsilon^4}{m^3} - \frac{11}{p} - \frac{43m^4}{(p-1)^{\nu-3}} , \tag{3.22}$$

$$t' \approx 4t , \tag{3.23}$$

$$m = (\nu-1)(q+q_{\mathrm{s}})+1 . \tag{3.24}$$

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

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

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

DWMS. Alper and Burdges [3] gave a 2-round MS scheme DWMS similar $\mathsf{MuSig2}$ that is also proved secure from OMDL in AGM. Their proof as given is non-concrete. However, tracing through their reduction, we obtained the following reduction loss: given an algebraic MS-UF adversary $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$, an q_{s}-OMDL adversary can be constructed with advantage ε' with running time t' where

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

and $t' \approx t$.

3.9 Forking lemma

We recall the general forking lemma of [14]. We restate it using the games of Figure 3.9. Each game has just one procedure, FIN, which takes no inputs. The games are parameterized by an algorithm $\mathcal{A}$ that is executed inside the game, and also by an algorithm IG called an input generator.

Lemma 3.9.1 [14] *Let $q \geq 1$ be an integer. Let C be a set of size $|C| \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on inputs $x, c_1, \ldots, c_q$ returns a pair, the first element of which is an integer in the range $0, \ldots, q$, and the second element of which we refer to as a side output. Let IG be a randomized algorithm that, as above, we call the input generator. Consider* Gm_0 *(called the*

Game Gm_0	Game Gm_1
FIN:	FIN:
$x \leftarrow\!\!\$\ \mathsf{IG}$	$x \leftarrow\!\!\$\ \mathsf{IG}$
$c_1, \ldots, c_q \leftarrow\!\!\$\ C$	$\rho \leftarrow\!\!\$\ \mathrm{rand}(\mathcal{A})$; $c_1, \ldots, c_q \leftarrow\!\!\$\ C$
$(I, \sigma) \leftarrow\!\!\$\ \mathcal{A}(x, c_1, \ldots, c_q)$	$(I, \sigma) \leftarrow\!\!\$\ \mathcal{A}(x, c_1, \ldots, c_q)$
Return $(I > 0)$	If $(I = 0)$ then return $(0, \varepsilon, \varepsilon)$
	$c'_I, \ldots, c'_q \leftarrow\!\!\$\ C$
	$(I', \sigma') \leftarrow\!\!\$\ \mathcal{A}(x, c_1, \ldots, c_{I-1}, c'_I, \ldots, c'_q)$
	Return $((I = I')$ and $(c_I \neq c'_I))$

Figure 3.9: Games referred to in Lemma 3.9.1. Both games have just one procedure, FIN, which does not take any input. These games run an algorithm $\mathcal{A}$ internally.

single run) and Gm_1 *(called the forked run) given in Fig. 3.9. Then:*

$$\Pr[\mathrm{Gm}_0] \leq \frac{q}{|C|} + \sqrt{q \cdot \Pr[\mathrm{Gm}_1]}\ . \tag{3.25}$$

3.10 Proof of Theorem 3.3.1

Proof of Theorem 3.3.1: Consider game Gm_0 given in the left panel of Fig. 3.10. By construction, it is the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}^{\mathrm{alg}}_{\mathrm{idl}})$. Next, consider game Gm_1, where the winning condition has been changed to checking that $(x = x')$, where x' is either computed on line 8 or 9 depending on whether $w = 0$. We claim that regardless of whether $w = 0$, game Gm_1 returns true as long as Gm_0 does. Assume Gm_0 returns $\mathtt{true}$, then b is set to true. If $w = 0$, then the game Gm_1 sets x' to x at line 8, so Gm_1 alsot returns $\mathtt{true}$. If $w \neq 0$, then the game Gm_1 computes x' as per line 12 and 13. Observe that if b is true, then

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

Game Gm_0, Gm_1, Gm_2

INIT:

$x \leftarrow\!\!\$\, \mathbb{Z}_p$; $X \leftarrow g^x$
$(I, z) \leftarrow\!\!\$\, \mathcal{A}_{\mathrm{idl}}^{\mathrm{CH}}(X)$
$b \leftarrow (g^z = R_I \cdot X^{c_I})$
$\underline{\mathrm{Gm}_0}$: Return b
$w \leftarrow (r_{I,2} + c_I)$
If $(w = 0)$ then $\mathsf{bad} \leftarrow \mathsf{true}$
  $\underline{\mathrm{Gm}_1}$:
    If b then $x' \leftarrow x$
    Else $x' \leftarrow \perp$
  $\underline{\mathrm{Gm}_2}$: $x' \leftarrow\!\!\$\, \mathbb{Z}_p$
Else
  $v \leftarrow (z - r_{I,1})$
  $x' \leftarrow v \cdot w^{-1} \mod p$
$\underline{\mathrm{Gm}_1, \mathrm{Gm}_2}$: Return $(x = x')$

$\underline{\mathrm{CH}(R, (r_1, r_2))}$:

$i \leftarrow i + 1$; $R_i \leftarrow R$
$r_{i,1} \leftarrow r_1$; $r_{i,2} \leftarrow r_2$
$c_i \leftarrow\!\!\$\, \mathbb{Z}_p$; Return c_i

Adversary $\mathcal{A}_{\mathrm{dl}}(X)$:

$(I, z) \leftarrow\!\!\$\, \mathcal{A}_{\mathrm{idl}}^{\mathrm{NwTar},\mathrm{CH}}(X)$
$w \leftarrow (r_{I,2} + c_I)$
If $(w = 0)$ then $x' \leftarrow\!\!\$\, \mathbb{Z}_p$
Else
  $v \leftarrow (z - r_{I,1})$
  $x' \leftarrow v \cdot w^{-1} \mod p$
Return x'

$\underline{\mathrm{CH}(R, (r_1, r_2))}$:

$i \leftarrow i + 1$; $R_i \leftarrow R$
$r_{i,1} \leftarrow r_1$; $r_{i,2} \leftarrow r_2$
$c_i \leftarrow\!\!\$\, \mathbb{Z}_p$; Return c_i

Figure 3.10: Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$ and adversary $\mathcal{A}_{\mathrm{dl}}$ the proof of Theorem 3.3.1.

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

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

which means that

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

So game Gm_1 must return true in this case as well. Hence

$$\Pr[\mathrm{Gm}_0] = \Pr[\mathrm{Gm}_1] . \tag{3.26}$$

Next, consider game Gm_2, which sets x' differently if $w = 0$. We have

$$\begin{aligned}\Pr[\mathrm{Gm}_1] &\leq \Pr[\mathrm{Gm}_2] + \Pr[\mathrm{Gm}_2 \text{ sets } \mathsf{bad}] \\ &\leq \Pr[\mathrm{Gm}_2] + \frac{q}{p} \ .\end{aligned} \tag{3.27}$$

Above, the calculation of $\Pr[\mathrm{Gm}_2 \text{ sets } \mathsf{bad}]$ is justified as follows. For each $\textsc{Ch}$ query, there is $1/p$ chance that $r_{i,2} + c_i = 0$, since c_i is uniform and independent of $r_{i,2}$. Hence, the probability that there is a choice of i to make $w = r_{i,2} + c_i$ zero is at most q/p using the union bound. Finally, we construct adversary $\mathcal{A}_{\mathrm{dl}}$, given in Fig. 3.10 such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) \ . \tag{3.28}$$

This is straight-forward, as $\mathcal{A}_{\mathrm{dl}}$ simulates $\textsc{Ch}$ and computes x' exactly as Gm_2. ▮

3.11 Proof of Theorem 3.3.2

Proof of Theorem 3.3.2: Consider games Gm_0 given in Fig. 3.11. Game Gm_0 pre-samples all the $c_1, \ldots, c_q$ values at line 2, but the game behaves otherwise exactly as $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}})$. We define $\Pr[\mathrm{Gm}_0]$ to be the probablity that the first component of the return value of Gm_0 is non-zero. Hence,

$$\Pr[\mathrm{Gm}_0] = \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}}) \ . \tag{3.29}$$

Next, consider Gm_1, which executes line 6 to 13 in addition to those executed by game Gm_0. Similar to Gm_0, we define $\Pr[\mathrm{Gm}_1]$ to be the probablity that the first component of the return value of Gm_1 is non-zero. We have constructed Gm_1 so that it is a forked run of Gm_0 (with $c_1, \ldots, c_q$ viewed as inputs) as defined by the forking lemma [14]. Specifically, line 8 to 10 freshly samples challenges $c'_I, \ldots, c'_{q_2}$ after the selected forgery index I before invoking $\mathcal{A}_{\mathrm{idl}}$ with these

Game Gm_0, Gm_1, Gm_2

INIT:

$x \leftarrow\!\!\$\; \mathbb{Z}_p$; $X \leftarrow g^x$
$\rho \leftarrow\!\!\$\; \mathrm{rand}(\mathcal{A}_{\mathrm{idl}})$; $c_1, \ldots, c_q \leftarrow\!\!\$\; \mathbb{Z}_p$
$(I, z) \leftarrow \mathcal{A}_{\mathrm{idl}}^{\mathrm{CH}_1}(X; \rho)$
$b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
If not b then $I \leftarrow 0$
$\underline{\mathrm{Gm}_0}$: Return $(I > 0)$
For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
$c'_I, c'_{I+1}, \ldots, c'_q \leftarrow\!\!\$\; \mathbb{Z}_p$
$i \leftarrow 0$; $(I', z') \leftarrow\!\!\$\; \mathcal{A}_{\mathrm{idl}}^{\mathrm{CH}_2}(X; \rho)$
$b' \leftarrow (g^{z'} = R_{I'} \cdot Y_{I'}^{c_{I'}})$
If not b' then $I' \leftarrow 0$
$\underline{\mathrm{Gm}_1}$:
  Return $((I = I' > 0)$ and $(c_I \neq c'_I))$
$\underline{\mathrm{Gm}_2}$:
  If $((I \neq I')$ or $(c_I = c'_I))$ then
    Return $\perp$
  $w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
  Return $(g^w = X)$

$\underline{\mathrm{CH}_1(R)}$:
$i \leftarrow i + 1$; $R_i \leftarrow R$
Return c_i

$\underline{\mathrm{CH}_2(R)}$:
$i \leftarrow i + 1$; $R'_i \leftarrow R$
Return c'_i

Adversary $\mathcal{A}_{\mathrm{dl}}(X)$:

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

$\underline{\mathrm{CH}_1(R)}$:
$i \leftarrow i + 1$; $R_i \leftarrow R$
Return c_i

$\underline{\mathrm{CH}_2(R)}$:
$i \leftarrow i + 1$; $R'_i \leftarrow R$
Return c'_i

Figure 3.11: Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$ and adversary $\mathcal{A}_{\mathrm{dl}}$ for proof of Theorem 3.3.2. $\rho \leftarrow\!\!\$\; \mathrm{rand}(\mathcal{A}_{\mathrm{idl}})$ denotes sampling the random coins of $\mathcal{A}_{\mathrm{idl}}$ and assigning it to ρ.

values programmed into CH_2. By the forking lemma, we have

$$\Pr[\mathrm{Gm}_0] \leq \frac{q_2}{p} + \sqrt{q_2 \cdot \Pr[\mathrm{Gm}_1]}\,. \tag{3.30}$$

We now move onto game Gm_2, which rewrites the winning condition of Gm_1 into line 15 to 18. We claim that game Gm_2 returns true as long as game Gm_1 returns true. This is because if both flags b and b' are ture, then

$$g^z = R_i X^{c_i}$$
$$g^{z'} = R_{i'} X^{c_{i'}} \ ,$$

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

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

which implies the the computed value of $w = (c_I - c'_I)^{-1}(z - z')$ (line 17) is the correct discrete log of X base g. As a result, Gm_2 must return true as well, and

$$\Pr[\mathrm{Gm}_2] \geq \Pr[\mathrm{Gm}_1] \ . \tag{3.31}$$

Finally, we construct adversary $\mathcal{A}_{\mathrm{dl}}$, given in Fig. 3.11, such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) \ . \tag{3.32}$$

Adversary $\mathcal{A}_{\mathrm{dl}}$ forwards its target point X to $\mathcal{A}_{\mathrm{idl}}$ and simulates Gm_2, starting from line 2 of Gm_2 and ending at line 17 of Gm_2, before outputting the computed value of w as the discrete log of target point X. Putting the above equations together, we obtain the claim in the theorem. ▮

Game Gm_0, Gm_1, Gm_2

INIT:

$x \leftarrow\!\!\$\; \mathbb{Z}_p$; $X \leftarrow g^x$

$e_1, \ldots, e_{q_1} \leftarrow\!\!\$\; \mathbb{Z}_p$; $c_1, \ldots, c_{q_2} \leftarrow\!\!\$\; \mathbb{Z}_p$

$(I, z) \leftarrow\!\!\$\; \mathcal{A}_{\text{xidl}}^{\text{NwTar,Ch}}(X)$

$b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$

Gm_0: Return b

$w \leftarrow (r_{I,2} + (s_{I,2} + e_J) \cdot c_I)$

If $(w = 0)$ then $\mathsf{bad} \leftarrow \mathsf{true}$

  Gm_1:

    If b then $x' \leftarrow x$

    Else $x' \leftarrow \perp$

  Gm_2: $x' \leftarrow\!\!\$\; \mathbb{Z}_p$

Else

  $v \leftarrow (z - r_{I,1} - s_{I,1} \cdot c)$

  $x' \leftarrow v \cdot w^{-1} \mod p$

Gm_1, Gm_2: Return $(x = x')$

NWTAR$(S, (s_1, s_2))$:

$j \leftarrow j + 1$; $S_j \leftarrow S$

$s_{j,1} \leftarrow s_1$; $s_{j,2} \leftarrow s_2$

$e_j \leftarrow\!\!\$\; \mathbb{Z}_p$; $T_j \leftarrow S_j \cdot X^{e_j}$

Return e_j

CH$(j_{\text{sel}}, R, (r_1, r_2))$:

Requires $1 \le j_{\text{sel}} \le j$

$i \leftarrow i + 1$; $R_i \leftarrow R$

$r_{i,1} \leftarrow r_1$; $r_{i,2} \leftarrow r_2$

$Y_i \leftarrow T_{j_{\text{sel}}}$; $\text{TJ}[i] \leftarrow j_{\text{sel}}$

$c_i \leftarrow\!\!\$\; \mathbb{Z}_p$; Return c_i

Adversary $\mathcal{A}_{\text{dl}}(X)$:

$(I, z) \leftarrow\!\!\$\; \mathcal{A}_{\text{xidl}}^{\text{NwTar,Ch}}(X)$

$J \leftarrow \text{TJ}[I]$

$w \leftarrow (r_2 + s_2 \cdot e_J \cdot c_I)$

If $(w = 0)$ then $x' \leftarrow\!\!\$\; \mathbb{Z}_p$

Else

  $v \leftarrow (z - r_{I,1} - s_{I,1} \cdot c)$

  $x' \leftarrow v \cdot w^{-1} \mod p$

Return x'

NWTAR$(S, (s_1, s_2))$:

$j \leftarrow j + 1$; $S_j \leftarrow S$

$s_{j,1} \leftarrow s_1$; $s_{j,2} \leftarrow s_2$

$e_j \leftarrow\!\!\$\; \mathbb{Z}_p$; $T_j \leftarrow S_j \cdot X^{e_j}$

Return e_j

CH$(j_{\text{sel}}, R, (r_1, r_2))$:

Requires $1 \le j_{\text{sel}} \le j$

$i \leftarrow i + 1$; $R_i \leftarrow R$

$r_{i,1} \leftarrow r_1$; $r_{i,2} \leftarrow r_2$

$Y_i \leftarrow T_{j_{\text{sel}}}$; $\text{TJ}[i] \leftarrow j_{\text{sel}}$

$c_i \leftarrow\!\!\$\; \mathbb{Z}_p$; Return c_i

Figure 3.12: Games $\text{Gm}_0, \text{Gm}_1, \text{Gm}_2$ and adversary $\mathcal{A}_{\text{dl}}$ the proof of Theorem 3.3.3.

3.12 Proof of Theorem 3.3.3

Proof of Theorem 3.3.3: We recall the convention that representation of each of the group elements S and R are additionally supplied when oracles NWTAR and CH are called. Specifically, each of its NWTAR queries must be of the form

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

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

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

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

Consider game Gm_0 given in the left panel of Fig. 3.12. By construction, it is the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}})$. Next, consider game Gm_1, where the winning condition has been changed to checking that $(x = x')$, where x' is either computed on line 9 or 10 depending on whether $w = 0$. We claim that regardless of the value of w, game Gm_1 returns true as long as Gm_0 does (Gm_0 returns the boolean value b). We check this by cases. First, if $w = 0$, then the games sets x' to x if b is true, so Gm_1 also returns true. If $w \neq 0$, then observe that if b is true, then

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

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

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

which means that

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

Hence

$$\Pr[\mathrm{Gm}_0] = \Pr[\mathrm{Gm}_1]\ . \tag{3.33}$$

Next, consider game Gm_2, which sets x' differently if $w = 0$. We have

$$\begin{aligned}\Pr[\mathrm{Gm}_1] &\leq \Pr[\mathrm{Gm}_2] + \Pr[\mathrm{Gm}_2 \text{ sets bad}] \\ &\leq \Pr[\mathrm{Gm}_2] + \frac{q_1 + q_2}{p} \ .\end{aligned} \tag{3.34}$$

Above, the calculation of $\Pr[\mathrm{Gm}_2 \text{ sets bad}]$ is justified as follows. First, the probability that $s_{j,2} + e_j = 0$ for any j is at most q_1/p, since e_j is uniform and independnet of $s_{j,2}$. Second, assuming $s_{j,2} + e_j \neq 0$ for all j, then the probability that $r_{i,2} + (s_{\mathrm{TJ}[i],2} + e_{\mathrm{TJ}[i]}) \cdot c_i = 0$ for some i is at most q_2/p, since c_i is uniform and independent of $r_{i,2}$. Finally, we construct adversary $\mathcal{A}_{\mathrm{dl}}$, given in the right panel of Fig. 3.12 such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{dl}}_{\mathbb{G},g}(\mathcal{A}_{\mathrm{dl}}) \ . \tag{3.35}$$

This is straight-forward, as $\mathcal{A}_{\mathrm{dl}}$ simulates $\textsc{NwTar}, \textsc{Ch}$ and computes x' exactly as Gm_2. ▮

3.13 Proof of Theorem 3.3.4

Proof of Theorem 3.3.4: Consider games Gm_0 given in Fig. 3.13. Game Gm_0 pre-samples all the e_j and c_i values at line 2 and 3, but the game behaves otherwise exactly as $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}})$. We define $\Pr[\mathrm{Gm}_0]$ to be the probablity that the first component of the return value of Gm_0 is non-zero. Hence,

$$\Pr[\mathrm{Gm}_0] = \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_1,q_2}(\mathcal{A}_{\mathrm{xidl}}) \ . \tag{3.36}$$

Next, consider Gm_1, which executes line 6 to 14 addition to those executed by game Gm_0. Similar to Gm_0, we define $\Pr[\mathrm{Gm}_1]$ to be the probablity that the first component of the return value of Gm_1 is non-zero. We have constructed Gm_1 so that it is a forked run of Gm_0 (with $c_1, \ldots, c_{q_2}$ viewed as inputs) as defined by the forking lemma [14]. Specifically, line 8 to 10 freshly samples

Game Gm_0, Gm_1, Gm_2

INIT:

$x \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_p$; $X \leftarrow g^x$; $\rho \leftarrow\!\!{\scriptstyle\$}\; \mathrm{rand}(\mathcal{A}_{\mathrm{xidl}})$
$e_1, \ldots, e_{q_1} \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_p$; $c_1, \ldots, c_{q_2} \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_p$
$(I, z) \leftarrow \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{Ch}}(X; \rho)$
$b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
If not b then $I \leftarrow 0$
Gm_0: Return $(I > 0)$
For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
$i, j \leftarrow 0$; $c'_I, c'_{I+1}, \ldots, c'_q \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_p$
$(I', z') \leftarrow\!\!{\scriptstyle\$}\; \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar},\mathrm{ChSim}}(X; \rho)$
$b' \leftarrow (g^{z'} = R_{I'} \cdot Y_{I'}^{c'_{I'}})$
If not b' then $i' \leftarrow 0$
$j \leftarrow \mathrm{TJ}[I]$; $j' \leftarrow \mathrm{TJ}[I']$
Gm_1:
  Return $((I = I' > 0)$ and $(c_I \neq c'_I))$
Gm_2:
  If $((I \neq I')$ or $(c_I = c'_I))$ then
    Return $\perp$
  $w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
  Return $(g^w = T_j)$

NWTAR(S):

$j \leftarrow j+1$; $S_j \leftarrow S$; $T_j \leftarrow S_j \cdot X^{e_j}$
Return e_j

$\mathrm{ChSim}_i(j_{\mathrm{sel}}, R)$: // $i \in \{1,2\}$

$i \leftarrow i+1$; $R_i \leftarrow R$
$Y_i \leftarrow T_{j_{\mathrm{sel}}}$; $\mathrm{TJ}[i] \leftarrow j_{\mathrm{sel}}$
ChSim_1 : Return c_i
ChSim_2 : Return c'_i

Adversary $\mathcal{A}_{\mathrm{idl}}^{\mathrm{Ch}}(X)$:

$c_1, \ldots, c_{q_2} \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_p$; $\rho \leftarrow\!\!{\scriptstyle\$}\; \mathrm{rand}(\mathcal{A}_{\mathrm{xidl}})$
$(I, z) \leftarrow\!\!{\scriptstyle\$}\; \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar}_1,\mathrm{ChSim}_1}(X; \rho)$
$b \leftarrow (g^z = R_I \cdot Y_I^{c_I})$
If not b then Return $\perp$
For $i = 1, \ldots, I-1$ do $c'_i \leftarrow c_i$
$i, j \leftarrow 0$; $c'_I, c'_{I+1}, \ldots, c'_q \leftarrow\!\!{\scriptstyle\$}\; \mathbb{Z}_p$
$(I', z') \leftarrow\!\!{\scriptstyle\$}\; \mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar}_2,\mathrm{ChSim}_2}(X; \rho)$
$b' \leftarrow (g^{z'} = R_{I'} \cdot Y_{I'}^{c'_{I'}})$
If not b' then Return $\perp$
If $((I \neq I')$ or $(c_I = c'_I))$ then
  Return $\perp$
$j \leftarrow \mathrm{TJ}[I]$; $j' \leftarrow \mathrm{TJ}[I']$
$w \leftarrow (c_I - c'_I)^{-1}(z - z') \mod p$
Return (j, w)

NWTAR$_1(S)$:

$j \leftarrow j+1$; $e_j \leftarrow\!\!{\scriptstyle\$}\; \mathrm{Ch}(S)$; $S_j \leftarrow S$
$T_j \leftarrow S_j \cdot X^{e_j}$; Return e_j

NWTAR$_2(S)$:

$j \leftarrow j+1$
If not e_j then $e_j \leftarrow \mathrm{Ch}(S)$
Return e_j

$\mathrm{ChSim}_i(j_{\mathrm{sel}}, R)$ // $i \in \{1,2\}$:

$i \leftarrow i+1$; $R_i \leftarrow R$
$Y_i \leftarrow T_{j_{\mathrm{sel}}}$; $\mathrm{TJ}[i] \leftarrow j_{\mathrm{sel}}$
ChSim_1 : Return c_i
ChSim_2 : Return c'_i

Figure 3.13: Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$ and adversary $\mathcal{A}_{\mathrm{idl}}$ for proof of Theorem 3.3.4.

challenges $c'_i, \ldots, c'_{q_2}$ after the selected forgery index i before invoking $\mathcal{A}_{\mathrm{xidl}}$ with these values reprogrammed into CH. We remark that the values of $e_1, \ldots, e_{q_1}$, which are outputs of NWTAR are *not* resampled across the two runs of $\mathcal{A}_{\mathrm{xidl}}$. By the forking lemma, we have

$$\Pr[\mathrm{Gm}_0] \le \frac{q_2}{p} + \sqrt{q_2 \cdot \Pr[\mathrm{Gm}_1]} \,. \tag{3.37}$$

We now move onto game Gm_2, which rewrites the winning condition of Gm_1 into line 16 to 19. We claim that game Gm_2 returns true as long as game Gm_1 returns true. This is because if both flags b and b' are ture, then

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

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

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

which implies the the computed value of w (line 18) is the correct discrete log of T_J base g. As a result, Gm_2 must return true as well, and

$$\Pr[\mathrm{Gm}_2] \ge \Pr[\mathrm{Gm}_1] \,. \tag{3.38}$$

Finally, we construct adversary $\mathcal{A}_{\mathrm{idl}}$, given in Fig. 3.13, such that

$$\Pr[\mathrm{Gm}_2] = \mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q_1}(\mathcal{A}_{\mathrm{idl}}) \,. \tag{3.39}$$

Crucially, in the construction of $\mathcal{A}_{\mathrm{idl}}$, NWTAR oracle need to be simulated differently for the two runs of $\mathcal{A}_{\mathrm{xidl}}$. In the first run, the oracle NWTAR_1 forwards the queries to CH (that is given to our reduction adversary from the game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q_1}$), while recording the responses $e_1,\ldots,e_j$. Then, in the second run, the oracle $\textsc{NwTar}_2$ will return previously recorded values of $e_1,\ldots,e_j$ as long as they are available, and only starts to forward queries when it runs out of previously recorded ones. This is to simulate the behavior of Gm_2, where there is one single fixed sequence of values $e_1,\ldots,e_{q_1}$, used by the oracle $\textsc{NwTar}$. Putting the above equations together, we obtain the claim in the theorem. ❚

3.14 Proof of Theorem 3.5.1

Proof of Theorem 3.5.1: The proof uses a game sequence. Our games will implement $\mathrm{H}_0,\mathrm{H}_1$ with lazy sampling, maintaining tables $\mathrm{HF}_0,\mathrm{HF}_1$ for this purpose. They will provide oracles $\textsc{Sign}_1,\textsc{Sign}_2$ for the first two rounds, but omit $\textsc{Sign}_3$, since this round returns to the adversary only a quantity it could itself compute already. In $\textsc{Fin}$ (for example Figure 3.14) we assume the query is non-trivial, meaning lines 6,7 of Figure 3.4 return true, and these lines are thus omitted. We start with games $\mathrm{Gm}_0,\mathrm{Gm}_1$ in Figure 3.14. Game Gm_0 includes the boxed code, and we claim that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})]\ . \tag{3.40}$$

Let us explain. We wish to move to a game where signing queries are answered without using the secret key sk. Naturally, we expect, for this, to use the zero-knowledge property of the Schnorr scheme. But certain obstacles must be removed before we can do this, and this will take a few steps. The first obstacle we address is that the BN-commitment $t_{u,k} = \mathrm{H}_0((k,R_{u,k}))$ may leak information about $R_{u,k}$. Rather than define $t_{u,k}$ in this way, games $\mathrm{Gm}_0,\mathrm{Gm}_1$ accordingly pick it at random at line 3. The reason for the boxed code at line 4 is that, under the "true" assignment $t_{u,k} = \mathrm{H}_0((k,R_{u,k}))$, having $R_{u,k_u} = R_{u',k_{u'}}$ would imply $t_{u,k_u} = t_{u',k_{u'}}$. At line 8, now

INIT: **/** Games Gm_0–Gm_7

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

$\mathrm{NS}(k, \boldsymbol{pk}, m)$: **/** Games $\boxed{\mathrm{Gm}_0}$, Gm_1

$u \leftarrow u+1$; $k_u \leftarrow k$; $\boldsymbol{pk}[1] \leftarrow pk$; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$; $m_u \leftarrow m$; $n_u \leftarrow |\boldsymbol{pk}|$

$\mathsf{CommitStage}_u \leftarrow \mathsf{true}$; $r_{u,k} \leftarrow\$ \mathbb{Z}_p$; $R_{u,k} \leftarrow g^{r_{u,k}}$; $t_{u,k} \leftarrow\$ \{0,1\}^\ell$

If $(\exists u' < u : R_{u,k_u} = R_{u',k_{u'}})$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{t_{u,k_u} \leftarrow t_{u',k_{u'}}}$

If $(\mathrm{HF}_0[(k, R_{u,1})] \neq \bot)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{t_{u,k} \leftarrow \mathrm{HF}_0[(k, R_{u,k})]}$

Return $t_{u,k}$

$\text{SIGN}_0(s, \boldsymbol{t})$: **/** Games $\mathrm{Gm}_0, \mathrm{Gm}_1$

$k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $\mathsf{CommitStage}_s \leftarrow \mathsf{false}$

$\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$; Return $R_{s,k}$

$\text{SIGN}_1(s, \boldsymbol{R})$: **/** Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$

$k \leftarrow k_s$; $\boldsymbol{R}[k] \leftarrow R_{s,k}$

For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$

If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\bot$

$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$

Return $z_{s,k}$

$\mathrm{H}_0(x)$: **/** Games $\boxed{\mathrm{Gm}_0}$, Gm_1

If $(\mathrm{HF}_0[x] \neq \bot)$ then Return $\mathrm{HF}_0[x]$

$\mathrm{HF}_0[x] \leftarrow\$ \{0,1\}^\ell$

If $(\exists u' : x = (k_{u'}, R_{u',k_{u'}})$ and $\mathsf{CommitStage}_{u'})$ then

  $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathrm{HF}_0[x] \leftarrow t_{u',k_{u'}}}$

Return $\mathrm{HF}_0[x]$

$\mathrm{H}_1(x)$: **/** Games Gm_0–Gm_7

If $(\mathrm{HF}_1[x] \neq \bot)$ then Return $\mathrm{HF}_1[x]$

$\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$; Return $\mathrm{HF}_1[x]$

$\text{FIN}(\boldsymbol{pk}, m, (R, z))$: **/** Games Gm_0–Gm_7

$n \leftarrow |\boldsymbol{pk}|$

For $i = 1, \ldots, n$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$

$X \leftarrow \prod_{i=1}^{n} \boldsymbol{pk}[i]^{c_i}$; Return $(g^z = RX)$

Figure 3.14: Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ for proof of Theorem 3.5.1. Some procedures will be included in later games, as indicated. A box around the name of a game following an oracle means the boxed code in that oracle is included in the game.

that the BN-commitments $\boldsymbol{t}$ of all players are known, the games ensure that $t_{u,k}$ indeed equals $\mathrm{H}_0((k, R_{u,k}))$. This is consistent with the real game only if the hash function was not already defined at this point, captured by setting bad at line 17. The boolean $\mathsf{CommitStage}$ ensures that bad is only set prior to the release of $R_{s,k}$, since the adversary can set it with probability one if it knows $R_{s,k}$. This justifies Eq. (3.40).

Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [19]

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

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

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

Game Gm_2 changes the $\mathrm{NS}, \textsc{Sign}_0, \mathrm{H}_0$ oracles as shown in Figure 3.15, maintaining the other oracles of Gm_1 from Figure 3.14. It drops redundant code, which allows it to move the choice of $R_{s,k}$ to line 28. At line 40, it also introduces a table HI to maintain an inverse of the hash function, but does not use this. We have

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

Game Gm_3 (oracles shown across Figures 3.15 and 3.14) aims to figure out the $R_{s,j}$-values of parties $j \neq k$ before having to supply $R_{s,k}$, because we will later need these to program H_1 values. It does this by "inverting" the BN-commitments, meaning at line 30 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 31. (Not finding the inverses is not yet a bad event. It can happen with high

$\mathrm{NS}(k, \boldsymbol{pk}, m)$: **/** Games Gm_2–Gm_7

24 $u \leftarrow u+1$; $k_u \leftarrow k$; $\boldsymbol{pk}[1] \leftarrow pk$; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$; $m_u \leftarrow m$; $n_u \leftarrow |\boldsymbol{pk}|$

25 $t_{u,1} \leftarrow\!\!\$\, \{0,1\}^{\ell}$; Return $t_{u,1}$

$\mathrm{SIGN}_0(s, \boldsymbol{t})$: **/** Game Gm_2

26 $\boldsymbol{t}[1] \leftarrow t_{s,1}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $r_{s,1} \leftarrow\!\!\$\, \mathbb{Z}_p$; $R_{s,1} \leftarrow g^{r_{s,1}}$; $\mathrm{HF}_0[(1, R_{s,1})] \leftarrow t_{s,1}$

27 Return $R_{s,1}$

$\mathrm{SIGN}_0(s, \boldsymbol{t})$: **/** Games $\mathrm{Gm}_3, \mathrm{Gm}_4$

28 $k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $r_{s,k} \leftarrow\!\!\$\, \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{r_{s,1}}$; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$

29 For $i = 1, \ldots, n_s$ do

30   If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$

31   Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\, \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$

32 Return $R_{s,k}$

$\mathrm{SIGN}_1(s, \boldsymbol{R})$: **/** Games Gm_3, $\boxed{\mathrm{Gm}_4}$

33 $k \leftarrow k_s$; $\boldsymbol{R}[k] \leftarrow R_{s,k}$

34 For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$

35 If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\perp$

36 If $(\boldsymbol{R} \neq \boldsymbol{R}_s^*)$ then bad $\leftarrow$ true ; $\boxed{\boldsymbol{R} \leftarrow \boldsymbol{R}_s^*}$

37 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$

38 Return $z_{s,k}$

$\mathrm{H}_0(x)$: **/** Games Gm_2–Gm_7

39 If $(\mathrm{HF}_0[x] \neq \perp)$ then Return $\mathrm{HF}_0[x]$

40 $\mathrm{HF}_0[x] \leftarrow\!\!\$\, \{0,1\}^{\ell}$; $(i, R) \leftarrow x$; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$; Return $\mathrm{HF}_0[x]$

Figure 3.15: Games for proof of Theorem 3.5.1.

probability. It becomes a bad event only at line 36 when the BN-commitments are verified.) The computation of t at that line is only to ensure that 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 36, which now includes the boxed code. The hope here is that the $\boldsymbol{R}_s^*$ obtained at lines 30,31 is correct with high probability. The boxed code

$\textsc{Sign}_0(s,\boldsymbol{t})$: **/** Game Gm_5

41 $k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $r_{s,k} \leftarrow\!\!\$\, \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{r_{s,k}}$; $\mathrm{HF}_0[(k,R_{s,k})] \leftarrow t_{s,k}$

42 For $i = 1,\ldots,n_s$ do

43   If $(\mathrm{HI}_0[i,\boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i,\boldsymbol{t}_s[i]]$

44   Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\, \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i,\boldsymbol{R}_s^*[i]))$

45 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$; $c_{s,k} \leftarrow \mathrm{H}_1((k,R_s,\boldsymbol{pk}_s,m_s))$; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$

46 Return $R_{s,k}$

$\textsc{Sign}_1(s,\boldsymbol{R})$: **/** Game $\mathrm{Gm}_5, \mathrm{Gm}_6, \mathrm{Gm}_7$

47 $k \leftarrow k_s$; $\boldsymbol{R}[k] \leftarrow R_{s,k}$

48 For $i = 1,\ldots,n_s$ do $y_i \leftarrow \mathrm{H}_0((i,\boldsymbol{R}[i]))$

49 If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\perp$ else Return $z_{s,k}$

$\textsc{Sign}_0(s,\boldsymbol{t})$: **/** Game $\boxed{\mathrm{Gm}_6}$, Gm_7

50 $k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$

51 $c_{s,k} \leftarrow\!\!\$\, \mathbb{Z}_p$; $z_{s,k} \leftarrow\!\!\$\, \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$; $\mathrm{HF}_0[(k,R_{s,k})] \leftarrow t_{s,k}$

52 For $i = 1,\ldots,n_s$ do

53   If $(\mathrm{HI}_0[i,\boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i,\boldsymbol{t}_s[i]]$

54   Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\, \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i,\boldsymbol{R}_s^*[i]))$

55 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$

56 If $(\mathrm{HF}_1((k,R_s,\boldsymbol{pk}_s,m_s)) \neq \perp)$ then bad $\leftarrow$ true ; $\boxed{c_{s,k} \leftarrow \mathrm{HF}_1[(k,R_s,\boldsymbol{pk}_s,m_s)]}$

57 $\mathrm{HF}_1[(k,R_s,\boldsymbol{pk}_s,m_s)] \leftarrow c_{s,k}$; Return $R_{s,k}$

Figure 3.16: Games for proof of Theorem 3.5.1.

ensures that in Gm_4, it is always correct. Since $\mathrm{Gm}_3, \mathrm{Gm}_4$ are identical-until-bad we have

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

Line 36 can only set bad if $y_i = \boldsymbol{t}_s[i]$ for all i, due to line 35. So it is set only if there is a collision in H_0-values, or no query hashing to $\boldsymbol{t}_s[i]$ was made prior to the latter being provided, but is made later. Thus

$$\Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets bad}] \leq \frac{q_0^2 + nq_0}{2^\ell} . \tag{3.41}$$

In game Gm_4, the $\boldsymbol{R}$ queried to $\textsc{Sign}_1$ is the same as the $\boldsymbol{R}^*$ determined in $\textsc{Sign}_0$, allowing game Gm_5 (Figure 3.16) to move line 37 into $\textsc{Sign}_0$ as line 45 and to simplify $\textsc{Sign}_1$. We have

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

Now that R_s is determined prior to the release of R_{s,k_s}, it becomes possible to successfully program H_1 via the zero-knowledge simulation. Game Gm_6 of Figure 3.16 does this, setting bad at line 56 if the programming was precluded by the hash value already being defined, and including the boxed code to correct. We have

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

Games $\mathrm{Gm}_6, \mathrm{Gm}_7$ (Figure 3.16) are identical-until-bad, so

$$\Pr[\mathrm{Gm}_6(\mathcal{A})] \leq \Pr[\mathrm{Gm}_7(\mathcal{A})] + \Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets } \mathsf{bad}] . \tag{3.42}$$

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

$$\Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_s q_1}{p} . \tag{3.43}$$

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

$$\mathbf{Adv}^{\mathrm{idl}}_{\mathbb{G},g,q}(\mathcal{A}_{\mathrm{idl}}) \geq \Pr[\mathrm{Gm}_7(\mathcal{A}_{\mathrm{ms}})] . \tag{3.44}$$

We specify $\mathcal{A}_{\mathrm{idl}}$ in Figure 3.17. It forwards the public key pk to $\mathcal{A}_{\mathrm{ms}}$. Simulating signatures without knowing the secret key, as $\mathcal{A}_{\mathrm{idl}}$ needs to do, is now easy because the oracles of games Gm_7 already did this, and $\mathcal{A}_{\mathrm{idl}}$ can just use the same code. Line 17 to 19 programs the challenge c_k of the target public key by first deriving commitment R_k, which is then submitted to $\textsc{Ch}$ to derive c_k. Since $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$ game also samples the challenge uniformly at random, this does not

Adversary $\mathcal{A}_{\mathrm{idl}}^{\mathrm{CH}}(pk)$:

$(\boldsymbol{pk}, m, (R,z)) \leftarrow\$ \mathcal{A}^{\mathrm{NS},\mathrm{SIGN}_0,\mathrm{SIGN}_1,\mathrm{H}_0,\mathrm{H}_1}(pk)$; Return $(\mathrm{TJ}[R], z)$

$\mathrm{NS}(k, \boldsymbol{pk}, m)$:

$u \leftarrow u+1$; $k_u \leftarrow k$; $\boldsymbol{pk}[1] \leftarrow pk$; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$; $m_u \leftarrow m$; $n_u \leftarrow |\boldsymbol{pk}|$

$t_{u,k} \leftarrow\$ \{0,1\}^{\ell}$; Return $t_{u,k}$

$\mathrm{SIGN}_0(s, \boldsymbol{t})$:

$k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$

$c_{s,k} \leftarrow\$ \mathbb{Z}_p$; $z_{s,k} \leftarrow\$ \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$

For $i = 1, \ldots, n_s$ do

  If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \bot)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$

  Else $\boldsymbol{R}_s^*[i] \leftarrow\$ \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$

$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$; $\mathrm{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)] \leftarrow c_{s,k}$; Return $R_{s,k}$

$\mathrm{SIGN}_1(s, \boldsymbol{R})$:

$k \leftarrow k_s$; $\boldsymbol{R}[k] \leftarrow R_{s,k}$

For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$

If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\bot$ else Return $z_{s,k}$

$\mathrm{H}_0(x)$:

If $(\mathrm{HF}_0[x] \neq \bot)$ then Return $\mathrm{HF}_0[x]$

$\mathrm{HF}_0[x] \leftarrow\$ \{0,1\}^{\ell}$; $(i, R) \leftarrow x$; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$; Return $\mathrm{HF}_0[x]$

$\mathrm{H}_1(x)$:

If $(\mathrm{HF}_1[x] \neq \bot)$ then Return $\mathrm{HF}_1[x]$

$(k, R, \boldsymbol{pk}, m) \leftarrow x$; $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$

If $(\boldsymbol{pk}[k] = pk)$ then

  $j \leftarrow j+1$; For $i = 2, \ldots, |\boldsymbol{pk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$

  $R_{j,k} \leftarrow R \cdot \prod_{i \neq k} \boldsymbol{pk}[i]^{c_i}$; $\mathrm{HF}_1[x] \leftarrow c_k \leftarrow \mathrm{CH}(R_{j,k})$; $\mathrm{TJ}[R] \leftarrow j$

Return $\mathrm{HF}_1[x]$

Figure 3.17: Adversary $\mathcal{A}_{\mathrm{idl}}$ for Theorem 3.5.1.

change the behavior of H_1. However, if a forgery $(\boldsymbol{pk}, m, (R,z))$, then it must be that

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

So $\mathcal{A}_{\mathrm{idl}}$ wins game $\mathrm{Gm}^{\mathrm{idl}}_{\mathbb{G},g,q}$. Eq. (3.3) is obtained by putting the above all together. ▮

3.15 Proof of Theorem 3.6.1

Let $\mathbb{G}$ be a group of prime order p with generator g. Let $\mathsf{MS} = \mathsf{MuSig}[\mathbb{G}, g, \ell]$ be the associated MuSig multi-signature scheme. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}$ of Figure 3.4. We shall fix these quantities for the rest of the proof. The first lemma relates the advantage of $\mathcal{A}_{\mathrm{ms}}$ against $\mathbf{G}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}$ to a simplied game $\mathrm{Gm}_{\mathrm{simp}}$ (given in Fig. 3.18).

Lemma 3.15.1 *Assume the execution of game* $\mathbf{G}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}$ *with* $\mathcal{A}_{\mathrm{ms}}$ *has at most* $q_0, q_1, q_2, q_{\mathrm{s}}$ *distinct queries to* $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$*, respectively, and the number of parties (length of verification-key vector) in queries to* NS *and* $\mathrm{F{\scriptstyle IN}}$ *is at most n. Let* $\alpha = q_{\mathrm{s}}(4q_0 + 2q_1 + q_{\mathrm{s}}) + 2q_1q_2$ *and* $\beta = q_0(q_0 + n)$*. Then,*

$$\mathbf{Adv}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) \leq \Pr[\mathrm{Gm}_{\mathrm{simp}}(\mathcal{A}_{\mathrm{ms}})] + \frac{\alpha}{2p} + \frac{\beta}{2^{\ell}} \,. \tag{3.45}$$

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

Lemma 3.15.2 *Assume the execution of game* $\mathbf{G}^{\mathrm{ms\text{-}uf}}_{\mathsf{MS}}$ *with* $\mathcal{A}_{\mathrm{ms}}$ *has at most* $q_0, q_1, q_2, q_{\mathrm{s}}$ *distinct queries to* $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}$*, respectively. We construct an adversary* $\mathcal{A}_{\mathrm{xidl}}$ *for game* $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$ *(shown explicitly in Figure 3.23) such that*

$$\Pr[\mathrm{Gm}_{\mathrm{simp}}(\mathcal{A}_{\mathrm{ms}})] \leq \mathbf{Adv}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}}) \,. \tag{3.46}$$

Proof of Lemma 3.15.1:

The proof uses a game sequence. Our games will implement $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ with lazy sampling, maintaining tables $\mathrm{HF}_0, \mathrm{HF}_1, \mathrm{HF}_2$ for this purpose. They will provide oracles $\mathrm{S{\scriptstyle IGN}}_0, \mathrm{S{\scriptstyle IGN}}_1$ while omitting $\mathrm{S{\scriptstyle IGN}}_2$, since this round returns to the adversary only a quantity it could itself compute already. In $\mathrm{F{\scriptstyle IN}}$ (for example Figure 3.19) we assume the query is non-trivial, meaning lines 6,7 of Figure 3.4 return true, and these lines are thus omitted. We start with games $\mathrm{Gm}_0, \mathrm{Gm}_1$

INIT:

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

$\text{NS}(k, \boldsymbol{pk}, m)$:

$u \leftarrow u+1$; $k_u \leftarrow k$; $\boldsymbol{pk}[1] \leftarrow pk$; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$; $m_u \leftarrow m$; $n_u \leftarrow |\boldsymbol{pk}|$

$t_{u,1} \leftarrow\!\!\$\ \{0,1\}^{\ell}$; Return $t_{u,1}$

$\text{SIGN}_1(s, \boldsymbol{R})$:

$k \leftarrow k_s$; $\boldsymbol{R}[k] \leftarrow R_{s,k}$

For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$

If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\perp$ else Return $z_{s,k}$

$\text{SIGN}_0(s, \boldsymbol{t})$:

$k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$

$c_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$; $z_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$

For $i = 1, \ldots, n_s$ do

    If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$

    Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\ \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$

$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}_s^*[i]$

$\mathrm{HF}_1[(k, R_s, \boldsymbol{pk}_s, m_s)] \leftarrow c_{s,k}$; Return $R_{s,k}$

$\mathrm{H}_0(x)$:

If $(\mathrm{HF}_0[x] \neq \perp)$ then Return $\mathrm{HF}_0[x]$

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

$\mathrm{H}_1(x)$:

If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$

$(R, apk, m) \leftarrow x$; $\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$

$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$; Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$:

If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$

$(k, \boldsymbol{pk}) \leftarrow x$; For $i = 1, \ldots, |\boldsymbol{pk}|$ do $\mathrm{HF}_2[(i, \boldsymbol{pk})] \leftarrow e_i \leftarrow\!\!\$\ \mathbb{Z}_p$

$apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i}$; For $y \in \mathrm{TV}[apk]$ do $\mathrm{HF}_1[y] \leftarrow \perp$

Return $\mathrm{HF}_2[x]$

$\text{FIN}(\boldsymbol{pk}, m, (R, z))$:

For $i = 1, \ldots, |\boldsymbol{pk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$; $e_i \leftarrow \mathrm{H}_2((i, \boldsymbol{pk}))$

$X \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i \cdot c_i}$; Return $(g^z = RX)$

Figure 3.18: Game $\mathrm{Gm}_{\mathrm{simp}}$ for proof of Theorem 3.6.1.

$\textsc{Init}$: **/** Games Gm_0–Gm_9

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

$\mathrm{NS}(k, \boldsymbol{pk}, m)$: **/** Games $\boxed{\mathrm{Gm}_0}$, Gm_1

$u \leftarrow u+1$; $k_u \leftarrow k$; $\boldsymbol{pk}[k] \leftarrow pk$; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$

$m_u \leftarrow m$; $n_u \leftarrow |\boldsymbol{pk}|$; $\mathsf{CommitStage}_u \leftarrow \mathsf{true}$

$r_{u,k} \leftarrow\!\!\$\ \mathbb{Z}_p$; $R_{u,k} \leftarrow g^{r_{u,k}}$; $t_{u,k} \leftarrow\!\!\$\ \{0,1\}^{\ell}$

If ($\exists u' < u : R_{u,k_u} = R_{u',k_{u'}}$) then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{t_{u,k_u} \leftarrow t_{u',k_{u'}}}$

If ($\mathrm{HF}_0[(k, R_{u,k})] \neq \perp$) then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{t_{u,k} \leftarrow \mathrm{HF}_0[(k, R_{u,k})]}$

Return $t_{u,k}$

$\textsc{Sign}_0(s, \boldsymbol{t})$: **/** Games $\mathrm{Gm}_0, \mathrm{Gm}_1$

$\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $\mathsf{CommitStage}_s \leftarrow \mathsf{false}$

$\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$; Return $R_{s,k}$

$\textsc{Sign}_1(s, \boldsymbol{R})$: **/** Games $\mathrm{Gm}_0, \mathrm{Gm}_1, \mathrm{Gm}_2$

$\boldsymbol{R}[k] \leftarrow R_{s,k}$

For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$

If ($\exists i : y_i \neq \boldsymbol{t}_s[i]$) then Return $\perp$

$R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$

Return $z_{s,k}$

$\mathrm{H}_0(x)$: **/** Games $\boxed{\mathrm{Gm}_0}$, Gm_1

If ($\mathrm{HF}_0[x] \neq \perp$) then Return $\mathrm{HF}_0[x]$

$\mathrm{HF}_0[x] \leftarrow\!\!\$\ \{0,1\}^{\ell}$; If ($\exists u' : x = (k_{u'}, R_{u',k_{u'}})$ and $\mathsf{CommitStage}_{u'}$) then

  $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathrm{HF}_0[x] \leftarrow t_{u',k_{u'}}}$

Return $\mathrm{HF}_0[x]$

$\mathrm{H}_1(x)$: **/** Games Gm_0–Gm_7

If ($\mathrm{HF}_1[x] \neq \perp$) then Return $\mathrm{HF}_1[x]$

$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$; Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$: **/** Games Gm_0–Gm_7

If ($\mathrm{HF}_2[x] \neq \perp$) then Return $\mathrm{HF}_1[x]$

$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$; Return $\mathrm{HF}_1[x]$

$\textsc{Fin}(k, \boldsymbol{pk}, m, (R, z))$: **/** Games Gm_0–Gm_9

For $i = 1, \ldots, |\boldsymbol{pk}|$ do $c_i \leftarrow \mathrm{H}_1((i, R, \boldsymbol{pk}, m))$; $e_i \leftarrow \mathrm{H}_2((i, \boldsymbol{pk}))$

$X \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i \cdot c_i}$; Return $(g^z = RX)$

Figure 3.19: Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ for proof of Theorem 3.6.1. Some procedures will be included in later games, as indicated. A box around the name of a game following an oracle means the boxed code in that oracle is included in the game.

in Figure 3.19. Game Gm_0 includes the boxed code, and we claim that

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\mathrm{Gm}_0(\mathcal{A})] \,. \tag{3.47}$$

Games $\mathrm{Gm}_0, \mathrm{Gm}_1$ are identical-until-bad, so by the Fundamental Lemma of Game Playing [19]

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

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

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

Game Gm_2 changes the $\mathrm{NS}, \textsc{Sign}_0, \mathrm{H}_0$ oracles as shown in Figure 3.20, maintaining the other oracles of Gm_1 from Figure 3.19. It drops redundant code, which allows it to move the choice of $R_{s,1}$ to line 29. At line 31, it also introduces a table HI to maintain an inverse of the hash function, but does not yet use this. We have

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

Game Gm_3 (oracles shown across Figures 3.20 and 3.19) aims to figure out the $R_{s,j}$-values of parties $j \neq k$ before having to supply $R_{s,k}$, because we will later need these to program 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 37. (Not finding the inverses is not yet a bad event. It can happen with high probability. It becomes a bad event only at line 37 when the BN-commitments are verified.) The computation of t at that line is only to ensure that H_0 has been called; this variable will not be

$\mathrm{NS}(k, \boldsymbol{pk}, m)$: **/** Games Gm_2–Gm_9

25 $u \leftarrow u+1$; $k_u \leftarrow k$; $\boldsymbol{pk}[.] \leftarrow pk$; $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$; $m_u \leftarrow m$; $n_u \leftarrow |\boldsymbol{pk}|$

26 $t_{u,k} \leftarrow\!\!\$\ \{0,1\}^{\ell}$; Return $t_{u,k}$

$\mathrm{SIGN}_0(s, \boldsymbol{t})$: **/** Game Gm_2

27 $k \leftarrow k_s$; $\boldsymbol{t}[.] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $r_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{r_{s,k}}$; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$

28 Return $R_{s,k}$

$\mathrm{SIGN}_0(s, \boldsymbol{t})$: **/** Games $\mathrm{Gm}_3, \mathrm{Gm}_4$

29 $k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $r_{s,k} \leftarrow\!\!\$\ \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{r_{s,k}}$; $\mathrm{HF}_0[(k, R_{s,k})] \leftarrow t_{s,k}$

30 For $i = 1, \ldots, n_s$ do

31   If $(\mathrm{HI}_0[i, \boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}_s^*[i] \leftarrow \mathrm{HI}_0[i, \boldsymbol{t}_s[i]]$

32   Else $\boldsymbol{R}_s^*[i] \leftarrow\!\!\$\ \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i, \boldsymbol{R}_s^*[i]))$

33 Return $R_{s,k}$

$\mathrm{SIGN}_1(s, \boldsymbol{R})$: **/** Games Gm_3, $\boxed{\mathrm{Gm}_4}$

34 $\boldsymbol{R}[k] \leftarrow R_{s,k}$

35 For $i = 1, \ldots, n_s$ do $y_i \leftarrow \mathrm{H}_0((i, \boldsymbol{R}[i]))$

36 If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\perp$

37 If $(\boldsymbol{R} \neq \boldsymbol{R}_s^*)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\boldsymbol{R} \leftarrow \boldsymbol{R}_s^*}$

38 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}[i]$; $c_{s,k} \leftarrow \mathrm{H}_1((k, R_s, \boldsymbol{pk}_s, m_s))$; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$

39 Return $z_{s,k}$

$\mathrm{H}_0(x)$: **/** Games Gm_2–Gm_9

40 If $(\mathrm{HF}_0[x] \neq \perp)$ then Return $\mathrm{HF}_0[x]$

41 $\mathrm{HF}_0[x] \leftarrow\!\!\$\ \{0,1\}^{\ell}$; $(i, R) \leftarrow x$; $\mathrm{HI}_0[i, \mathrm{HF}_0[x]] \leftarrow R$; Return $\mathrm{HF}_0[x]$

Figure 3.20: Games for proof of Theorem 3.6.1.

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 32,33 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

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

$\textsc{Sign}_0(s,\boldsymbol{t})$: **/** Game Gm_5

42 $k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$; $r_{s,k} \leftarrow\!\!{\scriptstyle\$}\, \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{r_{s,k}}$; $\mathrm{HF}_0[(k,R_{s,k})] \leftarrow t_{s,k}$

43 For $i = 1,\ldots,n_s$ do

44   If $(\mathrm{HI}_0[i,\boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}^*_s[i] \leftarrow \mathrm{HI}_0[i,\boldsymbol{t}_s[i]]$

45   Else $\boldsymbol{R}^*_s[i] \leftarrow\!\!{\scriptstyle\$}\, \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i,\boldsymbol{R}^*_s[i]))$

46 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}^*_s[i]$; $c_{s,k} \leftarrow \mathrm{H}_1((k,R_s,\boldsymbol{pk}_s,m_s))$; $z_{s,k} \leftarrow sk \cdot c_{s,k} + r_{s,k}$

47 Return $R_{s,k}$

$\textsc{Sign}_1(s,\boldsymbol{R})$: **/** Game Gm_5–Gm_9

48 $k \leftarrow k_s$; $\boldsymbol{R}[k] \leftarrow R_{s,k}$

49 For $i = 1,\ldots,n_s$ do $y_i \leftarrow \mathrm{H}_0((i,\boldsymbol{R}[i]))$

50 If $(\exists i : y_i \neq \boldsymbol{t}_s[i])$ then Return $\perp$ else Return $z_{s,k}$

$\textsc{Sign}_0(s,\boldsymbol{t})$: **/** Game $\boxed{\mathrm{Gm}_6}$, Gm_7–Gm_9

51 $k \leftarrow k_s$; $\boldsymbol{t}[k] \leftarrow t_{s,k}$; $\boldsymbol{t}_s \leftarrow \boldsymbol{t}$

52 $c_{s,k} \leftarrow\!\!{\scriptstyle\$}\, \mathbb{Z}_p$; $z_{s,k} \leftarrow\!\!{\scriptstyle\$}\, \mathbb{Z}_p$; $R_{s,k} \leftarrow g^{z_{s,k}} pk^{-c_{s,k}}$; $\mathrm{HF}_0[(k,R_{s,k})] \leftarrow t_{s,k}$

53 For $i = 1,\ldots,n_s$ do

54   If $(\mathrm{HI}_0[i,\boldsymbol{t}_s[i]] \neq \perp)$ then $\boldsymbol{R}^*_s[i] \leftarrow \mathrm{HI}_0[i,\boldsymbol{t}_s[i]]$

55   Else $\boldsymbol{R}^*_s[i] \leftarrow\!\!{\scriptstyle\$}\, \mathbb{G}$; $t \leftarrow \mathrm{H}_0((i,\boldsymbol{R}^*_s[i]))$

56 $R_s \leftarrow \prod_{i=1}^{n_s} \boldsymbol{R}^*_s[i]$

57 If $(\mathrm{HF}_1((k,R_s,\boldsymbol{pk}_s,m_s)) \neq \perp)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{c_{s,k} \leftarrow \mathrm{HF}_1[(k,R_s,\boldsymbol{pk}_s,m_s)]}$

58 $\mathrm{HF}_1[(k,R_s,\boldsymbol{pk}_s,m_s)] \leftarrow c_{s,k}$; Return $R_{s,k}$

Figure 3.21: Games for proof of Theorem 3.6.1.

Line 38 can only set bad if $y_i = \boldsymbol{t}_s[i]$ for all i, due to line 37. So it is set only if there is a collision in H_0-values, or no query hashing to $\boldsymbol{t}_s[i]$ was made prior to the latter being provided, but is made later. Thus

$$\Pr[\mathrm{Gm}_3(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_0^2 + nq_0}{2^\ell} \,. \tag{3.48}$$

In game Gm_4, the $\boldsymbol{R}$ queried to $\textsc{Sign}_1$ is the same as the $\boldsymbol{R}^*$ determined in $\textsc{Sign}_0$, allowing game Gm_5 (Figure 3.21) to move line 38 into $\textsc{Sign}_0$ as line 46 and to simplify $\textsc{Sign}_1$. We have

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

$\mathrm{H}_1(x)$: **/** Game $\mathrm{Gm}_8, \mathrm{Gm}_9$

If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$

$(R, apk, m) \leftarrow x$; $\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$

$\mathrm{HF}_1[x] \leftarrow\!\!\$\ \mathbb{Z}_p$; Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$: **/** Game Gm_8, $\boxed{\mathrm{Gm}_9}$

If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$

$(\cdot, \boldsymbol{pk}) \leftarrow x$; For $i = 1, \ldots, |\boldsymbol{pk}|$ do $\mathrm{HF}_2[(i, \boldsymbol{pk})] \leftarrow e_i \leftarrow\!\!\$\ \mathbb{Z}_p$

$apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{e_i}$

If $\mathrm{TV}[apk] \neq \perp$ then

  $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\text{For } y \in \mathrm{TV}[apk] \text{ do } \mathrm{HF}_1[y] \leftarrow \perp}$

Return $\mathrm{HF}_2[x]$

Figure 3.22: Games for proof of Theorem 3.6.1.

Now that R_s is determined prior to the release of R_{s,k_s}, it becomes possible to successfully program H_1 via the zero-knowledge simulation. Game Gm_6 of Figure 3.21 does this, setting bad at line 57 if the programming was precluded by the hash value already being defined, and including the boxed code to correct. We have

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

Games $\mathrm{Gm}_6, \mathrm{Gm}_7$ (Figure 3.21) are identical-until-bad, so

$$\Pr[\mathrm{Gm}_6(\mathcal{A})] \leq \Pr[\mathrm{Gm}_7(\mathcal{A})] + \Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets } \mathsf{bad}] \ . \tag{3.49}$$

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

$$\Pr[\mathrm{Gm}_7(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_s q_1}{p} \ . \tag{3.50}$$

Moving on, let us consider games Gm_8 and Gm_9 in Fig. 3.22, which differ from Gm_7 in modifications to oracles H_1 and H_2. Oracle H_1 now keeps track of a table TV, that stores for each aggregate key apk the set of H_1 queries that contain it. It otherwise behave identically to $\mathrm{Gm}_7.\mathrm{H}_1$.

Adversary $\mathcal{A}_{\mathrm{xidl}}^{\mathrm{CH}}(pk)$:

$(\boldsymbol{pk}, m, (R, z)) \leftarrow\!\!\$\; \mathcal{A}^{\mathrm{NS},\mathrm{SIGN}_0,\mathrm{SIGN}_1,\mathrm{H}_0,\mathrm{H}_1,\mathrm{H}_2}(pk)$

$apk \leftarrow \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$; Return $(\mathrm{TI}[(apk, R, m)], z)$

$\mathrm{H}_1(x)$:

If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$

$(R, apk, m) \leftarrow x$; $\mathrm{TV}[apk] \leftarrow \mathrm{TV} \cup \{x\}$

If $(\mathrm{TJ}[apk] = \perp)$ then Return $\mathrm{HF}_1[x] \leftarrow\!\!\$\; \mathbb{Z}_p$

$\iota \leftarrow \iota + 1$; $\mathrm{TI}[x] \leftarrow \iota$

$\mathrm{HF}_1[x] \leftarrow c_\iota \leftarrow\!\!\$\; \mathrm{CH}(\mathrm{TJ}[apk], R)$; Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$:

If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$

$(\cdot, \boldsymbol{pk}) \leftarrow x$; If $(pk \notin \boldsymbol{pk})$ then Return $\mathrm{HF}_2[x] \leftarrow\!\!\$\; \mathbb{Z}_p$

$j \leftarrow j + 1$; $k \leftarrow \mathsf{minInd}(pk, \boldsymbol{pk})$; If $(x \neq (k, \boldsymbol{pk}))$ then Return $\mathrm{HF}_2[x] \leftarrow\!\!\$\; \mathbb{Z}_p$

$S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$

$\mathrm{HF}_2[x] \leftarrow e_j \leftarrow \mathrm{NWTAR}(S)$; $apk \leftarrow S \cdot pk^{e_j}$; $\mathrm{TJ}[apk] \leftarrow j$

For $y \in \mathrm{TV}[apk]$ do $\mathrm{HF}_1[y] \leftarrow \perp$

Return $\mathrm{HF}_2[x]$

Figure 3.23: Adversary $\mathcal{A}_{\mathrm{xidl}}$ for Theorem 3.6.1. Oracles $\mathrm{NS}, \mathrm{SIGN}_0, \mathrm{SIGN}_1, \mathrm{H}_0$ are copied from game $\mathrm{Gm}_{\mathrm{simp}}$ (Fig. 3.18).

Oracle $\mathrm{Gm}_8.\mathrm{H}_2$ does not contain the boxed code, which makes the oracle behave identically to $\mathrm{Gm}_7.\mathrm{H}_2$. So, we have

$$\Pr[\mathrm{Gm}_7(\mathcal{A})] = \Pr[\mathrm{Gm}_8(\mathcal{A})] \,. \tag{3.51}$$

By construction, Gm_7 and Gm_8 are identical-until-bad, hence

$$\Pr[\mathrm{Gm}_8(\mathcal{A})] \leq \Pr[\mathrm{Gm}_9(\mathcal{A})] + \Pr[\mathrm{Gm}_8 \text{ sets } \mathsf{bad}] \tag{3.52}$$

$$\leq \Pr[\mathrm{Gm}_9(\mathcal{A})] + \frac{q_1 q_2}{p} \,, \tag{3.53}$$

where the last inequality is by the fact that each H_2 query has probability at most q_1/p of setting bad. Lastly, we note that Gm_9 and $\mathrm{Gm}_{\mathrm{simp}}$ are identical. This completes the proof of Lemma 3.15.1. ▮

Proof of Lemma 3.15.2: Consider $\mathcal{A}_{\text{xidl}}$ in Figure 3.23. It forwards the public key pk to $\mathcal{A}_{\text{ms}}$. Simulating signatures without knowing the secret key can be done exactly as Gm_{simp}. To break $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$, our adversary $\mathcal{A}_{\text{xidl}}$ needs to program H_1 and H_2. For each H_2 query, Line 10 to 12 programs the response e_j for the target public key by first deriving commitment $S = \prod_{i \neq k} \boldsymbol{pk}[i]^{e_i}$, which is then submitted to NwTar to derive e_k that is returned as the response. By construction, the corresponding aggregate public key $apk = S \cdot pk^{e_k}$ is exactly the target T_j recorded by $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$ for this NwTar query. For each H_1 query, our adversary first uses the aggregate public key apk find the corresponding H_2 query via table TJ. If possible, then the adversary proceeds to program in a challenge using the challenge oracle Ch of XIDL. If this is not possible, the advesary simply simulates H_1 honestly. If a forgery $(\boldsymbol{pk}, m, (R, z))$ is valid, then it must be that

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

where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\text{H}_2((i,\boldsymbol{pk}))}$. Observe that call involving a fresh vector $\boldsymbol{pk}$ to oracle H_2 erases the table HF_1 at every entry associated with the derived apk. Hence, our adversary can use the above relation to directly break XIDL. In other words, the value of z included in the forgery makes the following equation true in game $\text{Gm}^{\text{xidl}}_{\mathbb{G},g,q_2,q_1}$, $g^z = R \cdot T_j^{c_i}$, where $j = \text{TJ}[apk]$ and $i = \text{TI}[(R, apk, m)]$. This justifies Equation (3.46). ▮

3.16 Proof of Theorem 3.7.1

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

Lemma 3.16.1 *Let $\mathbb{G}$ be a group of prime order p with generator g. Let $\mathsf{MS} = \mathsf{HBMS}[\mathbb{G}, g]$ be the scheme specified in Fig. 3.8. Let $\mathcal{A}_{\mathrm{ms}}$ be an adversary for game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ of Fig. 3.4. Assume the execution of game $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$ with $\mathcal{A}_{\mathrm{ms}}$ has at most q_0, q_1, q_2 distinct queries to $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ respectively. Let $\rho \in [0,1]$ be a real number. Consider games Gm_0 and $\mathrm{Gm}_{1,\rho}$ give in Fig. 3.24. Then,*

$$\mathbf{Adv}^{\text{ms-uf}}_{\mathsf{MS}}(\mathcal{A}_{\mathrm{ms}}) = \Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})] \tag{3.54}$$

$$= \Pr[\mathrm{Gm}_{1,\rho}(\mathcal{A}_{\mathrm{ms}}) \mid \mathrm{Gm}_{1,\rho}(\mathcal{A}_{\mathrm{ms}}) \textit{ does not abort}\,] \,. \tag{3.55}$$

Moreover, the probability that game Gm_1 does not abort is

$$\Pr[\mathrm{Gm}_{1,\rho}(\mathcal{A}_{\mathrm{ms}}) \textit{ does not abort}\,] = \rho^{q_0} \,, \tag{3.56}$$

which is 1 if $\rho = 1$.

Proof of Lemma 3.16.1: Consider games Gm_0 and $\mathrm{Gm}_{1,\rho}$ given in Fig. 3.24. Game Gm_0 is simply a rewrite of $\mathbf{G}^{\text{ms-uf}}_{\mathsf{MS}}$, where $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ are lazily sampled. We fix the given adversary $\mathcal{A}_{\mathrm{ms}}$ for the rest of the proof and omit writing it in expression such as $\Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})]$ for simplicity. Game $\mathrm{Gm}_{1,\rho}$ is parameterized by a real number $\rho \in [0,1]$, and changes the code of NS, $\textsc{Sign}_1$ and H_0. The changes are made so that $\textsc{Sign}_1$ does not use the secret key sk, but will however preserve the output distribution of all oracles when it does not abort, as we will show below. In particular, for each H_0 query, game Gm_1 makes a guess, by flipping a biased coin $\mathsf{Coin}(\rho)$, which has probability ρ of returning 1 and probability $1-\rho$ of returning 0. If the coin flip returns 1, then we set the output of $\mathrm{H}_0(x)$ to be $g^{\beta_g} pk^{\beta_{pk}}$, otherwise we set the output of $\mathrm{H}_0(x)$ to be g^{β_g}. In either case, β_g and β_{pk} are uniformly chosen at random as per line 25.

Looking ahead, $\mathrm{Gm}_{1,\rho}$ will be able to simulate signatures for $\boldsymbol{pk}, m$ when $\mathrm{H}_0(\boldsymbol{pk}, m)$ is set to $g^{\beta_g} pk^{\beta_{pk}}$ (when the coin toss returns 1). In fact, ρ is set to 1 in deriving the AGM result

Game Gm_0, $\mathrm{Gm}_{1,\rho}$, $\mathrm{Gm}_{2,\rho}$

INIT:

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

NS$(k, \boldsymbol{pk}, m)$:

2 $\boldsymbol{pk}[k] \leftarrow pk$; $u \leftarrow u+1$

3 $k_u \leftarrow k$; $m_u \leftarrow m$

4 $\boldsymbol{pk}_u \leftarrow \boldsymbol{pk}$; $h \leftarrow\!\!\$\ \mathrm{H}_0((\boldsymbol{pk}, m))$

5 $apk_u \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$

6 $a_u, b_u \leftarrow\!\!\$\ \mathbb{Z}_p$; $T_{u,k} \leftarrow g^{a_u} h^{b_u}$

7 Return $T_{u,k}$

SIGN$_1(v, \mathbf{in})$:

8 $(T_{v,1}, \ldots, T_{v,n}) \leftarrow \mathbf{in}$; $T_v \leftarrow \prod_{i=1}^n T_{v,i}$

9 $c_v \leftarrow \mathrm{H}_1((T_v, apk_v, m_v))$

10 $e_v \leftarrow \mathrm{H}_2((k_v, \boldsymbol{pk}))$

11 <u>Gm_0:</u>

12 $\quad z_v \leftarrow a_v + sk \cdot e_v \cdot c_v \mod p$

13 $\quad s_v \leftarrow b_v$

14 <u>$\mathrm{Gm}_{1,\rho}, \mathrm{Gm}_{2,\rho}$:</u>

15 $\quad (w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[(\boldsymbol{pk}_v, m_v)]$

16 $\quad$ If $(w \neq pk)$ then abort

17 $\quad s_v \leftarrow b_v + e_v \cdot c_v \cdot \beta_{pk}^{-1} \mod p$

18 $\quad z_v \leftarrow a_v + \beta_g \cdot b_v - \beta_g \cdot s_v \mod p$

19 Return (s_v, z_v)

SIGN$_2(v, \mathbf{in})$:

20 $(t_1, \ldots, t_n) \leftarrow \mathbf{in}$; $t \leftarrow \sum_i t_i$

21 $(s, z) \leftarrow t$; Return (T_v, s, z)

$\mathrm{H}_0(x)$: // Gm_0

22 If $\mathrm{HF}_0[x] = \bot$ then $\mathrm{HF}_0[x] \leftarrow\!\!\$\ \mathbb{G}$

23 Return $\mathrm{HF}_0[x]$

$\mathrm{H}_0(x)$: // $\mathrm{Gm}_{1,\rho}, \mathrm{Gm}_{2,\rho}$

24 If $\mathrm{HF}_0[x] \neq \bot$ then Return $\mathrm{HF}_0[x]$

25 $\beta_g \leftarrow\!\!\$\ \mathbb{Z}_p$; $\beta_{pk} \leftarrow\!\!\$\ \mathbb{Z}_p^*$

26 If $(\mathsf{Coin}(\rho) = 1)$ then

27 $\quad \mathrm{HF}_0[x] \leftarrow g^{\beta_g} pk^{\beta_{pk}}$

28 $\quad \mathrm{TH}[x] \leftarrow (pk, \beta_g, \beta_{pk})$

29 Else

30 $\quad \mathrm{HF}_0[x] \leftarrow g^{\beta_g}$

31 $\quad \mathrm{TH}[x] \leftarrow (g, \beta_g, \beta_{pk})$

32 Return $\mathrm{HF}_0[x]$

$\mathrm{H}_i(x)$: // $i \in \{1, 2\}$

33 If $(\mathrm{HF}_i[x] = \bot)$ then $\mathrm{HF}_i[x] \leftarrow\!\!\$\ \mathbb{Z}_p$

34 Return $\mathrm{HF}_i[x]$

FIN$(\boldsymbol{pk}, m, (T, s, z))$:

35 If $(\boldsymbol{pk}[k] \neq pk)$ then return false

36 If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return false

37 $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$

38 <u>$\mathrm{Gm}_{2,\rho}$:</u>

39 $\quad (w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$

40 $\quad$ If $(w \neq g)$ then abort

41 $(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$

42 $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$

43 $c \leftarrow \mathrm{H}_1((T, apk, m))$

44 Return $(g^z h^s = T \cdot apk^c)$

Figure 3.24: Games Gm_0, $\mathrm{Gm}_{1,\rho}$, and $\mathrm{Gm}_{2,\rho}$, where $\rho \in [0,1]$ is a real number, used in Lemma 3.16.1 and proof of Theorem 3.7.2. Notation $\mathsf{Coin}(\rho)$ denotes flipping of a biased coin with probability ρ of giving 1 and $1-\rho$ of giving 0.

and the coin toss never returns 0. However, for the standard model result, we will need to make sure that the H_0 query corresponding to the forgery pk, m is programmed differently, namely that $\mathrm{H}_0((\boldsymbol{pk}, m)) = g^{\beta_g}$.

Game $\mathrm{Gm}_{1,\rho}$ could abort at line 16 (it is assumed that the adversary losses the game if

Gm_1 is aborted). By construction, we have

$$\Pr[\mathrm{Gm}_1 \text{ does not abort}] = \rho^{q_0} . \tag{3.57}$$

We claim that, for any value of ρ, if game Gm_1 does not abort, then it is indistinguishable from Gm_0 to the adversary. In particular, we claim

$$\Pr[\mathrm{Gm}_1 \mid \mathrm{Gm}_1 \text{ does not abort}] = \Pr[\mathrm{Gm}_0] . \tag{3.58}$$

Showing this amounts to showing that the outputs of $\textsc{Sign}_1$ oracle in either games are distributed identically. Observe that, in game Gm_0, the return value T_v of NS and (s_v, z_v) of $\textsc{Sign}_1$ are uniformly distributed subjected to the constraint that

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

We will show that this is also true in $\mathrm{Gm}_{1,\rho}$, namely that $\textsc{Sign}_0$ and $\textsc{Sign}_1$ in $\mathrm{Gm}_{1,\rho}$ also returns $T_{v,k}$ and (s_v, z_v) that are uniformly distributed subjected to the above equation. In game $\mathrm{Gm}_{1,\rho}$, if $w = pk$ at line 15, then $h = \mathrm{H}_0((\boldsymbol{pk}_v, m)) = g^{\beta_g} pk^{\beta_{pk}}$, by construction of H_0 (line 27). Hence, for a query $\textsc{Sign}_1(v, (T_{v,1}, \ldots, T_{v,n}))$ of game $\mathrm{Gm}_{1,\rho}$, it holds that

$$\begin{aligned} T_{v,k_v} \cdot pk^{e_v c_v} &= g^{a_v} \cdot h^{b_v} \cdot pk^{e_v c_v} = g^{a_v} \cdot (g^{\beta_g} pk^{\beta_{pk}})^{b_v} \cdot pk^{e_v c_v} \\ &= g^{a_v + \beta_g \cdot b_v} \cdot pk^{\beta_{pk} \cdot b_v + e_v c_v} . \end{aligned}$$

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

Adversary $\mathcal{A}_{\mathrm{dl}}(X)$:

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

Figure 3.25: Adversary $\mathcal{A}_{\mathrm{dl}}$ for Theorem 3.7.1, oracles $\mathrm{NS}, \mathrm{SIGN}_1, \mathrm{SIGN}_2, \mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2$ are implemented using the exact code as those in $\mathrm{Gm}_{1,1}$. Notation $\mathsf{Ext}(\cdot, g)$ and $\mathsf{Ext}(\cdot, X)$ are defined in the proof of Lemma 3.7.2. Computation of α_g and α_X are done modulo p.

make this true. To verify this, check that

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

Additionally, notice that s_v, z_v are both marginally uniform over $\mathbb{Z}_p$ by construction. This means the outputs of $\mathrm{SIGN}_0, \mathrm{SIGN}_1$ oracle from $\mathrm{Gm}_{1,\rho}$ has the same output distribution compared to that of Gm_0. This justifies Equation (3.58). ▮

Equipped with Lemma 3.16.1, we move on to prove Theorem 3.7.1. The proof constructs adversary $\mathcal{A}_{\mathrm{dl}}$ that simulates $\mathrm{Gm}_{1,1}$ (with ρ set to 1).

Proof of Theorem 3.7.1: Consider the games Gm_0 and $\mathrm{Gm}_{1,1}$ (with $\rho = 1$) in Fig. 3.24. We know that,

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

Moreover,

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

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

We first analyze the group elements involved in the inputs and outputs of oracles of $\mathrm{Gm}_{1,1}$. The u-th NS query takes in a list of group elements $\boldsymbol{pk}_u$. The v-th Sign_1 query takes in a list of group elements $(T_{v,1},\ldots,T_{v,n})$. The i-th H_2 query take in a list of group elements $\boldsymbol{pk}_{\mathrm{H}_2,i}$. The i-th H_1 query (T, apk, m) takes in group elements $T_{\mathrm{H}_1,i}$ and $apk_{\mathrm{H}_1,i}$. Above are the exhaustive list of group elements that are given to $\mathrm{Gm}_{1,1}$, let us denote this list by **out**, since they are the output of the adversary. The initial query to INIT outputs a group element pk. The u-th NS query gives out a group element T_{u,k_u}. The i-th H_0 query gives out a group element h_i. The last query to FIN gives group elements T (first component of the forged signature) and pk. Above (plus the group generator g) are the exhaustive list of group elements that are given out to the adversary $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$. Let us denote this list as **in**. Hence, the algebraic adversary $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$ gives, for each group element in the list **out**, a vector that is of dimension $|\mathbf{in}|$ which is a valid representation of the corresponding group element. Note that every group element in the list **in** is derived using only group operations on two group elements: g and pk (this is by the construction of game $\mathrm{Gm}_{1,1}$). As a result, every group element in the list **out** can be represent using g and pk only. For any $Y \in \mathbf{out}$, we use $\mathsf{Ext}(Y,g)$ and $\mathsf{Ext}(Y,pk)$ to denote this representation, i.e.

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

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

We move on to giving adversary $\mathcal{A}_{\mathrm{dl}}$, which simulates $\mathrm{Gm}_{1,1}$ for $\mathcal{A}_{\mathrm{ms}}^{\mathrm{alg}}$. Our adversary $\mathcal{A}_{\mathrm{dl}}$ is given in Fig. 3.25. Our adversary $\mathcal{A}_{\text{dl}}$ simulates oracles $\mathrm{NS}, \textsc{SignStage}_1, \textsc{SignStage}_2, \mathrm{H}_0, \mathrm{H}_1$ exactly as $\mathrm{Gm}_{1,1}$, hence their code are omitted. As stated above, since $\mathcal{A}_{\text{dl}}$ simulates $\mathrm{Gm}_{1,1}$, the representation of any group element $Y \in \mathbf{out}$ are available via scalars $\mathsf{Ext}(Y, g)$ and $\mathsf{Ext}(g, pk)$. Our adversary uses these scalars to compute the discrete log x'.

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

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

where $apk = \prod_{i=1}^{|\boldsymbol{pk}|} \boldsymbol{pk}[i]^{\mathrm{H}_2((i, \boldsymbol{pk}))}$. Since every group element in the above equation can be represented using g and X, one can solve for $\mathsf{DL}_{\mathbb{G},g}(X)$. Our adversary $\mathcal{A}_{\text{dl}}$ implements this intuition, computing value α_g and α_X (line 7 and 8) such that $g^{\alpha_g} = X^{\alpha_X}$. The only caveat is that α_X could be 0, in which case $\mathsf{DL}_{\mathbb{G},g}(X)$ cannot be solved for. When $\alpha_X = 0$ adversary $\mathcal{A}_{\text{dl}}$ sets bad, and we would like to upperbound the probability of this event. First, note that the view of adversary $\mathcal{A}_{\text{ms}}$ is independent of the value of β_{pk}. This is because the adversary is only given the value of $h = g^{\beta_g} pk^{\beta_{pk}}$. So, if the forgery is such that $s \neq 0$, then $\alpha_X = 0$ with probability at most $1/p$. If $s = 0$, then we need to make sure that $\mathsf{Ext}(T, X) + c \cdot (e_k + \sum_{i \neq k} \mathsf{Ext}(\boldsymbol{pk}[i], X) \cdot e_i)$ is not zero. We first bound the probability that there exists some query $\mathrm{H}_2((\cdot, \boldsymbol{pk}'))$ (which defines the values of $e'_1, \ldots, e'_{|\boldsymbol{pk}'|}$) such that $e'_k + \sum_{i \neq k} \mathsf{Ext}(\boldsymbol{pk}'[i], X) \cdot e'_i = 0$ (call this quantity $\gamma_{\boldsymbol{pk}'}$). This happens with probability at most q_2/p. Suppose the above does not happen, then for each query $\mathrm{H}_1((T', apk', m'))$ (which defines the value of c'), where apk' is the aggregate key of some vector $\boldsymbol{pk}'$, the probability that $\mathsf{Ext}(T', X) + c' \cdot \gamma_{\boldsymbol{pk}'} = 0$ is at most q_2/p, accounting for at most q_2 non-zero values that $\gamma_{\boldsymbol{pk}'}$ could take. This results in an overall bad probability of $q_2/p + q_1 q_2/p = (q_1 + 1) q_2/p$. This justifies Equation (3.7). ∎

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

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

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

$\mathrm{F{\scriptsize IN}}(\boldsymbol{pk}, m, (T, s, z))$: **/** Game Gm_3, $\boxed{\mathrm{Gm}_4}$
If $(\boldsymbol{pk}[k] \neq pk)$ then return false
If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return false
$(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$; If $(w \neq g)$ then abort
$(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$; $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i, \boldsymbol{pk}))}$
If $((T, apk, m) \in \mathbf{BadSet})$ then bad $\leftarrow$ true ; $\boxed{\mathrm{HF}_1[(T, apk, m)] \leftarrow \bot}$
$c \leftarrow \mathrm{H}_1((T, apk, m))$; $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$
Return $(g^z h^s = T \cdot apk^c)$

Figure 3.26: Games Gm_3 and Gm_4 for proof of Theorem 3.7.2. Oracles Init, NS, $\mathrm{S{\scriptsize IGN}}_1$, $\mathrm{S{\scriptsize IGN}}_2$, and H_0 are the same as those in $\mathrm{Gm}_{2,\rho}$. Parameter ρ is set to $(1 - (1 + q_s)^{-1})$ in oracle H_0.

3.17 Proof of Theorem 3.7.2

Proof of Theorem 3.7.2: We will start by considering $\mathrm{Gm}_{1,\rho}$ given in Fig. 3.24. By Lemma 3.16.1,

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

Towards construction of an adversary against XIDL, consider game $\mathrm{Gm}_{2,\rho}$ (Fig. 3.24), differ from $\mathrm{Gm}_{1,\rho}$ only at line 40–it aborts if the coin flip corresponding to the forgery target $(\boldsymbol{pk}, m)$ results in $w = g$. Marginally, $\mathrm{Gm}_{2,\rho}$ does not abort at line 40 with probability $(1 - \rho)$. We need to lower bound the probability of $\mathrm{Gm}_{2,\rho}$ not aborting overall, at either line 16 or line 40. Since there are overall q_s *unique* queries to NS in the execution of Gm_0 with $\mathcal{A}_{\text{ms}}$, then the probability that Gm_1

Adversary $\mathcal{A}_{\mathrm{xidl}}^{\mathrm{NwTar,Ch,Fin}}(X)$:

$pk \leftarrow X$; $(k, \boldsymbol{pk}, m, \sigma) \leftarrow\$ \mathcal{A}_{\mathrm{ms}}^{\mathrm{NS,Sign_1,Sign_2,H_0,H_1,H_2}}(pk)$
If $(\boldsymbol{pk}[k] \neq pk)$ then return $\perp$
If $(\boldsymbol{pk}, m) \in \{(\boldsymbol{pk}_i, m_i) : 1 \leq i \leq u\}$ then return $\perp$
$(w, \beta_g, \beta_{pk}) \leftarrow \mathrm{TH}[\boldsymbol{pk}, m]$; If $(w \neq g)$ then abort
$(pk_1, \ldots, pk_n) \leftarrow \boldsymbol{pk}$; $apk \leftarrow \prod_i^n pk_i^{\mathrm{H}_2((i,\boldsymbol{pk}))}$; $(T, s, z) \leftarrow \sigma$
If ($(T, apk, m) \in$ **BadSet**) then $\mathrm{HF}_1[(T, apk, m)] \leftarrow \perp$
$c \leftarrow \mathrm{H}_1((T, apk, m))$; $h \leftarrow \mathrm{H}_0((\boldsymbol{pk}, m))$; $i \leftarrow \mathrm{TI}[(T, apk, m)]$
Return $(i, (z + s \cdot \beta_g) \mod p)$

$\mathrm{H}_1(x)$:

If $(\mathrm{HF}_1[x] \neq \perp)$ then Return $\mathrm{HF}_1[x]$
$(T, apk, m) \leftarrow x$
$\mathrm{TV}[apk] \leftarrow \mathrm{TV}[apk] \cup \{x\}$
If $(\mathrm{TJ}[apk] = \perp)$ then
  Return $\mathrm{HF}_1[x] \leftarrow\$ \mathbb{Z}_p$
$\iota \leftarrow \iota + 1$; $\mathrm{TI}[x] \leftarrow \iota$
$\mathrm{HF}_1[x] \leftarrow c_\iota \leftarrow\$ \mathrm{Ch}(\mathrm{TJ}[apk], T)$
Return $\mathrm{HF}_1[x]$

$\mathrm{H}_2(x)$:

If $(\mathrm{HF}_2[x] \neq \perp)$ then Return $\mathrm{HF}_2[x]$
$(\cdot, \boldsymbol{pk}) \leftarrow x$; If $(pk \notin \boldsymbol{pk})$ then
  Return $\mathrm{HF}_2[x] \leftarrow\$ \mathbb{Z}_p$
$j \leftarrow j + 1$; $k \leftarrow \mathsf{minInd}(pk, \boldsymbol{pk})$
If $(x \neq (k, \boldsymbol{pk}))$ then
  Return $\mathrm{HF}_2[x] \leftarrow\$ \mathbb{Z}_p$
$S \leftarrow \prod_{i \neq k} \boldsymbol{pk}[i]^{\mathrm{H}_2((i,\boldsymbol{pk}))}$
$\mathrm{HF}_2[(k, \boldsymbol{pk})] \leftarrow e_j \leftarrow \mathrm{NwTar}(S)$
$apk \leftarrow S \cdot pk^{e_j}$; $\mathrm{TJ}[apk] \leftarrow j$
If $\mathrm{TV}[apk] \neq \perp$ then
  BadSet $\leftarrow$ **BadSet** $\cup\, \mathrm{TV}[apk]$
Return $\mathrm{HF}_2[x]$

Figure 3.27: Adversary $\mathcal{A}_{\mathrm{xidl}}$ used in Theorem 3.7.2. Oracles $\mathrm{NS}, \mathrm{Sign}_1, \mathrm{Sign}_2, \mathrm{H}_0$ are simulated exactly per code from Fig. 3.24.

does not abort is exactly

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

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

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

where we applied the fact that $(1-(1+n)^{-1})^n \geq e^{-1}$ for positive n. Since game Gm_2 can only abort more often than Gm_1 and that the aborting at line 40 is an event independent of whether $\mathcal{A}_{\mathrm{ms}}$ succeeds, Equation (3.58) gives us that

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

Hence,

$$\Pr[\mathrm{Gm}_{2,\rho}(\mathcal{A}_{\mathrm{ms}})] \geq \frac{1}{e(1+q_{\mathrm{s}})} \cdot \Pr[\mathrm{Gm}_0(\mathcal{A}_{\mathrm{ms}})] \, . \tag{3.59}$$

For the rest of the proof, we set $\rho = (1-(1+q_{\mathrm{s}})^{-1})$ and omit writing them in the subscript for games. Next, we need to further modify oracles H_1 and H_2 so that whenever H_2 derives a fresh aggregate key apk, it must not have been queried to H_1 (in the form of (T, apk, m) for *any* T and m). Formally, consider games Gm_3 and Gm_4 given in Fig. 3.26. These games also keep track of a set **BadSet**, which contains those H_1 queries (T, apk, m) such that the aggregate key apk is later derived in H_2 (line 51). By construction, if any H_1 query (T, apk, m) is not in **BadSet** (at the end of the game execution), the aggregate key apk is either previosly derived in H_2, or it has never been derived in any H_2 query. Game $\mathrm{Gm}_3.\textsc{Fin}$ does not contain the boxed code, which makes the oracle behave identically to $\mathrm{Gm}_2.\mathrm{H}_2$. So, we have

$$\Pr[\mathrm{Gm}_2(\mathcal{A})] = \Pr[\mathrm{Gm}_3(\mathcal{A})] \, . \tag{3.60}$$

Oracle $\mathrm{Gm}_4.\mathrm{H}_2$ contains the boxed code, which reset the oracle H_1 at the chosen forgery point (T, apk, m) if it is part of **BadSet**. This ensures the value $\mathrm{HF}_1[(T, apk, m)]$ to always be defined *after* the H_2 query that derives aggregate key apk. By construction, Gm_3 and Gm_4 are identical-until-bad. So,

$$\Pr[\mathrm{Gm}_3(\mathcal{A})] \leq \Pr[\mathrm{Gm}_4(\mathcal{A})] + \Pr[\mathrm{Gm}_4 \text{ sets bad}] \, . \tag{3.61}$$

We first compute that probability that **BadSet** is non-empty at line 57. Since each H_2 query has probability at most q_1/p probability of adding elements to **BadSet**, we can bound

$$\Pr[\mathbf{BadSet} \neq \emptyset \text{ at line 57 }] \leq \frac{q_1 q_2}{p} \,. \tag{3.62}$$

Note that flag bad can only be set if Gm_4 did not abort (in oracle H_0 or line 55), which happens with probability $1/(e(1+q_{\mathrm{s}}))$ by previous analysis. Furthermore, the view of the adversary is *independent* of whether game Gm_4 aborts. Hence,

$$\Pr[\mathrm{Gm}_4(\mathcal{A}) \text{ sets } \mathsf{bad}] \leq \frac{q_1 q_2}{ep(1+q_{\mathrm{s}})} \,. \tag{3.63}$$

We now move on to the construction of the adversary, given in Fig. 3.27. The adversary $\mathcal{A}_{\mathrm{xidl}}$ runs $\mathcal{A}_{\mathrm{ms}}$ while giving it simulated oracle $\mathrm{H}_0, \mathrm{H}_1, \mathrm{H}_2, \mathrm{NS}, \textsc{SignStage}_1, \textsc{SignStage}_2$. Code for $\mathrm{H}_0, \mathrm{NS}, \textsc{Sign}_1, \textsc{Sign}_2$ are copied from game Gm_4. The only new code here is in H_1 and H_2, which we now explain.

For each j-th H_2 query $x = (\cdot, \boldsymbol{pk})$, where $\mathrm{HF}_2[x]$ is not yet defined the adversary will sample $\mathrm{HF}_2[(i, \boldsymbol{pk})]$ for each $i = 1, \ldots, |\boldsymbol{pk}|$ as follows. If the target public key X is not in $\boldsymbol{pk}$, then these values are sampled honestly (line 15). Otherwise, let k be the smallest index such that $\boldsymbol{pk}[k] = X$. Our adversary will query the $\textsc{NwTar}$ oracle from $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$ game so that the resulting aggregate public key apk is the target point T_j generated by the game $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. This is done by first computing the partial aggregation value of S (line 17), before submitting it to the $\textsc{NwTar}$ oracle to obtain response e_j which is set as the output of H_2 (line 19).

For each H_1 query (T, apk, m), the adversary will submit the commitment to the oracle $\textsc{Ch}$, at the index that corresponds to the aggregate public key apk. This is done so that a forgery (T, s, z) corresponding to this H_1 query can be turned into a break against $\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}$. Here, we are also utilizing the fact that a successful forgery $(\boldsymbol{pk}, m, (T, s, z))$ is such that $\mathrm{H}_0((\boldsymbol{pk}, m))$ is a known power of g. Hence, the verification equation

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

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

$$\Pr[\mathrm{Gm}_4(\mathcal{A}_{\mathrm{ms}})] = \Pr[\mathrm{Gm}^{\mathrm{xidl}}_{\mathbb{G},g,q_2,q_1}(\mathcal{A}_{\mathrm{xidl}})] \,. \tag{3.64}$$

Putting Equation (3.59), (3.60), (3.61) and (3.64) together, we obtain the result claimed in the theorem. ▮

3.18 Acknowledgements

We thank the ASIACRYPT 2021 reviewers for their careful reading and valuable comments.

Bellare was supported in part by NSF grant CNS-1717640 and a gift from Microsoft. Dai was supported in part by a Powell Fellowship and grants of the Bellare.

Chapter 3, in full, is a repreint of the material as it appears in International Conference on the Theory and Application of Cryptology and Information Security, Asiacrypt 2021. Bellare, Mihir; Dai, Wei. Springer, 2021. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, April / May 2002.

[2] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 415–432. Springer, Heidelberg, December 2002.

[3] Handan Kilinc Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. Cryptology ePrint Archive, Report 2020/1245, 2020. `https://eprint.iacr.org/2020/1245`.

[4] Handan Kılınç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, August 2021. Springer, Heidelberg.

[5] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 113–134. Springer, Heidelberg, May / June 2010.

[6] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 36–54. Springer, Heidelberg, August 2009.

[7] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. Cryptology ePrint Archive, Report 2009/160, 2009. `https://eprint.iacr.org/2009/160`.

[8] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, October 2008.

[9] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

[10] Mihir Bellare and Wei Dai. The multi-base discrete logarithm problem: Tight reductions and non-rewinding proofs for Schnorr identification and signatures. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 529–552. Springer, Heidelberg, December 2020.

[11] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 373–402. Springer, Heidelberg, August 2016.

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

[13] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael 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.

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

[15] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, August 2002.

[16] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, Heidelberg, August 2004.

[17] Mihir Bellare, Bertram Poettering, and Douglas Stebila. From identification to signatures, tightly: A framework and generic transforms. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2016.

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

[19] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006.

[20] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

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

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

[23] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. Cryptology ePrint Archive, Report 2007/438, 2007. `https://eprint.iacr.org/2007/438`.

[24] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[25] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2018.

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

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

[28] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[29] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999.

[30] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 479–498. Springer, Heidelberg, February 2007.

[31] Feng Chen, David A Koufaty, and Xiaodong Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 181–192. ACM, 2009.

[32] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000.

[33] Ronald Cramer and Victor 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.

[34] `https://www.cryptopp.com/benchmarks.html`, 2015. Accessed: 2017-05-18.

[35] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

[36] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. Cryptology ePrint Archive, Report 2020/1110, 2020. `https://eprint.iacr.org/2020/1110`.

[37] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 143–159. Springer, Heidelberg, January 2010.

[38] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 225–244. Springer, Heidelberg, March 2006.

[39] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[40] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors 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.

[41] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 207–224. Springer, Heidelberg, March 2006.

[42] Rachid El Bansarkhani and Jan Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In Sara Foresti and Giuseppe Persiano, editors, *CANS 16*, volume 10052 of *LNCS*, pages 140–155. Springer, Heidelberg, November 2016.

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

[44] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, June 1988.

[45] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[46] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 444–460. Springer, Heidelberg, May 2013.

[47] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[48] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020.

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

[50] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[51] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 408–423. Springer, Heidelberg, August 1998.

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

[53] Javier Herranz and Germán Sáez. Forking lemmas for ring signature schemes. In Thomas Johansson and Subhamoy Maitra, editors, *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 266–279. Springer, Heidelberg, December 2003.

[54] Jung Yeon Hwang, Dong Hoon Lee, and Moti Yung. Universal forgery of the identity-based sequential aggregate signature scheme. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *ASIACCS 09*, pages 157–160. ACM Press, March 2009.

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

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

[57] Michael J Jacobson, Neal Koblitz, Joseph H Silverman, Andreas Stein, and Edlyn Teske. Analysis of the xedni calculus attack. *Designs, Codes and Cryptography*, 20(1):41–64, 2000.

[58] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, Heidelberg, August 2016.

[59] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Report 2020/852, 2020. `https://eprint.iacr.org/2020/852`.

[60] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 194–204. Springer, Heidelberg, May 1995.

[61] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, Heidelberg, May / June 2006.

[62] Changshe Ma, Jian Weng, Yingjiu Li, and Robert Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Designs, Codes and Cryptography*, 54(2):121–133, 2010.

[63] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. `https://eprint.iacr.org/2018/068`.

[64] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, 2019.

[65] McAfee. Stop data exfiltration, 2015. August 2015. `https://www.mcafee.com/us/resources/solution-briefs/sb-quarterly-threats-aug-2015-1.pdf`.

[66] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 245–254. ACM Press, November 2001.

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

[68] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261, 2020. `https://eprint.iacr.org/2020/1261`.

[69] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 2021. Springer, Heidelberg.

[70] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, November 2020.

[71] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057, 2020. `https://eprint.iacr.org/2020/1057`.

[72] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *ASIACRYPT'91*, volume 739 of *LNCS*, pages 139–148. Springer, Heidelberg, November 1993.

[73] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 354–369. Springer, Heidelberg, August 1998.

[74] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

[75] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005.

[76] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[77] Herbert Robbins. A remark on stirling's formula. *The American Mathematical Monthly*, 62(1):26–29, 1955.

[78] Lior Rotem and Gil Segev. Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 222–250, Virtual Event, August 2021. Springer, Heidelberg.

[79] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[80] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. Cryptology ePrint Archive, Report 2004/031, 2004. `https://eprint.iacr.org/2004/031`.

[81] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

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

[83] Joseph H. Silverman and Joe Suzuki. Elliptic curve discrete logarithms and the index calculus. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 110–125. Springer, Heidelberg, October 1998.

[84] Douglas R. Stinson and Reto Strobl. Provably secure distributed Schnorr signatures and a (t,n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *ACISP 01*, volume 2119 of *LNCS*, pages 417–434. Springer, Heidelberg, July 2001.

[85] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan 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.

[86] Aaram Yun. Generic hardness of the multiple discrete logarithm problem. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 817–836. Springer, Heidelberg, April 2015.