Efficient Zero-Knowledge Proofs and Applications

by

Ryan Henry

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Zero-knowledge proofs provide a means for a *prover* to convince a *verifier* that some claim is true *and nothing more*. The ability to prove statements while conveying zero information beyond their veracity has profound implications for cryptography and, especially, for its applicability to *privacy-enhancing technologies*. Unfortunately, the most common zero-knowledge techniques in the literature suffer from poor scalability, which limits their usefulness in many otherwise promising applications. This dissertation addresses the problem of designing communication- and computation-efficient protocols for zero-knowledge proofs and arguments of propositions that comprise many "simple" predicates. In particular, we propose a new formal model in which to analyze batch zero-knowledge protocols and perform the first systematic study of systems for *batch zero-knowledge proofs and arguments of knowledge*. In the course of this study, we suggest a general construction for batch zero-knowledge proof systems and use it to realize several new protocols suitable for proving knowledge of and relationships among large batches of discrete logarithm (DL) representations in prime-order groups. Our new protocols improve on existing protocols in several ways; for example, among the new protocols is one with lower asymptotic computation cost than any other such system in the literature. We also tackle the problem of constructing batch proofs of *partial knowledge*, proposing new protocols to prove knowledge of a DL that is equal to *at least* k-out-of-n other DLs, *at most* k-out-of-n other DLs, or *exactly* k-out-of-n other DLs. These constructions are particularly interesting as they prove some propositions that appear difficult to prove using existing techniques, *even when efficiency is not a primary consideration*. We illustrate the applicability of our new techniques by using them to construct efficient protocols for anonymous blacklisting and reputation systems.

Thesis examining committee:

- Ian Goldberg (PhD Advisor), Associate Professor, University of Waterloo
- Douglas R. Stinson, University Professor, University of Waterloo
- Alfred Menezes, Professor, University of Waterloo
- David Jao, Associate Professor, University of Waterloo
- Nicholas J. Hopper, Associate Professor, University of Minnesota

Acknowledgements

First and foremost, I thank my PhD advisor, Ian Goldberg, whose expertise, understanding, and patience have added considerably to my graduate experience. Our conversations over the past four years have extended beyond the content and direction of my research to teaching, publishing, writing grant proposals, "professoring", mentoring, and even tequila-ing. Ian's generosity in time, ideas, and funding has truly enabled me to flourish as a graduate student. I also thank the other members of my thesis examining committee — Professors Doug Stinson, Alfred Menezes, David Jao, and Nick Hopper — for critiquing this dissertation.

The Cryptography, Security, and Privacy (CrySP) lab has been a potent incubator for ideas and friendship. My conversations with fellow CrySPers have run the gamut from multivariate polynomials to politics and have always been fun and enlightening. I especially thank my collaborators on research that appears in this dissertation, Yizhou Huang and Femi Olumofin, in addition to Tariq Elahi, Sarah Harvey, Kevin Henry, Aniket Kate, Robin Smits, Sukhbir Singh, Colleen Swanson, Jalaj Upadhyay, Tao Wang, and Gregory Zaverucha, among others, for indulging me in countless discussions and debates.

I also thank my parents, Larry and Darlene Henry, for always believing in me, and last, but by no means least, I thank my lovely wife, Lindsay, and our cockapoo, Mocha, for their love, support, welcome distractions, tolerance, and encouragement. It is impossible to describe all the direct and indirect ways Lindsay and Mocha have helped me maintain some semblance of sanity as I pursued my PhD.

I was fortunate to receive generous financial support throughout my PhD studies from (i) the David R. Cheriton School of Computer Science through a (Type I) Cheriton Graduate Scholarship, from (ii) the Ontario Ministry of Training, Colleges and Universities through an Ontario Graduate Scholarship (OGS), from (iii) the Government of Ontario and Bell Canada through a GO-Bell Scholarship, and from (iv) the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Vanier Canada Graduate Scholarship (Vanier CGS).

Government of Canada
Vanier Canada Graduate Scholarships

Gouvernement du Canada
Bourses d'études supérieures du Canada Vanier

Canada

for Lindsay & Mocha

Table of Contents

FRONT MATTER

MAIN BODY

List of Tables

List of Figures

List of Related Publications

The publications listed below introduced key results in this dissertation. I once again thank my collaborators on those papers — Ian Goldberg, Yizhou Huang, and Femi Olumofin — without whom some of those results might never have been conceived.

[HG13a] **Ryan Henry** and Ian Goldberg. Batch proofs of partial knowledge. In *Proceedings of ACNS 2013*, volume 7954 of *LNCS*, pages 502–517, Banff, AB, Canada (June 2013).

[HG13c] **Ryan Henry** and Ian Goldberg. Thinking inside the BLAC box: Smarter protocols for faster anonymous blacklisting. In *Proceedings of WPES 2013*, pages 71–81, Berlin, Germany (November 2013).

[HHG13] **Ryan Henry**, Yizhou Huang, and Ian Goldberg. One (block) size fits all: PIR and SPIR with variable-length records via multi-block queries. In *Proceedings of NDSS 2013*, San Diego, CA, USA (February 2013).

[HG12] **Ryan Henry** and Ian Goldberg. All-but-k mercurial commitments and their applications. Technical Report CACR 2012-26, University of Waterloo, Waterloo, ON, Canada (November 2012).

[HOG11] **Ryan Henry**, Femi Olumofin, and Ian Goldberg. Practical PIR for electronic commerce. In *Proceedings of CCS 2011*, pages 677–690, Chicago, IL, USA (October 2011).

BibTeX entry for this thesis:

```
@phdthesis{thesis:Henry14,
    author = {Ryan Henry},
    title = {Efficient Zero-Knowledge Proofs and Applications},
    school = {University of Waterloo},
    address = {Waterloo, ON, Canada},
    month = {August},
    year = {2014},
}
```

This page is intentionally left blank.

Chapter 1

Introduction

1.1 Background and motivation

The Internet can be a dangerous place to visit. As the proportion of our day-to-day activities that occur online continues to increase, so too does our exposure to online privacy risks imposed on us by fraudsters and identity thieves, by intrusive advertising companies, by oppressive governments, and by countless unknown others. Privacy-enhancing technologies (PETs) are a class of services, applications, and mechanisms that help mitigate such online privacy threats by empowering users with control over the collection, dissemination, and use of information about themselves and about their day-to-day activities. Modern PETs employ sophisticated cryptographic primitives in order to provide *privacy-friendly* alternatives to otherwise *privacy-agnostic* technologies.

One important such primitive is *zero-knowledge proofs of knowledge*. Informally, a zero-knowledge proof of knowledge is a conversation between two mutually distrusting parties — a prover and a verifier — in which the prover convinces the verifier that some claim is true *and nothing more*. The prover holds some kind of "evidence" that "proves" its claim in the traditional sense (for example, the prover might hold an NP-witness[1]); however, throughout its conversation with the verifier, the prover will reveal neither this evidence nor any nontrivial information about it. Nonetheless, as a "proof of knowledge", the conversation must still, somehow, convince the verifier *beyond doubt* that the prover's claim holds.

[1] A *witness* for the claim "$\exists x\, R(x) = 1$" is just a specific value w in the domain of R for which $R(w) = 1$. An *NP-witness* is a witness for some such claim in the complexity class NP [Gol01; Definition 1.3.2].

What makes a zero-knowledge proof of knowledge special, therefore, is what the verifier learns from it: *nothing* beyond the veracity of the prover's claim.[2] The ability to prove statements while conveying nothing beyond their veracity has had profound implications for PETs, and for cryptography in general, since its introduction in a landmark 1985 paper by Goldwasser, Micali, and Rackoff [GMR85]; indeed, zero-knowledge proofs of knowledge are integral to numerous PET constructions in the literature, ranging from end-to-end verifiable voting schemes [Adi08, CH11, JCJ05], through to anonymous blacklisting and reputation systems [AKS12, ATK11, HG13c, LH10, TAKS10], private information retrieval protocols with novel features [HHG13, HOG11], threshold ring signatures [TWC+04], verifiable shuffles for mix networks [Gro10, Nef01, SK95], and cryptographically private auctions [Cac99, Bra06], among others. Other privacy-friendly cryptographic primitives, such as verifiable secret sharing schemes [Fel87, RBO89, Sta96] and systems for secure multi-party computation [KK09, CCD88, BOGW88], often employ zero-knowledge proofs of knowledge to obtain security against malicious, arbitrarily deviating adversaries using cryptographic building blocks that are only secure against *honest-but-curious* (or *semi-honest*) adversaries [GMW86, GMW87]. In these constructions, each participant must *prove* to all the others that it is correctly carrying out its own security-critical obligations, in which case the others can safely regard it as semi-honest. The zero-knowledge property is of course crucial here, as it is what conceals each participant's private information from its curious peers.

Alas, zero-knowledge does not come free. Each construction cited above gives rise to one or more zero-knowledge proofs of knowledge of statements whose "fan-in" (i.e., number of inputs) scales with a critical system parameter (such as the number of users [CH11, JCJ05, TWC+04], the size of a database [AKS12, ATK11, HG13c, HOG11, TAKS10], or the circuit complexity of a computation [KK09]). Such constructions tend to be inefficient in practice, as the communication and computation overhead imposed by the underlying zero-knowledge protocols typically grow in proportion to the fan-in of the statements under consideration. This might explain why, despite their general applicability to cryptography and PETs, as of June 2014 — when this dissertation was completed — not a single widely deployed PET uses high fan-in zero-knowledge proofs of knowledge to protect user privacy on the Internet.

[2] Of course, the verifier might be able to *deduce* more by combining its newfound conviction in said veracity with some prior knowledge. The zero-knowledge property cannot prevent the verifier from making such deductions; however, it does minimize, to the fullest extent possible, how much the verifier is able to deduce.

1.2 Thesis statement

This dissertation considers *batch zero-knowledge proof and verification* techniques, which attempt to reduce the communication and computation overhead imposed by certain zero-knowledge proofs of knowledge when they are used to prove statements exhibiting high fan-in. Such communication and computation overhead appears to be a significant — if not *the most* significant — barrier to the adoption of several promising PETs; thus, it is the author's hope that the batch zero-knowledge proof and verification techniques introduced herein will have a direct impact to online privacy by eliminating or reducing barriers to the adoption of new and existing PETs. More concretely, this dissertation will establish that *batch zero-knowledge proof and verification techniques can significantly reduce the communication and computation overhead imposed by the zero-knowledge protocols naturally arising in several privacy-enhancing technologies.*

The technical contributions contained herein focus specifically on systems for batch zero-knowledge proofs regarding knowledge of and relationships among batches of *discrete logarithms* (DLs) in prime-order groups. (Nonetheless, we state most of the definitions and theorems in a more general setting and some of our constructions for DLs in prime-order groups are immediately applicable to proofs about other algebraic objects.) Zero-knowledge proofs of knowledge about DLs in prime-order groups are by far the most widely utilized zero-knowledge protocols in the current generation of PET designs; in fact, every one of the high-fan-in proofs arising in the PET constructions cited in Section 1.1 concerns statements about such DLs. Therefore, using the new protocols contained in this dissertation (in place of the equivalent "schoolbook" protocols) can simultaneously reduce the communication overhead of, and speed up the bottleneck computations in, each of those PET constructions, among myriad others.

1.3 Research contributions

This section briefly summarizes the most notable research contributions contained herein.

1. **Formal treatment of batch verifiers and batch proof systems.** Chapter 3 proposes a new formal model with which to study *batch verifiers* and systems for *batch zero-knowledge proofs and arguments of knowledge*. Section 3.1 begins with a new formal definition

for batch verifiers, which improves on the existing definition by Bellare, Garay, and Rabin [BGR98b; Definition 2.1] in several important respects. That section also revisits Bellare et al.'s batch verifier constructions for DLs and analyzes them within this new formal model. Section 3.2 extends the new batch verifier definition to a definition of systems for batch zero-knowledge proofs (or arguments) of knowledge. The latter definition is a significant refinement of an existing definition proposed by the author and Ian Goldberg in a recent ACNS 2013 paper [HG13a; §4].

2. **Batch zero-knowledge proofs of complete knowledge.** Section 3.2 suggests a generic construction yielding systems for batch zero-knowledge proofs of complete knowledge for *linear relations*, and then uses it to construct several new batch variants of Schnorr's protocol [Sch89; §2] for proving knowledge of several DLs simultaneously. The new protocols provide greater flexibility in trading off communication versus computation cost and one of them has lower asymptotic computation cost than any previously suggested system for batch zero-knowledge proofs of knowledge in the literature.

3. **Lattice-based attack on batch proofs of partial knowledge.** Section 4.2 revisits Peng and Bao's batch "partial knowledge" variant of Chaum and Pedersen's protocol for proving knowledge of and equality among 1-out-of-n DL pairs [PB08]. Our analysis uncovers a critical flaw in Peng and Bao's protocol and we outline a practical, lattice-based attack to exploit it. The attack is practical even for computationally limited provers and, therefore, renders Peng and Bao's batch proof system completely insecure given reasonable assumptions about the prover's knowledge. Prior to the aforementioned ACNS 2013 paper [HG13a], Peng and Bao's protocol was the *only* such proof system in the literature. This attack first appeared in an appendix to the extended version of our ACNS 2013 paper [HG13b; Appendix A].

4. **A new variant of trapdoor mercurial commitments.** Section 4.3 proposes a new kind of cryptographic commitments that generalize trapdoor n-mercurial commitments [LY10]. These new commitments, which we call trapdoor $\mathscr{L}$-mercurial commitments, bind the committer to a *hidden subset of components* from a length-n sequence. Section 4.3.2 provides an efficient and provably secure construction for the special case of *trapdoor* $(n-k,n)$*-threshold mercurial commitments*, and then Section 4.4.1 uses this construction to implement a provably secure system for batch honest-verifier zero-knowledge arguments

of knowledge of k-out-of-n DLs, for any constant $k \in [1, n]$. Section 4.4.2 discusses how this latter protocol generalizes to a system for honest-verifier zero-knowledge arguments of partial knowledge over any monotone access structure from a large class of such access structures, given an efficient construction for a suitable $\mathscr{L}$-mercurial commitment scheme. The notion of $\mathscr{L}$-mercurial commitments presented herein is a significant generalization of the trapdoor $(n - k, n)$-threshold mercurial commitments recently proposed by the author in collaboration with Ian Goldberg [HG12].

5. **Batch arguments of knowledge for non-monotone access structures.** Section 4.4.3 discusses batch zero-knowledge arguments of knowledge and equality of DLs over certain *non-monotone* access structures. For instance, we present novel batch zero-knowledge arguments of knowledge of a DL that is equal to *exactly* k-out-of-n other DLs, to *at most* k-out-of-n other DLs, or to between k_1- and k_2-out-of-n other DLs. Using these protocols, it is possible to construct efficient arguments of knowledge for several additional non-monotone statements. The techniques in this section first appeared in a recent WPES 2013 paper [HG13c] by the author and Ian Goldberg, although the presentation herein extends those results in several important respects.

6. **Faster anonymous blacklisting without trusted third parties.** Section 5.1 introduces the anonymous blacklisting and reputation problem and discusses three variants of Tsang, Au, Kapadia, and Smith's Blacklistable Anonymous Credentials (BLAC) [TAKS10] system. Section 5.1.6 presents batch protocols for the bottleneck zero-knowledge proofs in each of the three variants [TAKS10, TAKS07, AKS12], thus yielding a suite of efficient and provably secure anonymous blacklisting and reputation systems with strong privacy guarantees and no trusted third parties that are capable of deanonymizing users at will. These improved constructions for BLAC and its variants first appeared in the aforementioned WPES 2013 paper [HG13c] with Ian Goldberg.

7. **Efficient zero-knowledge proofs about committed polynomials.** Appendix C introduces three novel systems for zero-knowledge proofs (and arguments) for claims about polynomials committed to using Kate, Zaverucha, and Goldberg's $\text{PolyCommit}_{\text{DL}}$ polynomial commitments [KZG10a]. While the original motivation for developing these protocols was to obtain provable security in our construction for trapdoor $(n - k, n)$-threshold mercurial commitments, the new proofs are more generally useful, especially in the setting of

verifiable secret sharing [CGMA85]. The zero-knowledge protocols in this section first appeared in the technical report that introduced trapdoor $(n - k, n)$-threshold mercurial commitments [HG12].

Chapter 2

Cryptographic preliminaries

This chapter reviews the basic mathematical notation and cryptographic primitives used in the rest of the dissertation.

2.1 Mathematical notation

Algebraic structures. Throughout, $\mathbb{G}$ will always denote a finite multiplicative group with τ-bit prime order q and a fixed generator $g \in \mathbb{G}$, and $\mathbb{G}^* = \mathbb{G} \setminus \{1\}$ will refer to its subset of non-identity elements. Likewise, $\mathbb{Z}_q$ will denote the field of integers modulo q, and $\mathbb{Z}_q^* = \mathbb{Z}_q \setminus \{0\}$ will refer to its multiplicative group of units. When convenient to do so, we will equate elements of $\mathbb{Z}_q$ with integers from $\{0, \ldots, q-1\}$ under arithmetic modulo q. We use $\mathbb{Z}_q[x]$ to denote the ring of polynomials with coefficients from $\mathbb{Z}_q$. $\mathbb{R}$ refers to the set of reals, $\mathbb{R}^+$ to the set of positive reals, $\mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$ to the set of integers, $\mathbb{N} = \{0, 1, \ldots\}$ to the set of non-negative integers, and $\mathbb{N}^+ = \{1, 2, \ldots,\}$ to the set of positive integers. Any reader not familiar with the above algebraic structures should consult a textbook on abstract algebra[3] for definitions and fundamental results about them. They will be used repeatedly throughout.

[3] The author recommends Gallian's *Contemporary Abstract Algebra* [Gal12] for this purpose.

If $a \in \mathbb{R}$, then $\lceil a \rceil$ is the smallest integer greater than or equal to a and $\lfloor a \rfloor$ is the largest integer less than or equal to a. For a given finite set U, $u \in_{\mathrm{R}} U$ denotes uniform random selection of an element u from U and $A \subseteq_d U$ indicates that the set A is a size-d subset of U. If $U \subseteq \mathbb{N}$, then $U_{(j)}$ denotes the j^{th} smallest element in U.

If Λ is a finite set, then the set of all length-n sequences of elements from Λ is denoted by Λ^n and the set of all finite sequences (of *any* finite, non-negative length) of elements from Λ is denoted by $\Lambda^* = \bigcup_{n \in \mathbb{N}} \Lambda^n$. The sequences in Λ^* are called finite *strings* over the *alphabet* Λ and subsets of Λ^* are called *languages* of strings over Λ. If the alphabet is $\Lambda = \{1\}$, then $\{1\}^*$ corresponds to $\mathbb{N}$ expressed in unary. As is customary in the literature, we drop the curly set brackets in this case, writing 1^τ to denote the integer $\tau \in \mathbb{N}$ encoded as a unary string. Given strings s and t, we use $s \| t$ to denote the concatenation of s and t. When s (or t) is *not* a string, it is understood that the concatenation acts on some canonical string representation of s (or t) and that the result of such concatenation is always a string. Finally, the notation $|\cdot|$ can have three distinct meanings, depending on context: (i) $|U|$ denotes the *cardinality* of a set U, (ii) $|s|$ denotes the *length* of a string s, and (iii) $|a|$ denotes the *absolute value* of a real number a. No confusion can arise as the precise meaning will always be clear from context.[4]

2.1.1 Asymptotic behaviour of functions

This section recalls important concepts about the asymptotic behaviour of functions. We begin with Landau's asymptotic notation, as espoused by Knuth [Knu76]. Given functions $f\colon \mathbb{N} \to \mathbb{R}$ and $h\colon \mathbb{N} \to \mathbb{R}$,

1. **little-O:** $f(n) \in \mathrm{o}(h(n))$ if, for each $c \in \mathbb{R}^+$, there exists $N_c \in \mathbb{N}$ such that $|f(n)| \leq c|h(n)|$ whenever $n > N_c$. The expression $f(n) \in \mathrm{o}(h(n))$ reads as "$f(n)$ is *little-O* of $h(n)$".
2. **big-O:** $f(n) \in \mathrm{O}(h(n))$ if there exist $c \in \mathbb{R}^+$ and $N_c \in \mathbb{N}$ such that $|f(n)| \leq c|h(n)|$ whenever $n > N_c$. The expression $f(n) \in \mathrm{O}(h(n))$ reads as "$f(n)$ is *big-O* of $h(n)$".
3. **(big-)Theta:** $f(n) \in \Theta(h(n))$ if both $f(n) \in \mathrm{O}(h(n))$ and $h(n) \in \mathrm{O}(f(n))$. The expression $f(n) \in \Theta(h(n))$ reads as "$f(n)$ is *(big-)Theta* of $h(n)$".

[4] In fact, an ambiguous case does arise throughout: whether to treat 1^τ as the integer τ expressed in unary or as a length-τ string of 1s. Fortunately, both of these interpretations result in the same value for $|1^\tau|$ and either choice is therefore acceptable.

4. **big-Omega:** $f(n) \in \Omega(h(n))$ if $h(n) \in \mathrm{O}(f(n))$. The expression $f(n) \in \Omega(h(n))$ reads as "$f(n)$ is *big-Omega* of $h(n)$".
5. **little-Omega:** $f(n) \in \omega(h(n))$ if $h(n) \in \mathrm{o}(f(n))$. The expression $f(n) \in \omega(h(n))$ reads as "$f(n)$ is *little-Omega* of $h(n)$".

Polynomial functions. A function $p\colon \mathbb{N} \to \mathbb{R}^+$ is said to be *polynomial in n* if there exists a constant $a \in \mathbb{R}^+$ such that $p(n) \in \mathrm{O}(n^a)$; likewise, $p(n)$ is *poly-logarithmic in n* if it is polynomial in $\lg n$ and it is *super-polynomial in n* if it is not polynomial in n. By $\mathit{poly}(n)$ we denote the set of all functions polynomial in n. An algorithm is (expected) *probabilistic polynomial-time* (PPT) if it is probabilistic and if its (expected) running time is polynomial in the length of its inputs. More generally, we call any algorithm (whether it be deterministic or probabilistic) *efficient* if its expected running time is polynomial in the length of its inputs.

Negligible functions. A function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if $\varepsilon(n) \in \mathrm{o}(1/n^a)$ for every $a \in \mathbb{R}^+$. Conversely, a function $\mu\colon \mathbb{N} \to \mathbb{R}^+$ is *non-negligible* if there exists some $a \in \mathbb{R}^+$ such that $\mu(n) \in \Omega(1/n^a)$. An event E occurs with *(non-)negligible probability* in n if the probability that it occurs is a (non-)negligible function of n, and E occurs with *overwhelming probability* in n if the probability that $\neg E$ occurs is negligible in n.

Note that a function being "not negligible" is not the same as a function being non-negligible. For example, given a negligible function ε and a non-negligible function μ, one can define many piecewise functions, for instance

$$\eta(n) = \begin{cases} \varepsilon(n) & \text{if } n \text{ is even, and} \\ \mu(n) & \text{if } n \text{ is odd,} \end{cases}$$

which are neither negligible nor non-negligible. The reader should keep this distinction in mind when interpreting definitions that reference functions that are negligible, non-negligible, or not negligible.

2.1.2 Languages and witnesses

Fix a finite alphabet Λ and let S and W be arbitrary languages over Λ. A collection of ordered pairs $R \subseteq S \times W$ is a *(binary) relation* on strings from Λ^*. We call the language of strings $L_R = \{s \in S \mid \exists w \in W, (s, w) \in R\}$ the *language induced by* R. In the following, we abuse notation by treating R as the Boolean-valued function indicating whether its input is in R as a relation; in other words, we treat $R(s, w)$ as a function evaluating to 1 if $(s, w) \in R$ and to 0 otherwise. Any pair of inputs $(s, w) \in S \times W$ on which R may be evaluated is called an *instance* for R. If $R(s, w) = 1$, then the string w is called a *witness* that $s \in L_R$. Given a string $s \in S$, the set $\mathcal{W}_R(s) = \{w \in W \mid R(s, w) = 1\}$ is called the *witness set* for $s \in L_R$. Of course, $\mathcal{W}_R(s) = \{\}$ if $s \notin L_R$. Throughout, we assume that there are efficient algorithms to test membership of arbitrary strings in S and W so that identifying instances for R is easy. We make no assumptions about the ease or difficulty of testing membership of a string s in L_R when no appropriate witness $w \in \mathcal{W}_R(s)$ has been provided. Intuitively, one should think of S as the subset of strings $s \in \Lambda^*$ for which one can *plausibly* claim that $s \in L_R$.

NP-relations and NP-languages. A language L_R is called an *NP-language* if it belongs to the complexity class NP; that is, if (i) there exists an efficient algorithm to evaluate $R(s, w)$ on any instance $(s, w) \in S \times W$, and (ii) there exists a function $p(n) \in poly(n)$ such that every $s \in L_R$ has at least one witness $w \in \mathcal{W}_R(s)$ satisfying $|w| \leq p(|s|)$. Any witness $w \in \mathcal{W}_R(s)$ that satisfies the latter bound is called an *NP-witness* that $s \in L_R$. If L_R is an NP-language, then we call the relation R an *NP-relation*. Viewing $\mathcal{W}_R(s)$ as the set of *proofs* that $s \in L_R$, we can interpret NP as the class of languages whose strings each have one or more "short" proofs of membership (that can be checked in polynomial time). Note that while, by definition, every NP-language has an efficient algorithm to *verify* its short proofs, it is widely believed that there are NP-languages for which no efficient algorithm can *find* its short proofs [For09]. We refer the reader to Goldreich's *Foundations of Cryptography – Volume 1, Basic Techniques* [Gol01] for further discussion about NP and its relevance to cryptography.

2.2 Interactive protocols

A *protocol* is a system of rules describing the sequence, syntax, and semantics of message exchange between two or more *interactive algorithms*.[5] A message transfer from one algorithm to another in a protocol constitutes a *move* and two consecutive moves (by different parties) constitute a *round*. Protocols comprising just one move are called *non-interactive protocols* and those comprising two or more moves (hence, comprising one or more rounds) are called *interactive protocols*.

2.2.1 Interactive proof systems

This dissertation concerns a special kind of two-party protocols called *interactive proof systems*, which involve a pair of interactive algorithms that play two distinct roles: one algorithm is the *prover* and the other algorithm is the *verifier*. The prover always has some advantage over the verifier, whether it be additional computational resources or access to private information about the common input; in particular, except where otherwise specified, the prover will always have *unbounded* computational capacity and the verifier, by contrast, will always be PPT in the common input. In general, many different interactive algorithms can send and receive messages according to either role in a given interactive proof system; nonetheless, we always have two specific "honest" algorithms — denoted by P and V — in mind. We use an asterisk to maintain a clear distinction between these honest algorithms and potential "dishonest" impostors, writing P^* and V^* to denote *arbitrary* algorithms taking on the prover and verifier roles in the protocol under consideration. (Note that P^* and V^* can refer both to the honest algorithms and to dishonest algorithms, whereas P and V can *only* refer to the honest algorithms.)

[5] Formally, an *interactive algorithm* is a Turing machine [AB09; §1.2] enhanced with a read-only *common input tape* and one or more read-and-appendive-write *communication tapes*. (In particular, each machine can read from and write to any of its shared communication tapes, but no machine can delete or overwrite data on a communication tape after it has been written.) All communicating algorithms share a common input tape and each pair of communicating algorithms shares a communication tape over which to exchange messages. In addition, each algorithm has three private tapes: a *random tape* containing an infinite sequence of unbiased coin tosses with which it can seed probabilistic computations, a *work tape* on which it can carry out intermediate calculations, and an *auxiliary input tape* on which it receives secret information that is not necessarily available to the other interacting algorithms.

If P is an interactive algorithm, then $\mathrm{P}(w)$ denotes P given the string $w \in \Lambda^*$ as its private auxiliary input. If V is a second interactive algorithm, then (P, V) denotes the protocol arising when P interacts with V, and $\langle \mathrm{P}, \mathrm{V} \rangle(s)$ denotes the random variable describing the output of V in such an interaction when the common input string is $s \in S$. (So, for example, in a given discussion, (P, V) and $(\mathrm{P}^*, \mathrm{V})$ reference the same protocol, but $\langle \mathrm{P}, \mathrm{V} \rangle(s)$ and $\langle \mathrm{P}^*, \mathrm{V} \rangle(s)$ generally do not reference the same random variables.) P always makes the final move in an interactive proof, after which V checks one or more *verification equations* to decide whether it should *accept* or *reject* the interaction. Given a common input string $s \in S$ for (P, V), we write $\Pr[1 \leftarrow \langle \mathrm{P}, \mathrm{V} \rangle(s)]$ for the probability of the event "V accepts" and we write $\Pr[0 \leftarrow \langle \mathrm{P}, \mathrm{V} \rangle(s)]$ for the probability of the event "V rejects". Intuitively, we want to ensure that, upon interacting with *any* arbitrarily cheating P^*, V accepts if and (essentially) only if $s \in L_R$. Such a property would imply that there is no cheating strategy using which some dishonest P^* can reliably "fool" V into concluding that $s \in L_R$ when in fact $s \notin L_R$. Definition 1 formalizes this intuitive requirement.

Definition 1. Fix an alphabet Λ and let S and W be infinite subsets of Λ^*. A two-party interactive protocol (P, V) is an *interactive proof system* for the infinite relation $R \subseteq S \times W$ if there exists a negligible function $\varepsilon \colon \mathbb{N} \to \mathbb{R}^+$ such that, for every $s \in S$ and for every (possibly dishonest) prover P^*, (P, V) provides the following two guarantees.

1. **Complete:** If $s \in L_R$, then $\Pr[1 \leftarrow \langle \mathrm{P}, \mathrm{V} \rangle(s)] = 1$.
2. **Sound:** If $s \notin L_R$, then $\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(s)] < \varepsilon(|s|)$.[6]

When P^* and V execute an interactive proof system on a given input $s \in S$, the resulting interaction is called an *interactive proof* that $s \in L_R$; the interactive proof is an *accepting proof* if V accepts and it is a *rejecting proof* otherwise.

2.2.2 Interactive argument systems

Because we assume that P has unbounded computation capacity, it follows that P can always compute a witness attesting to the membership of s in L_R, provided one exists. Sometimes it is useful to assume that P^* is PPT in the common input, in which case we must modify the

[6] An alternative, weaker soundness criterion requires only that $\Pr[0 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(s) \mid s \in S \setminus L_R]$ be non-negligible in $|s|$ for every (possibly cheating) prover P^*. Running some number in $\mathit{poly}(|s|)$ of independent trials of such a protocol and having V accept only if it accepts in every trial yields a new PPT protocol whose soundness error is negligible in $|s|$, as required by the stronger soundness criterion we use in Definition 1.

completeness criterion to provide honest P with an appropriate witness as private auxiliary input. (Indeed, the fact that $s \in L_R$ does not imply that a given PPT prover can find a witness to prove it.) The completeness criterion for PPT provers is therefore:

$$\text{“If } R(s, w) = 1, \text{ then } \Pr\left[1 \leftarrow \langle \mathrm{P}(w), \mathrm{V} \rangle(s)\right] = 1\text{”}.$$

This modified completeness criterion does *not* imply that a given PPT prover P^* can only make V accept when P^* has access to an appropriate witness $w \in \mathcal{W}_R(s)$; rather, it implies that *if* honest P is provided with such a witness, *then* honest P will always make V accept. In particular, if $s \in L_R$, then the definition does not preclude some other P^* from making V accept *even when P^* does not “know” any witness* that $s \in L_R$.

The interpretation — though not the statement — of the soundness criterion also changes when we restrict the P^* to be PPT: proving soundness with respect to all PPT provers implies that, for any *sufficiently long* common input $s \in S \setminus L_R$, it is *computationally infeasible* (rather than *impossible*) for a (possibly dishonest) P^* to make V accept with a probability that is not negligible in $|s|$. In such protocols, it typically happens that a given P^* can make V accept on input $s \in S \setminus L_R$ only if that same P^* can solve some (presumed) intractable problem *online, during the execution of the protocol*.

Because V’s confidence in concluding that “an accepting interaction implies that $s \in L_R$” is contingent on the assumption that P^* cannot solve a particular, concrete problem instance, such protocols involving PPT provers are called *interactive argument systems* so as to differentiate them from interactive proof systems. This notion is formalized in Definition 2.

Definition 2. Fix an alphabet Λ and let S and W be infinite subsets of Λ^*. A two-party interactive protocol (P, V) is an *interactive argument system* for the infinite relation $R \subseteq S \times W$ if there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for every $s \in S$ and for every (possibly dishonest) PPT prover P^*, (P, V) provides the following two guarantees.

1. **Complete:** If $R(s, w) = 1$, then $\Pr\left[1 \leftarrow \langle \mathrm{P}(w), \mathrm{V} \rangle(s)\right] = 1$.
2. **Sound:** If $s \notin L_R$, then $\Pr\left[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(s)\right] < \varepsilon(|s|)$.

When P^* and V execute an interactive argument system on a given input $s \in S$, the resulting interaction is called an *interactive argument* that $s \in L_R$; the interactive argument is an *accepting argument* if V accepts and it is a *rejecting argument* otherwise.

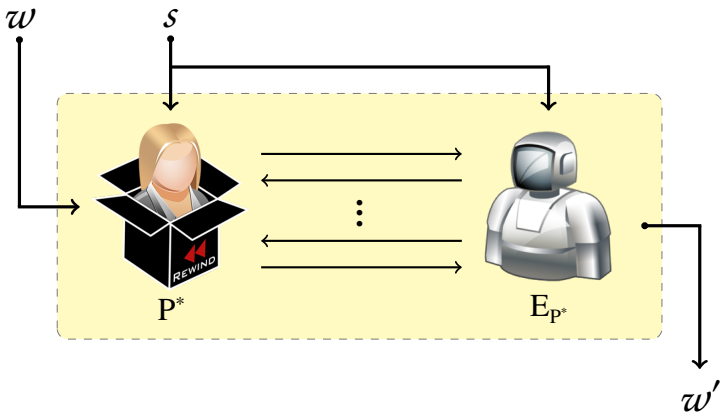

Figure 2.1: A *universal knowledge extractor* for (P, V) having rewinding black box oracle access to a prover P^*. If (P, V) is both a system for proofs of knowledge (in the sense of Definition 3) for the NP-language L_R and an interactive proof system (in the sense of Definition 1), then $\mathrm{E}_{\mathrm{P}^*}$ is expected PPT in $|s|$ and its output w' is an NP-witness that $s \in L_R$.

2.2.3 Extractability and proofs of knowledge

If P^* is computationally unbounded, then an accepting proof that $s \in L_R$ implies that P^* must, in some sense, "know" a witness that $s \in L_R$. After all, the proof establishes with overwhelming probability that $\mathcal{W}_R(s)$ is nonempty and the unboundedness of P^* ensures that P^* can easily be made to compute the elements of $\mathcal{W}_R(s)$. Of course, a *real* prover cannot be computationally unbounded. Nonetheless, it is often useful for a real, computationally bounded prover to demonstrate not only that $s \in L_R$ but also that it "knows" a witness in $\mathcal{W}_R(s)$. This idea is captured by the notion of *knowledge extraction*: we say that P^* knows a witness in $\mathcal{W}_R(s)$ if there is an efficient algorithm to *compute* such a witness from s, given special access to P^* as an oracle. We emphasize that, although the notion of extractability is *motivated* by PPT provers, we *do not assume* that the P^* are in fact PPT; in particular, we use extractability to define interactive proof systems that let honest P prove knowledge of a witness *even though P is only PPT* but that are sound *even when* P^* *is computationally unbounded*.

More precisely, we consider a probabilistic algorithm $\mathrm{E}_{\mathrm{P}^*}$ endowed with *rewinding black box oracle access* to P^*; in other words, $\mathrm{E}_{\mathrm{P}^*}$ may (i) send arbitrary messages to P^* and read the responses P^* generates (all in a single algorithmic time step), and (ii) "rewind" P^* to any earlier

internal state. Such a probabilistic algorithm $\mathrm{E}_{\mathrm{P}^*}$ having rewinding black box oracle access to an interactive algorithm P^* is called an *oracle machine* for P^*. As $\mathrm{E}_{\mathrm{P}^*}$ only has *black box* oracle access to P^*, it is *not* privy to P^*'s private inputs or internal state; however, through rewinding, $\mathrm{E}_{\mathrm{P}^*}$ can observe what moves P^* makes in response to several distinct verifier moves at a given point in a given protocol execution. This gives $\mathrm{E}_{\mathrm{P}^*}$ computational power somewhere between that of P^* and that of V: V is *strictly* PPT and P^* is potentially unbounded, whereas $\mathrm{E}_{\mathrm{P}^*}$ is *expected* PPT and is granted rewinding black box oracle access to P^*. Figure 2.1 illustrates the extraction process.

If the ability for P^* to make V accept with a probability non-negligible in $|s|$ implies that $\mathrm{E}_{\mathrm{P}^*}$ is an efficient algorithm for computing a witness $w' \in \mathcal{W}_R(s)$, then we say that an accepting proof (or argument) that $s \in L_R$ implies not only that $s \in L_R$ but also that P^* *knows* a witness $w \in \mathcal{W}_R(s)$; thus, we equate "knowledge of w" with "the ability to *efficiently* compute w". It is necessary to assume that L_R is an NP-language and that the output of $\mathrm{E}_{\mathrm{P}^*}$ is (essentially) always an NP-witness, as $\mathrm{E}_{\mathrm{P}^*}$ cannot possibly be efficient for any $s \in L_R$ that lacks an NP-witness. Note that if $|\mathcal{W}_R(s)| \geq 2$, then whatever witness P^* actually knows (for example, if $\mathrm{P}^*(w)$ was given w as private auxiliary input) might not be the same one that $\mathrm{E}_{\mathrm{P}^*}$ outputs: in other words, $w' \leftarrow \mathrm{E}_{\mathrm{P}^*(w)}(s)$ does not imply that $w = w'$.

Definition 3. Fix an alphabet Λ, let S and W be infinite subsets of Λ^*, and let $R \subseteq S \times W$ be an infinite NP-relation. A two-party interactive protocol (P, V) is a *system for proofs of knowledge* for L_R if there exists an oracle machine E, a positive function $\kappa\colon S \to \mathbb{R}^+$, and a constant $a \in \mathbb{R}^+$ such that, for every (possibly dishonest) prover P^*, (P, V) provides the following two guarantees.

1. **Complete:** If $s \in L_R$, then $\Pr[1 \leftarrow \langle \mathrm{P}, \mathrm{V} \rangle(s)] = 1$.
2. **Extractable:** Let $\rho(s) = \Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(s)]$. If $\rho(s) > \kappa(s)$, then $\mathrm{E}_{\mathrm{P}^*}(s)$ outputs a witness $w' \in \mathcal{W}_R(s)$ in an expected number of steps at most $|s|^a / (\rho(s) - \kappa(s))$.

   Note that the notation $\mathrm{E}_{\mathrm{P}^*}(s)$ here refers to the oracle machine E given the common input string s and rewinding black box oracle access to P^*.

An oracle machine E satisfying the extractability criterion in Definition 3 is called a *universal knowledge extractor* for (P, V) and the smallest function $\kappa(\cdot)$ for which the extractability criterion holds (for *every* possible knowledge extractor) is called the *knowledge error function* for (P, V). The knowledge error function measures the probability (inherent to the protocol) with which a

dishonest P^* can make V accept on input $s \in S$ without actually knowing a witness $w \in \mathcal{W}_R(s)$. Note that if the above denominator, $\rho(s) - \kappa(s)$, is non-negligible in $|s|$, then the expected running time $|s|^a / (\rho(s) - \kappa(s))$ of $E_{P^*}(s)$ is polynomial in $|s|$ and, consequently, the universal knowledge extractor is efficient. (In other words, if P^* causes honest V to accept with a probability non-negligibly greater than $\kappa(s)$, then $E_{P^*}(s)$ extracts a witness $w \in \mathcal{W}_R(s)$ in expected polynomial time.) In some instances, we must assume that the P^* are PPT in order to prove that $\rho(s) - \kappa(s)$ is non-negligible in $|s|$, in which case we call the protocol a system for *arguments of knowledge* for L_R. Similar remarks as those given for interactive argument systems apply to systems for arguments of knowledge. The probability $1 - \kappa(s)$ is sometimes called the *soundness* of (P, V). If (P, V) is also an interactive proof (or argument) system in the sense of Definition 1, then the knowledge error $\kappa(s)$ is negligible and (P, V) has soundness overwhelming in $|s|$.

2.2.4 Simulatability and zero-knowledge proofs

We now discuss the zero-knowledge property of a protocol (P, V). Informally, an interactive proof system for L_R is *zero-knowledge* if a trusted oracle merely proclaiming that $s \in L_R$ enables any (possibly dishonest) V^* to efficiently deduce anything it might have "learned" by engaging in (P, V^*) on common input $s \in L_R$. (We again equate "knowledge" with "the ability to efficiently compute" and note that such computation is not limited to evaluating functions — it also applies to generating *probability distributions*.) This idea is captured by the notion of *simulation*: we say that a given verifier V^* gains zero (extra) knowledge if, *without any help from P*, V^* can "simulate" an interaction that is "indistinguishable" from the real interactions that occur when honest P executes the protocol with that particular V^* on the same common input. This simulatability requirement ensures that there is no PPT cheating strategy using which V^* can "learn" things from P that it could not just as easily *compute on its own* given the common input s and a promise that $s \in L_R$.

To formalize this notion, we define the *transcript* of a (two-party) protocol execution between P and V^* as the tuple of messages exchanged between P and V^* up until V^* halts. An *accepting transcript* is a transcript of an accepting proof (or argument) and a *rejecting transcript* is a transcript of a rejecting proof (or argument). The *aggregate view* of V^* upon interacting with P, denoted $\mathsf{View}_{P,V^*}(s)$, consists of all the information that V^* "sees" in the protocol execution, including (i) the transcript, (ii) the common input string s, (iii) V^*'s own private inputs, and

(iv) V^*'s random coin flips (if any). At least one of P or V^* will always be probabilistic, so the aggregate view of V^* for a given set of inputs is a random variable induced by the random coin tosses of P and V^*. By parameterizing over all possible protocol executions, we obtain a collection $\{\mathsf{View}_{P,V^*}(s)\}_{s\in L_R}$ of random variables; such a parametrized collection of random variables is called an *ensemble of random variables*.

Definition 4. Fix an alphabet Λ and let $L_R \subseteq \Lambda^*$ be an infinite language over Λ. Given two probabilistic algorithm $\mathcal{A}$ and $\mathcal{B}$ operating on strings from Λ^*, the ensembles $\{\mathcal{A}(s)\}_{s\in L_R}$ and $\{\mathcal{B}(s)\}_{s\in L_R}$ are said to be

1. *perfectly indistinguishable* on L_R if, for all $s \in L_R$, the random variables $\mathcal{A}(s)$ and $\mathcal{B}(s)$ have the same distributions,
2. *statistically indistinguishable* on L_R if there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all $s \in L_R$,

$$\sum_{\alpha}\Big|\Pr[\alpha \leftarrow \mathcal{A}(s)] - \Pr[\alpha \leftarrow \mathcal{B}(s)]\Big| \leq \varepsilon(|s|),$$

   where α ranges over all the possible outputs of $\mathcal{A}$ and all the possible outputs of $\mathcal{B}$, and
3. *computationally indistinguishable* on L_R if there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all $s \in L_R$ and for every PPT algorithm D,

$$\sum_{\alpha\,|\,D(\alpha)=1}\Big|\Pr[\alpha \leftarrow \mathcal{A}(s)] - \Pr[\alpha \leftarrow \mathcal{B}(s)]\Big| \leq \varepsilon(|s|),$$

   where α ranges over all the possible outputs of $\mathcal{A}$ and all the possible outputs of $\mathcal{B}$ for which $D(\alpha) = 1$.

Intuitively, statistical indistinguishability ensures that, although the distributions of $\mathcal{A}(s)$ and $\mathcal{B}(s)$ may in fact be *slightly* different, they are sufficiently close that distinguishing between them is not feasible for any PPT algorithm (in an asymptotic sense), as doing so requires sampling from $\mathcal{A}(s)$ and $\mathcal{B}(s)$ a number of times super-polynomial in $|s|$. Likewise, computational indistinguishability ensures that, although the distributions of $\mathcal{A}(s)$ and $\mathcal{B}(s)$ may in fact be *quite* different, distinguishing between them becomes computationally infeasible for any PPT algorithm

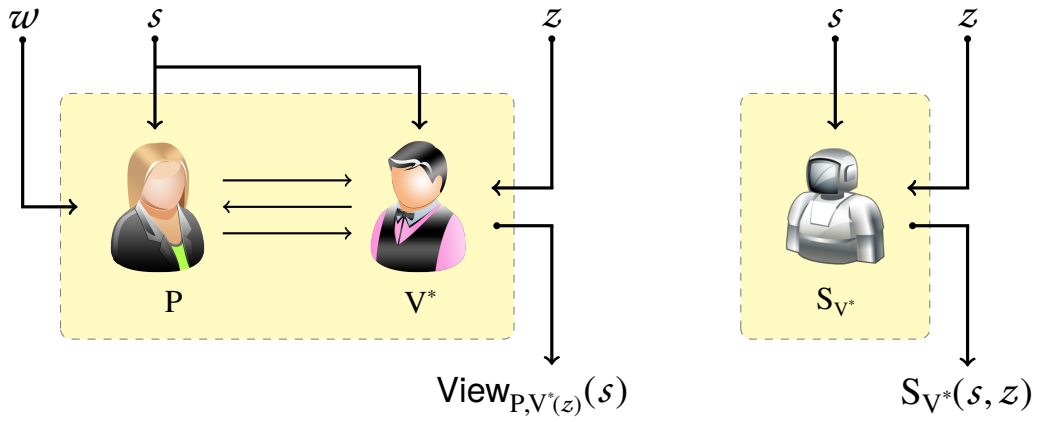

Figure 2.2: An interactive proof between P and V^* (left) and a *simulator* for V^* (right). If (P, V) is zero-knowledge on L_R, then $\mathsf{View}_{P,V^*(z)}(s)$ and $S_{V^*}(s, z)$ are sampled from indistinguishable random variables.

as $|s|$ grows sufficiently large. Proving that $\{\mathcal{A}(s)\}_{s\in L_R}$ and $\{\mathcal{B}(s)\}_{s\in L_R}$ are computationally indistinguishable — assuming that they are not also statistically indistinguishable — typically requires one or more assumptions about the intractability of certain mathematical problems. (We discuss such intractability assumptions in Section 2.3.)

Definition 5. Fix an alphabet Λ, let S and W be infinite subsets of Λ^*, and let $R \subseteq S \times W$ be an infinite relation. A two-party interactive protocol (P, V) is (*perfect*, *statistical*, or *computational*) *zero-knowledge* on L_R if, for every (possibly malicious) verifier V^* and for every auxiliary string $z \in \Lambda^*$, there exists an expected PPT algorithm S_{V^*} such that the ensembles $\{\mathsf{View}_{P,V^*(z)}(s)\}_{s\in L_R}$ and $\{S_{V^*}(s, z)\}_{s\in L_R}$ are (perfectly, statistically, or computationally) indistinguishable. An algorithm S_{V^*} with this property is called a (*perfect*, *statistical*, or *computational*) *simulator* for V^*.

The string z in Definition 5 plays the role of "prior knowledge" of V^*; hence, including it in the definition ensures that V^* cannot leverage prior knowledge to help it extract additional information from interactions with honest P. In particular, we demand that, for any given string z known to V^*, the simulator S_{V^*} can efficiently reproduce any conversation that might arise between V^* and P given only s and z as input. Moreover, providing V^* with access to an *arbitrary*

string z ensures that the zero-knowledge property holds under arbitrary sequential composition of zero-knowledge protocols, as z may include the view of V^* from all prior interactions with P.[7] Figure 2.2 illustrates a simulator for V^*.

The protocols that we consider in this dissertation are not zero-knowledge in the sense of Definition 5; rather, they satisfy a weaker notion of the zero-knowledge property under which there is a simulator for the *honest verifier* V, but not necessarily for any dishonest verifier V^*. This property is aptly called *honest-verifier zero-knowledge*. We sometimes refer to a zero-knowledge protocol (in the sense of Definition 5) as *general zero-knowledge* to distinguish it from a protocol that only satisfies this weaker "honest-verifier" version of the zero-knowledge property. Note that focusing on honest-verifier zero-knowledge protocols is not a limitation in practice; indeed, there exist many well-known techniques [GSV98, Gro04, FS86, DGOW95, Vad06] to convert honest-verifier zero-knowledge protocols into equivalent general zero-knowledge protocols while introducing only modest communication and computation overhead. Interested readers can find a concise description of one such construction in Ivan Damgård's lecture notes [Dam11; §8].

The Camenisch-Stadler notation

We use Camenisch and Stadler's ubiquitous notation [CS97; §4] to denote systems for honest-verifier zero-knowledge proofs of knowledge by expressing P's intent. The advantage of this notation is that it allows us to speak of zero-knowledge proofs of knowledge in the abstract, without specifying the actual protocols. For example, if L_R is an NP-language and if $s \in S$ is given as a common input to P^* and to honest V, then we write $\mathrm{PK}\{\, w : R(s, w) = 1 \,\}$ to denote an honest-verifier zero-knowledge proof (or argument) of knowledge of a witness $w \in \mathcal{W}_R(s)$. By convention, any values that appear to the left of the colon constitute a witness that P^* is proving knowledge of, and any values that appear only to the right of the colon are public (and, hence, known both to P^* and to V).

[7] In general, the zero-knowledge property does *not* hold under *parallel* composition; however, looking forward, it turns out that for the particular class of protocols that we focus on in this dissertation — called *sigma protocols* — the zero-knowledge property *does* indeed hold under arbitrary parallel and sequential composition.

2.2.5 The Fiat-Shamir heuristic and non-interactive arguments

An interactive proof system (P, V) is called *public coin* if the messages that honest V sends to P on common input $s \in S$ contain only uniform random strings from some publicly known "challenge" domain $C(s)$. (We typically assume that $C(s) = C(s')$ whenever $|s| = |s'|$.) All of the protocols that we consider in this dissertation are public-coin protocols.

Observation 2.1. To prove that a public-coin interactive proof system (P, V) is honest-verifier zero-knowledge, it suffices to prove indistinguishability among the ensembles of real and simulated interaction *transcripts* (rather than the verifier's entire aggregate view). Doing so suffices because the rest of the aggregate view is, by the very definition of a public-coin protocol, fixed before P^* and honest V ever exchange a message.[8]

We now describe a general construction for converting almost any public-coin system for interactive, honest-verifier zero-knowledge proofs of knowledge into a system for non-interactive, general zero-knowledge arguments of knowledge. This is accomplished by replacing V with a *cryptographically secure hash function* $\mathcal{H}_s$ with range $C(s)$; that is, an efficiently computable function $\mathcal{H}_s\colon \Lambda^* \to C(s)$ that is both *preimage-resistant* and *collision-intractable* [RS04; §3].[9] More precisely, in any move where honest V would normally send a (uniform random) message from $C(s)$ to P in the interactive proof, P instead *computes* the (still essentially random) message by evaluating $\mathcal{H}_s$ on the entire protocol transcript up to the current move. Typically, V (or some external, benevolent entity) will start off the transcript with an extra random nonce $M \in \Lambda^*$, whose length is polynomial in τ. This effectively forces P^* to compute the entire non-interactive proof of knowledge in real time.

If we reinterpret the nonce M as a *message*, then an accepting transcript with common input s and nonce M can be viewed as establishing that some prover holding a witness $w \in \mathcal{W}_R(s)$ has "signed" the message M. For this reason, such non-interactive protocols are sometimes

[8] If (P, V) is *not* a public-coin protocol, then any verifier-selected values that arise in the transcript may depend on P's earlier messages and V^*'s private input.

[9] Intuitively, a hash function $\mathcal{H}_s\colon \Lambda^* \to C(s)$ is *preimage-resistant* if no efficient algorithm can, given a random $c \in C(s)$, output a string $s \in \Lambda^*$ such that $\mathcal{H}_s(s) = c$, and it is *collision-intractable* if no efficient algorithm can produce distinct strings $s_1, s_2 \in \Lambda^*$ such that $\mathcal{H}_s(s_1) = \mathcal{H}_s(s_2)$. Formalizing the the latter collision-intractability property requires that the hash function be *keyed*; otherwise, an efficient algorithm can simply have one or more collisions "hard-wired" in [Rog06]. Note that the nonce M in the Fiat-Shamir transform serves the purpose of a key. For formal definitions relating to cryptographically secure hash functions the author suggests the treatment by Rogaway and Shrimpton [RS04].

called *signature proofs of knowledge* [CL06]. We indicate that a protocol is a signature proof of knowledge on the message M using a notation derived from the above Camenisch-Stadler notation by (i) replacing the prefix PK with SPK, and (ii) appending the suffix (M); thus, SPK$\{ w : R(s, w) = 1 \}(M)$ is the non-interactive form of PK$\{ w : R(s, w) = 1 \}$ using the above transformation and nonce M.

A special case of this transformation was first proposed by Fiat and Shamir [FS86]; for this reason, the transformation is often called the *Fiat-Shamir transform*. To argue that Fiat-Shamir transformed protocols are secure, Bellare and Rogaway [BR93] suggested modeling the hash function $\mathcal{H}_s$ as a *random oracle* — that is, as a function that pairs any given input from its domain with a uniform random output from its range. As P^* cannot predict the output of such a random oracle prior to choosing an input, the situation for P^* is essentially equivalent to that which arises when it interacts with honest V. (In both cases, no matter what P^* does, it receives nothing but apparently unrelated, random messages from $C(s)$ in response.) The key difference between the interactive protocol and the Fiat-Shamir transformed protocol is that, in the latter case, P^* can query the random oracle many times in hopes that some query will return a "lucky" message to which it knows how to respond. (Of course, this does not work in the interactive protocol, as V will reject in the overwhelmingly likely event that P^*'s first attempt fails.) However, if the domain from which V draws its random messages in the interactive proof — and, hence, the range of $\mathcal{H}_s$ in the non-interactive proof — has cardinality super-polynomial in $|s|$, then brute-force searching for such lucky outputs would require time super-polynomial in $|s|$. This explains why the resulting protocol is a system for non-interactive *arguments* of knowledge and not a system for non-interactive proofs of knowledge: we need to assume that P^* is PPT in the common input to argue that it cannot use a brute-force search to discover the negligible portion of inputs to the random oracle that yield lucky outputs.

Because obtaining overwhelming soundness in the non-interactive protocol requires that $|\mathcal{H}_s(\cdot)|$ be non-constant (indeed, super-polynomial in $|s|$), it is customary to assume that P and V have access to a single random oracle $\mathcal{H}$ that outputs an *infinite* random string for any given input and, moreover, that P and V can efficiently map such uniform random, infinite strings to uniform random elements of $C(s)$.[10] (In this case, P and V must prepend the common input s to

[10] If $\mathcal{H}$ is a real, cryptographically secure hash function with fixed-length outputs, then we still obtain *heuristic* soundness against any t-time probabilistic prover for certain, concrete values of $t \in \mathbb{N}^+$. For t sufficiently large, this level of heuristic security might be adequate in some real-world applications; therefore, the assumption that a given Fiat-Shamir-transformed protocol instantiation is secure in practice is known as the *Fiat-Shamir heuristic*.

each string they input to the random oracle.) Such non-interactive arguments are said to be sound in the *random oracle model*. To prove that the zero-knowledge property still holds, we must allow the random oracle to be *programmed* so that the simulator can choose some arbitrary inputs $s\|x$ in the domain of $\mathcal{H}$ and *specify* the corresponding outputs $\mathcal{H}(s\|x) \in C(s)$.

Theorem 2.2 (Pointcheval & Stern, 1996 [PS96]). *If the protocol denoted in Camenisch-Stadler notation by* $\mathrm{PK}\{\, w : R(s, w) = 1 \,\}$ *is a system for public-coin honest-verifier zero-knowledge proofs of knowledge for* L_R *in which all messages from V come from a set* $C(s)$ *whose cardinality is super-polynomial in* $|s|$*, then the protocol denoted by* $\mathrm{SPK}\{\, w : R(s, w) = 1 \,\}(M)$ *is a system for non-interactive, general zero-knowledge arguments of knowledge for* L_R*, with soundness holding in the random oracle model.*

Of course, real hash functions do not behave like programmable random oracles, a fact vividly illustrated by Canetti, Goldreich, and Halevi [CGH04], who constructed several (contrived) protocols each provably secure in the random oracle model yet trivially insecure when the random oracle is instantiated by any efficiently computable function. Nonetheless, the non-interactive zero-knowledge arguments arising from the Fiat-Shamir transform are generally believed to be sound in practice (with respect to PPT provers), provided the random oracle is instantiated using a hash function that is both preimage-resistant and collision-intractable.

2.3 Discrete logarithms and the DL assumption

In the rest of this dissertation, we consider systems for zero-knowledge proofs of knowledge about *discrete logarithms* (DLs) in prime-order multiplicative groups. This section briefly introduces the DL problem and its generalization to several bases. Some other variations on the DL problem appear in Appendix A and will be introduced in the text as needed. Whenever we define such a problem, we begin with a concrete problem definition, stated with respect to a particular multiplicative group $\mathbb{G}$, and we follow with an asymptotic definition of the associated intractability assumption. For the latter asymptotic formulations, we require the notion of a *group-generating algorithm*.

Definition 6. A *group-generating algorithm* $\mathcal{G}$ is a PPT algorithm that, on input 1^τ, outputs a description of a finite multiplicative group $\mathbb{G}$, its τ-bit prime order q, and a fixed generator $g \in \mathbb{G}^*$.

We write $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ to indicate that the tuple $(\mathbb{G}, q, g)$ is obtained from $\mathcal{G}$ by sampling from its output distribution on input 1^τ and we write $(\mathbb{G}, q, g_1, \ldots, g_k) \leftarrow \mathcal{G}(1^\tau; k)$ as shorthand for the process of sampling $(\mathbb{G}, q, g_1) \leftarrow \mathcal{G}(1^\tau)$ and having a trusted entity choose $k - 1$ additional generators $g_2, \ldots, g_k \in_R \mathbb{G}^*$ uniformly at random. Intuitively, the quantity $\tau \in \mathbb{N}^+$ is a *security parameter* that controls the difficulty of solving random instances of certain mathematical problems in $\mathbb{G}$.

2.3.1 The DL problem

The *discrete logarithm (DL) problem* in $\mathbb{G}$ is:

DL problem: Given as input a pair $(g, h) \in \mathbb{G}^* \times \mathbb{G}$, output $x \in \mathbb{Z}_q$ such that $h = g^x$.

The exponent x is called the *discrete logarithm* of h with respect to the base g, which we express in mathematical notation as $x = \log_g h$. Unlike computing traditional *continuous* logarithms in $\mathbb{R}^+$, computing $x = \log_g h$ from $(\mathbb{G}, q, g, h)$ appears to be a difficult problem for many groups $\mathbb{G}$;[11] on the other hand, given any pair $(h, x) \in \mathbb{G} \times \mathbb{Z}_q$, one can efficiently verify that $x = \log_g h$ by computing g^x and checking that the result equals h. It is therefore natural to view the DL problem in $\mathbb{G}$ as defining, for each generator $g \in \mathbb{G}^*$, an NP-relation $R_g = \{(h, x) \in \mathbb{G} \times \mathbb{Z}_q \mid h = g^x\}$.

An algorithm $\mathcal{A}$ solves the DL problem in $\mathbb{G}$ with *advantage* ε if

$$\Pr[x \leftarrow \mathcal{A}(g, h) \mid h = g^x] = \varepsilon,$$

where the probability is over the random choice of $(g, h) \in_R \mathbb{G}^* \times \mathbb{G}$ and the random bits consumed by $\mathcal{A}$. We say that the DL problem is (t, ε)*-intractable* in $\mathbb{G}$ if no t-time probabilistic algorithm solves the DL problem in $\mathbb{G}$ with advantage greater than ε.

Our first intractability assumption, the so-called DL assumption, concerns the infeasibility of solving DLs in the groups output by a fixed group-generating algorithm $\mathcal{G}$. The DL assumption has been one of the most studied and widely used intractability assumptions in the cryptographic literature since its introduction in Diffie and Hellman's seminal paper [DH76; §3].

[11] In fact, not only is the DL problem apparently difficult in the *worst case* for many groups, but, in some settings, one can exploit a property called *random self-reducibility* to show that the DL problem is equally difficult in the *average case* [JMV05].

Definition 7. The *DL assumption* holds for a group-generating algorithm $\mathcal{G}$ if there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$, then $\mathcal{A}$ solves the DL problem in $\mathbb{G}$ with advantage at most $\varepsilon(\tau)$.

The DL assumption is widely believed to hold for several natural classes of group-generating algorithms [Bon98; §1.1]. Most commonly, these group-generating algorithms output either (i) the group of points on an elliptic curve over a finite field, or (ii) a multiplicative subgroup of the integers modulo a prime. Throughout this dissertation, we will always assume that the DL assumption holds for $\mathcal{G}$ without concerning ourselves with the specific representation of the groups that $\mathcal{G}$ outputs.

2.3.2 The k-DLREP problem

The DL problem in $\mathbb{G}$ generalizes naturally to the k-base *DL-representation (k-DLREP) problem* [Bra00; §2.3.2] in $\mathbb{G}$:

> ***k*-DLREP problem:** Given as input a $(k+1)$-tuple $(g_1, \ldots, g_k, h) \in (\mathbb{G}^*)^k \times \mathbb{G}$, output $(x_1, \ldots, x_k) \in (\mathbb{Z}_q)^k$ such that $h = g_1^{x_1} \cdots g_k^{x_k}$.

The tuple $(x_1, \ldots, x_k)$ is called a *DL-representation* of h with respect to the basis $(g_1, \ldots, g_k)$. The bases g_i in the basis are typically pairwise distinct, though the definition does not require this. Unlike DLs, the DL-representations of h with respect to a given basis are generally not unique.

An algorithm $\mathcal{A}$ solves the k-DLREP problem in $\mathbb{G}$ with advantage ε if

$$\Pr\left[(x_1, \ldots, x_k) \leftarrow \mathcal{A}(g_1, \ldots, g_k, h) \mid h = g_1^{x_1} \cdots g_k^{x_k}\right] = \varepsilon,$$

where the probability is over the random choices of $(g_1, \ldots, g_k) \in_R (\mathbb{G}^*)^k$ and $h \in_R \mathbb{G}$, and the random bits consumed by $\mathcal{A}$. In particular, the probability is *not* over the specific k-DLREP $(x_1, \ldots, x_k)$ that $\mathcal{A}$ outputs; indeed, the advantage of $\mathcal{A}$ measures the probability that $\mathcal{A}$ can output *any* — not just some *particular* — k-DLREP of h with respect to a random basis $(g_1, \ldots, g_k)$. We say the k-DLREP problem is (t, ε)*-intractable* in $\mathbb{G}$ if no t-time probabilistic algorithm solves the k-DLREP problem in $\mathbb{G}$ with advantage greater than ε.

Definition 8. The *DLREP assumption* holds for a group-generating algorithm $\mathcal{G}$ if, for every positive integer-valued function $k(\tau) \in \mathit{poly}(\tau)$, there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $(\mathbb{G}, q, g_1, \ldots, g_{k(\tau)}) \leftarrow \mathcal{G}(1^\tau; k(\tau))$, then $\mathcal{A}$ solves the $k(\tau)$-DLREP problem in $\mathbb{G}$ with advantage at most $\varepsilon(\tau)$.

Theorem 2.3 (Brands, 2000 [Bra00; Proposition 2.3.3]). *If the DL assumption holds for $\mathcal{G}$, then the DLREP assumption also holds for $\mathcal{G}$.*

The converse of Theorem 2.3 holds trivially.

2.4 Cost model

As the focus of this dissertation is efficient zero-knowledge proof systems for DLs, we take this opportunity to introduce the cost model we employ in the analysis of such protocols.

Computation cost

In implementations of zero-knowledge proof systems for DLs, the CPU time required to compute powers (exponentiation) and products of powers (multiexponentiation) dominates the running time. We measure the costs of both kinds of operations by counting the expected number of multiplications in $\mathbb{G}$ that they require. (For simplicity, we ignore the cost of arithmetic in $\mathbb{Z}_q$ required, for example, to determine which exponents to use in a given exponentiation.) Following Bellare, Garay, and Rabin [BGR98b; §2.3], we write $\mathrm{ExpCost}_{\mathbb{G}}^{(m)}(\tau)$ to denote the expected cost of raising a single generator $g \in \mathbb{G}^*$ to some number $m \in \mathbb{N}^+$ of pairwise distinct, random τ-bit powers. When $m = 1$, we omit it from the notation.

The classic "square-and-multiply" algorithm [MvOV96; Algorithm 2.143] provides an upper bound of

$$\mathrm{ExpCost}_{\mathbb{G}}(\tau) \leq 1.5\tau \tag{2.1a}$$

multiplications in $\mathbb{G}$, and more sophisticated windowing methods [BC89, SD92] easily reduce the coefficient in this bound to about 1.2. Asymptotically, Erdös proved [Erd61] that, if one uses the best exponentiation algorithm theoretically possible, then $\mathrm{ExpCost}_{\mathbb{G}}(\tau)$ converges to

$$\tau + \tau/\lg \tau \tag{2.1b}$$

multiplications in $\mathbb{G}$ as τ grows large. An algorithm proposed by Brauer [Bra39] two decades prior to Erdös' proof meets this asymptotic bound and is always within a factor of about $\lg \tau$ from optimal [Ber02]. The author suggests Bernstein's presentation of Brauer's algorithm [Ber02; §3], which incorporates a number of concrete optimizations to the original algorithm.

For $m > 1$, we note the trivial bound

$$\mathrm{ExpCost}^{(m)}_{\mathbb{G}}(\tau) \leq m \,\mathrm{ExpCost}_{\mathbb{G}}(\tau) \tag{2.2a}$$

and remark that well-known techniques from the literature [Lim00, BGMW92, LL94, MN96, Yao76] can make the inequality in this bound strict. Pippenger [Pip76; Theorem 2] generalized Erdös' result to establish that, using the best possible algorithm, $\mathrm{ExpCost}^{(m)}_{\mathbb{G}}(\tau)$ converges to

$$\tau + m\tau/\lg(m\tau) \tag{2.2b}$$

multiplications in $\mathbb{G}$ as m and τ grow large, subject to $\lg m \in o(\tau)$. Yao [Yao76] proposed a concrete algorithm for such computations (later rediscovered and popularized by Brickell, Gordon, McCurley, and Wilson [BGMW92]) whose cost approaches $\tau + m\tau/\lg \tau$ as m grows large.

For products of powers (i.e., multiexponentiations), we replace the bit length τ by a list of ordered pairs, writing $\mathrm{ExpCost}^{(m)}_{\mathbb{G}}((n_1, \tau_1), \ldots, (n_k, \tau_k))$ to denote the expected cost to compute m products of powers of a common set of $n = \sum_{i=1}^{k} n_i$ bases of which, in each product, the same n_i bases are raised to distinct, random τ_i-bit exponents. Straus [Str64] suggested the multiexponentiation algorithm in Figure 2.3, which yields a concrete upper bound of

$$\mathrm{ExpCost}^{(m)}_{\mathbb{G}}((n_1, \tau_1), \ldots, (n_k, \tau_k)) \leq m\Big(\max_{1 \leq i \leq k} \{\tau_i\} + \tfrac{1}{2}\sum_{i=1}^{k} n_i \tau_i\Big) \tag{2.3a}$$

Input: $(g_1, x_1), \ldots, (g_n, x_n) \in \mathbb{G}^* \times \mathbb{Z}_q^*$
Output: $h = \prod_{i=1}^{n} g_i^{x_i}$

```
h := 1                                   // h = multiplicative identity
τ := ⌈ lg max{x_1, ..., x_n} ⌉           // τ = bit length of longest x_i
for (j = τ down to 1) do
    for (i = 1 up to n) do               // N.B.: x_i[j] is the j^th bit of x_i
        if (x_i[j] == 1) then            //       x_i[j] = 0 when j > lg x_i
            h := h * g_i                 // multiply conditionally
        end
    end
    h := h * h                           // but always square
end
return h                                 // h = g_1^x_1 ··· g_n^x_n
```

Figure 2.3: A simple multiexponentiation algorithm attributed to Straus [Str64]. The algorithm uses exactly $\lceil \lg \max\{x_1, \ldots, x_n\} \rceil$ squarings and an expected $\frac{1}{2}\sum_{i=1}^{n} \lg x_i$ additional multiplications in $\mathbb{G}$ when the bases g_i and the exponents x_i are all pairwise distinct.

multiplications in $\mathbb{G}$. One can further reduce the cost of multiexponentiation by using precomputation [LL94, Lim00, BGMW92], at the expense of additional storage for the precomputed values. Pippenger's analysis [Pip80] establishes general asymptotic bounds for $\mathrm{ExpCost}_{\mathbb{G}}^{(m)}((n,\tau))$ as m, n, and τ each tend to infinity subject to $\lg m \in o(\tau)$ and $\lg n \in o(\tau)$; specifically, Pippenger showed that $\mathrm{ExpCost}_{\mathbb{G}}^{(m)}((n,\tau))$ approaches $\tau \cdot \min\{m,n\} + mn\tau/\log(mn\tau)$ multiplications in $\mathbb{G}$ [Pip80]. Setting $n = \sum_{i=1}^{k} n_i$ and $\tau = \max_{1 \le i \le k}\{\tau_i\}$ yields the inequality

$$\mathrm{ExpCost}_{\mathbb{G}}^{(m)}((n_1,\tau_1),\ldots,(n_k,\tau_k)) \le \mathrm{ExpCost}_{\mathbb{G}}^{(m)}((n,\tau)),$$

which provides (loose) asymptotic bounds for some additional multiexponentiation operations. The author suggests a technical report by Lim [Lim00] for a concrete algorithm establishing that

$$\mathrm{ExpCost}_{\mathbb{G}}^{(m)}((n,\tau)) \le 2mn\tau/\lg n \tag{2.3b}$$

multiplications in $\mathbb{G}$ whenever n is sufficiently large.

In subsequent chapters, a frequently arising cost is $\mathrm{ExpCost}_{\mathbb{G}}((1,\tau),(n,\lambda))$ with $\lambda \in \mathbb{N}^+$, which is bounded above asymptotically by

$$\tau + n\lambda/\lg n + \tau/\lg \tau$$

multiplications in $\mathbb{G}$ as τ and n grow large subject to $\lg n \in o(\tau)$.

We are interested both in the asymptotic behaviour of algorithms and in their concrete cost for specific "realistic" parameter choices. Whenever we make an asymptotic comparison between the cost of two algorithms, we will assume the best possible asymptotic cost for all operations involved; however, when we make concrete comparisons between the costs of two algorithms on specific "small" inputs, we will assume that exponentiations are evaluated using square-and-multiply and that multiexponentiations are evaluated using Straus' generalization of square-and-multiply (as presented in Figure 2.3). We summarize the expected concrete and asymptotic costs for various common exponentiation operations in Table 2.1.

Communication cost

For communication costs, we separately count the numbers of $\mathbb{G}$ and $\mathbb{Z}_q$ elements that are sent both from P to V and from V to P, and the quantity (and lengths) of all other strings that P and V must exchange.

Round complexity

All of the protocols that we consider consist of a (small) constant number of rounds. Furthermore, they can all be made non-interactive using the Fiat-Shamir transform.

2.5 Schnorr's protocol for DLs

A fundamental building block of almost every zero-knowledge proof system in this dissertation is Claus-Peter Schnorr's system for honest-verifier zero-knowledge proofs of knowledge of a DL [Sch89]. The protocol is denoted in Camenisch-Stadler notation by $\mathrm{PK}\{\, x : h = g^x \,\}$. The

Table 2.1: The expected number of multiplications in $\mathbb{G}$ (both concrete and asymptotic) required to evaluate common exponentiation operations. For all rows, we assume that each of m, n, and τ tends to infinity subject to $\lg n \in o(\tau)$ and $\lg m \in o(\tau)$.

Operation	Concrete cost	Asymptotic cost
$\mathbf{ExpCost}_{\mathbb{G}}(\tau)$	$3\tau/2$	$\tau + \tau/\lg \tau$
$\mathbf{ExpCost}^{(m)}_{\mathbb{G}}(\tau)$	$3m\tau/2$	$\tau + m\tau/\lg(m\tau)$
$\mathbf{ExpCost}_{\mathbb{G}}((n,\tau))$	$\tau + n\tau/2$	$\tau + n\tau/\lg(n\tau)$
$\mathbf{ExpCost}^{(m)}_{\mathbb{G}}((n,\tau))$	$m\tau + mn\tau/2$	$\tau \cdot \min\{m,n\} + mn\tau/\lg(mn\tau)$

common input (which was heretofore always denoted by 's') to Schnorr's protocol in $\mathbb{G}$ is a pair of group elements $(g,h) \in \mathbb{G}^* \times \mathbb{G}$. Hence, in the notation of Section 2.2, we have that both $S = \mathbb{G}^* \times \mathbb{G}$ and $L_R = \mathbb{G}^* \times \mathbb{G}$ and that $W = \mathbb{Z}_q$. As Schnorr's protocol is well-defined for any prime-order group $\mathbb{G}$, we also treat $(\mathbb{G},q)$ as part of the common input so that $(\mathbb{G},q,g)$ describes the particular DL relation $R_g = \{(h,x) \in \mathbb{G} \times \mathbb{Z}_q \mid h = g^x\}$ with respect to which P wishes to prove knowledge of a witness $x = \log_g h$. We assume that it is easy to check membership of (g,h) in $\mathbb{G}^* \times \mathbb{G}$ (and that V always does so); hence, as $\mathbb{G}$ has prime order q, it follows that g is a generator for $\mathbb{G}$ and that there *exists* an exponent $x \in \mathbb{Z}_q$ with $h = g^x$; however, it does not necessarily follow that either of P or V actually *knows* the exponent $x \in \mathbb{Z}_q$.

Schnorr's protocol, which we illustrate in Figure 2.4, provides a means for P to prove knowledge of $x = \log_g h$ without revealing anything about x to V. In such protocol figures, the actions performed by P appear within the leftmost box and the actions performed by V appear within the rightmost box. Each move in the protocol is depicted by an arrow from the sender's box to the receiver's box, with the message that is sent appearing as a label above the arrow. The protocol figures should be read from top to bottom. V will accept if and only if every verification equation after P's last move holds.

P's expected computation cost in Schnorr's protocol is $\mathrm{ExpCost}_{\mathbb{G}}(\tau) < 3\tau/2$ multiplications in $\mathbb{G}$ and, after some rearrangement in the verification equation, V's expected computation cost is $\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) < 2\tau$ multiplications in $\mathbb{G}$. For communication cost, P sends one $\mathbb{G}^*$ element and one $\mathbb{Z}_q$ element to V, and V sends one $\mathbb{Z}_q$ element to P.

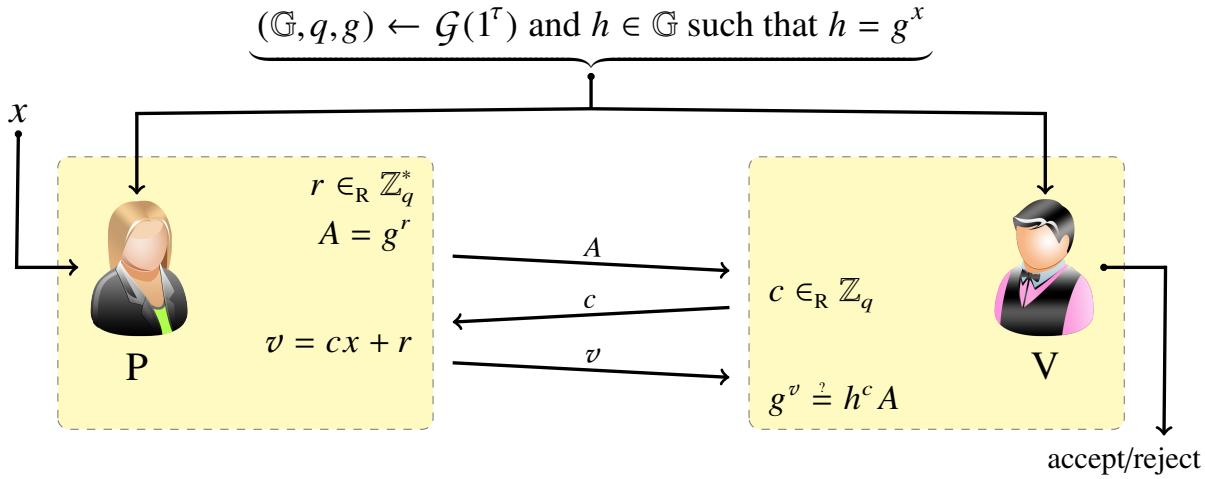

Figure 2.4: A system for honest-verifier perfect zero-knowledge proofs of knowledge of a DL due to Schnorr [Sch89]. The protocol is c-simulatable and 2-extractable (see Definitions 9 and 10). An accepting transcript is a tuple $(A, c, v) \in \mathbb{G}^* \times (\mathbb{Z}_q)^2$ such that $g^v = h^c A$. The protocol is denoted by $\mathrm{PK}\{\, x : h = g^x \,\}$.

2.5.1 Sigma protocols

Schnorr's protocol exemplifies a well-studied class of three-move public-coin protocols called *sigma protocols* [Cra96]. The general structure of a sigma protocol is as follows:

1. P sends a randomized *announcement* (A) to V,
2. V issues a uniform random *challenge* (c) to P, and then
3. P computes a *response* (v) for V.

The transcript of a sigma protocol is the announcement-challenge-response triple (A, c, v).

Definition 9. A sigma protocol (P, V) is (*perfectly*, *statistically*, or *computationally*) *c-simulatable* on L_R if there exists a (perfect, statistical, or computational) simulator $\mathrm{S_V}$ for honest V such that, for any c in the domain of possible challenges, if $C^{-1}(c)$ denotes the subset of inputs $s \in L_R$ to which $c \in C(s)$ is a possible challenge from honest V, then the ensemble $\{\mathrm{S_V}(s, c)\}_{s \in C^{-1}(c)}$ is (perfectly, statistically, or computationally) indistinguishable from the ensemble $\{\mathsf{View}_{\mathrm{P,V}}(s) \mid \text{V issues challenge } c \in C(s)\}_{s \in C^{-1}(c)}$.

In other words, a sigma protocol (P, V) is c-simulatable if there is a simulator S_{V} for honest V that takes both $s \in L_R$ and $c \in C(s)$ as input, and outputs transcripts in which the common input is s and the challenge is c. Another common name for this property is *special honest-verifier zero-knowledge*. The following observation is immediate.

Observation 2.4. If (P, V) is (perfectly, statistically, or computationally) c-simulatable on L_R, then (P, V) is honest-verifier (perfect, statistical, or computational) zero-knowledge on L_R.

Definition 10. A sigma protocol (P, V) is *k-extractable* on L_R if there exists an efficient algorithm that, on input a sequence of k accepting transcripts of (P, V) on the same common input $s \in L_R$ and in which P^* uses the same announcement A but V issues pairwise distinct random challenges $c_1, \ldots, c_k$, outputs a witness $w' \in \mathcal{W}_R(s)$ with probability overwhelming in $|s|$.

In other words, a sigma protocol (P, V) is k-extractable if the universal knowledge extractor E_{P}, having rewinding black box oracle access to honest P, can (essentially) always output a witness after rewinding P to an earlier state at most $k - 1$ times. The special $k = 2$ case of this property is often called *special soundness*. Again, the following observation is immediate.

Observation 2.5. If (P, V) is k-extractable on L_R for some constant $k \in \mathbb{N}^+$, then (P, V) is a system for proofs of knowledge for L_R.

Both c-simulatability and 2-extractability are invariant under parallel composition [Dam11; Lemma 1]; in particular, given c-simulatable and 2-extractable protocols (P, V) and $(\mathrm{P}', \mathrm{V}')$, the protocol obtained by running (P, V) and $(\mathrm{P}', \mathrm{V}')$ in parallel is a c-simulatable and 2-extractable sigma protocol. Moreover, it is easy to verify that, if the challenge domains $C(s)$ and $C'(s')$ from which V and V′ issue their uniform random challenges are the same, then the verifier in such a parallel composition can issue *a single challenge* from $C(s) = C'(s')$ to be used in both parallel sub-protocols. In this case, the soundness bound from Theorem 2.6 is useful.

Theorem 2.6 (Cramer, 1996 [Cra96; Proposition 2.1]). *If (P, V) is a sigma protocol that is both c-simulatable and 2-extractable on L_R, then (P, V) is a system for honest-verifier zero-knowledge proofs of knowledge for L_R. Moreover, if s is the common input and $C(s)$ is the set of possible challenges for common input s, then (P, V) has knowledge error function $\kappa(s) = 1/|C(s)|$.*

The proof of Theorem 2.6 uses the following technical lemma.

Lemma 2.7. *Let (P,V) be a sigma protocol whose transcripts on common input $s \in S$ are elements of $A(s) \times C(s) \times V(s)$. For a given prover P^*, let $D_A \colon \mathbb{N} \to A(s)$ be the function describing the announcement output by P^* in its i^{th} invocation, and let $E \colon \mathbb{N} \times C(s) \to V(s)$ be the function that describes the response of P^* in its i^{th} invocation when the challenge is $c \in C(s)$. If $\Pr[1 \leftarrow \langle P^*, V\rangle(s)] = \epsilon(s)$, then for $i \in \mathbb{N}^+$, we have, with probability exceeding $\epsilon(s)/2$, that $(D_A(i), c, E(i,c))$ is accepting for at least a fraction $\epsilon(s)/2$ of the $c \in C(s)$.*

Proof. Let H denote the subset of announcements $A \leftarrow D_A(i)$ from P^* such that $(A, c, E(i,c))$ is accepting for at least a fraction $\epsilon(s)/2$ of the $c \in C(s)$, and let h be the probability (from the perspective of V) that D_A outputs an element from H in a given invocation. Then $\epsilon < h + (1-h)\epsilon/2$ so that $h > \epsilon/(2(1-\epsilon/2)) \geq \epsilon/2$, as desired. ∎

Proof (of Theorem 2.6). That (P,V) is a system for honest-verifier zero-knowledge proofs of knowledge for L_R follows directly from Observations 2.4 and 2.5; hence, we only need to prove that (P,V) has knowledge error function $\kappa(s) = 1/|C(s)|$. It is clear that $\kappa(s) \geq 1/|C(s)|$, since a prover P^* that responds correctly for one and only one challenge $c \in C(s)$ for any given announcement A will cause V to accept with probability $1/|C(s)|$, yet $\mathrm{E}_{\mathrm{P}^*}$ will never succeed to extract a witness from P^*. Now, assume that $\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V}\rangle(s)] = \epsilon(s)$, where $\epsilon(s) = (1/|C(s)|) + \mu(s)$ and $\mu(s)$ is non-negligible in $|s|$.

As in the setup for Lemma 2.7, let $D_A \colon \mathbb{N} \to A(s)$ be the function describing the announcement output by P^* in its i^{th} invocation, and let $E \colon \mathbb{N} \times C(s) \to V(s)$ be the function that describes the response of P^* in its i^{th} invocation when the challenge is $c \in C(s)$. Upon receiving an announcement $A_1 \leftarrow D_A(1)$ from P^*, $\mathrm{E}_{\mathrm{P}^*}$ can probe P^* with a random challenge $c_1^{(1)} \in C(s)$ to learn $E(1, c_1^{(1)})$ and, through rewinding, $\mathrm{E}_{\mathrm{P}^*}$ can probe P^* with a second random challenge $c_1^{(2)} \in C(s) \setminus \{c_1^{(1)}\}$ to learn $E(1, c_1^{(2)})$. Likewise, if $\mathrm{E}_{\mathrm{P}^*}$ runs the protocol to completion, it can start a new invocation and, thereby, receive a second announcement $A_2 \leftarrow D_A(2)$ and response $E(2, c_2^{(1)})$ from P^*. The goal is for P^* to locate a triple $(A_i, c_i^{(j)}, c_i^{(k)})$ such that $A_i \leftarrow D_A(i)$ and $c_i^{(j)} \neq c_i^{(k)}$, and such that $(A_i, c_i^{(j)}, E(i, c_i^{(j)}))$ and $(A_i, c_i^{(k)}, E(i, c_i^{(k)}))$ are both accepting, in which case the 2-extractability of (P,V) allows $\mathrm{E}_{\mathrm{P}^*}$ to compute the desired witness with probability overwhelming in $|s|$. For simplicity, we assume that $C(s) = \{0, \ldots, |C(s)| - 1\}$ and, for each $i \in \mathbb{N}^+$, we define a uniform random permutation π_i on the elements of $C(s)$.

The extractor $\mathrm{E}_{\mathrm{P}^*}$ probes P^* for the following responses in the following order:

$E(1,\pi_1(1))$,
$E(1,\pi_1(2))$, $E(2,\pi_2(1))$,
$E(1,\pi_1(3))$, $E(2,\pi_2(2))$, $E(3,\pi_3(1))$,
$E(1,\pi_1(4))$, $E(2,\pi_2(3))$, $E(3,\pi_3(2))$, $E(4,\pi_4(1))$,

and so on, until it locates a triple (i,j,k) such that the transcripts $(D_A(i),\pi_i(j),E(D_A(i),\pi_i(j)))$ and $(D_A(i),\pi_i(k),E(D_A(i),\pi_i(k)))$ are both accepting.

By Lemma 2.7, if $\mathrm{E}_{\mathrm{P}^*}$ queries P^* for $E(i,\pi_i(j))$ for each $i+j<B$, which requires $B(B+1)$ queries, then the expected value of B at completion time is $2/\epsilon(s) + 2/\epsilon(s) = 4/\epsilon(s)$. In particular, $\mathrm{E}_{\mathrm{P}^*}$ succeeds after about $16/\epsilon(s)^2$ probes to P^*, on average. Since $\epsilon(s)$ is a non-negligible function in $|s|$ by assumption, it follows that $16/\epsilon(s)^2 \in \mathit{poly}(|s|)$, as desired. ∎

2.5.2 Security analysis for Schnorr's protocol

We now prove that Schnorr's protocol is, in fact, a system for honest-verifier zero-knowledge proofs of knowledge for the various DL relations arising in $\mathbb{G}$.

Theorem 2.8. *Let $(\mathbb{G},q,g) \leftarrow \mathcal{G}(1^r)$ for a fixed group-generating algorithm $\mathcal{G}$. Schnorr's protocol, as depicted in Figure 2.4, is a system for honest-verifier perfect zero-knowledge proofs of knowledge for the language induced by the DL relation $R_g = \{(h,x) \in \mathbb{G}\times\mathbb{Z}_q \mid h = g^x\}$. Furthermore, it is a c-simulatable and 2-extractable sigma protocol and so has constant knowledge error function $\kappa(\mathbb{G},q,g,h) = 1/q$.*

Proof. **Complete:** $g^v = g^{cx+r} = g^{xc+r} = (g^x)^c g^r = h^c g^r = h^c A$.

2-Extractable: Given two accepting transcripts $(A,c_1,v_1) \in \mathbb{G}^* \times (\mathbb{Z}_q)^2$ and $(A,c_2,v_2) \in \mathbb{G}^* \times (\mathbb{Z}_q)^2$ that use the same announcement A but distinct challenges c_1 and c_2, we have that $g^{v_1} = h^{c_1}A$ and $g^{v_2} = h^{c_2}A$ so that $g^{v_1-v_2} = h^{c_1-c_2}$. Taking logarithms with respect to base g reveals that $v_1 - v_2 = (c_1-c_2)\log_g h$ with $c_1 \neq c_2$; thus, $\mathrm{E}_{\mathrm{P}^*}$ can easily compute $x = (v_1-v_2)/(c_1-c_2)$. This establishes that (P,V) is indeed 2-extractable.

***c*-Simulatable:** Given $(\mathbb{G},q)$ and $(g,h,c) \in \mathbb{G}^*\times\mathbb{G}\times\mathbb{Z}_q$ as input, S_{V} chooses a random response $v \in_{\mathrm{R}} \mathbb{Z}_q$, and then it computes the corresponding announcement $A = g^v/h^c$. (In the unlikely event that $A = 1$, S_{V} chooses a different $v \in_{\mathrm{R}} \mathbb{Z}_q$ and tries again.) The simulated transcript

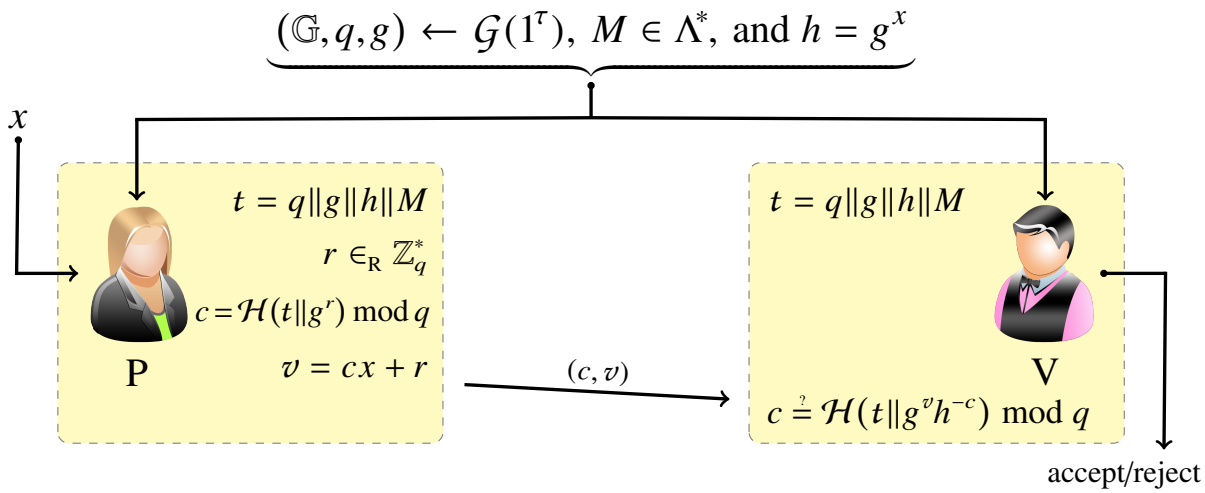

Figure 2.5: A system for non-interactive honest-verifier zero-knowledge arguments of knowledge of a DL obtained from Schnorr's protocol [Sch89] using the Fiat-Shamir transform [FS86]. The protocol is denoted by SPK$\{ x : h = g^x \}(M)$.

is $(A, c, v) \in \mathbb{G}^* \times (\mathbb{Z}_q)^2$. We must prove that S_V's output follows the same probability distribution as the transcripts in V's aggregate view of a real interaction using the same common input $(\mathbb{G}, q, g, h)$ and the same challenge c. But this is clearly the case: since the values in $(\mathbb{G}, q, g, h, c, v)$ uniquely determine A, and since each of these values but v is fixed before the simulation commences, it suffices to note that the one and only free variable, v, is distributed uniformly at random in both the real transcripts and the simulated transcripts. (In the real transcripts, P computes $v = cx + r$ for a uniform random $r \in \mathbb{Z}_q^*$ and, in the simulated transcripts, S_V chooses $v \in_R \mathbb{Z}_q$ directly.)

By inspection, (P, V) is a public-coin sigma protocol; hence, the claimed knowledge error function $\kappa(\mathbb{G}, q, g, h) = 1/q$ follows from Theorem 2.6. ∎

We also point out that, since the set $C(s) = \mathbb{Z}_q$ of possible challenges for a given common input $(\mathbb{G}, q, g, h)$ to Schnorr's protocol has cardinality super-polynomial (indeed, exponential) in $\tau \approx \lg q$, the Fiat-Shamir transform applies to produce a system for non-interactive zero-knowledge arguments of knowledge for the languages induced by DL relations in $\mathbb{G}$. In particular, we can instantiate the protocol denoted by SPK$\{ x : h = g^x \}(M)$ as follows: Given $M \in \Lambda^*$,

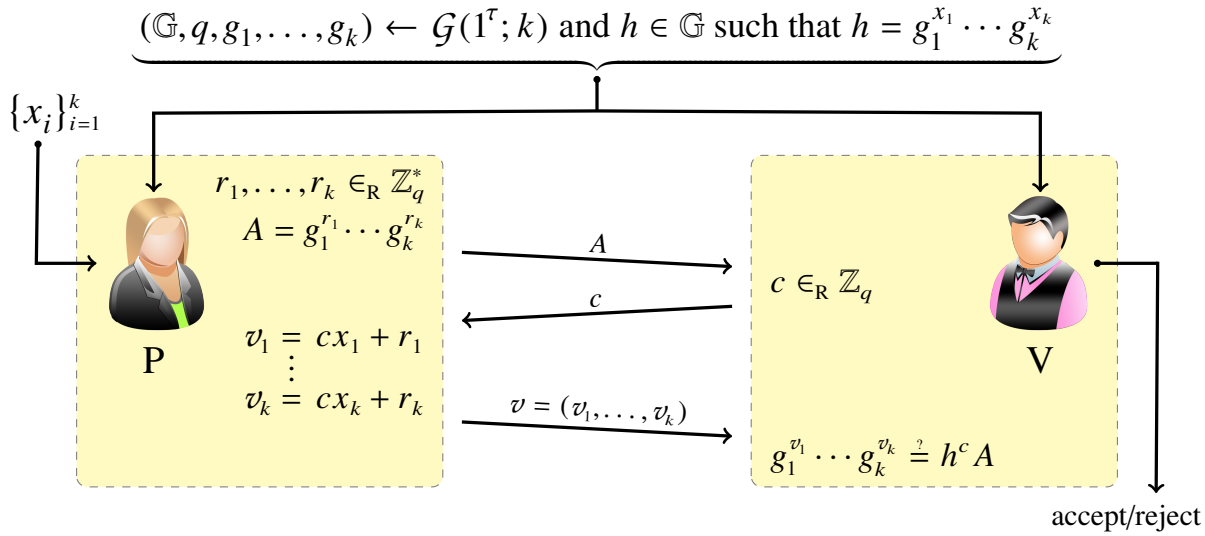

Figure 2.6: A system for honest-verifier perfect zero-knowledge proofs of knowledge of a k-DLREP due to Brands [Bra00]. An accepting transcript is a tuple $(A, c, v_1, \ldots, v_k) \in \mathbb{G}^* \times (\mathbb{Z}_q)^{k+1}$ such that the verification equation $g_1^{v_1} \cdots g_k^{v_k} = h^c A$. The protocol is denoted by $\mathrm{PK}\{(x_1, \ldots, x_k) : h = g_1^{x_1} \cdots g_k^{x_k}\}$.

(i) P chooses $r \in_R \mathbb{Z}_q^*$, and then (ii) P computes the challenge $c = \mathcal{H}(q \| g \| h \| M \| g^r) \bmod q$ and the response $v = cx + r$, and (iii) P outputs the proof $(M, c, v) \in \Lambda^* \times (\mathbb{Z}_q)^2$. To verify such a proof, V simply checks if $c = \mathcal{H}(q \| g \| h \| M \| g^v h^{-c}) \bmod q$.

As random oracles (and, we hope, cryptographically secure hash functions) are *collision intractable* [Bra00; Definition 2.3.1], P need not send the announcement A to V; indeed, if $A \neq g^v h^{-c}$, then with probability overwhelming in τ, we have $c \not\equiv \mathcal{H}(q \| g \| h \| M \| g^v h^{-c}) \bmod q$. The Fiat-Shamir transformed Schnorr protocol is illustrated in Figure 2.5.

2.5.3 Brands' protocol for DLREPs

Schnorr's protocol generalizes naturally to a system for proofs of knowledge for the languages induced by k-DLREP relations in $\mathbb{G}$ for any constant $k \in \mathbb{N}^+$. Figure 2.6 illustrates such a generalization due to Brands [Bra00; §2.4.3].

Theorem 2.9. *Let* $(\mathbb{G}, q, g_1, \ldots, g_k) \leftarrow \mathcal{G}(1^\tau; k)$ *for a fixed group-generating algorithm* $\mathcal{G}$. *Brands' protocol, as depicted in Figure 2.6, is a system for honest-verifier perfect zero-knowledge proofs of knowledge for the language induced by the* k*-DLREP relation* $R = \{(h, x_1, \ldots, x_k) \in \mathbb{G} \times (\mathbb{Z}_q)^k \mid h = g_1^{x_1} \cdots g_k^{x_k}\}$. *Furthermore, it is a* c*-simulatable and* 2*-extractable sigma protocol and so has constant knowledge error function* $\kappa(\mathbb{G}, q, g_1, \ldots, g_k, h) = 1/q$.

The proof of Theorem 2.9 is a straightforward generalization of that for Theorem 2.8.

Chapter 3

Batch proof and verification

This chapter introduces batch verification and systems for batch zero-knowledge proofs of knowledge for NP-languages. It also presents several batch verifiers for DL relations in prime-order groups and describes a new methodology for converting such batch verifiers into systems for conjunctive batch zero-knowledge proofs of knowledge for languages induced by linear relations.

3.1 Batch tests and batch verifiers

Fix a finite alphabet Λ, let S and W be subsets of Λ^*, and let $R \subseteq S \times W$ be an NP-relation. An *n-instance* for R refers to any sequence of n instances $(s_i, w_i)_{i=1}^n$ for R, and a *batch instance* for R refers to any n-instance for R with $n \geq 2$. The quantity n is called the *fan-in* of the n-instance. An n-instance is *correct* if $R(s_i, w_i) = 1$ for every $i \in [1, n]$ and it is *incorrect* otherwise. If $\mathcal{I}$ is an incorrect n-instance for R, then any component instance (s_j, w_j) of $\mathcal{I}$ for which $R(s_j, w_j) = 0$ is called a *bad* component instance.

Intuitively, a batch verifier is a probabilistic test that tries to verify the correctness of an n-instance faster than doing each of the n component verifications individually. Bellare, Garay, and Rabin [BGR98a, BGR98b] studied the batch verification problem, proposing several batch verifiers for fixed-base DL relations in prime-order groups and providing the first (and, apparently, only)

formal treatment of batch verifiers in the literature. Below we review Bellare et al.'s batch verifiers, each of which is suitable for verifying n-instances for a *particular DL relation* in a *particular finite group*; thus, each of their batch verifiers comes with an implicit, finite upper bound on the fan-in of n-instances that it can check. We wish to study the asymptotic behaviour of batch verifiers as the fan-ins of their inputs grow without bound. We therefore extend Bellare et al.'s formal model so as to define batch verifiers with respect to *countably infinite families* $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ in which each $R_\tau \subseteq S_\tau \times W_\tau$ is a finite NP-relation. In particular, we assume that $S_\tau \subseteq \Lambda^{p(\tau)}$ for some fixed, positive integer-valued function $p(\tau) \in poly(\tau)$. Given such an infinite family $\mathcal{R}$ and an index $\tau \in \mathbb{N}^+$, we call any n-instance for R_τ an (n, τ)-*instance* (or simply a *batch instance*) for $\mathcal{R}$. We assume that, given any batch instance $\mathcal{I}$ for $\mathcal{R}$, it is easy to determine the index τ of the particular relation R_τ that $\mathcal{I}$ is a batch instance for.

3.1.1 The naive verifier

We now introduce the *naive verifier* for $\mathcal{R}$ as a baseline against which to compare the expected computation cost of competing verifiers. Definition 12 below requires that every batch verifier has expected computation cost asymptotically lower than that of its corresponding naive verifier.

Definition 11. The *naive verifier* for $\mathcal{R}$ is the algorithm that, given an (n, τ)-instance $(s_i, w_i)_{i=1}^n$ for $\mathcal{R}$, evaluates *each* instance using the lowest-cost algorithm available, and then outputs 1 if no instance was incorrect and outputs 0 otherwise.

The naive verifier does not merely invoke the lowest cost algorithm for a *single* R_τ instance n times; rather, it uses the lowest cost algorithm to evaluate R_τ on the entire (n, τ)-instance simultaneously. (In other words, given an (n, τ)-instance for $\mathcal{R}$, the naive verifier always evaluates the entire sequence $R_\tau(s_1, w_1), \ldots, R_\tau(s_n, w_n)$, without taking any "shortcuts", but it does so as efficiently as possible.) The naive verifier never makes mistakes: it always outputs 1 given a correct batch instance (perfect completeness) and it always outputs 0 given an incorrect batch instance (perfect soundness). Moreover, if the given batch instance is incorrect, then not only does the naive verifier always determine this, but, in doing so, it automatically learns *which* of its component instances are bad.

A *batch verifier* for $\mathcal{R}$, in contrast, tries to determine $\prod_{i=1}^{n} R_\tau(s_i, w_i)$ directly, at a much lower cost than that of evaluating the entire sequence $R_\tau(s_1, w_1), \ldots, R_\tau(s_n, w_n)$. If the batch verifier outputs 1, then the given batch instance is *probably* correct. If the batch verifier outputs 0, then the given batch instance is *definitely* incorrect. Unlike the naive verifier, the batch verifier in this latter case cannot necessarily infer anything about *which* particular component instances are bad. Zaverucha and Stinson [ZS09] observed that locating the bad instances in a known incorrect batch instance is a special case of combinatorial group testing; their paper suggests and analyzes several algorithms from the group testing literature that are suitable for identifying the bad instances in an incorrect batch [ZS09; Section 2].

3.1.2 Defining batch verifiers

In the following discussion and definitions, $\mathcal{B}_\tau$ denotes the set of batch instances for R_τ, $\mathcal{B} = \bigcup_{\tau \in \mathbb{N}^+} \mathcal{B}_\tau$ denotes the set of batch instances for $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$, and $\mathcal{V}_\tau(n)$ denotes the expected computation cost for the naive verifier to check a random (n, τ)-instance for $\mathcal{R}$.

A *batch test* for $\mathcal{R}$ refers to any Boolean-valued PPT algorithm $\mathbf{R}\colon \mathcal{B} \to \{0,1\}$ taking as input batch instances for $\mathcal{R}$. The naive verifier for $\mathcal{R}$ is therefore always a batch test for $\mathcal{R}$. If a batch test $\mathbf{R}$ has the properties listed in Definition 12, then we call it a batch verifier for $\mathcal{R}$.

Definition 12. Let $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ be an infinite family of NP-relations. A batch test $\mathbf{R}\colon \mathcal{B} \to \{0,1\}$ is a *batch verifier* for $\mathcal{R}$ if there exists a constant $\delta \in [0, 1/2]$ and a function $\lambda\colon \mathbb{N}^+ \to \mathbb{R}^+$, with $|\lambda(\tau) - \delta|$ negligible in τ, such that, for every positive integer-valued function $n(\tau) \in \mathit{poly}(\tau)$, if $\mathcal{I} \in \mathcal{B}_\tau$ is an $(n(\tau), \tau)$-instance for $\mathcal{R}$, then $\mathbf{R}$ provides the following three guarantees.

1. **Complete:** If $\mathcal{I}$ is correct, then $\Pr[1 \leftarrow \mathbf{R}(\mathcal{I})] = 1$.
2. **Sound:** If $\mathcal{I}$ is incorrect, then $\Pr[1 \leftarrow \mathbf{R}(\mathcal{I})] \leq \lambda(\tau)$.
3. **Asymptotically efficient:** $\mathbf{R}(\mathcal{I})$ has expected computation cost in $o(\mathcal{V}_\tau(n(\tau)))$.

The smallest function $\lambda\colon \mathbb{N}^+ \to \mathbb{R}^+$ satisfying the soundness bound in Definition 12 is called the *soundness error function* for $\mathbf{R}$, and the constant δ is its *absolute soundness error*. The soundness error function measures the probability that $\mathbf{R}$ outputs 1 given an incorrect (n, τ)-instance as input. The probability is over τ and the random coin tosses of $\mathbf{R}$; in particular, assuming n

is appropriately bounded by some fixed polynomial in τ, the same soundness error probability must apply to all (n, τ)-instances. Depending on the verifier, δ may be zero or it may be a small constant, which is sometimes specified using a *soundness parameter* $\lambda_0 \in \mathbb{N}^+$. For $\mathbf{R}$ to be useful in practice, δ must be small enough that $\mathbf{R}$ will almost certainly *never* — throughout its deployed lifetime — output 1 on input an incorrect batch instance. If δ is not sufficiently small for this to be the case, then we can construct a new batch verifier with lower absolute soundness error by running several independent trials of $\mathbf{R}$ and then outputting 1 only if each trial outputs 1. The Chernoff bound [Gol01; §1.2] ensures that the soundness error of the resulting batch verifier drops off exponentially in the number of trials.[12] Usually, decreasing δ correspondingly increases the expected cost to evaluate $\mathbf{R}$; thus, the soundness parameter (if there is one) should be selected with care.

The batch verifiers that we consider in this section all have positive absolute soundness error; for example, we will see batch verifiers with constant soundness error function $2^{-\lambda_0}$ and others with soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$ for a given soundness parameter $\lambda_0 \in \mathbb{N}^+$ and τ-bit prime q. Here we can set λ_0 quite low by cryptographic standards, say $\lambda_0 = 40$ or 60 so that $\delta = 2^{-40}$ or 2^{-60}, and still obtain acceptable soundness error for many practical applications. (Note that we can safely do this only because the soundness error is independent of the particular input instance under consideration.)

Definition 12 is similar to the definition for batch verifiers that Bellare et al. [BGR98b; Definition 2.1] proposed; however, the new definition differs from the old one in two important respects. First, the new definition allows much greater flexibility in the acceptable soundness error. Bellare et al.'s definition insists that the soundness error be a particular, positive probability (namely, $\lambda(\tau) \leq 2^{-\lambda_0}$ regardless of $\tau \in \mathbb{N}^+$); however, it is perfectly meaningful (indeed, potentially useful) to consider the behaviour of batch verifiers with other soundness errors. Our definition accommodates this by representing the soundness error as a *function* rather than as a scalar. In practice, we often still aim for soundness error below a concrete bound of $2^{-\lambda_0}$, but we can tolerate and reason about batch tests with any soundness error and applied to any sized batch instance. Notably, our model can accommodate batch verifiers with soundness error functions in $o(1)$ so that $\delta = 0$.

[12] Note that our insistence that δ comes from $[0, 1/2]$ in Definition 12 is somewhat arbitrary: as $\mathbf{R}$ has perfect completeness, the same Chernoff bound argument applies for any constant $\delta \in [0, 1)$. On the other hand, if δ gets too close to 1, then the required number of trials can grow prohibitively large for use in practical applications. Having $\delta \leq 1/2$ guarantees that we can get absolute soundness error at most $2^{-\lambda_0}$ by invoking $\mathbf{R}$ at most λ_0 times.

The second difference between our Definition 12 and Bellare et al.'s definition [BGR98b; Definition 2.1] is that our definition requires that every batch verifier be asymptotically faster than its corresponding naive verifier. This is, of course, exactly what Bellare et al. had in mind; however, it is not possible to formulate such a requirement in their model, which provides no meaningful way to speak of infinite sequences of n-instances "growing large" without bound.

Definition 12 is inherently asymptotic. In practice, we are also concerned with the concrete performance and soundness error of a batch verifier evaluated on an (n, τ)-instance for a given choice of n and τ. We therefore propose Definition 13, which is closer to Bellare et al.'s definition.

Definition 13. A batch verifier $\mathbf{R}\colon \mathcal{B} \to \{0,1\}$ is a (n, τ, λ_0)-*batch verifier* for $\mathcal{R}$ if, on input an (n, τ)-instance, its expected computation cost is strictly less than $\mathcal{V}_\tau(n)$ and its soundness error function satisfies $\lambda(\tau) \leq 2^{-\lambda_0}$.

3.1.3 Families of linear relations

An infinite family of relations $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ is a *family of linear relations* if each component relation R_τ describes a linear map; that is, if, for each $\tau \in \mathbb{N}^+$, (i) $R_\tau \subseteq S_\tau \times W_\tau$ for vector spaces S_τ and W_τ, (ii) there exists a function $f_\tau\colon S_\tau \to W_\tau$ with $R_\tau(s, w) = 1$ if and only if $f_\tau(s) = w$, and (iii) there exist efficient operators $\odot_\tau\colon S_\tau \times S_\tau \to S_\tau$ and $\oplus_\tau\colon W_\tau \times W_\tau \to W_\tau$, such that, for all $s_1, s_2 \in S_\tau$,

1. **Linear:** $f_\tau(s_1) \oplus_\tau f_\tau(s_2) = f_\tau(s_1 \odot_\tau s_2)$, and
2. **Scalar multiplication:** $k f_\tau(s_1) = f_\tau(k s_1)$ for any integer scalar k.

Note that there need not exist an efficient algorithm to *evaluate* the functions $(f_\tau)_{\tau \in \mathbb{N}^+}$; all that is needed is for such functions to exist. There must, however, be algorithms, both efficient with respect to τ, to evaluate the operators $\odot_\tau$ and $\oplus_\tau$.

The DL relations. Our canonical example of a (finite) linear relation is the base-g DL relation in $\mathbb{G}$; that is, the relation $R_g = \{(h, x) \in \mathbb{G} \times \mathbb{Z}_q \mid h = g^x\}$. Here, the linear map that R describes is the base-g DL function $f_\tau(\cdot) = \log_g(\cdot)$ mapping each $h \in \mathbb{G}$ to the exponent $x \in \mathbb{Z}_q$ satisfying $h = g^x$. Thus, we have that $S_\tau = \mathbb{G}$ and $W_\tau = \mathbb{Z}_q$, where $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$, and that the operators $\odot_\tau$ and $\oplus_\tau$ respectively denote multiplication in $\mathbb{G}$ and addition in $\mathbb{Z}_q$.

3.1.4 Batch tests for families of linear relations

This section presents several batch tests for families of linear relations. We present each batch test using the notation customary for the DL relations induced by $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ with $\tau \in \mathbb{N}^+$; in addition, we analyze the soundness error of each batch test in this setting and, in doing so, we prove that all but one of them is a batch verifier for the DL relations in prime-order groups. Thus, except where otherwise specified, our infinite family of relations in this section is always a family of fixed-base DL relations $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ where, for each $\tau \in \mathbb{N}^+$, $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ and $R_\tau = \{(h, x) \in \mathbb{G} \times \mathbb{Z}_q \mid h = g^x\}$. We write $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ to indicate that $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ is constructed in this way using the group-generating algorithm $\mathcal{G}$. In cryptographic applications, the DL assumption would typically hold for $\mathcal{G}$, possibly among other intractability assumptions.

Observation 3.1. The naive verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ has expected cost $\text{ExpCost}^{(n)}_{\mathbb{G}}(\tau)$, which approaches $\tau + n\tau/\lg(n\tau)$ multiplications in $\mathbb{G}$ as τ and n tend to infinity, subject to $\lg n \in o(\tau)$.

More generally, we write $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*; k)$ to denote that $\mathcal{R}$ is an infinite family of fixed-basis k-DLREP relations induced by $\mathcal{G}$. The naive verifier for this latter family has expected computation cost $\text{ExpCost}^{(n)}_{\mathbb{G}}((k, \tau))$, which approaches $k\tau + kn\tau/\lg(n\tau)$ multiplications in $\mathbb{G}$ as τ and n tend to infinity, subject to $\lg n \in o(\tau)$.

Note that the batch tests we present below are applicable to linear relations other than the fixed-base DL and fixed-basis k-DLREP relations in prime-order groups; however, for each new relation that one wishes to apply a given batch test to, the soundness error must be reassessed.

3.1.4.1 The atomic random subset (ARS) test

Consider the following batch test.

> **Product test:** On input an (n, τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, compute the instance (h_0, x_0) in which $h_0 = \prod_{i=1}^n h_i$ and $x_0 = \sum_{i=1}^n x_i$, and then output 1 if (h_0, x_0) is correct (that is, if $h_0 = g^{x_0}$) and output 0 otherwise.

It is easy to check that the product test is complete: if $h_i = g^{x_i}$ for each $i = 1, \ldots, n$, then $\prod_{i=1}^{n} h_i = \prod_{i=1}^{n} g^{x_i} = g^{\sum_{i=1}^{n} x_i}$ and the product test will always output 1. Moreover, it is easy to check that the product test is asymptotically efficient: by inspection, its expected cost on input a random (n, τ)-instance for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ is just $\text{ExpCost}_{\mathbb{G}}(\tau)$ plus an additional $(n-1)$ multiplications in $\mathbb{G}$, which is much lower than the naive verifier's expected cost of $\text{ExpCost}_{\mathbb{G}}^{(n)}(\tau)$. Unfortunately, the product test is not sound for $\mathcal{R}$. For example, choosing any $r \in \mathbb{Z}_q^*$ and setting $x_j = \log_g h_j + r$ and $x_k = \log_g h_k - r$ in an otherwise correct n-instance causes the product test to output 1 despite $h_j \neq g^{x_j}$ and $h_k \neq g^{x_k}$. This attack easily generalizes to larger collections of bad instances; all that is required is that the "error terms" in the bad component instances collectively sum to zero.

The following batch test attempts to thwart such attacks by splitting up the bad instances in a batch. It does this by only applying the product test to a *random subset* of the component instances [BGR98b; §3.1].

> **Atomic random subset (ARS) test:** On input an (n, τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, choose a subset $U \subseteq [1, n]$ uniformly at random, compute the instance (h_0, x_0) in which $h_0 = \prod_{i \in U} h_i$ and $x_0 = \sum_{i \in U} x_i$, and then output 1 if (h_0, x_0) is correct (that is, if $h_0 = g^{x_0}$) and output 0 otherwise.

The ARS test uses (slightly) fewer multiplications than the product test, and it has absolute soundness error $\delta = 1/2$.

Theorem 3.2. *The ARS test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. It has absolute soundness error $\delta = 1/2$ and a constant soundness error function $\lambda(\tau) = 1/2$.*

Proof. Completeness and asymptotic efficiency easily follow by inspection. To prove that the ARS test has soundness error $\lambda(\tau) = 1/2$ for all $\tau \in \mathbb{N}^+$, assume without loss of generality that (h_j, x_j) is a bad instance. Partition the power set of the n component instances into two sets: the set of subsets that contain (h_j, x_j) and the set of subsets that do not. For each subset that contains (h_j, x_j), if the ARS test outputs 1 on that subset, then a simple calculation confirms that the ARS test outputs 0 on the corresponding subset with (h_j, x_j) removed. Similarly, for each subset that does not contain (h_j, x_j), if the ARS test outputs 1 on that subset, then the ARS test outputs 0 on the corresponding subset with (h_j, x_j) added. This establishes that at most half of all subsets

can possibly make the ARS test output 1 when the input is incorrect; hence, the soundness error of the ARS test can be at most $1/2$. To see that the bound $1/2$ is tight, consider the case where (h_j, x_j) is the *only* bad instance. Here the ARS test outputs 1 on *exactly* half the sets: the sets that do not contain (h_j, x_j).[13] ∎

3.1.4.2 The random multiple product (RMP) test

Given a soundness parameter $\lambda_0 \in \mathbb{N}^+$, we can construct a batch verifier for $\mathcal{R}$ with absolute soundness error $2^{-\lambda_0}$ by performing λ_0 independent trials of the ARS test and outputting 1 only if all ARS trials output 1. (Hence the "*atomic*" qualifier on the ARS test.) Bellare et al. call this λ_0-fold parallelized ARS test the *random subset* (RS) test [BGR98b; §3.1]. We prefer to call it the *random multiple product (RMP) test* because it invokes the well-studied *multiple product problem* [Ber02; §6]: Given a set of n group elements $h_1, \ldots, h_n$ and a set of λ_0 subsets $U_1, \ldots, U_{\lambda_0}$ of $[1, n]$, compute the λ_0 products $(\prod_{i \in U_1} h_i), \ldots, (\prod_{i \in U_{\lambda_0}} h_i)$.

> **Random multiple product (RMP) test:** On input an (n, τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, choose λ_0 subsets $U_1, \ldots, U_{\lambda_0} \subseteq [1, n]$, each uniformly at random, compute the (λ_0, τ)-instance $(h_{U_k}, x_{U_k})_{k=1}^{\lambda_0}$ in which $h_{U_k} = \prod_{i \in U_k} h_i$ and $x_{U_k} = \sum_{i \in U_k} x_i$ for each $k = 1, \ldots, \lambda_0$, and then output 1 if $(h_{U_k}, x_{U_k})_{k=1}^{\lambda_0}$ is correct (that is, if $h_{U_k} = g^{x_{U_k}}$ for each $k = 1, \ldots, \lambda_0$) and output 0 otherwise.

We denote the RMP test instantiated with soundness parameter $\lambda_0 \in \mathbb{N}^+$ by $\mathbf{R}_{\text{RMP}}^{(\lambda_0)}$. Note that the last step in the RMP test is to evaluate a "random" (λ_0, τ)-instance for $\mathcal{R}$; thus, the naive verifier for $\mathcal{R}$ has lower cost (*and* lower soundness error) than that of the RMP test whenever $n < \lambda_0$ (assuming also that $\lambda_0 < \tau$). On the other hand, when n is large, the RMP test can have much lower cost than that of its naive counterpart. More precisely, the expected cost to evaluate the RMP test on input a random (n, τ)-instance for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ is at most about $\text{ExpCost}_{\mathbb{G}}^{(\lambda_0)}((n, 1)) + \text{ExpCost}_{\mathbb{G}}^{(\lambda_0)}(\tau)$. Pippenger established [Pip76; Theorem 1] that $\text{ExpCost}_{\mathbb{G}}^{(\lambda_0)}((n, 1))$ converges to $n\lambda_0 / \lg(n\lambda_0)$ as n increases under the assumption that $\lambda_0 \in \omega(\lg n)$. In our case, $\lambda_0 \in \mathbb{N}^+$ is a

[13] We can do *slightly* better by never choosing the empty set; however, the soundness error with this optimization in place is still $2^{n-1}/(2^n - 1)$, which approaches the absolute soundness error $\delta = 1/2$ exponentially fast as n increases.

small constant and so the best we can say is that the expected cost to evaluate the RMP test on input an (n, τ)-instance for $\mathcal{R}$ is fewer than

$$2n\lambda_0/\lg n + 3\tau\lambda_0/2$$

multiplications in $\mathbb{G}$ when n is sufficiently large. Fortunately, this bound is good enough to establish the RMP test as a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. We summarize the results just discussed in a theorem.

Theorem 3.3. *The RMP test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. For a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and a constant soundness error function $\lambda(\tau) = 2^{-\lambda_0}$.*

Concretely, the RMP test is an (n, τ, λ_0)-batch verifier for $\mathcal{R}$ approximately when

$$n\lambda_0/2 + 3\tau\lambda_0/2 < 3n\tau/2$$

so that

$$n(3\tau - \lambda_0) < 3\tau\lambda_0.$$

If $\tau \gg \lambda_0$, then the latter inequality very roughly states that the RMP test is an (n, τ, λ_0)-batch verifier for $\mathcal{R}$ whenever $n > \lambda_0$.

3.1.4.3 The random multiexponentiation (RME) test

Consider the following batch test.

> **Random multiexponentiation (RME) test:** On input an (n, τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, choose $t_1, \ldots, t_n \in_{\text{R}} [0, 2^{\lambda_0} - 1]$, compute the instance (h_0, x_0) in which $h_0 = \prod_{i=1}^n h_i^{t_i}$ and $x_0 = \sum_{i=1}^n t_i x_i$, and then output 1 if (h_0, x_0) is correct (that is, if $h_0 = g^{x_0}$) and output 0 otherwise.

Bellare, Garay, and Rabin call this test the *small exponent (SE) test.* We call it the *random multiexponentiation (RME) test* because it reduces n exponentiations into a random n-base multiexponentiation (albeit with "small exponents"). We denote the RME test instantiated with soundness parameter $\lambda_0 \in \mathbb{N}^+$ by $\mathbf{R}^{(\lambda_0)}_{\text{RME}}$. On input a random (n,τ)-instance for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, the RME test has soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$[14] and expected computation cost $\text{ExpCost}_{\mathbb{G}}((1,\tau),(n,\lambda_0))$, which is fewer than

$$3\tau/2 + 2n\lambda_0/\lg n$$

multiplications in $\mathbb{G}$ when n is sufficiently large.

Theorem 3.4. *The RME test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. For a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.*

Before proving Theorem 3.4, we first prove the following lemma.

Lemma 3.5. *Let $(h_1,\ldots,h_n) \in (\mathbb{G})^n$ and $(x_1,\ldots,x_n) \in (\mathbb{Z}_q)^n$, and let $j \in [1,n]$ such that $h_j \neq g^{x_j}$. For every possible assignment of the $t_i \in [0, 2^{\lambda_0} - 1]$ for $i \in [1,n] \setminus \{j\}$, there exists a unique choice for $t_j \in \mathbb{Z}_q$ such that $\prod_{i=1}^n h_i^{t_i} = g^{\sum_{i=1}^n t_i x_i}$.*

Proof. Suppose that

$$\prod_{i=1}^n h_i^{t_i} = g^{\sum_{i=1}^n t_i x_i}.$$

Setting $u_i = \log_g h_i$ for each $i = 1,\ldots,n$, the left-hand side of this expression is also equal to $\prod_{i=1}^n (g^{u_i})^{t_i} = g^{\sum_{i=1}^n t_i u_i}$; hence,

$$g^{\sum_{i=1}^n t_i u_i} = g^{\sum_{i=1}^n t_i x_i}.$$

Taking logarithms with respect to base g yields

$$\sum_{i=1}^n t_i u_i = \sum_{i=1}^n t_i x_i.$$

[14] If $2^{\lambda_0} > q$, then the exponents t_i should be selected at random from $[0, q-1]$ and not from $[0, 2^{\lambda_0} - 1]$. Otherwise, both the soundness error and the computation cost will be somewhat higher than is necessary.

Now, $u_j - x_j \neq 0$ by assumption, and we see that

$$t_j = (\sum_{i=1, i\neq j}^{n} t_i(u_i - x_i))/(u_j - x_j) \bmod q.$$

■

Proof (of Theorem 3.4). Completeness is by inspection and efficiency follows from the analysis above. To prove that the RME test has soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$, define $u_i = \log_g h_i$ for each $i = 1, \ldots, n$ and assume without loss of generality that (h_j, x_j) is a bad instance, so that $u_j \neq x_j$. By Lemma 3.5, for any fixed assignment of the t_i with $i \in [1, n] \setminus \{j\}$, there exists a *unique* choice for $t_j \in [0, q-1]$ (and, consequently, there exists *at most one* choice for $t_j \in [0, 2^{\lambda_0} - 1]$) for which the RME test outputs 1. Since t_j is selected uniformly at random from $[0, 2^{\lambda_0} - 1]$ independent of the other t_i, the probability of selecting this particular value for t_j is at most $2^{-\lambda_0}$. (If $2^{\lambda_0} > q$ and the t_i are drawn from $\mathbb{Z}_q$, then the probability is exactly $1/q$.) Finally, as this same argument holds for *every* fixed assignment of the t_i with $i \in [1, n] \setminus \{j\}$, it also holds when every t_i is selected uniformly at random. This establishes that the soundness error of the RME test can be at most $2^{-\lambda_0}$. To see that the bound $2^{-\lambda_0}$ is tight, consider the case where (h_j, x_j) is the *only* bad instance. Here the RME test outputs 1 if and only if $t_j = 0$ (see Lemma 3.6 below), which happens with probability $2^{-\lambda_0}$. ■

Concretely, the RME test is an (n, τ, λ_0)-batch verifier for $\mathcal{R}$ approximately when

$$3\tau/2 + n\lambda_0/2 < 3n\tau/2,$$

which is the same as

$$n\lambda_0 < 3(n-1)\tau.$$

The latter inequality holds for all $n \geq 2$, provided $\lambda_0 < 3\tau/2$.

Observe that the expected cost to evaluate the RME test is asymptotically equal to that of the RMP test; nevertheless, the RME test outperforms the RMP test quite handily when n is reasonably small. Notably, whereas the RMP test is *never* a $(\lambda_0, \tau, \lambda_0)$-batch verifier, the RME test *almost always is*. Recalling that the last step of the RMP test is to evaluate a (λ_0, τ)-instance for $\mathcal{R}$, the preceding observation suggests that we might be able to speed up the RMP test by verifying this latter instance using the RME test. Of course, applying a batch test to the output of a

batch test in this manner introduces additional soundness error. To maintain the original absolute soundness error of $\delta = 2^{-\lambda_0}$ in the composed test, the underlying RMP and RME tests each need to be instantiated with their soundness parameters set to $\lambda_0 + 1$. We refer to the resulting batch test as the *RMP⁺ test* and denote it by $\mathbf{R}^{(\lambda_0)}_{\text{RMP}^+}$. For small values of n, the RMP⁺ test is more efficient than the RMP test; however, as n grows sufficiently large, the additional cost from incrementing the soundness parameter in the RMP test eventually exceeds the cost savings due to RME verification. Moreover, neither the RMP test nor the RMP⁺ test *ever* outperforms the RME test for a given choice of (n, τ, λ_0); as such, it almost always makes more sense to simply use the RME test instead of the RMP or RMP⁺ tests.

Modified random multiexponentiation test. We point out the following trivial modification to the RME test, which very slightly improves on its computation cost. The modification follows from Lemma 3.6. Unlike the standard RME test, the modified RME test has the nice property of reducing to standard (non-batch) verification when evaluated on an instance with fan-in $n = 1$. In addition, the modified RME test saves about $\lambda_0/2$ multiplications compared to the standard RME test for $\mathcal{R}$. (Nonetheless, we revert to using the standard RME test in most of our discussions. We do this purely for notational convenience; indeed, in implementations it is usually better, if only slightly, to use the modified RME test.)

Lemma 3.6. *Given as input an incorrect (n, τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ in which only a single component instance, say (h_j, x_j), is bad, the RME test outputs 1 if and only if $t_j = 0$.*

Proof. Define $u_i = \log_g h_i$ for each $i = 1, \ldots, n$. Since

$$\prod_{i=1}^{n} h_i^{t_i} = g^{\sum_{i=1}^{n} t_i x_i},$$

we can take logarithms with respect to base g to find that

$$\log_g \prod_{i=1}^{n} h_i^{t_i} = \sum_{i=1}^{n} t_i x_i$$

or, equivalently, that

$$\sum_{i=1}^{n} t_i (u_i - x_i) = 0.$$

But $u_i - x_i = 0$ for each $i \in [1,n] \setminus \{j\}$ by assumption; hence, $t_j(u_j - x_j) = 0$. Finally, as $u_j - x_j \neq 0$ by assumption, it follows that $t_j = 0$. ∎

We now state the modified RME test:

> **Modified RME test:** On input an (n,τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}(1^*)$, choose $t_2, \ldots, t_n \in_R [0, 2^{\lambda_0} - 1]$, compute the instance (h_0, x_0) in which $h_0 = h_1 \prod_{i=2}^n h_i^{t_i}$ and $x_0 = x_1 + \sum_{i=2}^n t_i x_i$, and then output 1 if (h_0, x_0) is correct (that is, if $h_0 = g^{x_0}$) and output 0 otherwise.

In other words, the modified RME test is exactly like the regular RME test, except with the first exponent t_1 always set to 1. We denote the modified RME test instantiated with soundness parameter $\lambda_0 \in \mathbb{N}^+$ by $\mathbf{R}_{\mathrm{RME}^+}^{(\lambda_0)}$.

Theorem 3.7. *The modified RME test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$. For a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.*

Proof (Sketch). The proof of Theorem 3.7 is almost identical to that of Theorem 3.4. The only difference arises when the first instance in the batch, (h_1, x_1), is bad. If (h_1, x_1) is the *only* bad instance, then because $t_1 \neq 0$ by construction, Lemma 3.6 implies that the RME test will never output 1; otherwise, if there exists some $j > 1$ such that (h_j, x_j) is also bad, then the proof given for Theorem 3.4 holds, with the caveat that the argument must focus on a bad instance such as (h_j, x_j) with $j > 1$, and not on (h_1, x_1). ∎

3.1.4.4 The atomic random m-partition (m-ARP) test

Consider a variant of the ARS test in which the verifier selects a subset $U \subseteq [1,n]$ uniformly at random, but then, unlike in the ARS test, it outputs 1 only if both

$$\prod_{i \in U} h_i = g^{\sum_{i \in U} x_i} \quad \text{and} \quad \prod_{i \in [1,n] \setminus U} h_i = g^{\sum_{i \in [1,n] \setminus U} x_i}.$$

(In other words, rather than choosing a *random subset* of the instances and running the product test on it, the verifier chooses a *random partition* of the instances into two subsets and runs the product test on each of them.) Given an incorrect batch instance that contains only one bad component instance, the ARS test outputs 1 with probability $1/2$ whereas this modified test always outputs 0. In fact, it is easy to see that for any given input, the above modified ARS test has soundness error less than or equal to that of the standard ARS test (albeit, with expected computation cost nearly double that of the standard ARS test).

This idea generalizes naturally to a test using random partitions of $[1,n]$ into m subsets, for any choice of $m \in \mathbb{N}^+$.

> **Atomic random m-partition (m-ARP) test:** On input an (n,τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, partition $[1,n]$ randomly into m subsets $U_1, \ldots, U_m$ by placing each $i \in [1,n]$ into one of $U_1, \ldots, U_m$ uniformly at random, compute the (m,τ)-instance $(h_{U_k}, x_{U_k})_{k=1}^m$ in which $h_{U_k} = \prod_{i \in U_k} h_i$ and $x_{U_k} = \sum_{i \in U_k} x_i$ for each $k = 1, \ldots, m$, and then output 1 if $(h_{U_k}, x_{U_k})_{k=1}^m$ is correct (that is, if $h_{U_k} = g^{x_{U_k}}$ for each $k = 1, \ldots, m$) and output 0 otherwise.

On input a random (n,τ)-instance for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, the expected cost to evaluate the m-ARP test is $\text{ExpCost}_{\mathbb{G}}^{(m)}(\tau)$, plus an additional $(n-m)$ multiplications in $\mathbb{G}$. This is somewhat fewer than

$$3m\tau/2 + (n-m),$$

multiplications in $\mathbb{G}$. If we let m grow large subject to $m \leq n$ and $\lg n \in o(\tau)$, then the expected cost approaches

$$\tau + m\tau/\lg(m\tau) + (n-m)$$

multiplications in $\mathbb{G}$ [Erd61]. When $m = 1$, the m-ARP test reduces to the product test (which is not sound for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$); for any $m \geq 2$, however, it yields a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$.

Theorem 3.8. *The m-ARP test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ whenever $m \geq 2$. For any fixed number of partitions $m \in \mathbb{N}^+$, it has absolute soundness error $\delta = 1/m$ and a constant soundness error function $\lambda(\tau) = 1/m$.*

Proof. Completeness is by inspection and asymptotic efficiency follows from the analysis presented above. To prove that the m-ARP test has a constant soundness error function $\lambda(\tau) = 1/m$, define $u_i = \log_g h_i$ for each $i = 1, \ldots, n$ and assume, without loss of generality, that instance (h_j, x_j) is bad, so that $u_j \neq x_j$. Suppose the partitioning of $[1, n]$ is into pairwise disjoint subsets $U_1, \ldots, U_m \subseteq [1, n]$. Call the subset U_k *bad* if $\prod_{i \in U_k} h_i \neq g^{\sum_{i \in U_k} x_i}$ and call it *good* otherwise. Finally, set $U'_k = U_k \setminus \{j\}$ for each $k = 1, \ldots, m$. There are three cases to consider: (i) no U'_k is bad, (ii) one U'_k is bad, and (iii) several U'_k are bad.

If no U'_k is bad, then it is easy to see that the U_k containing j is bad; moreover, if several U'_k are bad, then at least one U_k is also bad, as $U_k = U'_k$ for all but one value of k. In either case, the m-ARP test clearly outputs 0. For the remaining case, suppose that U'_k is the only bad subset and, moreover, that $U'_k \cup \{j\}$ is good. If $U_k = U'_k \cup \{j\}$, then the m-ARP test outputs 1, despite the input being incorrect. However, because each index j was placed in one and only one of the U_k uniformly at random, the probability that $U_k = U'_k \cup \{j\}$ is just $1/m$. This establishes that the m-ARP test can have soundness error at most $1/m$. To see that the $1/m$ bound is tight, consider an incorrect batch instance that has exactly two bad component instances, say (h_j, x_j) and (h_k, x_k), with $h_j \neq g^{x_j}$ and $h_k \neq g^{x_k}$ yet $h_j h_k = g^{x_j + x_k}$. The m-ARP test outputs 1 for such an instance if and only if the two bad component instances appear in the same subset U_k, which happens with probability $1/m$. ∎

3.1.4.5 The parallel random partition (PRP) test

Given a soundness parameter $\lambda_0 \in \mathbb{N}^+$, we can construct a batch verifier with soundness error less than $2^{-\lambda_0}$ by performing at least $\lambda_0 / \lg m$ independent trials of the m-ARP test and outputting 1 only if each trial outputs 1. (Note that we are choosing the number of trials $\ell = \lceil \lambda_0 / \lg m \rceil$ to be the smallest positive integer ℓ satisfying $m^{-\ell} \leq 2^{-\lambda_0}$.) The expected cost to evaluate all ℓ trials of the m-ARP test concurrently is $\mathrm{ExpCost}_{\mathbb{G}}^{(\ell m)}(\tau)$, plus the cost of evaluating the ℓm products. Determining the cost of these latter products is tricky: we could call the cost $\mathrm{ExpCost}_{\mathbb{G}}^{(\ell m)}((n, 1))$ just to get an easy estimate; however, this bound is quite loose when $m > 2$, as each of the ℓm products involves about n/m multiplicands, rather than the $n/2$ multiplicands one expects in a random multiple product problem whose cost is described by $\mathrm{ExpCost}_{\mathbb{G}}^{(\ell m)}((n, 1))$. It is clear that the actual cost to evaluate these products is always less than or equal to $\ell(n - m)$ multiplications in $\mathbb{G}$; in fact, the cost is likely closer to $\mathrm{ExpCost}_{\mathbb{G}}^{(\ell m)}((2n/m, 1))$, although we emphasize that this

is only a crude approximation. Nonetheless, given an assignment of (n, τ, λ_0), we can use such an approximation to solve for the number of partitions m (and the resulting number of trials ℓ) that minimize the expected cost of the entire verification procedure while obtaining an absolute soundness error of at most $2^{-\lambda_0}$. We call the batch test obtained by parallelizing the m-ARP test using the optimal choice for (m, ℓ) the *parallel random partition (PRP) test.*

Parallel random partition (PRP) test: On input an (n, τ)-instance $(h_i, x_i)_{i=1}^n$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$,

1. solve for m and $\ell = \lceil \lambda_0 / \lg m \rceil$ to minimize the cost of performing steps 2 and 3 below, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$,
2. for each $j = 1, \ldots, \ell$, partition $[1, n]$ into m subsets $U_{j1}, \ldots, U_{jm}$ by placing each $i \in [1, n]$ into one of the U_{jk} uniformly at random, and then
3. compute the $(\ell m, \tau)$-instance $((h_{U_{1k}}, x_{U_{1k}})_{k=1}^m)_{j=1}^\ell$ in which $h_{U_{jk}} = \prod_{i \in U_{jk}} h_i$ and $x_{U_{jk}} = \sum_{i \in U_{jk}} x_i$ for each $j = 1, \ldots, \ell$ and $k = 1, \ldots, m$.

Output 1 if $((h_{U_{1k}}, x_{U_{1k}})_{k=1}^m)_{j=1}^\ell$ is correct (that is, if $h_{U_{jk}} = g^{x_{U_{jk}}}$ for each $j = 1, \ldots, \ell$ and $k = 1, \ldots, m$) and output 0 otherwise.

We denote the PRP test instantiated with soundness parameter $\lambda_0 \in \mathbb{N}^+$ by $\mathbf{R}_{\text{PRP}}^{(\lambda_0)}$. The PRP test is both complete and sound by inspection and it is asymptotically efficient because the underlying m-ARP test is asymptotically efficient and because $\ell \leq \lambda_0$. This proves that the PRP test is a batch verifier for $\mathcal{R}$. We summarize this result in a theorem.

Theorem 3.9. *The PRP test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. For a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function satisfying $\lambda(\tau) \leq 2^{-\lambda_0}$ for all $\tau \in \mathbb{N}^+$.*

The exact soundness error function for the PRP test depends on the particular choice of m (and ℓ), which in turn depends on how one approximates the cost of computing the ℓm products. Using the approximation suggested above, we get that the PRP test is a (n, τ, λ_0)-batch verifier approximately when

$$\text{ExpCost}_{\mathbb{G}}^{(\ell m)}(\tau) + \text{ExpCost}_{\mathbb{G}}^{(\ell m)}((2n/m, 1)) \leq 3n\tau/2$$

multiplications in $\mathbb{G}$. For small values of m and n, we can approximate this inequality using

$$3m\lambda_0\tau/2 \lg m + \lambda_0(n - m)/\lg m \leq 3n\tau/2,$$

which, for $\tau \gg \lambda_0$, is roughly equivalent to

$$m/\lg m \leq n/\lambda_0.$$

If $n > k\lambda_0$ for some positive integer $k > 1$, then any choice of $m \in [2, k]$ will satisfy the latter inequality.

When n is large, the cost of the PRP test can be very low compared to that of the RMP and RME tests. Moreover, as with the RMP test, the final step in the PRP test involves verifying a new "random" batch instance for $\mathcal{R}$; however, the fan-in ℓm of this latter batch instance is potentially quite large and, unlike with the RMP test, this fan-in actually *increases with* n, at least until n grows exponentially large relative to λ_0. When ℓm is very large, a recursive application of the PRP test might be the most efficient way to verify that the latter $(\ell m, \tau)$-instance is correct. However, for the values of ℓm that are likely to arise in practice, the RME test is usually more efficient. Bellare et al. called this latter combination of the PRP and RME tests the *bucket test* [BGR98b; §3.4]; we call it the *PRP⁺ test* and denote it by $\mathbf{R}^{(\lambda_0)}_{\text{PRP}^+}$. (Note that, as in the RMP⁺ test, we must instantiate the underlying PRP and RME tests with soundness parameter $\lambda_0 + 1$.) The PRP⁺ is considerably more efficient than the RMP and RME tests when the fan-in is very large.

3.1.5 Comparison of batch verifiers

Table 3.1 compares the expected computation cost for various batch verifiers for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ when $\tau = 256$ and $\lambda_0 = 40$ (so that $\delta \leq 2^{-40}$). The RME test is most efficient when the fan-in n is relatively small and the PRP⁺ is most efficient when the fan-in n is large.

3.1.6 Batch verifying Schnorr's protocol

Recall Schnorr's protocol (Figure 2.4 on Page 30). Schnorr's protocol is a system for honest-verifier zero-knowledge proofs of knowledge for the languages induced by $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. The common input to P and V in Schnorr's protocol is $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ and $h \in \mathbb{G}$, and the transcript of the resulting interaction is $(A, c, v) \in \mathbb{G}^* \times (\mathbb{Z}_q)^2$ such that $g^v = h^c A$; thus, to verify a run of Schnorr's protocol, V checks if the transcript is in a particular NP-relation $R = \{(A, c, v) \in \mathbb{G}^* \times (\mathbb{Z}_q)^2 \mid g^v = h^c A\}$ induced by the common input $(\mathbb{G}, q, g, h)$.

Table 3.1: The (approximate) expected computation cost for various batch verifiers for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. In each case, we measure the cost of a batch verifier as the number of multiplications in $\mathbb{G}$ required to get soundness error less than 2^{-40} on input a random $(n, 256)$-instance for $\mathcal{R}$. (Thus, $\tau = 256$ and $\lambda_0 = 40$ so that $\delta = 2^{-40}$.) For each fan-in n that the table displays, the cell containing the lowest expected cost to check (n, τ)-instances is highlighted.

n	5	10	50	100	500	1 000	5 000	10 000	50 000
Naive	896	1 536	6 656	13 056	64 256	128 256	640 256	1 280 256	6 400 256
RMP	5 476	5 576	6 376	7 376	15 376	25 376	105 376	205 376	1 005 376
RME	484	584	1 384	2 384	10 384	20 384	100 384	200 384	1 000 384
PRP	10 292	10 422	11 416	12 416	20 416	28 115	70 402	112 768	377 980
RMP$^+$	1 327	1 429	2 250	3 275	11 475	21 725	103 724	206 225	1 026 225
PRP$^+$	1 996	2 126	2 994	3 808	9 107	13 488	43 098	73 494	278 409

Common-base RME verification. We first show how V can use the RME test to speed up verification for an n-fold parallelized variant of Schnorr's protocol in which the common input is $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ and $(h_1, \ldots, h_n) \in (\mathbb{G})^n$ and in which P proves knowledge of $x_i = \log_g h_i$ for each $i = 1, \ldots, n$. The private input to P, therefore, is the n-tuple $(x_1, \ldots, x_n) \in (\mathbb{Z}_q)^n$. The resulting protocol is sometimes called the *common-base* parallelization of Schnorr's protocol, since the base g of the DL relation is common to all component instances. The parallelized protocol is still a sigma protocol, in which

1. P sends a randomized announcement $A = (A_1, \ldots, A_n)$ to V,
2. V issues a uniform random challenge c to P, and then
3. P computes a response $v = (v_1, \ldots, v_n)$ for V.

Here $A_i = g^{r_i}$ and $v_i = cx_i + r_i$, where $r_i \in_{\text{R}} \mathbb{Z}_q^*$ for each $i = 1, \ldots, n$. Observe that V issues a *single* challenge $c \in \mathbb{Z}_q$ to which P must respond for each $i = 1, \ldots, n$. For the verification equations, V checks that $g^{v_i} = h_i^c A_i$ for each $i = 1, \ldots, n$; if one or more of these expressions does not hold, then V rejects the proof. The following observation is immediate.

Observation 3.10. The naive verifier for the common base parallelization of Schnorr's protocol has expected cost $\mathrm{ExpCost}_{\mathbb{G}}^{(n)}(\tau) + n\,\mathrm{ExpCost}_{\mathbb{G}}(\tau)$, plus an additional n multiplications in $\mathbb{G}$. This cost approaches $(n+1)\tau + n\tau(1/\lg\tau + 1/\lg(n\tau))$ multiplications in $\mathbb{G}$ as τ and n tend to infinity, subject to $\lg n \in o(\tau)$.

Since the challenge c is common to all the verification equations, we can in fact treat the right-hand sides of the verification equations as constants $H_i = h_i^c A_i$; thus, V needs to check if $H_i = g^{v_i}$ for every $i = 1\ldots,n$, a task for which the RME test is well suited. Applying the RME test to the expressions $H_i = g^{v_i}$ for $i = 1,\ldots,n$ yields

$$\begin{aligned} g^{\sum_{i=1}^n t_i v_i} &\stackrel{?}{=} \prod_{i=1}^n H_i^{t_i} \\ &= \prod_{i=1}^n (h_i^c A_i)^{t_i}, \end{aligned}$$

and, after rearranging and regrouping, the right-hand side of this expression becomes

$$= \Big(\prod_{i=1}^n h_i^{t_i}\Big)^c \Big(\prod_{i=1}^n A_i^{t_i}\Big),$$

where $t_i \in_{\mathrm{R}} [0, 2^{\lambda_0} - 1]$ for each $i = 1,\ldots,n$. The expected computation cost to evaluate this expression is at most about $2\,\mathrm{ExpCost}_{\mathbb{G}}((n,\lambda_0)) + 2\,\mathrm{ExpCost}_{\mathbb{G}}(\tau)$, which is fewer than

$$3\tau + 4n\lambda_0/\lg n$$

multiplications in $\mathbb{G}$ when n is sufficiently large. This is a substantial reduction from the cost of the naive verifier, even when n is fairly small, provided $\tau > \lambda_0$. Moreover, it follows from Theorem 3.4 that the resulting batch-verified protocol has knowledge error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.

Common-exponents RME verification. Next, we show how V can use the RME test to speed up verification for a different n-fold parallelized variant of Schnorr's protocol in which the common input is $(\mathbb{G}, q, g_1, \ldots, g_n) \leftarrow \mathcal{G}(1^\tau; n)$ and $(h_1,\ldots,h_n) \in (\mathbb{G})^n$ and in which P proves knowledge of a single exponent $x \in \mathbb{Z}_q^*$ such that $x = \log_{g_i} h_i$ for every $i = 1,\ldots,n$. In other words, the protocol is a system for proofs of knowledge and *equality* among several DL pairs. The special $n = 2$ case of this protocol (without batch verification) was proposed by Chaum and Pedersen [CP92; §3.2]. The more general protocol for any $n > 1$ is illustrated in Figure 3.1.

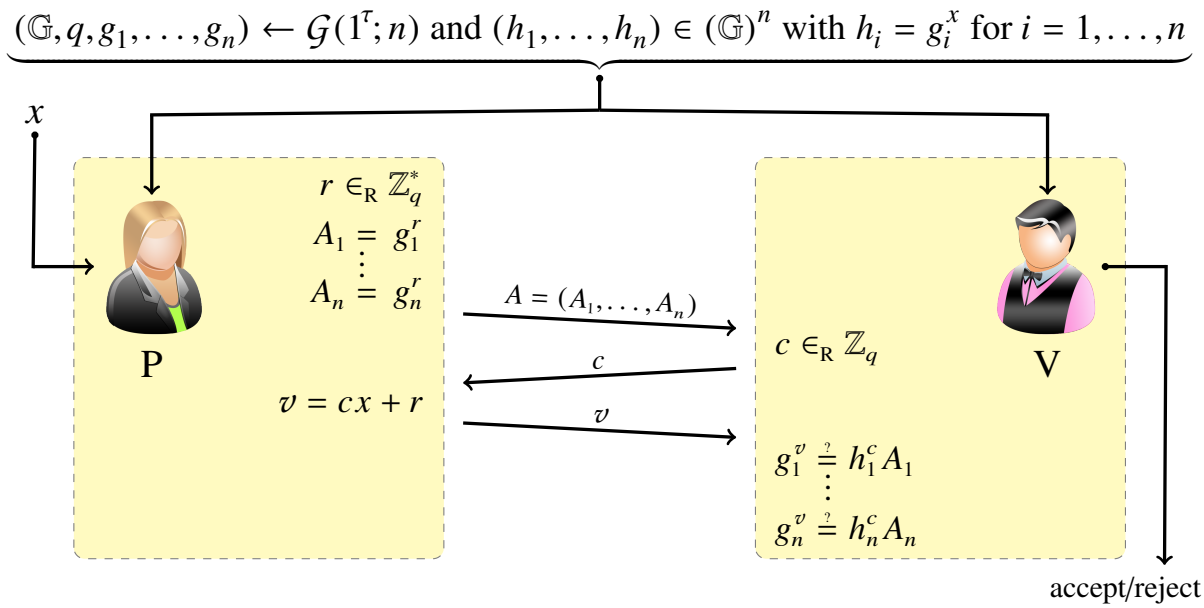

Figure 3.1: A system for honest-verifier zero-knowledge proofs of knowledge and equality among DLs due to Chaum and Pedersen [CP92]. The protocol is denoted by $\mathrm{PK}\{ x : \bigwedge_{i=1}^{n} (h_i = g_i^x) \}$.

Similar to the common-base parallelization, the common-exponent protocol is a sigma protocol in which V issues a single challenge $c \in \mathbb{Z}_q$; however, unlike in the common-base case, P only computes a *single* response $v \in \mathbb{Z}_q$. This is because P can — indeed, to prove that the given DLs are pairwise equal, P *must* — use the same random exponent $r \in_R \mathbb{Z}_q^*$ to produce each commitment $A_i = g_i^r$ in the announcement. This results in P computing an identical response $v = cx + r$ for each of the n component instances. Setting each $H_i = h_i^c A_i$ as before, the verification equations now ask whether $H_i = g_i^v$ for each $i = 1, \ldots, n$.

Observation 3.11. The naive verifier for the common-exponent parallelization of Schnorr's protocol has expected cost $n \, \mathrm{ExpCost}_{\mathbb{G}}((2, \tau))$. This is fewer than $2n\tau$ multiplications in $\mathbb{G}$ for any positive integers τ and n, using Straus' algorithm.

Given an arbitrary base $g \in \mathbb{G}^*$, we can define $u_i = \log_g g_i$ and then rewrite $g_i = g^{u_i}$ so that the verification equations ask whether $g^{u_i v} = h_i^c A_i$ or, equivalently, whether $(g^v)^{u_i} = H_i$, for each $i = 1, \ldots, n$. Applying the RME test to this latter expression yields

$$(g^v)^{\sum_{i=1}^n t_i u_i} \stackrel{?}{=} \prod_{i=1}^n H_i^{t_i}.$$

Of course, we cannot assume that V knows the exponents $u_i = \log_g g_i$. Fortunately, this does not present a problem in practice, as V can still evaluate the left-hand side of this expression *without knowing the* u_i via $\prod_{i=1}^n (g_i^v)^{t_i} = (\prod_{i=1}^n g_i^{t_i})^v$. We thus obtain the following verification equation:

$$\begin{aligned}(\prod_{i=1}^n g_i^{t_i})^v &\stackrel{?}{=} \prod_{i=1}^n H_i^{t_i} \\ &= \prod_{i=1}^n (h_i^c A_i)^{t_i} \\ &= (\prod_{i=1}^n h_i^{t_i})^c (\prod_{i=1}^n A_i^{t_i}),\end{aligned}$$

where $t_i \in_{\mathrm{R}} [0, 2^{\lambda_0} - 1]$ for each $i = 1, \ldots, n$. The expected computation cost to evaluate this expression is at most $2\,\mathrm{ExpCost}_{\mathbb{G}}(\tau) + 3\,\mathrm{ExpCost}_{\mathbb{G}}((n, \lambda_0))$, which is fewer than

$$3\tau + 6n\lambda_0/\lg n$$

multiplications in $\mathbb{G}$ when n is sufficiently large. Again, this is a substantial reduction from the naive verifier's expected cost. Moreover, by Theorem 3.4, the batch-verified protocol has knowledge error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.

3.1.6.1 Batch tests for DLREP relations

The RME test applies more generally: V can apply the RME test to batch verify any parallelized variant of Schnorr's protocol without any specific assumptions about uniqueness — or lack thereof — among the bases and exponents arising in the component verification equations. If all of the bases *and* all of the exponents are pairwise distinct, then the RME test saves only a modest

number of multiplications compared to the naive verifier; however, if some subset of verification equations share either a common base or a common exponent, then V can regroup terms in the resulting batch verification equation, as above, to make the savings more substantial.

In fact, the RME test naturally generalizes from the fixed-base DL relations in $\mathbb{G}$ to arbitrary k-DLREP relations in $\mathbb{G}$ for any $k \in \mathbb{N}^+$: given a sequence of verification equations involving k-DLREPs in $\mathbb{G}$, say

$$\begin{aligned} h_1 &\stackrel{?}{=} \prod_{j=1}^{k} g_{1j}^{x_{1j}} \\ &\vdots \\ h_n &\stackrel{?}{=} \prod_{j=1}^{k} g_{nj}^{x_{nj}}, \end{aligned}$$

we can fix an arbitrary generator $g \in \mathbb{G}^*$ and define $u_{ij} = \log_g g_{ij}$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, k$ to yield the equivalent sequence of verification equations,

$$\begin{aligned} h_1 &\stackrel{?}{=} g^{\sum_{j=1}^{k} u_{1j} x_{1j}} \\ &\vdots \\ h_n &\stackrel{?}{=} g^{\sum_{j=1}^{k} u_{nj} x_{nj}}. \end{aligned}$$

Applying the RME test to this latter sequence results in the expression

$$\prod_{i=1}^{n} h_i^{t_i} \stackrel{?}{=} g^{\sum_{i=1}^{n} t_i \left(\sum_{j=1}^{k} u_{ij} x_{ij}\right)}.$$

Theorem 3.4 establishes that this verification procedure has soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$. As before, V can evaluate the right-hand side without knowing the exponents u_{ij} via, for example,

$$\prod_{i=1}^{n} h_i^{t_i} \stackrel{?}{=} \prod_{i=1}^{n} \left(\prod_{j=1}^{k} g_{ij}^{x_{ij} t_i}\right).$$

In the worst case, evaluating this expression has expected cost about $\mathrm{ExpCost}_{\mathbb{G}}((n, \lambda_0), (nk, \tau))$, which is around

$$2n(\lambda_0 + k\tau)/\lg n$$

multiplications in $\mathbb{G}$ when n is sufficiently large.

The actual cost can be substantially lower if the same exponent x_{ij} or the same base g_{ij} appears several times. By comparison, the naive verifier has expected cost $n\,\mathrm{ExpCost}_{\mathbb{G}}((k,\tau))$ in the above general case, which is closer to $2nk\tau$ multiplications in $\mathbb{G}$ when k is constant.

The results just described for the RME test can be generalized to prove the following.

Observation 3.12. Fix a positive integer-valued function $k(\tau) \in \mathit{poly}(\tau)$ and let $\mathcal{R}$ denote an infinite family of generalized $k(\tau)$-DLREP relations induced by the group-generating algorithm $\mathcal{G}$; that is, let $\mathcal{R} = (R_\tau)_{\tau\in\mathbb{N}^+}$ such that, for each $\tau \in \mathbb{N}^+$, $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ and

$$R_\tau = \left\{ ((g_1,\ldots,g_{k(\tau)}), h, (x_1,\ldots,x_{k(\tau)})) \in (\mathbb{G}^*)^{k(\tau)} \times \mathbb{G} \times (\mathbb{Z}_q)^{k(\tau)} \;\middle|\; h = \prod_{i=1}^{k(\tau)} g_i^{x_i} \right\}.$$

If $\mathbf{R}$ is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$ having soundness error function $\lambda(\tau)$, then $\mathbf{R}$ is a batch verifier for $\mathcal{R}$ also having soundness error function $\lambda(\tau)$.

3.2 Batch proofs of knowledge

Fix an infinite family $\mathcal{R} = (R_\tau)_{\tau\in\mathbb{N}^+}$ of finite NP-relations $R_\tau \subseteq S_\tau \times W_\tau$. An *$(n,\tau)$-predicate* over $\mathcal{R}$ is a sequence of n strings $(s_1,\ldots,s_n)$ from S_τ, and a *batch predicate* over $\mathcal{R}$ is an (n,τ)-predicate over $\mathcal{R}$ for any $\tau \in \mathbb{N}^+$ and $n \geq 2$. The quantity n is called the *fan-in* of the (n,τ)-predicate. If $R_\tau(s_i, w_i) = 1$ for each $i = 1,\ldots,n$, then the length-n sequence of NP-witnesses $(w_1,\ldots,w_n)$ is called an *n-witness* for $(s_1,\ldots,s_n)$ with respect to $\mathcal{R}$. In other words, $(w_1,\ldots,w_n)$ is an n-witness for $(s_1,\ldots,s_n)$ with respect to $\mathcal{R}$ if $(s_i, w_i)_{i=1}^n$ is a correct (n,τ)-instance for $\mathcal{R}$. Likewise, we say that the (n,τ)-predicate $(s_1,\ldots,s_n) \in S_\tau^n$ is a *correct batch predicate* over $\mathcal{R}$ if there exists a sequence of witnesses $(w_1,\ldots,w_n) \in W_\tau^n$ for which the (n,τ)-instance $(s_i, w_i)_{i=1}^n$ is correct or, equivalently, if $(s_1,\ldots,s_n) \in (L_{R_\tau})^n$.

The remainder of this chapter considers zero-knowledge protocols that take (n,τ)-predicates as common input and scale efficiently with n. Several prior works [BDD07, GLSY04, Gro10, HOG11, PB10, SK95] have proposed such protocols, calling them systems for "batch zero-knowledge proofs of knowledge". However, with no generally agreed upon definition of what constitutes a system for batch zero-knowledge proofs of knowledge in the literature, each work assumes its own *ad hoc* definition. Peng, Boyd, and Dawson suggested one semi-formal definition [PBD07; Definition 1], which they presumably modeled after Bellare et al.'s definition for batch verifiers. Their

definition speaks to the correctness and soundness of a system for batch zero-knowledge proofs of knowledge; however, it is silent with regards to the *asymptotic efficiency* of such protocols and it only applies to systems for *conjunctive proofs of knowledge* in which no component instance is allowed to be bad. In any case, their definition was never embraced by the cryptographic community. In a recent collaboration with Ian Goldberg, the author of this dissertation proposed an asymptotic definition [HG13a; Definition 3] covering systems both for batch zero-knowledge proofs of knowledge and for batch zero-knowledge arguments of knowledge. This section presents a new formal definition, which is a significant refinement of that definition and a direct analog of Definition 12 for batch verifiers.

3.2.1 The naive conjunctive proof system

We first need to define a suitable notion of the *naive system for (conjunctive) zero-knowledge proofs of knowledge* for the language of correct batch predicates over $\mathcal{R}$, which will be our baseline against which to compare the cost of competing protocols. Just as Definition 12 requires that every batch verifier has cost asymptotically lower than that of its naive counterpart, Definition 17 below requires that every system for batch zero-knowledge proofs or arguments of knowledge has expected cost asymptotically lower than that of its naive (conjunctive) counterpart.

Recall that the naive verifier for $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ is the *lowest cost* algorithm that checks the correctness of a batch instance $(s_i, w_i)_{i=1}^n$ by evaluating the sequence $R_\tau(s_1, w_1), \ldots, R_\tau(s_n, w_n)$. We wish to translate this notion directly into the setting of zero-knowledge proofs of knowledge; hence, the naive system for (conjunctive) zero-knowledge proofs of knowledge for $\mathcal{R}$ should be the "lowest cost" protocol that "proves knowledge of each witness" in an n-witness (with respect to $\mathcal{R}$) for its common input. To be more precise than this, we require some additional definitions. In the following — indeed, whenever we speak about a naive system for conjunctive zero-knowledge proofs of knowledge — we use $\hat{\mathrm{P}}$ to refer to the (honest) naive prover and $\hat{\mathrm{V}}$ to refer to the (honest) naive verifier. This convention avoids confusion when we speak of a batch proof system (P, V) and its naive counterpart $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ in the same discussion.

Definition 14. Let $T = (t_1, \ldots, t_m)$ be a transcript from a two-party interactive protocol (P, V) in which each t_i is a distinct "message". Given any nonempty subset $U \subseteq_d [1, m]$, we let T_U denote the ordered subsequence $(t_{U_{(1)}}, \ldots, t_{U_{(d)}})$ of T. (Recall that $U \subseteq_d [1, n]$ denotes that U is a size-d subset of $[1, n]$ and that, for any set U of positive integers, $U_{(j)}$ denotes the j^{th} smallest element in U.) Such a subsequence T_U of T is called a *sub-transcript* of T.

Note that the sub-transcripts of T may comprise a *non-contiguous* subset of messages from T. The following *separability* criterion characterizes what it means for $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ to "prove knowledge of each witness" in an n-witness for the common input.

Definition 15. Let (P, V) be a system for honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over $\mathcal{R}$ and, for each $n \in \mathbb{N}^+$, let $m(n)$ denote the number of distinct messages in transcripts of (P, V) when the common input is an (n, τ)-predicate over $\mathcal{R}$. We say that (P, V) is *separable* if, for every $n \in \mathbb{N}^+$ and for each $j \in [1, n]$, there exists a fixed subset $U_j \subseteq [1, m(n)]$ such that the ensemble of random variables describing the sub-transcripts T_{U_j} arising from (P, V) when the common input is $(s_1, \ldots, s_n) \in S_\tau^n$ is statistically indistinguishable from the ensemble of random variables describing the transcripts T' arising in (P, V) when the common input is just s_j.

As a concrete example, if $T = (A_1, A_2, c, v_1, v_2) \in (\mathbb{G}^*)^2 \times (\mathbb{Z}_q)^3$ is an accepting transcript from a run of the common base parallelization of Schnorr's protocol in Section 3.1.6 when the common input is $(\mathbb{G}, q, g, h_1, h_2)$, then the sub-transcripts (A_1, c, v_1) and (A_2, c, v_2) correspond to accepting transcripts for the first component instance $(\mathbb{G}, q, g, h_1)$ and second component instance $(\mathbb{G}, q, g, h_2)$, respectively. In fact, it is easy to verify that both of the parallel variants of Schnorr's protocol discussed in the last section are separable.

The notion of the "lowest cost" protocol is more difficult to pin down because the cost of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ can be measured in several ways, including (i) the expected computation cost for $\hat{\mathrm{P}}$, (ii) the expected computation cost for $\hat{\mathrm{V}}$, (iii) the communication cost from $\hat{\mathrm{P}}$ to $\hat{\mathrm{V}}$, (iv) the communication cost from $\hat{\mathrm{V}}$ to $\hat{\mathrm{P}}$, (v) the number of rounds, or (vi) any function of these, and possibly other, metrics. All of the constructions that we deal with are proofs of knowledge for DL or DLREP relations in prime-order multiplicative groups. To simplify our analysis, we assume that the appropriate generalization of Schnorr's protocol always has the "lowest cost" among the separable protocols for such statements.

Definition 16. Let $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ be an infinite family of finite NP-relations. A protocol $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ is the *naive system for conjunctive honest-verifier zero-knowledge proofs of knowledge* for $\mathcal{R}$ if (i) it is a system for honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over $\mathcal{R}$, (ii) it is separable, and (iii) it has the lowest cost among the protocols satisfying both (i) and (ii).

It is understood that $\hat{\mathrm{V}}$ always uses the appropriate naive verifier (and never a batch verifier) to check the verification equations in $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$.

3.2.2 Defining batch proofs of knowledge

For each $\tau \in \mathbb{N}^+$, let $\varphi_\tau \colon \mathcal{B}_\tau \to \{0,1\}^*$ denote the function that maps each (n,τ)-instance $\mathcal{I} = (s_i, w_i)_{i=1}^n$ for $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ to the n-bit string

$$\varphi_{R_\tau}(\mathcal{I}) = R_\tau(s_1, w_1) \| R_\tau(s_2, w_2) \| \cdots \| R_\tau(s_n, w_n).$$

For a given binary NP-language $\Gamma \subseteq \{0,1\}^*$, we say that $\mathcal{I}$ is *Γ-correct* with respect to $\mathcal{R}$ if $\varphi_\tau(\mathcal{I}) \in \Gamma$. Likewise, we say that an (n,τ)-predicate $(s_1, \ldots, s_n) \in S_\tau^n$ over $\mathcal{R}$ is Γ-correct if there exists a sequence of witnesses $(w_1, \ldots, w_n) \in W_\tau^n$ for which $(s_i, w_i)_{i=1}^n$ is Γ-correct. In this case, $(w_1, \ldots, w_n)$ is called a *(Γ, n)-witness* for $(s_1, \ldots, s_n)$ with respect to $\mathcal{R}$.

Below, Definition 17 defines systems for batch zero-knowledge proofs of knowledge with respect to pairs $(\mathcal{R}, \Gamma)$ in which $\mathcal{R}$ is an infinite family of NP-relations and Γ is an infinite binary NP-language. The protocol associated with the pair $(\mathcal{R}, \Gamma)$ is a system for (honest-verifier) zero-knowledge proofs of knowledge for the language of Γ-correct batch predicates over $\mathcal{R}$; in other words, it is a system for proofs of knowledge of a (Γ, n)-witness for the common input. Thus, in general, P is proving *partial knowledge* of an n-witness for the common input, with the bit strings in Γ reflecting which subsets of component witnesses P might actually be proving to know. For conjunctive ("AND") proofs, Γ is the language of finite bit strings comprised entirely of 1s; for disjunctive ("OR") proofs, Γ is the language of finite bit strings that are not comprised entirely of 0s; for k-out-of-n threshold proofs, Γ is the language of finite bit strings with Hamming weight k or greater; and so on.

In the following definitions, let $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ denote the naive system for conjunctive (honest-verifier) zero-knowledge proofs of knowledge for $\mathcal{R}$. Furthermore, let

(a) $\mathcal{V}_\tau(n)$ denote the expected computation cost for $\hat{\mathrm{V}}$ in $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$,

(b) $\mathcal{P}_\tau(n)$ denote the expected computation cost for $\hat{\mathrm{P}}$ in $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$, and

(c) $\mathcal{C}_\tau(n)$ denote the length of the transcripts of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$,

where, in each case, the common input is a random, correct (n, τ)-predicate over $\mathcal{R}$.

Definition 17. Let $\mathcal{R}$ be an infinite family of finite NP-relations and let $\Gamma \subseteq \{0,1\}^*$ be an infinite NP-language. An interactive protocol (P, V) is a system for *batch (honest-verifier) zero-knowledge proofs of knowledge* for the language of Γ-correct batch predicates over $\mathcal{R}$ if there exists a constant $\delta \in [0, 1/2]$ and a function $\lambda : \mathbb{N}^+ \to \mathbb{R}^+$, with $|\lambda(\tau) - \delta|$ negligible in τ, such that, for every positive integer-valued function $n(\tau) \in \mathit{poly}(\tau)$ and for every (possibly dishonest) prover P^*, (P, V) provides the following four guarantees.

1. **Proof of knowledge:** (P, V) is a system for proofs of knowledge for the language of Γ-correct $(n(\tau), \tau)$-predicates over $\mathcal{R}$.
2. **Zero-knowledge:** (P, V) is (honest-verifier) zero-knowledge on the language of Γ-correct $(n(\tau), \tau)$-predicates over $\mathcal{R}$.
3. **Sound:** If $(s_1, \ldots, s_n) \in S_\tau^n$ is not Γ-correct, then $\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(s_1, \ldots, s_n)] \leq \lambda(\tau)$.
4. **Asymptotically efficient:** On input an (n, τ)-predicate $(s_1, \ldots, s_n) \in S_\tau^n$ over $\mathcal{R}$,
   (a) V has expected computation cost in $\mathrm{o}(\mathcal{V}_\tau(n(\tau)))$,
   (b) if $(w_1, \ldots, w_n) \in W_\tau^n$ is a (Γ, n)-witness for $(s_1, \ldots, s_n)$ with respect to $\mathcal{R}$, then $\mathrm{P}(w_1, \ldots, w_n)$ has expected computation cost in $\mathrm{o}(\mathcal{P}_\tau(n(\tau)))$, and
   (c) the transcript of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ has length in $\mathrm{O}(\mathcal{C}_\tau(n(\tau)))$.

If $\Gamma = 1^*$, then (P, V) is a system for *conjunctive* batch proofs of knowledge. Systems for conjunctive proofs of knowledge are sometimes called systems for proofs of *complete* knowledge so as to distinguish them from systems for proofs of *partial* knowledge, which are the focus of Chapter 4. When $\Gamma \neq 1^*$, we call a system for (batch) proofs of knowledge for the language of Γ-correct batch predicates over $\mathcal{R}$ as a system for *(batch) proofs of Γ-partial knowledge* over $\mathcal{R}$.

The smallest function $\lambda : \mathbb{N}^+ \to \mathbb{R}^+$ satisfying the soundness bound in Definition 17 is called the *soundness error function* for (P, V) and the constant δ is its *absolute soundness error*. The soundness error function measures the probability that a dishonest P^* can make V accept when the common input is not Γ-correct. As before, lower values of δ usually necessitate a higher cost to run (P, V).

The first two asymptotic efficiency criteria require no explanation. The third criterion uses the length of an interaction transcript as a proxy for the *bidirectional communication cost* of the interaction that produced that transcript. It would also make sense to consider the unidirectional communication cost in each direction, though no batch protocol in the literature would satisfy this stronger definition. (In particular, no protocol in the literature reduces the outgoing communication cost from V to P relative to that from $\hat{\mathrm{V}}$ to $\hat{\mathrm{P}}$, if doing so is even possible; in fact, all known batch protocols actually *increase* this cost, and sometimes substantially so.)

Definition 17 is inherently asymptotic. The following definition is analogous to Definition 13. It requires, for a particular choice of (n, τ, λ_0), that a system (P, V) for batch zero-knowledge proofs of knowledge have cost strictly lower than that of its naive (conjunctive) counterpart $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$.

Definition 18. Let $\mathcal{R}$ be an infinite family of finite NP-relations, let $\Gamma \subseteq \{0,1\}^*$ be an infinite NP-language, and let (P, V) be a system for (honest-verifier) batch zero-knowledge proofs of Γ-partial knowledge over $\mathcal{R}$. We call (P, V) a system for (n, τ, λ_0)-*batch zero-knowledge proofs of Γ-partial knowledge* over $\mathcal{R}$ if the soundness error function for (P, V) satisfies $\lambda(\tau) \leq 2^{-\lambda_0}$ and if, in any accepting interaction in which the common input is an (n, τ)-predicate over $\mathcal{R}$, each of the following is true.

(a) **Prover efficient:** The expected computation cost for V is strictly less than $\mathcal{V}_\tau(n)$.

(b) **Verifier efficient:** The expected computation cost for P is strictly less than $\mathcal{P}_\tau(n)$.

(c) **Communication efficient:** The transcript of (P, V) has length strictly less than $\mathcal{C}_\tau(n)$.

The next section introduces several systems for conjunctive batch honest-verifier zero-knowledge proofs of knowledge for linear relations. First, we introduce the following notion of *component-wise k-extractability*. As its name suggests, component-wise k-extractability is a

component-wise generalization of k-extractability (see Definition 10 on Page 31), which stipulates that the universal knowledge extractor should be able to extract *any single component witness of its choosing* by rewinding honest P at most k times. Extracting additional witnesses from P may require rewinding P more times.

Definition 19. A protocol (P,V) is *component-wise k-extractable* for the language of correct batch predicates over $\mathcal{R}$ if there exists a knowledge extractor E_{P}, whose running time is PPT in τ, such that, for every (n,τ)-predicate $(s_1,\ldots,s_n)$ over $\mathcal{R}$, for any accepting transcript (A,c_1,v_1) of (P,V) on common input $(s_1,\ldots,s_n)$, and for any index $j\in[1,n]$, there exist super-polynomially (in λ_0) many $(k-1)$-tuples of challenges $(c_2,\ldots,c_k)$, such that, given $(v_1,\ldots,v_k)$ with (A,c_i,v_i) accepting for each $i=1,\ldots,k$, there is an efficient algorithm to compute a witness $w_j\in\mathcal{W}_{R_\tau}(s_j)$.

Note that the first challenge c_1 in Definition 19 is arbitrary, whereas the subsequent choices for $c_2,\ldots,c_k$ may depend both on the first challenge c_1 and on the index j for which a witness is sought.

Observation 3.13. If (P,V) is component-wise k-extractable for the language of correct batch predicates over $\mathcal{R}$, then (P,V) is a system for proofs of knowledge for the language of correct batch predicates over $\mathcal{R}$.

3.2.3 Conjunctive batch proofs for families of linear relations

This section presents systems for batch zero-knowledge proofs of *complete* knowledge for families of linear relations $\mathcal{R}=(R_\tau)_{\tau\in\mathbb{N}^+}$. The most obvious construction for such systems involves replacing the naive protocol $(\hat{\mathrm{P}},\hat{\mathrm{V}})$, denoted in Camenisch-Stadler notation by

$$\mathrm{PK}\Big\{(w_1,\ldots,w_n):\bigwedge_{i=1}^{n}R_\tau(s_i,w_i)=1\Big\},$$

with the protocol (P,V) denoted by

$$\mathrm{PK}\Big\{(w_1,\ldots,w_n):\mathbf{R}((s_1,w_1),\ldots,(s_n,w_n))=1\Big\},$$

where $\mathbf{R}\colon \mathcal{B} \to \{0,1\}$ is a suitable *batch verifier* for $\mathcal{R}$. In other words, instead of directly proving knowledge of $(w_1, \ldots, w_n)$ such that $R_\tau(s_i, w_i) = 1$ for each $i = 1, \ldots, n$, P proves knowledge of $(w_1, \ldots, w_n)$ such that the (n,τ)-instance $(s_i, w_i)_{i=1}^n$ passes a random trial of a suitable batch test for $\mathcal{R}$.

Of course, for the batch test $\mathbf{R}$ to be convincing, it is crucial that all the random coin tosses it requires are indeed uniform random; thus, we must have V (or, perhaps, a random oracle) provide these random values to P^* at some point *after both parties receive the common input predicate*. Depending on the underlying family of relations $\mathcal{R}$, this might necessitate an additional opening move in which V sends one or more uniform random strings to P; for other families, P does not require (or, in some instances, is not allowed to see) the random strings immediately, and V therefore includes them as part of some later move.

Peng et al. [PBD07] instantiated the above idea with the RME test to obtain systems for conjunctive batch honest-verifier zero-knowledge proofs of knowledge and equality among several DLs in a prime-order group. We discuss their protocols below, along with several new systems for conjunctive batch honest-verifier zero-knowledge proofs of knowledge that we constructed in a similar fashion using other linear batch verifiers. We also discuss a batch protocol due to Gennaro, Leigh, Sundaram, and Yerazunis [GLSY04], which is not based on a linear batch verifier.

As before, we present and analyze each system for batch zero-knowledge proofs of knowledge with respect to an infinite family of DL relations $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$ induced by a fixed group-generating algorithm $\mathcal{G}$. The generalizations to families of k-DLREP relations $\mathcal{R} \leftarrow \mathcal{G}(1^*; k)$ are all straightforward.

Observation 3.14. Let $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ be the naive system for conjunctive honest-verifier zero-knowledge proofs of knowledge for $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$. Given an (n,τ)-predicate over $\mathcal{R}$ as input,

(a) $\hat{\mathrm{V}}$ has expected computation cost $\mathcal{V}_\tau(n) = \mathrm{ExpCost}_{\mathbb{G}}^{(n)}(\tau) + n\,\mathrm{ExpCost}_{\mathbb{G}}(\tau)$, plus an additional n multiplications in $\mathbb{G}$,

(b) $\hat{\mathrm{P}}$ has expected computation cost $\mathcal{P}_\tau(n) = \mathrm{ExpCost}_{\mathbb{G}}^{(n)}(\tau)$, and

(c) the transcript of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ is an element of $(\mathbb{G})^n \times (\mathbb{Z}_q)^{n+1}$.

Furthermore, as τ and n grow large subject to $\lg n \in \mathrm{o}(\tau)$,

(a) $\mathcal{V}_\tau(n)$ approaches $(n+1)\tau + n\tau/\lg(n\tau) + n\tau/\lg\tau$ multiplications in $\mathbb{G}$, and

(b) $\mathcal{P}_\tau(n)$ approaches $\tau + n\tau/\lg(n\tau)$ multiplications in $\mathbb{G}$.

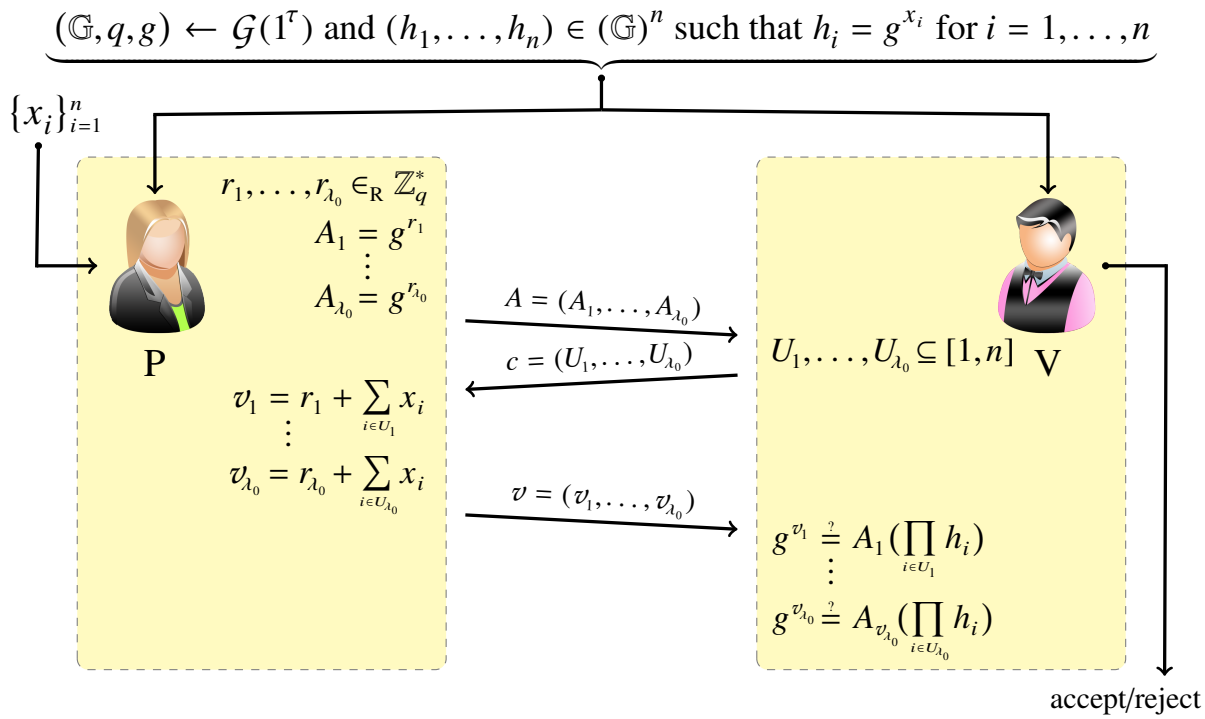

Figure 3.2: A common-base batch variant of Schnorr's protocol based on the RMP test. The protocol is c-simulatable and component-wise 2-extractable and is denoted by $\mathrm{PK}\{(x_1, \ldots, x_n) : \mathbf{R}_{\mathrm{RMP}}^{(\lambda_0)}((h_1, x_1), \ldots, (h_n, x_n))\}$.

3.2.3.1 The RMP-based common-base Schnorr protocol

The first protocol we discuss is a new batch variant of Schnorr's protocol based on the RMP test. In particular, for a given a soundness parameter $\lambda_0 \in \mathbb{N}^+$, we construct a system for conjunctive batch honest-verifier zero-knowledge proofs of knowledge with a constant soundness error function $\lambda(\tau) = 2^{-\lambda_0}$ by executing the protocol denoted in Camenisch-Stadler notation by

$$\mathrm{PK}\Big\{(x_1, \ldots, x_n) : \mathbf{R}_{\mathrm{RMP}}^{(\lambda_0)}((h_1, x_1), \ldots, (h_n, x_n)) = 1\Big\}.$$

Given $(x_1, \ldots, x_n) \in (\mathbb{Z}_q)^n$ and tasked with verifying that $h_i = g^{x_i}$ for each $i = 1, \ldots, n$ using the RMP test, V would choose λ_0 subsets $U_1, \ldots, U_{\lambda_0} \subseteq [1, n]$, each uniformly at random, and then output 1 only if $\prod_{i \in U_k} h_i = g^{\sum_{i \in U_k} x_i}$ for each $k = 1, \ldots, \lambda_0$. The batch proof of knowledge is similar, except V chooses the random subsets $U_1, \ldots, U_{\lambda_0} \subseteq [1, n]$ and P merely proves knowledge of $(\varpi_1, \ldots, \varpi_{\lambda_0})$ such that $\prod_{i \in U_k} h_i = g^{\varpi_k}$ for each $k = 1, \ldots, \lambda_0$. In the latter proofs of knowledge, V uses $c = (U_1, \ldots, U_{\lambda_0})$ as its challenge, so that P responds with $v_k = r_k + \varpi_k$ for $k = 1, \ldots, \lambda_0$, where each $r_k \in_{\mathrm{R}} \mathbb{Z}_q$. A simple calculation confirms that, for each $k = 1, \ldots, \lambda_0$, the verification equation $g^{\varpi_k} = \prod_{i \in U_k} h_i$ holds if and only if $\varpi_k = \sum_{i \in U_k} x_i$. Figure 3.2 illustrates the interaction.

Theorem 3.15. *The RMP-based variant of Schnorr's protocol depicted in Figure 3.2 is a system for conjunctive batch honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over* $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$. *It is c-simulatable and component-wise* 2*-extractable and, for a fixed soundness parameter* $\lambda_0 \in \mathbb{N}^+$*, it has absolute soundness error* $\delta = 2^{-\lambda_0}$ *and soundness error function* $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.

Proof. **Complete:** For each $k = 1, \ldots, \lambda_0$, we have that

$$g^{v_k} = g^{r_k + \sum_{i \in U_k} x_i} = g^{r_k} \left(\prod_{i \in U_k} g^{x_i}\right) = g^{r_k} \left(\prod_{i \in U_k} h_i\right) = A_k \left(\prod_{i \in U_k} h_i\right).$$

Component-wise 2-extractable: To extract $x_j = \log_g h_j$, the universal knowledge extractor $\mathrm{E}_{\mathrm{P}^*}$ first challenges P^* on a uniform random choice of $c = (U_1, \ldots, U_{\lambda_0})$ to get an announcement $A = (A_1, \ldots, A_{\lambda_0})$ and a response $v = (v_1, \ldots, v_{\lambda_0})$ satisfying

$$g^{v_k} = A_k \left(\prod_{i \in U_k} h_i\right)$$

for each $k = 1, \ldots, \lambda_0$. Suppose that some subset U_θ in c does *not* contain the index j. In this case, $\mathrm{E}_{\mathrm{P}^*}$ rewinds P^* and issues a new challenge $c' = (U'_1, \ldots, U'_{\lambda_0})$ in which $U'_\theta = U'_\theta \cup \{j\}$. If P^* responds with an accepting $v' = (v'_1, \ldots, v'_{\lambda_0})$ for c', then

$$\begin{aligned} g^{v'_\theta} &= A_\theta \left(\prod_{i \in U'_\theta} h_i\right) \\ &= A_\theta \, h_j \left(\prod_{i \in U_\theta} h_i\right), \end{aligned}$$

and E_{P^*} can divide out the corresponding expression from P^*'s first response to get $g^{v'_\theta - v_\theta} = h_j$; the exponent sought is then $x_j = v'_\theta - v_\theta$. In the unlikely event that $j \in U_k$ for every $k = 1, \ldots, \lambda_0$ after the first step, E_{P^*} instead issues the second challenge having $U'_\theta = U_\theta \setminus \{j\}$, and then it computes $x_j = v_\theta - v'_\theta$. Note that for any initial challenge $c = (U_1, \ldots, U_{\lambda_0})$, there are $(2^n)^{\lambda_0 - 1} \notin poly(\lambda_0)$ additional challenges for which a valid response enables E_{P^*} to extract the desired exponent $x_j = \log_g h_j$.

Sound: Consider a protocol run using soundness parameter $\lambda_0 = 1$. We will show that the soundness error function is $\lambda(\tau) = 1/2$; our desired result for $\lambda_0 > 1$ then follows by a standard union-bound argument [Dam11; Lemma 1]. Fix $j \in [1, n]$ and let M_j denote the binary matrix with a row for each possible announcement $A \in \mathbb{G}^*$ from P^* and a column for each possible challenge $U_1 \subseteq [1, n]$ from V, subject to $j \notin U_1$. (Hence, there are $q - 1$ rows and 2^{n-1} columns.) The entry (A, U_1) of M_j is 1 if P^* makes V accept for *both* of the challenges U_1 and $U'_1 = U_1 \cup \{j\}$ when the announcement is A. Using P^* as an oracle, E_{P^*} can probe P^* for its response to a random challenge U'_1 and, through rewinding, E_{P^*} can also probe P^* for its response to the challenge $U_1 \cup \{j\}$. The extractor's goal is to locate a challenge U_1 for which the entry (A, U_1) of M_j is 1, in which case E_{P^*} can compute $x_j = \log_g h_j$ using the component-wise 2-extractability of (P, V).

Note that $\Pr[1 \leftarrow \langle P^*, V \rangle(h_1, \ldots, h_n)]$ is related to the fraction of 1-entries in M_j as follows: P^* can make V accept with probability at most $\frac{1}{2}(1 + \mu(\tau))$, where $\mu(\tau) \geq 0$ is the fraction of 1-entries in M_j. We have no information about the distribution of 1-entries in M_j; in particular, some rows of M_j might contain fewer than a fraction $\mu(\tau)$ of 1-entries. Nonetheless, if $\mu(\tau) > 0$, then E_{P^*} can locate a 1-entry in M_j by rewinding P^* about $1/\mu(\tau)$ times on average; hence, if $\mu(\tau)$ is non-negligible in λ_0, then E_{P^*} outputs x_j in expected polynomial time. This proves that $\lambda(\tau) \leq 1/2$ when $\lambda_0 = 1$. This bound is clearly tight; for example, if P^* makes V accept on challenge $U_1 \subseteq [1, n]$ if and only if $j \notin U_1$, then V accepts with probability $1/2$, but E_{P^*} will always fail to extract $x_j = \log_g h_j$ from P^*.

***c*-Simulatable:** Given $c = (U_1, \ldots, U_{\lambda_0})$ as input, $S_V(c)$ chooses $v_1, \ldots, v_{\lambda_0} \in_R \mathbb{Z}_q$, and then it computes $A_k = g^{v_k} / (\prod_{i \in U_k} h_i)$ for each $k = 1, \ldots, \lambda_0$. (In the unlikely event that $A_k = 1$ for some $k = 1, \ldots, \lambda_0$, S_V chooses a different $v_k \in_R \mathbb{Z}_q$ and tries again.) The simulated transcript is $(A_1, \cdots, A_{\lambda_0}, U_1, \cdots, U_{\lambda_0}, v_1, \cdots, v_{\lambda_0})$. The proof that S_V's output follows the correct distribution is a direct adaptation of the corresponding proof for Schnorr's protocol: for each $k = 1, \ldots, \lambda_0$, the sub-transcript $(\mathbb{G}, q, g, h_1, \ldots, h_n, U_k, v_k)$ uniquely determines the corresponding

announcement $A_k = g^{v_k}/(\prod_{i \in U_k} h_i)$, and each of these values but v_k is fixed before the simulation begins; thus, our desired result follows from the fact that v_k is distributed uniformly at random in both the real and simulated transcripts.

Asymptotically efficient: By inspection, P's expected computation cost is $\mathrm{ExpCost}_{\mathbb{G}}^{(\lambda_0)}(\tau) \in o(\mathcal{P}_\tau(n))$. This cost is *independent of the fan-in* n.[15] The expected computation cost for V on common input $(\mathbb{G}, q, g, h_1, \ldots, h_n)$ is about $\mathrm{ExpCost}_{\mathbb{G}}^{(\lambda_0)}(\tau) + \mathrm{ExpCost}_{\mathbb{G}}^{(\lambda_0)}((n,1)) \in o(\mathcal{V}_\tau(n))$, which is almost identical to the expected cost for evaluating $\mathbf{R}_{\mathrm{RMP}}^{(\lambda_0)}((h_1, x_1), \ldots, (h_n, x_n))$ given access to the exponents $(x_1, \ldots, x_n)$. Finally, the transcripts are elements of $(\mathbb{G}^*)^{\lambda_0} \times (\{0,1\}^n)^{\lambda_0} \times (\mathbb{Z}_q)^{\lambda_0}$. (Recall that the transcripts of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ are elements of $(\mathbb{G}^*)^n \times (\mathbb{Z}_q)^{n+1}$.) If we let $\ell_{\mathbb{G}}(\tau)$ denote the bit-length of $\mathbb{G}^*$ elements, then the transcripts of (P, V) are shorter than those of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$ whenever the fan-in n exceeds $(\ell_{\mathbb{G}}(\tau)\lambda_0 + \tau(\lambda_0 - 1))/(\ell_{\mathbb{G}}(\tau) + \tau + \lambda_0)$.

We could also implement the above system for batch zero-knowledge proofs using the RMP$^+$ test. Doing so almost always reduces both the transcript length and the expected computation cost for P relative to the RMP-based protocol. Depending on the fan-in n, using RMP$^+$ sometimes increases the computation cost for V and it sometimes decreases the computation cost for V; in either case, the magnitude of the change for V is usually small relative to the overall cost of the protocol.

3.2.3.2 The RME-based common-base Schnorr protocol

Peng et al. [PBD07] considered RME-based batch protocols for the common-base parallelization of Schnorr's protocol and for the common exponent parallelization of Chaum and Pedersen's protocol. In Figure 3.3, we apply the same construction to Schnorr's protocol, thus instantiating the protocol denoted in Camenisch-Stadler notation by

$$\mathrm{PK}\{(x_1, \ldots, x_n) : \mathbf{R}_{\mathrm{RME}}^{(\lambda_0)}((h_1, x_1), \ldots, (h_n, x_n))\}.$$

[15] Of course, P must still compute the responses v_k, each of which involves a summation of about $n/2$ addends; however, summation in $\mathbb{Z}_q$ is typically *much* faster than multiplication in $\mathbb{G}$ and is, in any case, not counted under our cost model.

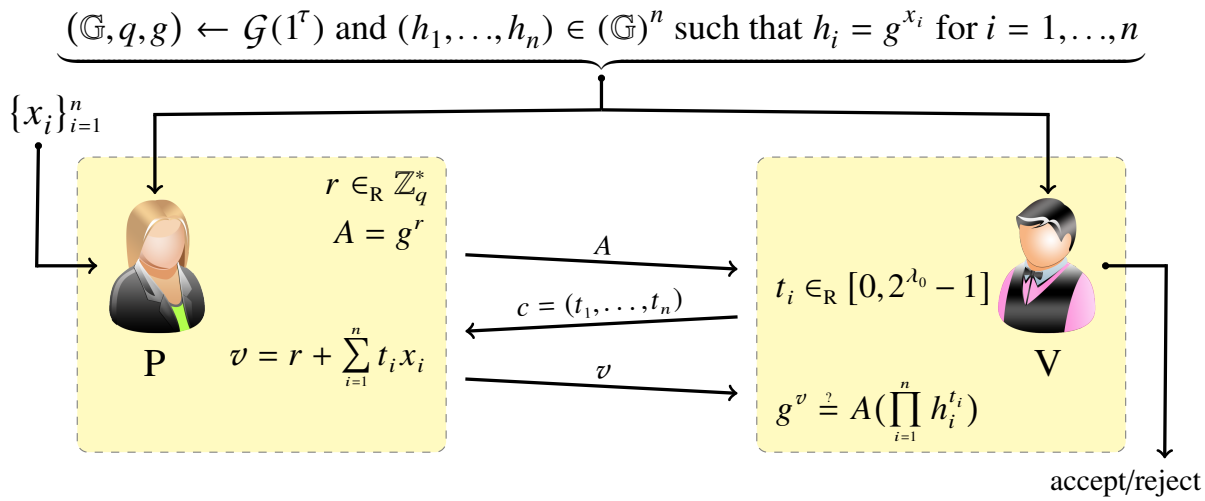

Figure 3.3: A common-base batch variant of Schnorr's protocol based on the RME test. The protocol is c-simulatable and component-wise 2-extractable and is denoted by $\mathrm{PK}\{(x_1, \ldots, x_n) : \mathbf{R}_{\mathrm{RME}}^{(\lambda_0)}((h_1, x_1), \ldots, (h_n, x_n))\}$.

Theorem 3.16. *The RME-based common-base Schnorr protocol depicted in Figure 3.3 is a system for conjunctive batch honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$. It is c-simulatable and component-wise 2-extractable and, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.*

Proof. **Complete:** $g^v = g^{r + \sum_{i=1}^n t_i x_i} = g^r (\prod_{i=1}^n g^{t_i x_i}) = g^r (\prod_{i=1}^n g^{x_i t_i}) = g^r (\prod_{i=1}^n h_i^{t_i}) = A(\prod_{i=1}^n h_i^{t_i})$.

Component-wise 2-extractable: Suppose that $(h_i, x_i)_{i=1}^n$ is a correct (n, τ)-instance for $\mathcal{R}$. To extract $x_j = \log_g h_j$, the universal knowledge extractor $\mathrm{E}_{\mathrm{P}^*}$ first challenges P^* on a uniform random choice of $c = (t_1, \ldots, t_n)$ to get an announcement A and response v such that

$$g^v = A(\prod_{i=1}^n h_i^{t_i}).$$

$\mathrm{E}_{\mathrm{P}^*}$ then rewinds P^* and issues the challenge $c' = (t'_1, \ldots, t'_n)$ such that $t'_j \in_{\mathrm{R}} [0, 2^{\lambda_0} - 1]$ and $t'_i = t_i$ for each $i \in [1,n] \setminus \{j\}$, thereby obtaining a second response v' such that

$$\begin{aligned} g^{v'} &= A(\prod_{i=1}^{n} h_i^{t'_i}) \\ &= A\, h_j^{t'_j - t_j} (\prod_{i=1}^{n} h_i^{t_i}). \end{aligned}$$

Dividing out the corresponding expression from P^*'s first response, we get $g^{v'-v} = h_j^{t'_j - t_j}$; the exponent sought is then $x_j = (v' - v)/(t'_j - t_j)$. Note that for any initial challenge $c = (t_1, \ldots, t_n)$, there are $2^{\lambda_0} - 1 \notin \mathit{poly}(\lambda_0)$ distinct second challenges, any one of which is sufficient to extract $x_j = \log_g h_j$.

Sound: Suppose P^* outputs announcement $A \in \mathbb{G}^*$. Define the binary matrix M_A with a column for each possible assignment of $t_j \in [0, 2^{\lambda_0} - 1]$ and a row for each possible assignment of the $t_i \in [0, 2^{\lambda_0} - 1]$ with $i \in [1,n] \setminus \{j\}$. (Hence, there are $(2^{\lambda_0})^{n-1}$ rows and 2^{λ_0} columns.) The entry $(t_1, \ldots, t_n)$ of M_A is 1 if P^* makes V accept when the challenge is $c = (t_1, \ldots, t_n)$, and it is 0 otherwise. Using P^* as an oracle, $\mathrm{E}_{\mathrm{P}^*}$ can probe P^* with a random challenge in hopes of finding a 1-entry in M_A and, through rewinding, $\mathrm{E}_{\mathrm{P}^*}$ can probe P^* with a different challenge in hopes of finding a second 1-entry on the same row of M_A. (In particular, $\mathrm{E}_{\mathrm{P}^*}$ probes P^* with any challenge $c' = (t'_1, \ldots, t'_n)$ in which $t_i = t'_i$ for each $i \in [1,n] \setminus \{j\}$.) $\mathrm{E}_{\mathrm{P}^*}$'s goal is to locate a pair of challenges (c, c') each corresponding to 1-entries on the same row of M_A and such that $t_j \neq t'_j$, in which case $\mathrm{E}_{\mathrm{P}^*}$ can compute $x_j = \log_g h_j$ using the component-wise 2-extractability of (P, V).

Suppose $\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(h_1, \ldots, h_n)] = \epsilon(\tau)$ such that $\epsilon(\tau) = 2^{-\lambda_0} + \mu(\tau)$ for some $\mu(\tau) \geq 0$ and note that $\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle(h_1, \ldots, h_n) \mid \mathrm{P}^*\text{'s announcement is } A]$ is equal to the fraction of 1-entries in the matrix M_A. By Lemma 2.7, with a probability exceeding $\epsilon(\tau)/2$, M_A has a fraction at least $\epsilon(\tau)/2$ of 1-entries, in which case a second application of Lemma 2.7 implies that a fraction exceeding $\epsilon(\tau)/4$ of the rows of M_A have a fraction at least $\epsilon(\tau)/4$ of 1-entries. In particular, $\mathrm{E}_{\mathrm{P}^*}$ can locate such a row after rewinding P^* an expected $4/\epsilon(\tau)$ times. Such a row has no fewer than $2^{\lambda_0}(2^{-\lambda_0 - 2} + \mu(\tau)/4) - 1 > 1/4 + 2^{\lambda_0 - 2}\mu(\tau) - 1$ additional 1-entries. It therefore follows that, if $\mu(\tau)$ is non-negligible in λ_0, then $\mathrm{E}_{\mathrm{P}^*}$ can find a pair of 1-entries in such a row using an expected number of probes of P^* polynomial in λ_0.

***c*-Simulatable:** Given $c = (t_1, \ldots, t_n)$ as input, $S_V(c)$ chooses $v \in_R \mathbb{Z}_q$ and computes $A = g^v / (\prod_{i=1}^n h_i^{t_i})$. (In the unlikely event that $A = 1$, S_V chooses a different $v \in_R \mathbb{Z}_q$ and tries again.) The simulated transcript is (A, c, v). We must show that S_V's output follows the same probability distribution as that of the transcripts in V's aggregate view of a real interaction with the same common input $(\mathbb{G}, q, g, h_1, \ldots, h_n)$ and challenge $c = (t_1, \ldots, t_n)$. As we have seen before, the values in $(\mathbb{G}, q, g, h_1, \ldots, h_n, c, v)$ uniquely determine A and each of these values but v is fixed prior to the simulation; thus, because v is distributed uniformly at random in both the real and simulated transcripts, it follows that the distributions for real and simulated transcripts are described by identical random variables.

Asymptotically efficient: P's expected computation cost is $\mathrm{ExpCost}_{\mathbb{G}}(\tau) \in o(\mathcal{P}_\tau(n))$, which is independent of n and, in fact, is identical to its expected computation cost in Schnorr's protocol when the common input has fan-in equal to 1. The expected computation cost for V on input $(h_1, \ldots, h_n)$ is about $\mathrm{ExpCost}_{\mathbb{G}}((1, \tau), (n, \lambda_0)) \in o(\mathcal{V}_\tau(n))$, which is similar to the expected cost to evaluate $\mathbf{R}_{\mathrm{RME}}^{(\lambda_0)}((h_1, x_1), \ldots, (h_n, x_n))$ given access to the exponents $(x_1, \ldots, x_n)$. Finally, the transcripts are elements of $\mathbb{G}^* \times (\{0,1\}^n)^{\lambda_0} \times \mathbb{Z}_q$ and, in particular, are always strictly shorter than the corresponding transcripts of $(\hat{P}, \hat{V})$.

Generalizing Figure 3.3 into an RME-based common-bases parallelization of Chaum and Pedersen's protocol is straightforward. Suppose the common input comprises $(\mathbb{G}, q, g_1, \ldots, g_k) \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*; k)$ and a k-tuple $(h_{i1}, \ldots, h_{ik}) \in (\mathbb{G})^k$ for each $i = 1, \ldots, n$, and that P knows an n-witness $(x_1, \ldots, x_n) \in (\mathbb{Z}_q)^n$ such that $g_j^{x_i} = h_{ij}$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, k$. For its announcement, P chooses $r \in \mathbb{Z}_q^*$ and then it computes $A = (A_1, \ldots, A_k)$, where each $A_j = g_j^r$; V checks that $g_j^v = A_j (\prod_{i=1}^n h_{ij}^{t_i})$ for each $j = 1, \ldots, k$. Peng et al. proposed and analyzed the $k = 2$ case of the resulting protocol [PBD07; Figure 2].

3.2.3.3 The RME-based common-exponent Schnorr protocol

Next, we discuss the common exponent parallelization of Schnorr's protocol [PBD07; Figure 4], which requires an extra round of interaction.[16] The protocol is illustrated in Figure 3.4.

[16] As a general rule of thumb, conjunctive parallelizations involving common exponents always require an additional round of interaction, whereas conjunctive parallelizations that only involve common bases typically do not.

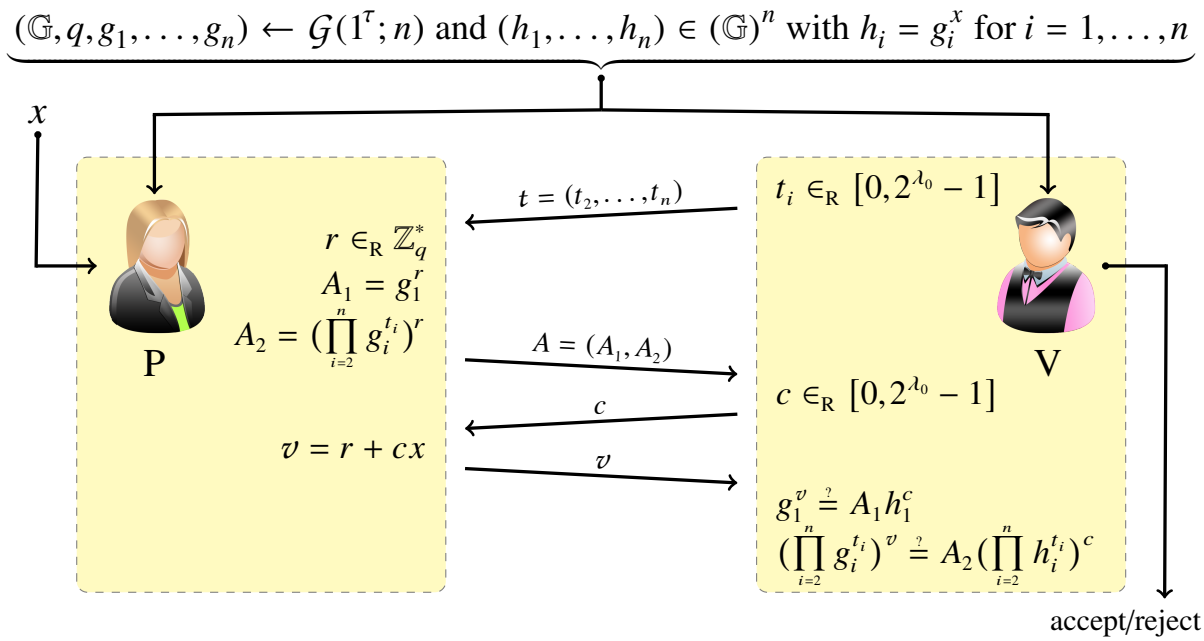

Figure 3.4: A common-exponent batch variant of Schnorr's protocol based on the RME test. The protocol is c-simulatable and component-wise 2-extractable and is denoted by $\mathrm{PK}\{ x : \mathbf{R}_{\mathrm{RME}}^{(\lambda_0)}((h_1, x), \ldots, (h_n, x)) \}$.

Theorem 3.17. *The RME-based common-exponent Schnorr protocol depicted in Figure 3.4 is a system for conjunctive batch honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates for the equality of DLs relations induced by $\mathcal{G}_{\mathrm{DL}}(1^*)$. It is c-simulatable and component-wise 2-extractable and, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$.*

The proof of Theorem 3.17 is similar to the proofs of Theorems 3.15 and 3.16. Note that having two commitments in the announcement and two linearly independent verification equations is necessary to ensure that P^* has zero degrees of freedom in computing the response v. (See our attack on Peng and Bao's protocol in Section 4.2 for an example of why this is important.) We could easily generalize the protocol to prove knowledge and equality among a sequence of k-DLREPs induced by $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*; k)$. In this case, P^*'s response would be a k-tuple $v = (v_1, \ldots, v_k)$ and we would therefore require $k + 1$ linearly independent verification equations in order to guarantee that P has zero degrees of freedom with which to compute v.

3.2.3.4 The PRP$^+$-based common-base Schnorr protocol

One can easily construct a similar variant of Schnorr's protocol from the PRP$^+$ test[17] by implementing the protocol denoted in Camenisch-Stadler notation by

$$\mathrm{PK}\{(x_1,\ldots,x_n) : \mathbf{R}^{(\lambda_0)}_{\mathrm{PRP}^+}((h_1,x_1),\ldots,(h_n,x_n))\}.$$

The expected computation cost for V in the PRP$^+$-based Schnorr protocol on common input $(h_1,\ldots,h_n)$ is slightly lower than the expected cost to evaluate $\mathbf{R}^{(\lambda_0)}_{\mathrm{PRP}^+}((h_1,x_1),\ldots,(h_n,x_n))$ given access to the exponents $(x_1,\ldots,x_n)$; thus, the PRP$^+$-based Schnorr protocol has the lowest expected verification cost among the batch protocols we have considered. In fact, the PRP$^+$-based Schnorr protocol has lower verification cost than any protocol previously proposed in the literature. (Additionally, the expected cost for P is $\mathrm{ExpCost}_{\mathbb{G}}(\tau)$, which is clearly the best possible.)

The communication cost of the PRP$^+$-based Schnorr protocol, however, is much higher than that of the RMP- or RME-based protocols. In particular, V challenges P with ℓ random partitions of $[1,n]$, each into m subsets, plus an additional ℓm random (λ_0+1)-bit scalars for the final RME verification. Each partitioning of $[1,n]$ into m sets requires $n\lceil \lg m\rceil$ bits to describe; thus, as $\ell \approx (\lambda_0+1)/\lg m$, the overall length of V's challenge is about

$$\ell n \lg m + \ell m(\lambda_0+1) \approx n(\lambda_0+1) + m(\lambda_0+1)^2/\lg m \text{ bits};$$

moreover, since $\lg m$ approaches λ_0+1 as n grows large, this latter expression converges to

$$\approx (\lambda_0+1)(n+2^{\lambda_0+1}) \text{ bits}.$$

Theorem 3.18. *The PRP$^+$-based common-base Schnorr protocol is a system for conjunctive batch honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$. It is c-simulatable and component-wise 2-extractable and, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta \le 2^{-\lambda_0}$.*

[17] The diagram for this protocol is quite messy and we therefore omit it.

3.2.4 Communication-efficient conjunctive batch proofs

The communication costs of the RMP-, RME-, and PRP$^+$-based Schnorr protocols are all linear in the fan-in n of their common inputs. This cost may be prohibitively high for some potential use cases; in particular, the batch tests we used to construct the above tests all require the verifier to produce at least $n\lambda_0$ uniform random bits, which become at least $n\lambda_0$ uniform random *challenge bits* that V must send to P in the resulting batch Schnorr protocol. The high communication cost associated with sending so many challenge bits is not the only concern; indeed, merely *producing* such a large number of uniform random bits can be burdensome for V, especially if V is implemented on a server that must regularly engage in high fan-in interactive proofs with many different provers. This section presents some common-base batch variants of Schnorr's protocol that use only *sublinear* random challenge bits. For instance, the next protocol we consider uses $2\lceil\sqrt{n}\rceil(\lambda_0+1)$ random challenge bits, while its most general form uses just $\lceil\lg n\rceil(\lambda_0+\lceil\lg\lg n\rceil)$ random challenge bits, and the last protocol we present requires only $\lambda_0+\lceil\lg n\rceil$ random challenge bits.

3.2.4.1 The $\sqrt[k]{\text{RME}}$-based common-base Schnorr protocol

For our next batch proof system, we introduce a new batch verifier that extends the RME test to use fewer random bits in exchange for a slightly higher expected computation cost. As a first approximation to the new verifier, which we call the $\sqrt[k]{RME}$ *test*, we consider the simple sub-case with $k = 2$. For ease of exposition, we assume that the fan-in of the input predicate is n^2, a square number.

> **$\sqrt{\text{RME}}$ test:** On input an (n^2,τ)-instance $(h_i,x_i)_{i=1}^{n^2}$ for $\mathcal{R}$ in which $n \geq 2$, choose $t_1,\ldots,t_n \in_{\text{R}} [0,2^{\lambda_0+1}-1]$ and $t'_0,\ldots,t'_{n-1} \in_{\text{R}} [0,2^{\lambda_0+1}-1]$, compute the instances (h_0,x_0) in which $h_0 = \prod_{j=0}^{n-1}(\prod_{i=1}^{n} h_{nj+i}^{t_i})^{t'_j}$ and $x_0 = \sum_{j=0}^{n-1} t'_j(\sum_{i=1}^{n} t_i x_{nj+i})$, and then output 1 if (h_0,x_0) is correct (that is, if $h_0 = g^{x_0}$) and output 0 otherwise.

Intuitively, the $\sqrt{\text{RME}}$ test divides an (n^2,τ)-instance into a length-n sequence of (n,τ)-instances, and then it "collapses" each of the resulting (n,τ)-instances into a single instance by computing a random multiexponentiation (and linear combination) of its components, exactly

as in the RME test. The result is a sequence of n (non-batch) instances for $\mathcal{R}$, which we regard as a new (n, τ)-instance to be verified using a second application of the RME test. Note that each of the n random multiexponentiations (and linear combinations) in the first step can use the same sequence of random exponents, while the final RME verification must use a fresh random sequence of exponents; hence, the $\sqrt{\text{RME}}$ test requires the verifier to produce $2n(\lambda_0 + 1)$ uniform random bits, which grows much slower than the $n^2\lambda_0$ uniform random bits required by the standard RME test. (If we swap in the RME^+ test, then we can reduce this cost slightly, to $2(n-1)(\lambda_0 + 1)$ uniform random bits.)

If some component instance (h_{nj+i}, x_{nj+i}) is bad, then, by Lemma 3.5, the sub-instance $((h_{nj+1}, x_{nj+1}), \ldots, (h_{nj+n}, x_{nj+n}))$ that contains it will "collapse" to an incorrect instance in the first step, except with probability at most $2^{-(\lambda_0+1)}$. Likewise, if the input to the second application of the RME test contains a bad instance, then, by a second application of Lemma 3.5, the output of the $\sqrt{\text{RME}}$ test will be 0, except with probability at most $2^{-(\lambda_0+1)}$. The absolute soundness error of the $\sqrt{\text{RME}}$ test therefore satisfies $\delta \leq 2^{-(\lambda_0+1)} + 2^{-(\lambda_0+1)} = 2^{-\lambda_0}$.

The expected computation cost for the $\sqrt{\text{RME}}$ test is at most about $(n+1)\,\text{ExpCost}_{\mathbb{G}}((n, \lambda_0 + 1))$, which is fewer than

$$(n+1)(2n(\lambda_0+1))/\lg n \approx 4n^2\lambda_0/\lg n^2$$

multiplications in $\mathbb{G}$ when n is sufficiently large. Note that this cost is essentially always less (and usually *much* less) than twice that of the standard RME test; hence, the $\sqrt{\text{RME}}$ test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. We summarize these results in the following theorem.

Theorem 3.19. *The $\sqrt{\text{RME}}$ test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. For a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{2/q, 2^{-\lambda_0}\}$.*

The above $\sqrt{\text{RME}}$ test naturally generalizes to a verifier for batch instances for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ with fan-in n^k for arbitrary positive integers $n \geq 2$ and $k \geq 1$:

$\sqrt[k]{\text{RME}}$ test: On input an (n^k, τ)-instance $(h_i, x_i)_{i=1}^{n^k}$ for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$, set $\mathcal{I}_k = (h_i, x_i)_{i=1}^{n^k}$ and then repeat the following steps (in descending order) for $j = k, \ldots, 1$.

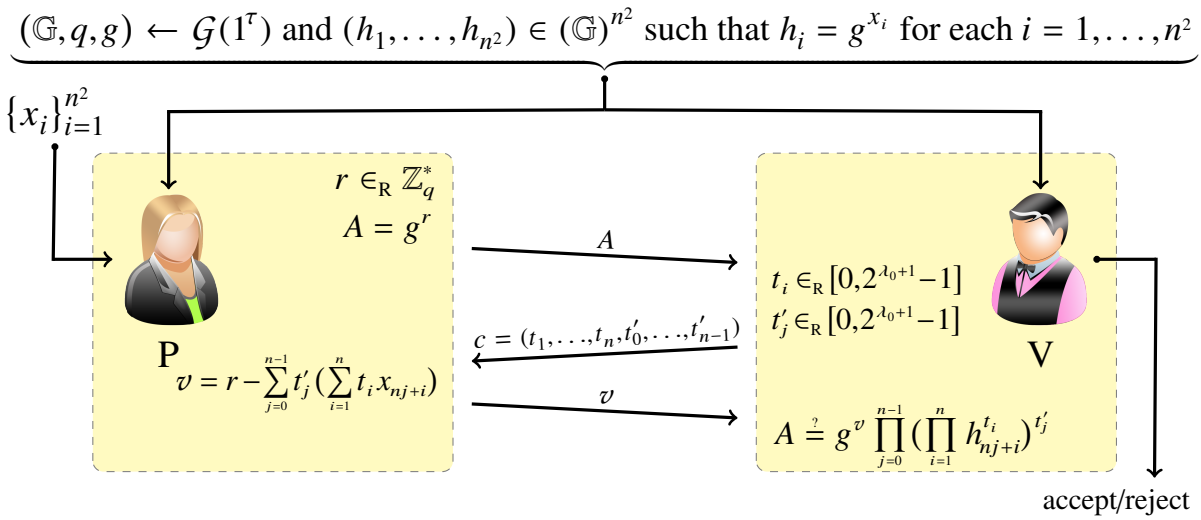

Figure 3.5: A common-base batch variant of Schnorr's protocol based on the $\sqrt{\text{RME}}$ test. This protocol requires V to send just $2n(\lambda_0 + 1)$ uniform random challenge bits to P.

1. Partition $\mathcal{I}_j$ into a length-(n^{j-1}) sequence of (n, τ)-instances $(\mathcal{I}_{ji})_{i=1}^{n^{j-1}}$,
2. choose random exponents $t_1^{(j)}, \ldots, t_n^{(j)} \in_R [0, k\, 2^{\lambda_0} - 1]$,
3. collapse each (n, τ)-instance $\mathcal{I}_{ji}$ into a single instance using the common-base RME test with exponents $(t_1^{(j)}, \ldots, t_n^{(j)})$, and then
4. combine the resulting n^{j-1} instances resulting from step 3 into an (n^{j-1}, τ)-instance, and call it $\mathcal{I}_{j-1}$.

Output 1 if $\mathcal{I}_0 = (h_0, x_0)$ is correct (that is, if $h_0 = g^{x_0}$) and output 0 otherwise.

When $k = 1$, the $\sqrt[k]{\text{RME}}$ test reduces to the RME test. By a similar argument as given above for the $\sqrt{\text{RME}}$ test, if instance $\mathcal{I}_j$ is incorrect, then instance $\mathcal{I}_{j-1}$ can be correct with probability at most $2^{-(\lambda_0 + \lg k)}$; hence, by the union bound, if the original batch instance was not correct, then the final instance $\mathcal{I}_0$ is correct with probability at most $2^{-(\lambda_0+\lg k)} + \cdots + 2^{-(\lambda_0+\lg k)} = k \cdot 2^{-(\lambda_0+\lg k)} = 2^{-\lambda_0}$. In particular, the $\sqrt[k]{\text{RME}}$ test has absolute soundness error satisfying $\delta \leq 2^{-\lambda_0}$ and soundness error function satisfying $\lambda(\tau) \leq \max\{k/q, 2^{-\lambda_0}\}$.

Suppose the input instance has fan-in $N = n^k$. When $n = 2$ so that $k = \lceil \lg N \rceil$, the expected computation cost in the j^{th} level of recursion is about

$$(3 \cdot 2^{j-1}/2)(\lambda_0 + \lceil \lg \lg N \rceil).$$

When $k \in \mathbb{N}^+$ is constant so that $n = \lceil \sqrt[k]{N} \rceil$, the expected computation cost in the j^{th} level of recursion approaches

$$n^{j-1}(1 + (n-1)/\lceil \lg n \rceil)(\lambda_0 + \lceil \lg k \rceil).$$

In both expressions, the only term depending on j is n^{j-1}. Summing this term over $j = 1, \ldots, k$ yields the j^{th} partial sum of a geometric series; thus, this coefficient simplifies to $(n^j - 1)/(n-1) = (N-1)/(n-1)$. When $n = 2$ is fixed and k grows large, the total expected computation cost is around

$$(3(N-1)/2)(\lambda_0 + \lceil \lg \lg N \rceil)$$

multiplications in $\mathbb{G}$. When k is a constant and n grows large, the total expected computation cost is around

$$((N-1)/\lceil \lg n \rceil)(\lambda_0 + \lceil \lg k \rceil)$$

multiplications in $\mathbb{G}$.

Note that it is not strictly required that the input instance have fan-in exactly n^k, provided the actual fan-in N is not larger than n^k. In fact, we can even implement an unbalanced design in which the different levels of recursion partition the batch into different sizes. All that is required is that each level of recursion uses a fresh sequence of uniform random challenges and that these challenges each be at least $\lambda_0 + \lceil \lg k \rceil$ bits long when there are k such recursion levels.

We can use the modified RME test from Section 3.1.4.3 (on Page 48) so that $t_1^{(j)} = 1$ in each recursion level. The total number of random challenge bits required to check an (N, τ)-instance with the resulting $\sqrt[k]{\text{RME}}$ test is then $k(\lceil \sqrt[k]{N} \rceil - 1)(\lambda_0 + \lceil \lg k \rceil)$. We can minimize this cost by setting $k = \lceil \lg N \rceil$ so that the test requires just

$$\lceil \lg N \rceil (\lambda_0 + \lceil \lg \lg N \rceil)$$

random challenge bits. This is a dramatic reduction compared to the $N\lambda_0$ random challenge bits needed for the standard RME test.

Theorem 3.20. *The $\sqrt[k]{\text{RME}}$ test is a batch verifier for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ for any fixed $k \in [1, \lg N]$, where N is the fan-in of the input. For any fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{k/q, 2^{-\lambda_0}\}$.*

We can, of course, construct the corresponding $\sqrt[k]{\text{RME}}$-based common base Schnorr protocol; Figure 3.5 illustrates the special $k = 2$ case of this construction.

Theorem 3.21. *The $\sqrt[k]{\text{RME}}$-based common-base Schnorr protocol is a system for batch honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. It is c-simulatable and, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{k/q, 2^{-\lambda_0}\}$.*

3.2.4.2 The polynomial-based common-base Schnorr protocol

The batch protocols that we have seen so far each require that $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ be a family of *linear* relations $R_\tau \subseteq S_\tau \times W_\tau$. The next protocol, due to Gennaro, Leigh, Sundaram, and Yerazunis [GLSY04] requires further that each domain W_τ of witnesses forms a *commutative ring* and, consequently, $W_\tau[x]$ is a ring of polynomials. Note that $\mathbb{Z}_q$ is a ring (indeed, a field) so that Schnorr's protocol and its variants satisfy this additional requirement.

Consider the single-input Schnorr protocol from the following perspective: The common input $h = g^m$ is a *commitment* to the *slope* $m \in \mathbb{Z}_q$ of a line in $\mathbb{Z}_q[x]$, and P's announcement $A = g^b$ in the first move is a commitment to a random (non-zero) *y-intercept* $b \in \mathbb{Z}_q^*$. In particular, we can view the pair (h, A) as a commitment to the line $\ell(x) = mx + b$ in the polynomial ring $\mathbb{Z}_q[x]$. In the second move, then, V challenges P with a random x-coordinate, $c \in \mathbb{Z}_q$, and P responds in the third move with the corresponding y-coordinate, $v = mc + b$. Finally, in the verification equation, V checks that (c, v) is indeed a point on the committed line $\ell(x)$ by taking advantage of the linearity of the DL relation: $g^v = h^c A = (g^m)^c g^b = g^{mc+b} = g^{\ell(c)}$. Intuitively, since knowledge of any two points on a line is sufficient to recover that line and, hence, its slope, the fact that P successfully produces $v = mc + b$ for a given random, verifier-select challenge $c \in_{\text{R}} \mathbb{Z}_q$ provides overwhelming evidence that P indeed knows the slope m. (In other words, the

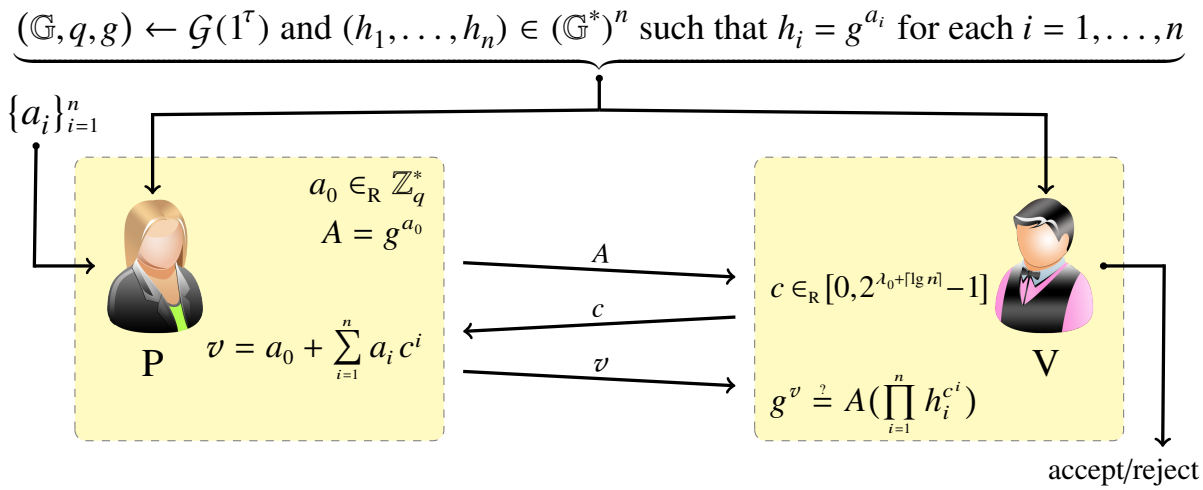

Figure 3.6: A common-base batch variant of Schnorr's protocol due to Gennaro, Leigh, Sundaram, and Yerazunis [GLSY04; §3]. The protocol is c-simulatable and $(n+1)$-extractable.

protocol is 2-extractable.) On the other hand, knowledge of only one point on $\ell(x) = mx + b$ reveals nothing about the slope m and so it would appear that V gains zero knowledge from the interaction. (In other words, the protocol is honest-verifier zero-knowledge.)

In the standard, non-batch parallelization of Schnorr's protocol, P commits to a length-n sequence of random y-intercepts using the announcement $A = (A_1, \ldots, A_n)$ in which each $A_i = g^{b_i}$ for some $b_i \in_R \mathbb{Z}_q^*$; hence, each tuple (h_i, A_i) is a commitment to a line $\ell_i(x) = m_i x + b_i$. V then challenges P to produce the sequence $\ell_1(c), \ldots, \ell_n(c)$. In Chaum and Pedersen's protocol, P commits to the *same line* using two or more distinct generators, and then V checks that all committed lines intersect at the point (c, v). In this case, the fact that the lines intersect at a random, verifier-selected $c \in \mathbb{Z}_q$ provides overwhelming evidence that the lines are all equal and, therefore, that they have the same slope.

Each of the systems for batch zero-knowledge proofs of knowledge that we have seen so far instead have P commit to, and evaluate, some random, verifier-selected *linear combinations* of the above lines $\ell_1(x), \ldots, \ell_n(x) \in \mathbb{Z}_q[x]$. The intuition here is that, if P knows each such line, then it can easily compute any given linear combination of those lines; however, if P^* does not know one or more of the lines, then it knows at most a small fraction of the possible linear combinations of those lines.

Gennaro et al. propose a different way to batch Schnorr's protocol, which we illustrate in Figure 3.6. In particular, they propose to think of the sequence $(h_1, \ldots, h_n)$ not as commitments to the slopes of n lines in $\mathbb{Z}_q[x]$ but, rather, as commitments to the *coefficients* of a degree-n polynomial $f(x) = a_0 + \sum_{i=1}^{n} a_i x^i$ in which $a_0 \in_{\mathrm{R}} \mathbb{Z}_q^*$ and $a_i = \log_g h_i$ for each $i = 1, \ldots, n$. V still challenges P^* to evaluate $f(c)$ for a random, verifier-selected challenge $c \in_{\mathrm{R}} \mathbb{Z}_q$. In this case, P^* may know how to respond to up to n distinct challenges without actually knowing the coefficients of $f(x)$; however, if P^* can respond for any $n+1$ distinct challenges $c_1, \ldots, c_{n+1}$, then it can easily interpolate the points $(c_i, f(c_i))_{i=1}^{n+1}$ to recover the polynomial $f(x) \in \mathbb{Z}_q[x]$ and, thereby, to learn its coefficients. Thus, if P^* does not know $f(x)$, then it can respond to a given challenge with probability at most about n/q. If V selects the challenges uniformly at random from $[0, T-1]$ for some $T < q$, then the absolute soundness error increases to $\delta \leq n/T$. In particular, to get absolute soundness error $\delta = 2^{-\lambda_0}$ we should use $T \geq 2^{\lambda_0 + \lceil \lg n \rceil}$ [GLSY04; Theorem 2].

The polynomial proof is not component-wise 2-extractable: no universal knowledge extractor can extract a witness from P using fewer than n rewinds; however, any set of $n+1$ distinct challenge-response pairs suffices to extract all n witnesses. The polynomial-based Schnorr protocol is therefore $(n+1)$-extractable, just not *component-wise* k-extractable for any $k < n+1$.

Theorem 3.22. *The polynomial-based common-base Schnorr protocol depicted in Figure 3.6 is a system for batch honest-verifier zero-knowledge proofs of knowledge for the language of correct batch predicates over* $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$. *It is c-simulatable and* $(n+1)$*-extractable and, for a fixed soundness parameter* $\lambda_0 \in \mathbb{N}^+$, *it has absolute soundness error* $\delta \leq 2^{-\lambda_0}$ *and soundness error function satisfying* $\lambda(\tau) \leq \max\{n/q, 2^{-\lambda_0}\}$.

Proof. **Complete:** $g^v = g^{a_0 + \sum_{i=1}^{n} a_i c^i} = g^{a_0} \left(\prod_{i=1}^{n} g^{a_i c^i}\right) = g^{a_0} \left(\prod_{i=1}^{n} h_i^{c^i}\right) = A \left(\prod_{i=1}^{n} h_i^{c^i}\right)$.

$(n+1)$-Extractable: Any $n+1$ responses for pairwise distinct challenges $c_1, \ldots, c_{n+1}$ is sufficient for $\mathrm{E}_{\mathrm{P}^*}$ to interpolate the polynomial $f(x)$ and, thereby, to determine its coefficients $a_0, a_1, \ldots, a_n \in \mathbb{Z}_q$. As the coefficients of the polynomial (except for a_0, which is random) correspond to the witnesses that P is proving knowledge of, it follows that the protocol is $(n+1)$-extractable.

Sound: Let M be a binary matrix with a row for each possible announcement $A \in \mathbb{G}^*$ from P^* and a column for each possible challenge $c \in [0, 2^{\lambda_0 + \lceil \lg n \rceil} - 1]$ from V. (Hence, there are $q-1$ rows and $2^{\lambda_0 + \lceil \lg n \rceil}$ columns.) An entry (A, c) of M is 1 if P^* makes V accept in interactions

beginning (A, c) and it is 0 otherwise. Upon receiving the announcement A from P^*, $\mathrm{E}_{\mathrm{P}^*}$ can probe P^* using a random challenge $c_1 \in [0, 2^{\lambda_0+\lceil \lg n \rceil} - 1]$ to learn the value in entry (A, c_1) of M and, through rewinding, $\mathrm{E}_{\mathrm{P}^*}$ can probe P^* using another random challenge $c_2 \in [0, 2^{\lambda_0+\lceil \lg n \rceil} - 1]$ to learn the value of entry (A, c_2) of M. The goal is for $\mathrm{E}_{\mathrm{P}^*}$ to locate a set of $n + 1$ distinct challenges $(c_1, \ldots, c_{n+1})$ for which the values of entries $(A, c_1), \ldots, (A, c_{n+1})$ of M are all 1, in which case the $(n + 1)$-extractability of (P, V) allows $\mathrm{E}_{\mathrm{P}^*}$ to compute the desired exponents.

Suppose the fraction of 1-entries in M is $\epsilon = ((n + 1)/2^{\lambda_0+\lceil \lg n \rceil}) + \varepsilon(s)$ for some $\varepsilon(s) > 0$. As before, at least half of all 1-entries in M reside in rows in which a fraction $\epsilon/2$ or more of the entries are 1. In particular, if $\mathrm{E}_{\mathrm{P}^*}$ locates a 1-entry at location (A, c_1) in M, which happens after an expected $1/\epsilon$ probes, then, with probability at least $1/2$, row A of M contains no fewer than $\epsilon 2^{\lambda_0+\lceil \lg n \rceil}/2 - 1$ additional 1-entries.

***c*-Simulatable:** Given $c \in [0, 2^{\lambda_0+\lceil \lg n \rceil} - 1]$ as input, $\mathrm{S}_{\mathrm{V}}(c)$ chooses $v \in_{\mathrm{R}} \mathbb{Z}_q$, and then it computes $A = g^v / (\prod_{i=1}^{n} h_i^{c^i})$. (If $A = 1$, it selects a new $v \in_{\mathrm{R}} \mathbb{Z}_q$ and tries again.) It is easy to verify that the simulated transcripts (A, c, v) follow the same distribution as the real transcripts.

Asymptotically efficient: The expected computation cost for P is just $\mathrm{ExpCost}_{\mathbb{G}}(\tau) \in \mathrm{o}(\mathcal{P}_\tau(n))$. The expected computation cost for V on input $(h_1, \ldots, h_n)$ is $n \,\mathrm{ExpCost}_{\mathbb{G}}(\lambda_0 + \lceil \lg n \rceil) \leq 3n(\lambda_0 + \lceil \lg n \rceil)/2$ using Horner's method: $\big(h_1(h_2(\cdots(h_{n-1}h_n^c)^c \cdots)^c)^c\big)^c$. As $\lg n \in \mathrm{o}(\tau)$ by assumption, this latter cost is in $\mathrm{o}(\mathcal{V}_\tau(n))$. Finally, the transcript is an element of $\mathbb{G}^* \times \{0,1\}^{\lambda_0+\lceil \lg n \rceil} \times \mathbb{Z}_q$, which is shorter than the corresponding transcripts of $(\hat{\mathrm{P}}, \hat{\mathrm{V}})$, provided $\tau > \lambda_0$.

3.2.5 Cost comparison for batch Schnorr protocols

Tables 3.2–3.4 respectively compare the expected computation cost for P, the expected computation cost for V, and the aggregate communication cost for each of the batch Schnorr variants discussed in this section. From Table 3.2 we see that all but the RMP- and PRP-based Schnorr protocols have constant expected computation cost for P and, moreover, that this cost is identical to P's expected cost in a regular Schnorr protocol with a non-batch input. From Table 3.3 we see that the lowest expected computation cost for V is realized by either the RME- or the PRP^+-based Schnorr protocols, depending on the fan-in of the input predicate. In particular, the RME-based

Table 3.2: The expected computation cost for P in several batch variants of Schnorr's protocol. Each protocol is new, unless otherwise indicated. The costs are measured by the number of multiplications in $\mathbb{G}$ required to get soundness error less than 2^{-40} for a random $(n, 256)$-instance. (Thus, we are using $\tau = 256$ and $\lambda_0 = 40$ so that $\delta \leq 2^{-40}$.) The $\sqrt[k]{\text{RME}}$ row always uses $k = \lceil \lg n \rceil$. For each value of n displayed, the cells containing the lowest expected cost on for (n, τ)-predicates are highlighted.

n	5	10	50	100	500	1 000	5 000	10 000	50 000
Naive	896	1 536	6 656	13 056	64 256	128 256	640 256	1 280 256	6 400 256
RMP	5 376	5 376	5 376	5 376	5 376	5 376	5 376	5 376	5 376
RME[a]	384	384	384	384	384	384	384	384	384
PRP	10 240	10 240	10 496	10 496	10 496	15 232	25 600	33 024	78 592
PRP$^+$	384	384	384	384	384	384	384	384	384
$\sqrt{\text{RME}}$	384	384	384	384	384	384	384	384	384
$\sqrt[3]{\text{RME}}$	384	384	384	384	384	384	384	384	384
$\sqrt[k]{\text{RME}}$	384	384	384	384	384	384	384	384	384
Poly[b]	384	384	384	384	384	384	384	384	384

[a]The RME-based Schnorr protocol was proposed by Peng, Boyd, and Dawson [PBD07].
[b]The polynomial-based Schnorr protocol was proposed by Gennaro, Leigh, Sundaram and Yerazunis [GLSY04].

protocol is most efficient for relatively small fan-ins, while the PRP$^+$-based protocol eventually becomes more efficient when the fan-in grows large. (Note that both the RME- and PRP$^+$-based protocols exhibit constant expected cost for P.)

The $\sqrt[k]{\text{RME}}$- and polynomial-based Schnorr protocols do not appear to be very competitive on Table 3.3; however, we see from Table 3.4 that both protocols have significantly lower communication cost than their peers. The most striking difference is between the cost of the PRP$^+$- and polynomial-based Schnorr protocols: the former requires V to send nearly *3 ooo times* the number of random challenge bits of the latter when the fan-in is $n = 50\,000$. On the other hand, V's expected computation cost in the PRP$^+$-based protocol is only about $(1/16)^{\text{th}}$ that of its cost in the polynomial-based protocol. The $\sqrt[k]{\text{RME}}$-based protocols provide a tuneable trade-off

Table 3.3: The expected computation cost for V in several batch variants of Schnorr's protocol. Each protocol is new, unless otherwise indicated. The costs are measured by the number of multiplications in $\mathbb{G}$ required to get soundness error less than 2^{-40} for a random $(n, 256)$-instance. (Thus, we are using $\tau = 256$ and $\lambda_0 = 40$ so that $\delta \leq 2^{-40}$.) The $\sqrt[k]{\text{RME}}$ row always uses $k = \lceil \lg n \rceil$. For each value of n displayed, the cell containing the lowest expected cost for (n, τ)-predicates is highlighted.

n	5	10	50	100	500	1 000	5 000	10 000	50 000
Naive	2 821	5 386	25 906	51 556	256 756	513 256	2 565 256	5 130 256	25 650 256
RMP	5 476	5 576	6 376	7 376	51 376	25 376	105 376	205 376	1 005 376
RME[a]	484	584	1 384	2 384	10 384	20 384	100 384	200 384	1 000 384
PRP	10 292	10 422	11 416	12 416	20 416	28 115	70 402	112 768	377 980
PRP$^+$	2 035	2 165	3 039	3 856	9 166	13 596	43 301	73 839	279 147
$\sqrt{\text{RME}}$	630	794	1 860	2 639	11 700	22 032	105 180	207 434	1 033 584
$\sqrt[3]{\text{RME}}$	825	1 476	2 589	4 290	14 181	26 025	137 241	245 265	1 123 170
$\sqrt[k]{\text{RME}}$	825	1 329	4 448	8 576	34 110	67 902	540 990	1 081 662	4 325 694
Poly[b]	707	1 044	3 834	7 434	37 134	75 384	397 884	810 384	4 200 384

[a] The RME-based Schnorr protocol was proposed by Peng, Boyd, and Dawson [PBD07].
[b] The polynomial-based Schnorr protocol was proposed by Gennaro, Leigh, Sundaram and Yerazunis [GLSY04].

between these two extremes; for instance, the $\sqrt{\text{RME}}$-based protocol requires about 23 times as many random challenge bits as the polynomial-based protocol (compared to 3 000 times as many for the PRP$^+$-based protocol) and has expected cost for V less than 24% that of its cost in the polynomial-based protocol. The $\sqrt[3]{\text{RME}}$ test reduces the communication cost to just over 6 times that of polynomial-based protocol with an expected cost for V that is still around 26% that of its expected cost in the polynomial-based protocol.

Table 3.4: The bidirectional communication cost for several batch variants of Schnorr's protocol. Each protocol is new, unless otherwise indicated. The costs are measured by the number of bits transmitted to get soundness error less than 2^{-40} for a random $(n, 256)$-instance. We assume that $\mathbb{G}$ is an elliptic curve group and count each $\mathbb{G}$ element as 512 bits and each $\mathbb{Z}_q$ element as 256 bits. (Thus, we are using $\tau = 256$ and $\lambda_0 = 40$ so that $\delta \leq 2^{-40}$.) The $\sqrt[k]{\text{RME}}$ row always uses $k = \lceil \lg n \rceil$. For each value of n displayed, the cell containing the lowest expected cost for (n, τ)-predicates is highlighted.

n	5	10	50	100	500	1 000	5 000	10 000	50 000
Naive	3 840	7 680	38 400	76 800	384 000	768 000	3 840 000	7 680 000	38 400 000
RMP	968	1 168	2 768	4 768	20 768	40 768	200 768	400 768	2 000 768
RME[a]	968	1 168	2 768	4 768	20 768	40 768	200 768	400 768	2 000 768
PRP	60 104	60 304	63 440	65 440	81 440	129 856	352 064	596 608	2 470 016
PRP$^+$	4 088	4 288	5 968	7 968	23 968	45 448	208 688	411 008	2 025 248
$\sqrt{\text{RME}}$	932	1 014	1 342	1 506	2 572	3 310	6 508	8 886	19 054
$\sqrt[3]{\text{RME}}$	894	1 020	1 146	1 272	1 650	1 902	2 910	3 414	5 304
$\sqrt[k]{\text{RME}}$	894	936	1 026	1 069	1 164	1 208	1 340	1 384	1 472
Poly[b]	811	812	814	815	817	818	821	822	824

[a] The RME-based Schnorr protocol was proposed by Peng, Boyd, and Dawson [PBD07].
[b] The polynomial-based Schnorr protocol was proposed by Gennaro, Leigh, Sundaram and Yerazunis [GLSY04].

3.3 Chapter summary

This chapter proposed a new formal model with which to study *batch verifiers* and systems for *batch zero-knowledge proofs and arguments of knowledge*. The new definitions for batch verification improve on the existing definitions proposed by Bellare, Garay, and Rabin [BGR98b] in several important respects, while the new definitions of systems for batch zero-knowledge proofs (or arguments) of knowledge were sorely lacking from the literature. We then suggested a generic construction yielding systems for batch zero-knowledge proofs of complete knowledge for *linear relations*, and we used it to construct several new batch variants of Schnorr's protocol

to prove knowledge of several DLs simultaneously. Compared to existing systems for batch zero-knowledge proofs of knowledge in the literature, the new protocols provide greater flexibility in trading off communication versus computation cost and one of them, the PRP^+-based protocol, has the lowest asymptotic computation cost of any known protocol.

Chapter 4

Batch proofs of partial knowledge

This chapter explores systems for batch zero-knowledge proofs of *partial* knowledge. Our investigation into systems for batch proofs of partial knowledge reveals that the only such protocols in the literature are flawed. To harden those protocols, we introduce a new variant of *trapdoor mercurial commitments* [CHL+13] with a novel binding property. The latter construction also has some interesting implications for zero-knowledge proof systems in general, even when efficiency is not a primary concern.

4.1 Non-batch proofs of partial knowledge

We first discuss systems for proofs of partial knowledge that are *not* batch protocols. Suppose we are given a 2-extractable sigma protocol (P, V) for an infinite family $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ in which each $R_\tau \colon S_\tau \times W_\tau$ is a finite NP-relation. Cramer, Damgård, and Schoenmakers [CDS94] proposed a general methodology for constructing from (P, V) a system for proofs of Γ-partial knowledge over $\mathcal{R}$, where $\Gamma \subseteq \{0,1\}^*$ can be any monotone NP-language. At a high level, Cramer et al.'s method involves converting a system for proofs of *partial* knowledge into a related system for proofs of *complete* knowledge by having P *simulate* the sub-transcripts corresponding to witnesses that it is not proving to know. V checks that each sub-transcript is correct, all the while oblivious to which sub-transcripts are simulated and which sub-transcripts are real. Simultaneously, P proves

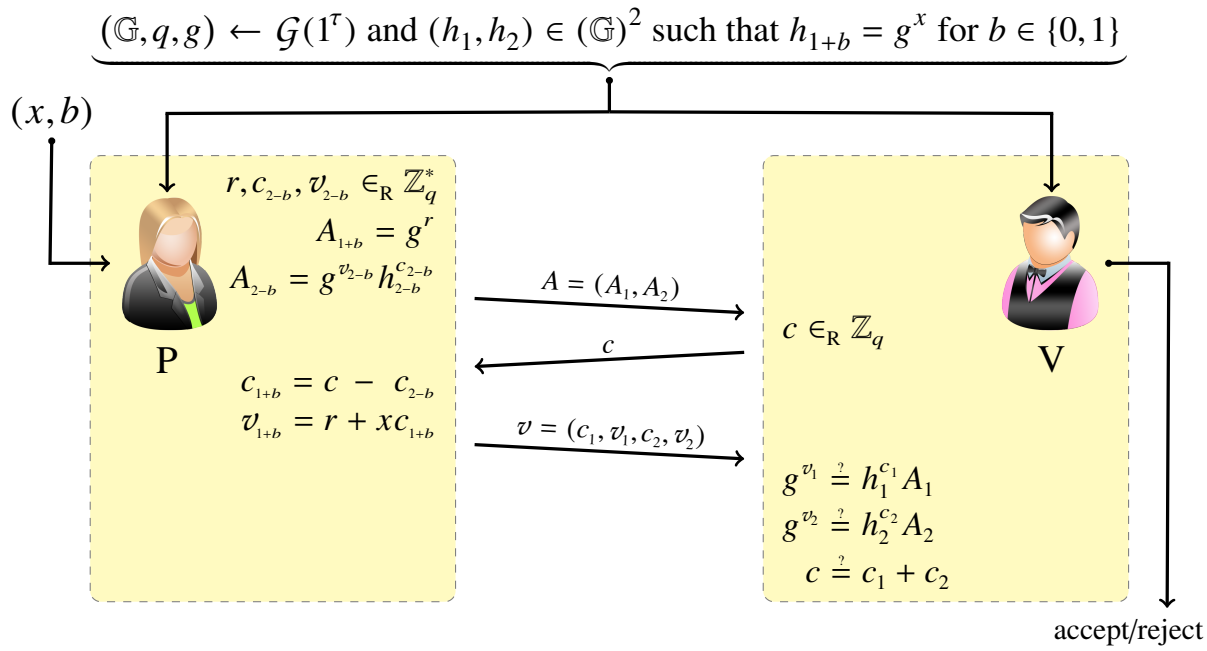

Figure 4.1: A disjunctive variant of Schnorr's protocol due to Cramer, Damgård, and Schoenmakers [CDS94]. The protocol is c-simulatable and 2-extractable. It is denoted by PK$\{ x : (h_1 = g^x) \vee (h_2 = g^x) \}$.

that the *indices* of the simulated sub-transcripts also index the zero bits of some length-n string from Γ. Thus, if the common input is $(s_1, \ldots, s_n) \in S_\tau^n$, then the resulting interaction establishes that P holds a (Γ, n)-witness $(w_1, \ldots, w_n) \in W_\tau^n$ for $(s_1, \ldots, s_n)$ with respect to $\mathcal{R}$.

It is not immediately clear from the above description how to instantiate Cramer et al.'s methodology. After all, if P invokes S_V to simulate a proper subset of the sub-transcripts, then presumably P must choose (or be told) whatever challenges appear in those simulated sub-transcripts *before the simulations complete*; however, for the non-simulated proofs to be sound, it is crucial that V — and not some potentially dishonest P^* — selects the challenges that appear in each of the non-simulated sub-transcripts and, moreover, that it does so *after it receives P's announcements*. P, of course, cannot output announcements for the simulated sub-transcripts until after it runs the simulations; thus, the only option seems to be for P to choose (and commit to, by way of the simulations) the challenges that will appear in a subset of the sub-transcripts, and then for V to issue the remaining challenges to P in a subsequent move. Given the complete set of challenges, P

must then prove the required correspondence between the simulated sub-transcripts and the zero bits of a (secret) string from Γ, all the while ensuring that V cannot later distinguish between the simulated and non-simulated sub-transcripts based on the challenges that appear in them.

Cramer et al. propose an ingenious way to do all this [CDS94]: have the challenge associated with each sub-transcript be a *secret share* from a "suitable" *secret sharing scheme*. We provide a formal definition for secret sharing schemes and clarify what we mean by "suitable" in the next subsection; first, however, we illustrate a *disjunctive* ("OR") variant of Schnorr's protocol, which is constructed from the method just described using a very simple, additive secret sharing scheme.

4.1.1 A disjunctive Schnorr protocol from additive secret sharing

Figure 4.1 illustrates a disjunctive variant of Schnorr's protocol in which P proves knowledge of $x \in \mathbb{Z}_q$ such that either $h_1 = g^x$ or $h_2 = g^x$, without revealing which of the latter two expressions actually holds. The generalization to input predicates with fan-in $n > 2$ is straightforward. If the common input is $(\mathbb{G}, q, g, h_1, \ldots, h_n)$ and the private input to P is $j \in [1,n]$ and $x \in \mathbb{Z}_q$ such that $h_j = g^x$, then we implement the protocol (P, V) denoted in Camenisch-Stadler notation by $\mathrm{PK}\{\, x : \bigvee_{i=1}^{n} (h_i = g^x) \,\}$ as follows.

Protocol 4.1 (Disjunctive variant of Schnorr's protocol).

Common input: $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^r)$ and $(h_1, \ldots, h_n) \in (\mathbb{G})^n$
P's private input: $x \in \mathbb{Z}_q$ and $j \in [1,n]$ such that $h_j = g^x$

P1: P chooses $r \in_{\mathrm{R}} \mathbb{Z}_q^*$ and uses it to compute the announcement $A_j = g^r$. Then, for each $i \in [1,n] \setminus \{j\}$, P selects $c_i \in_{\mathrm{R}} \mathbb{Z}_q$ and runs the simulator to get $(A_i, c_i, v_i) \leftarrow \mathrm{S}_{\mathrm{V}}(\mathbb{G}, q, g, h_i, c_i)$. P announces $A = (A_1, \ldots, A_n)$ to V.

V2: V issues a challenge $c \in_{\mathrm{R}} \mathbb{Z}_q$ to P.

P3: P computes the challenge $c_j = c - \sum_{i \in [1,n] \setminus \{j\}} c_i$ and response $v_j = r + c_j x$ for the non-simulated sub-transcript (A_j, c_j, v_j), and then it sends $v = (c_1, v_1, \ldots, c_n, v_n)$ to V.

V4: V accepts if $g^{v_i} = h_i^{c_i} A_i$ for each $i = 1, \ldots, n$ and if $c = \sum_{i=1}^{n} c_i$; otherwise, it rejects.

The first n verification equations ensure to V that each sub-transcript is accepting and the last verification equation provides overwhelming evidence that at least one sub-transcript was not simulated. In particular, if dishonest P^* simulates every sub-transcript, then the sum of the

challenges $\sum_{i=1}^{n} c_i$ is determined before V chooses $c \in_{\mathrm{R}} \mathbb{Z}_q$; hence, the probability that V's choice for $c \in_{\mathrm{R}} \mathbb{Z}_q$ equals $\sum_{i=1}^{n} c_i$ is just $1/q$. Nonetheless, the interaction transcript (A, c, v) contains no information to help V determine which particular sub-transcripts (A_i, c_i, v_i) are simulated and which sub-transcripts are not.

Theorem 4.2. *Let* $\Gamma_1 = \{t \in \{0,1\}^* \mid t \neq 0 \cdots 00\}$ *be the language of finite bit strings that are not comprised entirely of* 0*s. The disjunctive variant of Schnorr's protocol described in Protocol 4.1 is a system for honest-verifier perfect zero-knowledge proofs of* Γ_1*-partial knowledge over* $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$.

Damgård's lecture notes [Dam11; Theorem 2] contain a detailed proof of Theorem 4.2.

4.1.2 Secret sharing schemes

Let $U \subseteq \mathbb{N}^+$ be a finite subset of the positive integers (typically $U = [1, n]$). An *access structure* Δ on U refers to any collection of subsets of U. If Δ has the property that $H \in \Delta$ and $H' \subseteq U$ with $H \subseteq H'$ implies that $H' \in \Delta$, then Δ is called a *monotone access structure* on U.

Definition 20 (Cramer, Damgård & Schoenmakers, 1994 [CDS94; Definition 5]). Let $U \subseteq \mathbb{N}^+$ be a finite subset of the positive integers and let Δ be a monotone access structure on U. Given a finite commitment domain C, a *smooth secret sharing scheme* for C with access structure Δ is a PPT algorithm that, on input a *secret* $c \in C$, outputs a sequence $(c_1, \ldots, c_{|U|})$ of *secret shares*, such that the following conditions each hold.

1. **Completable:** For any $H \in \Delta$, given the subsequence of shares $(c_i)_{i \in H}$ indexed by H, there is an efficient algorithm that computes the secret c and the entire sequence of shares $(c_1, \ldots, c_{|U|})$.
2. **Independent:** For every $H' \subseteq U$ such that $H' \notin \Delta$, the distribution of the subsequence of shares $(c_i)_{i \in H'}$ indexed by H' is statistically independent of the secret c.
3. **Uniform:** For every $H' \subseteq U$ such that $H' \notin \Delta$, each share in the subsequence of shares $(c_i)_{i \in H'}$ indexed by H' is distributed uniformly in its domain, independent of all the others.

A PPT algorithm that satisfies all but the third condition (uniformity) is called a *semi-smooth secret sharing scheme*. If $(c_1, \ldots, c_n) \in C^n$ so that each share is the same size as the secret, then the secret sharing scheme is said to be *perfect*.

The typical use case for a secret sharing scheme is as follows: A trusted dealer distributes shares of some secret $c \in C$ to a set of $|U|$ shareholders indexed by U. Later, any *authorized subset* of shareholders — that is, any subset of shareholders indexed by a subset of U in the access structure Δ — can cooperate to (efficiently) reconstruct the secret c, but no *unauthorized subset* of shareholders can do so. In fact, no unauthorized subset of shareholders can use their shares to deduce *anything at all* about c because the set of shares they hold is statistically independent of c by definition.

4.1.3 Proofs of partial knowledge for monotone languages

Let ψ_U denote the function that maps each bit string $t \in \{0,1\}^{|U|}$ to the subset $\psi_U(t)$ such that $U_{(i)} \in \psi_U(t)$ if and only if the i^{th} bit of t is equal to 1. Given an arbitrary Boolean language $\Gamma \subseteq \{0,1\}^*$, we call $\Delta = \{\psi_U(t)\}_{t \in \Gamma \cap \{0,1\}^{|U|}}$ the *access structure induced by* Γ on U.

Definition 21. A binary language $\Gamma \subseteq \{0,1\}^*$ is a *monotone language* if, for every finite set $U \subseteq \mathbb{N}^+$, the access structure induced by Γ on U is monotone.

If Δ is an access structure on U, then the *dual* of Δ over U, denoted Δ^*, is the set of subsets of U whose *complements* are not in Δ; that is, $\Delta^* = \{H^* \subseteq U \mid U \setminus H^* \notin \Delta\}$. It is straightforward to verify that the dual of any monotone access structure is itself monotone and, moreover, that $(\Delta^*)^* = \Delta$ [SJM91]. The following observation is another immediate consequence of the definition of a dual access structure.

Observation 4.3 (Jackson and Martin, 1994 [JM94; Result 1]). If Δ is a monotone access structure on U, then a given subset $H \subseteq U$ is in Δ if and only if $H \cap H^*$ is nonempty for every $H^* \in \Delta^*$.

We are especially interested in the equivalent (inverse) formulation of Observation 4.3: $H^* \notin \Delta^*$ if and only if there exists $H \in \Delta$ for which $H \cap H^*$ is empty; thus, if Δ is induced by the monotone language $\Gamma \subseteq \{0,1\}^*$, then the zero bits in a given string $t \in \{0,1\}^{|U|}$ are indexed by a subset $H^* \subseteq U$ that is *not* in Δ^* if and only if $t \in \Gamma$. We can now formally state what we meant earlier by a "suitable" secret sharing scheme for use in Cramer et al.'s construction.

Proposition 4.4 (Cramer, Damgård & Schoenmakers, 1994 [CDS94; Theorems 8 & 9]). *Let (P, V) be a 2-extractable sigma protocol for $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$, let $\Gamma \subseteq \{0,1\}^*$ be any infinite monotone NP-language, and, for each $n \in \mathbb{N}^+$, let Δ_n be the access structure induced by Γ on $[1,n]$. Applying Cramer et al.'s construction to (P, V) results in a system for honest-verifier zero-knowledge proofs of Γ-partial knowledge over $\mathcal{R}$ if there exists a sequence $(C_\tau)_{\tau \in \mathbb{N}^+}$ of challenge domains, with $|C_\tau|$ super-polynomial in τ, such that, on input an (n, τ)-predicate over $\mathcal{R}$, one of the following is true:*

1. *each challenge c_i is a secret share (of the verifier-selected challenge c) in a* smooth *secret sharing scheme for C_τ with access structure Δ_n^*, or*
2. *(P, V) is c-simulatable and each challenge c_i is a secret share (of the verifier-selected challenge c) in a* semi-smooth *secret sharing scheme for C_τ with access structure Δ_n^*.*

To avoid confusion, we herein refer to the prover-selected shares c_i as *sub-challenges* and to the verifier-selected secret c as the *challenge*. Both the simple additive secret sharing scheme that we encountered in Figure 4.1 (and in Protocol 4.1) and Shamir's threshold secret sharing scheme, which we review momentarily, are smooth and perfect. Benaloh and Leichter [BL88] provide a general construction for a semi-smooth secret sharing schemes with any access structure described by a given Boolean formula comprising only AND, OR, and threshold operators. It is easy to see that such formulas are always monotone and, therefore, always describe monotone access structures. The overhead of Benaloh and Leichter's construction depends on the *size* the chosen Boolean function f;[18] that is, it depends on the combined total number of occurrences of all variables in f. In particular, if each variable $s_i \in \{0,1\}$ occurs $m_i \in \mathbb{N}^+$ times in the formula $f(s_1, \ldots, s_n)$, then the i^{th} share in Benaloh and Leichter's construction is an m_i-tuple $c_i \in C^{m_i}$. In the corresponding proof of partial knowledge, P must produce an appropriate announcement and response for each of the $m = \sum_{i=1}^{n} m_i$ challenges; hence, the communication and computation cost of the resulting interaction are each proportional to the size of f. We say that $(f_n)_{n \in \mathbb{N}^+}$ is an infinite family of *polynomial-sized* Boolean formulas if there exists a positive integer-valued function $p(n) \in \mathit{poly}(n)$ such that each $f_n \colon \{0,1\}^n \to \{0,1\}$ has size at most $p(n)$.

[18] The problem of decomposing a given Boolean formula into a set of low-fan-in threshold operators has been well-studied under the name *threshold logic synthesis of Boolean formulas* [SJF08, SR94].

Theorem 4.5. *If a Boolean formula f_n describes the monotone access structure Δ_n, then the formula f_n^* obtained from f_n by replacing each (k,n)-threshold operator with an $(n-k+1,n)$-threshold operator describes the dual Δ_n^* of Δ_n.*[19]

It therefore follows that, given a c-simulatable, 2-extractable honest-verifier zero-knowledge sigma protocol for $\mathcal{R}$ and a monotone language $\Gamma \subseteq \{0,1\}^*$ inducing access structures described by a family of polynomial-sized Boolean formulas, Cramer et al.'s methodology yields a system for honest-verifier zero-knowledge proofs of Γ-partial knowledge over $\mathcal{R}$. Notably, Schnorr's protocol and its common generalizations are all c-simulatable, 2-extractable sigma protocols; thus, we obtain the following corollary to Proposition 4.4.

Corollary 4.6 (Cramer, Damgård & Schoenmakers, 1994 [CDS94; Proposition 6]). *For any language $\Gamma \subseteq \{0,1\}^*$ inducing access structures described by a family of polynomial-sized monotone Boolean formulas, Cramer et al.'s methodology yields systems for honest-verifier zero-knowledge proofs of Γ-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$.*

In Section 4.4.2, we extend the result of Corollary 4.6 to systems for *batch* honest-verifier zero-knowledge proofs of partial knowledge for the above class of monotone languages, and then, in Section 4.4.3, we extend it further to batch honest-verifier zero-knowledge proofs of partial knowledge for certain *non-monotone* languages.

4.1.4 Shamir's (k,n)-threshold secret sharing scheme

A special class of monotone access structures on U are the $(k,|U|)$-*threshold access structures*; that is, the class of access structures of the form $\Delta = \{H \subseteq U \mid |H| \geq k\}$. Shamir's threshold secret sharing scheme [Sha79] is an elegant construction for such threshold access structures. It is easy to confirm that the $(k,|U|)$-threshold access structure on U is dual to the $(|U|-k+1,|U|)$-threshold access structure on U; thus, Shamir's construction also yields a secret sharing scheme with access structure dual to the $(k,|U|)$-threshold access structure on U, for any $k \in [1,|U|]$. We can therefore use Shamir's secret sharing scheme to implement systems for zero-knowledge proofs of "(k,n)-threshold knowledge" for arbitrary thresholds $k \in [1,n]$. The

[19] Note that AND is an (n,n)-threshold operator and OR is a $(1,n)$-threshold operator.

secret sharing procedure for Shamir's (k,n)-threshold construction is described in Algorithm 4.7; reconstruction can use any polynomial interpolation algorithm (see Berrut and Trefethen [BT04], for example). For simplicity, we assume that $U = [1,n]$.

Algorithm 4.7 (Shamir's (k,n)-threshold secret sharing scheme).

Input: A finite field C of possible secrets and a secret value $c \in C$
Output: A sequence $(c_1,\ldots,c_n) \in C^n$ of (k,n)-threshold secret shares of c

1. Select $k-1$ random coefficients $a_1,\ldots,a_{k-1} \in_{\mathrm{R}} C$.
2. Construct the degree-$(k-1)$ polynomial $f(x) = c + \sum_{i=1}^{k-1} a_i x^i$ in $C[x]$.
3. Output the sequence $(c_1,\ldots,c_n) \in C^n$ in which $c_i = f(i)$ for each $i = 1,\ldots,n$.

Given any $H \subseteq [1,n]$ such that $|H| \geq k$, the subsequence of shares $(c_i)_{i\in H}$ is sufficient to interpolate the secret sharing polynomial f and, thereby, to determine both the shared secret $c = f(0)$ and the complete sequence of shares $(f(1),\ldots,f(n)) \in C^n$. However, given any $H' \subseteq [1,n]$ such that $|H'| < k$, the subsequence of shares $(c_j)_{j\in H'}$ is distributed uniformly at random in $C^{|H'|}$, independent of c.

Threshold secret sharing without a secret. Note that in a proof of partial knowledge, there is no actual "secret" to be shared; rather, the "secret" is just V's challenge c, which it broadcasts to P as its only move. In particular, we treat c in a proof of partial knowledge not as a secret, but as *just another share* that happens to have been selected by V instead of P; indeed, just as any subset of k shares is sufficient to reconstruct both the secret c and the entire sequence $(c_1,\ldots,c_n)$, so is any $k-1$ shares and c sufficient to reconstruct the entire sequence $(c_1,\ldots,c_n)$. It is this latter sequence of shares — and not the secret c — that P and V seek to reconstruct in a zero-knowledge proof of partial knowledge.

4.1.4.1 (k,n)-threshold Schnorr protocol from Shamir secret sharing

Let $H \subseteq_k [1,n]$ and let $\overline{H} = [1,n] \setminus H$. As noted above, in a system for proofs of (k,n)-threshold knowledge, the prover-selected sub-challenges should be $(n-k+1,n)$-threshold secret shares of the challenge c. This allows P to select a subsequence $(c_i)_{i\in\overline{H}}$ comprising up to $n-k$ shares for simulations and then, upon receiving c from V, to construct the appropriate degree-$(n-k)$ secret sharing polynomial f with which to compute the remaining k shares, $(c_j)_{j\in H}$. Note that,

although V cannot identify H from the sequence $(c_1, \ldots, c_n)$, it can easily confirm that $|H| \geq k$ by (i) interpolating $(c_1, \ldots, c_n)$ to find f, and then (ii) checking that $\deg f \leq n - k$ and that $f(0) = c$. (In practice, it is more efficient to have P send V the $n - k$ free coefficients of $f(x)$ instead of the length-n sequence $(c_1, \ldots, c_n)$ of its evaluations. V then uses f to compute the sub-challenges $c_i = f(i)$ for each $i = 1, \ldots, n$.)

Given the above, we can instantiate the protocol denoted in Camenisch-Stadler notation by $\mathrm{PK}\{(H, x_1, \ldots, x_k) : H \subseteq_k [1,n] \wedge (\bigwedge_{i=1}^{k} h_{H_{(i)}} = g^{x_i})\}$ using Protocol 4.8.

Protocol 4.8 ((k, n)-threshold variant of Schnorr's protocol).

Common input: $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ and $(h_1, \ldots, h_n) \in (\mathbb{G})^n$
P's private input: $H \subseteq_k [1,n]$ and $(x_1, \ldots, x_k) \in (\mathbb{Z}_q)^k$ with $h_{H_{(j)}} = g^{x_j}$ for $j = 1, \ldots, k$

P1: Let $\overline{H} = [1,n] \setminus H$. For each $j = 1, \ldots, k$, P chooses $r_j \in_{\mathrm{R}} \mathbb{Z}_q$ and uses it to compute the announcement $A_{H_{(j)}} = g^{r_j}$. Then, for each $i \in \overline{H}$, P selects a sub-challenge $c_i \in_{\mathrm{R}} \mathbb{Z}_q$ and runs the simulator to get $(A_i, c_i, v_i) \leftarrow \mathrm{S}_{\mathrm{V}}(\mathbb{G}, q, g, h_i, c_i)$. P announces $A = (A_1, \ldots, A_n)$ to V.

V2: V issues a challenge $c \in_{\mathrm{R}} \mathbb{Z}_q$ to P.

P3: P uses polynomial interpolation to compute the degree-$(n-k)$ polynomial $f \in \mathbb{Z}_q[x]$ passing through $(0, c)$ and through each point in $\{(i, c_i)\}_{i \in \overline{H}}$. Then, for each $j = 1, \ldots, k$, it computes the sub-challenge $c_{H_{(j)}} = f(H_{(j)})$ and response $v_{H_{(j)}} = r_j + c_{H_{(j)}} x_j$ for the non-simulated sub-transcript $(A_{H_{(j)}}, c_{H_{(j)}}, v_{H_{(j)}})$. P sends $v = (f, v_1, \ldots, v_n)$ to V.

V4: V accepts if $g^{v_i} = h_i^{f(i)} A_i$ for each $i = 1, \ldots, n$ and if $\deg f \leq n - k$ with $f(0) = c$, and it rejects otherwise.

Theorem 4.9 (Cramer, Damgård & Schoenmakers, 1994 [CDS94; Corollary 12]). *Let $(k, n) \in \mathbb{N} \times \mathbb{N}^+$ with $k \leq n$ and let $\Gamma_k \subseteq \{0,1\}^*$ denote the language of finite bit strings with Hamming weight k or greater. The (k, n)-threshold variant of Schnorr's protocol described in Protocol 4.8 is a system for honest-verifier zero-knowledge proofs of Γ_k-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$.*

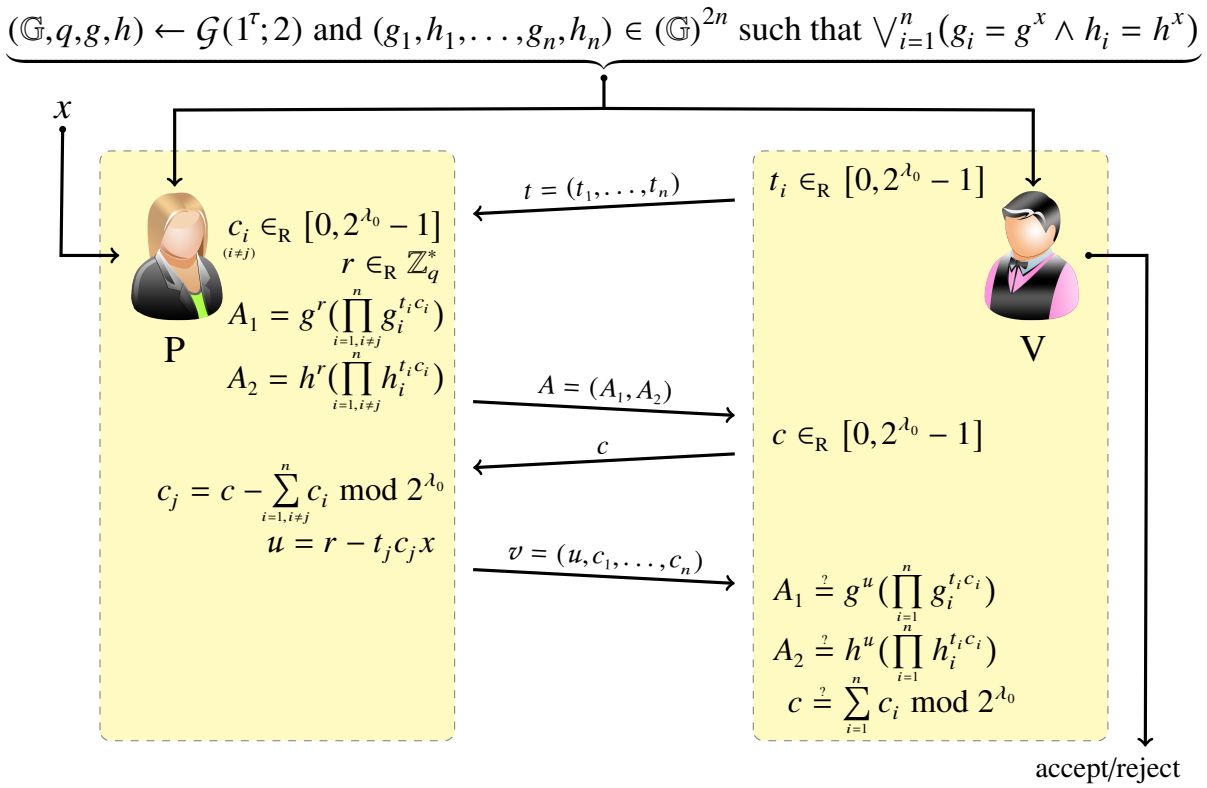

Figure 4.2: An interactive protocol due to Peng and Bao [PB08; Figure 2]. The protocol was intended to be a disjunctive batch variant of Chaum and Pedersen's protocol for honest-verifier zero-knowledge proofs of knowledge and equality among DLs; *however, the protocol as specified is not sound.*

4.2 Peng and Bao's proofs of disjunctive knowledge

The protocol (P,V) depicted in Figure 4.2 is due to Peng and Bao [PB08; §5.1]. The common input to (P,V) is $(\mathbb{G},q,g,h) \leftarrow \mathcal{G}(1^\tau;2)$ and $(g_1,h_1,\ldots,g_n,h_n) \in (\mathbb{G})^{2n}$; the protocol is intended to be a common-base batch variant of Chaum and Pedersen's protocol in which P proves knowledge of an exponent $x \in \mathbb{Z}_q$ and a secret index $j \in [1,n]$ for which $g_j = g^x$ and $h_j = h^x$. A review of the literature reveals no other protocols that purport to be systems for batch zero-knowledge proofs of *partial* knowledge, aside from two similar protocols in Peng and Bao's same paper and one protocol [CY08] that is known from previous work to be insecure [Pen12].

Conceptually, P proceeds just like it would in the non-batch protocol: it first chooses $r \in_R \mathbb{Z}_q^*$ and uses it to compute the "real" announcements $A_{j1} = g^r$ and $A_{j2} = h^r$ for the non-simulated sub-transcript. Then, for each $i \in [1,n] \setminus \{j\}$, P invokes the simulator S_V to produce the sub-transcript $(A_{i1}, A_{i2}, c_i, v_i) \leftarrow S_V(\mathbb{G}, q, g, h, g_i, h_i, c_i)$ satisfying $A_{i1} = g^{v_i} g_i^{c_i}$ and $A_{i2} = h^{v_i} h_i^{c_i}$. If P were to announce $A' = (A_{11}, A_{12}, \ldots, A_{n1}, A_{n2})$ to V at this point, then the two could complete the proof by engaging in a straightforward RME-based variant of Chaum and Pedersen's protocol using A' as the common input; however, computing and transmitting such a linear-sized announcement A' is very costly. To avoid this cost, Peng and Bao have P — *without ever showing the individual commitments A_{ik} to V* — batch the announcement as $A = (A_1, A_2)$ such that

$$A_1 = \prod_{i=1}^{n} A_{i1}^{t_i c_i} \quad \text{and} \quad A_2 = \prod_{i=1}^{n} A_{i2}^{t_i c_i}$$

where each exponent $t_i \in [0, 2^{\lambda_0} - 1]$ is selected in an opening move by V.

In practice, P need not actually compute the individual A_{ik} as it can compute A_1 and A_2 directly at a much lower cost. P sends the RME-batched announcement $A = (A_1, A_2)$ to V, and the two proceed — *as if V had seen the individual A_{ik} and subsequently computed (A_1, A_2) on its own* — just as they would in the aforementioned RME-based variant of Chaum and Pedersen's protocol with common input $A' = (A_{11}, A_{12}, \ldots, A_{n1}, A_{n2})$.

Theorem 1 in Peng and Bao's paper [PB08] states that "Soundness in [the protocol depicted in Figure 4.2] only fails with an overwhelmingly small probability [in the soundness parameter λ_0]." Their proof derives an upper bound of $\delta = 2^{-\lambda_0}$ on the absolute soundness error of the protocol using an argument similar to the one we used to prove Theorem 3.4; however, their proof assumes (incorrectly) that the RME-batched announcement $A = (A_1, A_2)$ binds P^* to a subsequence of *all but one* sub-challenge c_j from the sequence $(c_1, \ldots, c_n)$. If that were true, then P would have zero degrees of freedom to compute $c_j = c - \sum_{i \in [1,n] \setminus \{j\}} c_i$, and Peng and Bao's argument would hold to establish that the protocol is 2-extractable with absolute soundness error $\delta = 2^{-\lambda_0}$. Unfortunately, *the announcement $A = (A_1, A_2)$ does not provide the needed binding property*; indeed, P^* can sometimes exploit this observation in order to solve for several sequences $(c_1, \ldots, c_n)$ that are each consistent both with the announcement and with the challenge issued by V. In such cases, P^* may cause V to accept even when the common input is not Γ_1-correct. We give a high-level description of the attack below; additional details on how to implement

the attack are located in Appendix B. (We also note that the attack naturally extends to attacks against the other two systems for batch zero-knowledge proofs of Γ_1-partial knowledge in Peng and Bao's paper [PB08; Figures 3 & 4].)

4.2.1 Attacking Peng and Bao's protocol

Suppose that $\log_g g_i \neq \log_h h_i$ for each $i = 1, \ldots, n$, but P^* knows several pairs of exponents $(x_j, u_j) \in (\mathbb{Z}_q)^2$ for which $g_j = g^{x_j}$ and $h_j = h^{u_j}$.[20] Partition $[1,n]$ into a pair $(H, \overline{H})$ of disjoint sets in which H comprises (a subset of) the indices $j \in [1,n]$ for which P^* knows the above (x_j, u_j) pair and $\overline{H} = [1,n] \setminus H$ comprises (a superset of) the indices $i \in [1,n]$ for which P does not know any such pair. Note that in some reasonable settings, P^* may know *every* such pair so that H can be all of $[1,n]$. In its first move, P^* chooses $c_i \in_{\mathrm{R}} [0, 2^{\lambda_0} - 1]$ only for the $i \in \overline{H}$, and it likewise computes the announcement $A = (A_1, A_2)$ only over $\overline{H}$, so that

$$A_1 = g^r \Big(\prod_{i \in \overline{H}} g_i^{t_i c_i}\Big)$$

and

$$A_2 = h^r \Big(\prod_{i \in \overline{H}} h_i^{t_i c_i}\Big).$$

Such an announcement provides P^* with $|H| - 1$ degrees of freedom when it comes time to compute the missing sub-challenges $(c_j)_{j \in H}$. In particular, P can solve for $(c_j)_{j \in H}$ to satisfy the following system of two linear equations in $k = |H|$ unknowns:

$$0 = \sum_{j \in H} c_j t_j (x_j - u_j) \tag{4.1}$$

and

$$c' = \sum_{j \in H} c_j \bmod 2^{\lambda_0}, \tag{4.2}$$

where $c' = c - \sum_{i \in \overline{H}} c_i \bmod 2^{\lambda_0}$. Equation (4.1) implies that $\sum_{j \in H} c_j t_j x_j = \sum_{j \in H} c_j t_j u_j$; hence, setting $u = r - \sum_{j \in H} c_j t_j x_j$, the response $v = (u, c_1, \ldots, c_n)$ satisfies each of V's verification equations, yet P^* does not know any (Γ_1, n)-witness for the common input.

[20] Of course, the attack works the same when $\log_g g_j = \log_h h_j = x_j$ for some $j \in [1,n]$ but P^* knows no such x_j.

Of course, if P* just naively solves the above system of equations and obtains a solution $(c_j)_{j\in H}$ having some $c_{j'} \geq 2^{\lambda_0}$, then V can detect that P* is cheating, since all sub-challenges should be elements of $[0, 2^{\lambda_0} - 1]$. Therefore, what P* really wants to do is find a solution to the above system subject to the additional restriction that $c_j \in [0, 2^{\lambda_0} - 1]$ for each $j \in H$.

A counting argument suggests that "sufficiently small" solutions are plentiful whenever $k\lambda_0$ is "sufficiently large" compared to $\tau \approx \lg q$.[21] If $\mathbf{X}$ is an instance of the above system induced by an actual interaction between P* and honest V, then we heuristically expect the distribution of solutions of $\mathbf{X}$ to be uniform among all possible $(c_{j_1}, \ldots, c_{j_k}) \in (\mathbb{Z}_q)^k$; in particular, we expect the proportion of solutions that are sufficiently small to be about $(2^{\lambda_0}/q)^k$. Now, only q^{k-1} of the $(c_{j_1}, \ldots, c_{j_k}) \in (\mathbb{Z}_q)^k$ can satisfy Equation (4.1) and, of those, only about $q^{k-1}/2^{\lambda_0}$ can simultaneously satisfy Equation (4.2). This leads us to conclude that $\mathbf{X}$ has around

$$(q^{k-1}/2^{\lambda})(2^{\lambda}/q)^k = (2^{\lambda})^{k-1}/q$$

sufficiently small solutions. Appendix B discusses how P* can find one of these solutions by solving a short vector search problem in a particular lattice of dimension $k + 3$. When k is reasonably small, P* can use a standard basis reduction algorithm, such as the celebrated Lenstra-Lenstra-Lovász algorithm [LLL82], to find a sufficiently small solution very quickly. For example, if $\lambda_0 = 40$ and $\lg q \approx 256$, then P* only needs to know $k = 8$ exponent pairs to find a sufficiently small solution. In our experiments, solving the above short vector search problem with $\lambda_0 = 40$, $\tau = 256$, and $k = 8$ took about 12 ms per trial, on average.

Towards repairing Peng and Bao's protocol

The attack just outlined is only possible because cheating P* can wait until after it receives the challenge c to choose the sub-challenges $(c_j)_{j\in H}$ for a subset of indices H such that $|H| > 1$. In a standard, non-batch proof of partial knowledge, this does not pose any problem because each sub-challenge that P* defers choosing introduces an additional linear equation to the system that

[21] Recall that $k = |H|$ is a lower bound on the number of exponent pairs that P* knows and that λ_0 is the *soundness parameter*. Higher values of λ_0 are *supposed* to result in better soundness; however, what we find is just the opposite: higher values of λ_0 only make sufficiently small solutions to the above system more plentiful and easier for cheating P* to compute.

P^* must ultimately solve in order to produce an accepting response. If we wish to maintain a fixed number of verification equations, as in Peng and Bao's protocol, then we must ensure that the announcement binds P^* to a specific choice for all but one of the sub-challenges.

Such a fix would require a special kind commitment scheme with which P can commit to all but one sub-challenge from the sequence $(c_1, \ldots, c_n)$ as part of its announcement and then, upon receiving the challenge c from V, compute the missing c_j and open the commitment to the *entire* sequence $(c_1, \ldots, c_n)$ without revealing the index j. Similarly, given a commitment scheme with which P can commit to *all but* k sub-challenges from $(c_1, \ldots, c_n)$ and later open the commitment to all of $(c_1, \ldots, c_n)$ without revealing the set of k shares that were chosen after receiving c — yet explicitly revealing the number k of such shares — we could easily generalize the repaired protocol to an RME-based system for proofs of (k, n)-threshold knowledge.

More generally still, given an arbitrary monotone language $\Gamma \subseteq \{0,1\}^*$ and an efficient commitment scheme with a suitable generalization of the above binding property, we could implement a system for batch honest-verifier zero-knowledge proofs of Γ-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. In the next section, we formally define such commitments for arbitrary binary languages $\mathcal{L} \subseteq \{0,1\}^*$, and in Section 4.4.1 we present an efficient construction for languages inducing $(n-k, n)$-threshold access structures.

4.3 $\mathcal{L}$-mercurial commitments

Let $C \subseteq \mathbb{N}^+$ denote a fixed, finite set of positive integers called the *commitment domain*. Standard cryptographic commitments provide a means by which a committer $\mathcal{A}$ can output a short string — called a *commitment* — that *binds* it to a particular choice of $c \in C$, while temporarily *hiding* its choice from the recipient $\mathcal{B}$. $\mathcal{A}$ can subsequently *open* the commitment to c, thereby convincing $\mathcal{B}$ that $\mathcal{A}$ had indeed chose that particular value prior to outputting the commitment. A *trapdoor mercurial commitment scheme* [CHL+05; §2.2] is a special type of commitment scheme with a carefully relaxed binding property that provides $\mathcal{A}$ with two distinct options:

1. $\mathcal{A}$ can "*hard commit*" to a specific value $c \in C$, much like it would do using a standard cryptographic commitment, or
2. $\mathcal{A}$ can "*soft commit*" to *nothing at all*.

A "soft commitment" *looks* just like a "hard commitment" (indeed, the two kinds of commitments are perfectly indistinguishable), but, unlike a hard commitment, a soft commitment does not bind P to any specific value and P can, therefore, open the soft commitment arbitrarily. For such a scheme to be useful, there needs to be an efficient way for P to (optionally) assert, upon opening a commitment, that the given commitment is not a soft commitment. Whereas opening a commitment to x together with such an assertion provides overwhelming evidence that P indeed chose x prior to outputting the commitment, opening that same commitment to x with no such assertion merely provides overwhelming evidence that P did not choose any different value $y \neq x$; in particular, it could be that P hard-committed to x, or it could be that P soft committed to nothing at all.

A variant of mercurial commitments called *trapdoor n-mercurial commitments* [CFM08] provide a means by which $\mathcal{A}$ can commit to an entire length-n sequence of values from C using a single *fixed-length* commitment. $\mathcal{A}$ can open such a trapdoor n-mercurial commitment either *component-wise* or *vector-wise*. Note that the binding property here is an "all-or-nothing" generalization of the above mercurial binding property: trapdoor n-mercurial commitments always binds $\mathcal{A}$ either to an *entire* length-n sequence of values from C or to nothing at all — there are no intermediate possibilities.

The commitment scheme we seek in order to fix Peng and Bao's protocol is not a trapdoor n-mercurial commitment scheme; nevertheless, whatever the scheme, it must have a decidedly mercurial-esque binding property. We propose the new commitments below, calling them *trapdoor $\mathcal{L}$-mercurial commitments* to highlight this connection to trapdoor n-mercurial commitments. In a recent technical report, the author and Ian Goldberg [HG12] provided a thorough exposition for the special case of *trapdoor $(n-k,n)$-threshold mercurial commitments* and discussed several possible applications for the same. (The technical report refers to such commitments as *all-but-k mercurial commitments*.) The discussion below is both more general and more limited than the one in that technical report. In particular, below we restrict our attention to the subset of functionality we need to implement systems for batch zero-knowledge proofs of partial knowledge, providing a somewhat simplified definition and construction. The more general definition in our technical report considers both *vector-wise* and *component-wise* openings of trapdoor $(n-k,n)$-threshold mercurial commitments, while this dissertation only considers the former

vector-wise openings.[22] On the other hand, the definition in the technical report only considers trapdoor $(n-k,n)$-threshold mercurial commitments and here we extend this notion to trapdoor $\mathcal{L}$-mercurial commitments for arbitrary Boolean languages $\mathcal{L} \subseteq \{0,1\}^*$.

4.3.1 Formal definition

Intuitively, a *trapdoor $\mathcal{L}$-mercurial commitment scheme* is a generalization of a trapdoor n-mercurial commitment scheme with a binding property indicated by the access structures Δ_n induced by $\mathcal{L}$ on $[1,n]$ for each $n \in \mathbb{N}^+$. In particular, a trapdoor $\mathcal{L}$-mercurial commitment binds the committer to a *hidden subsequence* of a sequence from C^n: given an authorized subset $\overline{H} \in \Delta_n$, $\mathcal{A}$ can *hard commit* to the subsequence $(c_i)_{i \in \overline{H}}$ and subsequently open the commitment to any sequence $(c_1, \ldots, c_n)$ consistent with this initial commitment.

Upon opening the commitment, $\mathcal{A}$ can optionally reveal (with proof) the set $\overline{H}$ indexing the hard-committed subsequence. If $\mathcal{A}$ does not choose to *explicitly* reveal the set $\overline{H}$, then the *$\mathcal{L}$-mercurial hiding property* ensures that $\mathcal{B}$ learns nothing about $\overline{H}$ beyond that $\overline{H} \in \Delta_n$, a fact which is guaranteed by the *$\mathcal{L}$-mercurial binding property*. We observe that

1. trapdoor n-mercurial commitments are equivalent to trapdoor $\mathcal{L}$-mercurial commitments for the language $\mathcal{L} = \{0\}^* \cup \{1\}^*$,[23] and
2. *non-mercurial* vector commitments are equivalent to trapdoor $\mathcal{L}$-mercurial commitments for the language $\mathcal{L} = 1^*$.

We therefore view trapdoor $\mathcal{L}$-mercurial commitments as a substantial generalization both of trapdoor n-mercurial commitments and of standard, non-mercurial vector commitments.

Definition 22. Let $\mathcal{L} \subseteq \{0,1\}^*$ be an infinite binary language and, for each $n \in \mathbb{N}^+$, let Δ_n be the access structure induced by $\mathcal{L}$ on $[1,n]$. A *trapdoor $\mathcal{L}$-mercurial commitment scheme* is a 4-tuple of PPT algorithms $(\mathrm{KeyGen}_{\mathcal{L}}, \mathrm{Com}, \mathrm{Open}, \mathrm{Ver})$, which are executed by entities taking on three roles — the *trusted initializer*, the *committer*, and the *recipient*.

[22] Removing support for component-wise opening makes the construction both more efficient and easier to analyze.
[23] Asserting that such a commitment is not soft corresponds to revealing (with proof) that $\overline{H} = [1,n]$.

1. $\mathrm{KeyGen}_{\mathscr{L}}(C, 1^\tau, n)$ is run once by the *trusted initializer* to produce public parameters for commitments that open to sequences from C^n. Given a finite commitment domain C, a security parameter 1^τ, and a sequence length $n \in \mathbb{N}^+$, it outputs public parameters $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n)$, securely deletes its own memory, and then permanently exits the scene.

The trapdoor $\mathscr{L}$-mercurial commitments created with $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n)$ and opening to sequences in C^n are called $(\mathscr{L}, n)$-*mercurial commitments*. The next three algorithms take $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n)$ as an *implicit* input.

2. $\mathrm{Com}((c_i)_{i \in \overline{H}})$ is run by the *committer* to commit to a subsequence of values $(c_i)_{i \in \overline{H}}$ in which each $c_i \in C$. If $\overline{H} \in \Delta_n$, then it outputs a trapdoor $(\mathscr{L}, n)$-mercurial commitment $\mathscr{C}$ binding $\mathscr{A}$ to $(c_i)_{i \in \overline{H}}$ and *auxiliary information* γ for the later opening of $\mathscr{C}$; otherwise, it halts without output.
3. $\mathrm{Open}(\mathscr{C}, \gamma, c_1, \ldots, c_n)$ is run by the *committer* to open a trapdoor $(\mathscr{L}, n)$-mercurial commitment $\mathscr{C}$ to the sequence $(c_1, \ldots, c_n) \in C^n$. If $(c_1, \ldots, c_n)$ is consistent with the initial subsequence $(c_i)_{i \in \overline{H}}$ committed to by $\mathscr{C}$, then it outputs the *decommitment* π proving this fact; otherwise, it halts without output.
4. $\mathrm{Ver}(\mathscr{C}, \pi, c_1, \ldots, c_n)$ is run by the *recipient* to verify that $(c_1, \ldots, c_n) \in C^n$ is a valid opening of $\mathscr{C}$. It outputs 1 if the decommitment π is correct and it outputs 0 otherwise.

Definition 23. A trapdoor $\mathscr{L}$-mercurial commitment scheme $(\mathrm{KeyGen}_{\mathscr{L}}, \mathrm{Com}, \mathrm{Open}, \mathrm{Ver})$ is a *secure trapdoor $\mathscr{L}$-mercurial commitment scheme* if it provides the following three guarantees.

1. **Complete:** Let $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n) \leftarrow \mathrm{KeyGen}_{\mathscr{L}}(C, 1^\tau, n)$ and let Δ_n be the access structure induced by $\mathscr{L}$ on $[1, n]$. For all $(c_1, \ldots, c_n) \in C^n$ and for all $\overline{H} \in \Delta_n$, if $(\mathscr{C}, \gamma) \leftarrow \mathrm{Com}((c_i)_{i \in \overline{H}})$ and $\pi \leftarrow \mathrm{Open}(\mathscr{C}, \gamma, c_1, \ldots, c_n)$, then $\mathrm{Ver}(\mathscr{C}, \pi, c_1, \ldots, c_n) = 1$.
2. **(Unconditional) $\mathscr{L}$-mercurial hiding:** There exists a negligible function $\varepsilon \colon \mathbb{N} \to \mathbb{R}^+$ such that no (possibly unbounded) recipient $\mathcal{B}$ can win the following indistinguishability game against an honest committer $\mathscr{A}$ with a probability exceeding $1/2 + \varepsilon(\tau)$.[24]

[24] Despite hiding being unconditional, we permit $\mathcal{B}$ to win with success probability $1/2 + \varepsilon(\tau)$ for a negligible function $\varepsilon(\tau)$ because the $\mathscr{L}$-mercurial hiding property in our own trapdoor $(n-k, n)$-threshold mercurial commitments relies on a protocol that is *statistical* zero-knowledge.

Game: $\mathscr{L}$-mercurial indistinguishability game

Initialization: A trusted initializer generates $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n) \leftarrow \text{KeyGen}_{\mathscr{L}}(C, 1^\tau, n)$, and then it passes $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n)$ to $\mathcal{A}$ and $\mathcal{B}$.

(a) $\mathcal{B}$ chooses $(c_1, \ldots, c_n) \in C^n$ and $\overline{H}_1, \overline{H}_2 \in \Delta_n$ and sends them to $\mathcal{A}$.

(b) $\mathcal{A}$ tosses an unbiased coin to obtain $b \in_{\text{R}} \{0,1\}$, and then it computes $(\mathscr{C}, \mathfrak{y}) \leftarrow \text{Com}\big((c_i)_{i \in \overline{H}_b}\big)$ and $\pi \leftarrow \text{Open}(\mathscr{C}, \mathfrak{y}, c_1, \ldots, c_n)$ and sends $(\mathscr{C}, \pi)$ to $\mathcal{B}$.

(c) $\mathcal{B}$ outputs a bit $b' \in \{0,1\}$.

Outcome: $\mathcal{B}$ wins the game if and only if $b = b'$.

3. **(Computational) $\mathscr{L}$-mercurial binding:** For any positive integer-valued function $n(\tau) \in \mathit{poly}(\tau)$, there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n(\tau)) \leftarrow \text{KeyGen}_{\mathscr{L}}(C, 1^\tau, n(\tau))$ is output by a trusted initializer and if $\mathscr{C} \leftarrow \mathcal{A}(\textsc{Pub}_{\mathscr{L}}(C, 1^\tau, n(\tau)))$, then there exists a fixed set $\overline{H} \in \Delta_{n(\tau)}$ such that, if $\big((\pi^{(0)}, c_1^{(0)}, \ldots, c_{n(\tau)}^{(0)}), (\pi^{(1)}, c_1^{(1)}, \ldots, c_{n(\tau)}^{(1)})\big) \leftarrow \mathcal{A}(\mathscr{C})$ with $\text{Ver}(\mathscr{C}, \pi^{(0)}, c_1^{(0)}, \ldots, c_{n(\tau)}^{(0)}) = 1$ and $\text{Ver}(\mathscr{C}, \pi^{(1)}, c_1^{(1)}, \ldots, c_{n(\tau)}^{(1)}) = 1$, then

$$\Pr\Big[\, \overline{H} \setminus \{\, i \in [1, n(\tau)] \;\mid\; c_i^{(0)} = c_i^{(1)} \,\} \neq \emptyset \Big] \leq \varepsilon(\tau).$$

Intuitively, the computational $\mathscr{L}$-mercurial binding property states that, for a given $(\mathscr{L}, n)$-mercurial commitment $\mathscr{C}$ output by a PPT committer $\mathcal{A}$, there exists a fixed subset $\overline{H} \in \Delta_n$ such that the sequences to which $\mathcal{A}$ can feasibly open $\mathscr{C}$ all "agree" on the subsequence indexed by $\overline{H}$. As equality is transitive, this trivially extends to openings of $\mathscr{C}$ to $m > 2$ different sequences.

4.3.2 An efficient $(n - k, n)$-threshold construction

Let $\Gamma_{\text{-}k} \subseteq \{0,1\}^*$ denote the language of finite bit strings $t \in \{0,1\}^*$ with Hamming weight at least $|t| - k$. We now provide an efficient construction for trapdoor $\Gamma_{\text{-}k}$-mercurial commitments (colloquially: $(n-k, n)$-threshold mercurial commitments). In particular, a trapdoor $\Gamma_{\text{-}k}$-mercurial commitment binds P to a subsequence of at least $n - k$ components of a sequence from C^n. Our construction uses Kate, Zaverucha, and Goldberg's $\text{PolyCommit}_{\text{DL}}$ polynomial commitment scheme [KZG10a; §3.2] as a building block.

4.3.2.1 PolyCommit$_{\text{DL}}$ polynomial commitments

Let $p \in \mathbb{N}^+$ be a τ-bit prime and let $\mathbb{Z}_p[x]$ be the ring of polynomials over the finite field $\mathbb{Z}_p$.[25] A *polynomial commitment scheme* is a cryptographic commitment scheme with which a committer $\mathcal{A}$ can commit to a polynomial $f \in \mathbb{Z}_p[x]$ and later open the commitment either *polynomial-wise* to f or *point-wise* to a point $(i, f(i))$ on f, for an arbitrary input $i \in \mathbb{Z}_p$. Kate et al.'s PolyCommit$_{\text{DL}}$ scheme [KZG10a; §3.2] is a *bilinear pairing-based* construction (see Definition 24 below) for polynomial commitments in which the size of a commitment does not depend on the degree of the committed polynomial. Using a long-lived reference string comprising $n + 1$ elements from an order-p bilinear group $\widetilde{\mathbb{G}}$, $\mathcal{A}$ can commit to any polynomial $f \in \mathbb{Z}_p[x]$ of degree at most $n < \sqrt{p}$ using only a *single* $\widetilde{\mathbb{G}}$ element.

Definition 24. A *(symmetric) bilinear pairing* on the pair of groups $(\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}})$ of prime order p is an efficient algorithm $e\colon \widetilde{\mathbb{G}} \times \widetilde{\mathbb{G}} \to \mathbb{G}_{\text{T}}$ that satisfies the following two conditions.

1. **Bilinearity:** For all $\tilde{g}_1, \tilde{g}_2 \in \widetilde{\mathbb{G}}$ and for all $x \in \mathbb{Z}_p$, $e(\tilde{g}_1^x, \tilde{g}_2) = e(\tilde{g}_1, \tilde{g}_2^x) = e(\tilde{g}_1, \tilde{g}_2)^x$.
2. **Non-degeneracy:** For all $\tilde{g} \in \widetilde{\mathbb{G}}^*$, $e(\tilde{g}, \tilde{g}) \neq 1$.

If $e\colon \widetilde{\mathbb{G}} \times \widetilde{\mathbb{G}} \to \mathbb{G}_{\text{T}}$ is a bilinear pairing, then the pair of groups $(\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}})$ is called a *bilinear pair*.

We use a symmetric pairing for ease of exposition; however, generalizing our constructions to use asymmetric (specifically, Type III) pairings is not difficult and is substantially more efficient in practice. For an overview of the available choices of cryptographic pairings, we refer the reader to Galbraith, Paterson, and Smart [GPS08].

A degree-n PolyCommit$_{\text{DL}}$ common reference string is a tuple of the form

$$\textsc{Poly}_\tau(n) = ((\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}}, p, \tilde{g}, e), (\tilde{g}^{\alpha^j} \mid j \in [1, n])),$$

where $(\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}})$ is a bilinear pair in which $\widetilde{\mathbb{G}}$ and $\mathbb{G}_{\text{T}}$ have τ-bit prime order p, $e\colon \widetilde{\mathbb{G}} \times \widetilde{\mathbb{G}} \to \mathbb{G}_{\text{T}}$ is an efficient bilinear pairing on $(\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}})$, $\tilde{g} \in \widetilde{\mathbb{G}}^*$ is a generator of $\widetilde{\mathbb{G}}$, and $\alpha \in \mathbb{Z}_p^*$ is a random *trapdoor exponent*. On input the security parameter 1^τ, a trusted initializer generates $(\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}}, p, \tilde{g}, e) \leftarrow \widetilde{\mathcal{G}}(1^\tau)$ using a suitable *bilinear group-generating algorithm* $\widetilde{\mathcal{G}}$ (as defined

[25] We use p to denote our prime in this subsection in order to emphasize that we are *not* working in the field $\mathbb{Z}_q$ whose order matches our ubiquitous multiplicative group $\mathbb{G}$.

in Definition 25 in Appendix A) with respect to which certain computational intractability assumptions hold. (We discuss the necessary assumptions below.) It then selects the trapdoor exponent $\alpha \in_R \mathbb{Z}_p^*$, uses it to compute the power basis $(\tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^n})$, and then securely discards it immediately thereafter.[26]

To commit to a polynomial $f(x) = \sum_{j=0}^{d} a_j x^j \in \mathbb{Z}_p[x]$ of degree $d \leq n$, $\mathcal{A}$ computes $\mathscr{C} = \prod_{j=0}^{d} (\tilde{g}^{\alpha^j})^{a_j} = \tilde{g}^{f(\alpha)}$ using the appropriate values from $\text{Poly}_\tau(n)$. $\mathcal{A}$ can of course open $\mathscr{C}$ to f by simply revealing f to $\mathcal{B}$ and having $\mathcal{B}$ redo the calculation. Alternatively, $\mathcal{A}$ can open $\mathscr{C}$ to a point $(i, f(i))$ on f by appealing to the Polynomial Remainder Theorem [Gal12; Corollary 2 to Theorem 16.2] and the properties of the bilinear pairing, as outlined in Protocol 4.10.

Protocol 4.10 (Point-wise opening in PolyCommit$_{\text{DL}}$ [KZG10a; §3.2]).

Common input:	$(\mathscr{C}, i) \in \widetilde{\mathbb{G}} \times \mathbb{Z}_p \setminus \{\alpha\}$ and $((\widetilde{\mathbb{G}}, \mathbb{G}_T, p, \tilde{g}, e), (g^{\alpha^j} \mid j \in [1, n]))$
$\mathcal{A}$'s private input:	$f \in \mathbb{Z}_p[x]$ such that $\deg f = d$ and $\mathscr{C} = \tilde{g}^{f(\alpha)}$

$\mathcal{A}1$: Write $f(x) = Q(x)(x - i) + f(i)$, where $Q(x) = \sum_{j=0}^{d-1} b_j x^j$ is the polynomial quotient obtained upon dividing $f(x) - f(i)$ by $(x - i)$. $\mathcal{A}$ computes the *evaluation witness* element $w_i = \prod_{j=0}^{d-1} (\tilde{g}^{\alpha^j})^{b_j} = \tilde{g}^{Q(\alpha)}$ and sends $(i, f(i), w_i)$ to $\mathcal{B}$.

$\mathcal{B}2$: $\mathcal{B}$ outputs 1 if $e(\mathscr{C}/\tilde{g}^{f(i)}, \tilde{g}) = e(w_i, \tilde{g}^{\alpha}/\tilde{g}^{i})$, and outputs 0 otherwise.

Note that the evaluation witness w_i is itself a PolyCommit$_{\text{DL}}$ commitment whose length, like that of $\mathscr{C}$, is independent of $\deg f$. The point-wise and polynomial-wise opening procedures are each complete by inspection. Moreover, one can show that if $\mathcal{A}$ selects $f \in \mathbb{Z}_p[x]$ uniformly at random (subject only to $\deg f \leq d$), then (i) $\mathscr{C}$ is unconditionally hiding when $\mathcal{B}$ knows at most $d - 1$ evaluations of f (and the associated witnesses), (ii) $\mathscr{C}$ is computationally hiding under the DL assumption when $\mathcal{B}$ knows exactly d evaluations of f (and the associated witnesses) and that $\deg f \leq d$, and (iii) $\mathscr{C}$ is trivially non-hiding when $\mathcal{B}$ knows $d + 1$ or more

[26] We can view PolyCommit$_{\text{DL}}$ as a *trapdoor commitment scheme*: given access to the trapdoor exponent α, $\mathcal{A}$ can open a PolyCommit$_{\text{DL}}$ commitment to an *arbitrary* point $(i, y_i) \in (\mathbb{Z}_p \setminus \{\alpha\}) \times \mathbb{Z}_p$. Note that although knowledge of the trapdoor exponent affects the binding property in PolyCommit$_{\text{DL}}$, it does *not* affect the hiding property; thus, it is sometimes acceptable for the *recipient* $\mathcal{B}$ to also be the trusted initializer. Alternatively, the trusted initializer can be replaced by a secure multiparty computation between, say, all of the committers and recipients that intend to use $\text{Poly}_\tau(n)$.

evaluations of f.[27] Point-wise openings are computationally binding under a pair of DL-like assumptions — the *strong Diffie-Hellman (SDH) assumption* [BB08; §2.3] and the *polynomial Diffie-Hellman (polyDH) assumption* [KZG10a; Definition 2] — and polynomial-wise openings are computationally binding under the DL assumption. See Appendix A for definitions of the SDH and polyDH assumptions. We refer the reader to Kate et al.'s paper [KZG10a] for further details on $\text{PolyCommit}_{\text{DL}}$ commitments, including proofs of the above security claims.

Change of basis in the $\text{PolyCommit}_{\text{DL}}$ common reference string

Given a polynomial $f \in \mathbb{Z}_p[x]$ of degree n whose coefficients $(a_0, \ldots, a_n)$ are each bounded above by a small constant, say 2^{λ_0}, we can use Straus' algorithm to compute a commitment $\mathscr{C} = \tilde{g}^{f(\alpha)}$ to f using fewer than $\lambda_0 + (n+1)\lambda_0/2$ multiplications in $\widetilde{\mathbb{G}}$, on average. In our construction below, it is not the coefficients $(a_0, \ldots, a_n)$ of f that are each bounded above by 2^{λ_0}, but the evaluations $(f(0), \ldots, f(n))$ at $x = 1, \ldots, n$. It is therefore advantageous for us to perform a "change of basis" from the power basis $(\tilde{g}, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^n})$ in the $\text{PolyCommit}_{\text{DL}}$ common reference string to the appropriate *Lagrange basis*. (In fact, we keep both bases around as we still find the power basis useful for efficiently committing to low-degree polynomials.) In particular, the trusted initializer precomputes the sequence of $\text{PolyCommit}_{\text{DL}}$ commitments $(\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_n)$ in which each $\tilde{g}_j = \tilde{g}^{\ell_j(\alpha)}$ for the Lagrange coefficient

$$\ell_j(x) = \prod_{i=0, i\neq j}^{n} \frac{x - i}{j - i}. \tag{4.3}$$

The committer can then use Straus' algorithm with this basis to evaluate $\mathscr{C} = \prod_{i=0}^{n} \tilde{g}_i^{f(i)}$ using fewer than $\lambda_0 + (n+1)\lambda_0/2$ multiplications in $\widetilde{\mathbb{G}}$, on average.

[27] More precisely, solving for $f \in_{\text{R}} \mathbb{Z}_p[x]$ given (i) a $\text{PolyCommit}_{\text{DL}}$ commitment $\tilde{g}^{f(\alpha)}$ to f, (ii) a set of d distinct points on f (and the associated witnesses), and (iii) a promise that $\deg f = d$, is equivalent to solving a random DL problem instance in $\widetilde{\mathbb{G}}$ (or in $\mathbb{G}_{\text{T}}$, if $\mathcal{B}$ prefers). The correct interpretation for the above binding property is therefore: if the bilinear pairs $(\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}})$ are output by a *bilinear group-generating algorithm* $\widetilde{\mathcal{G}}$ with respect to which the DL assumption holds (see Definition 25 in Appendix A), then no PPT algorithm can solve for a randomly selected f from $\tilde{g}^{f(\alpha)}$ and a set of $\deg f$ distinct points on f, except with probability negligible in $\tau \approx \lg p$. A similar interpretation holds for the computational binding properties.

Note that this optimization *does not affect the security of the underlying PolyCommit$_{DL}$ construction*, as $\mathcal{A}$ can easily compute the commitments $(\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_n)$ in polynomial time from the power basis $(\tilde{g}, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^n})$. Indeed, we only have the trusted initializer output $(\tilde{g}_0, \ldots, \tilde{g}_n)$ as part of the public parameters for convenience.

4.3.2.2 Zero-knowledge protocols for PolyCommit$_{\text{DL}}$ commitments

To facilitate our construction, we have developed three new zero-knowledge protocols for proving (or arguing) assertions about polynomials committed to in PolyCommit$_{\text{DL}}$ commitments. Although we developed these protocols specifically for the following trapdoor $(n-k, n)$-threshold mercurial commitments, they will likely be useful in other applications, especially those related to verifiable secret sharing [CGMA85]. The new protocols include:

1. a system for honest-verifier *statistical* zero-knowledge proofs of knowledge of a witness-evaluation pair for a given commitment $\mathscr{C}$ and input $i \in \mathbb{Z}_p \setminus \{\alpha\}$ (colloquially: a system for proofs of knowledge of a point on a committed polynomial), which we denote by $\text{PK}\{(w_i, \gamma_i) : e(\mathscr{C}/\tilde{g}^{\gamma_i}, \tilde{g}) = e(w_i, \tilde{g}^{\alpha}/\tilde{g}^{i})\}$;
2. a system for honest-verifier *statistical* zero-knowledge *arguments* of knowledge of a committed polynomial, which we denote by $\text{PK}\{\, f : \mathscr{C} = \tilde{g}^{f(\alpha)} \,\}$; and
3. a system for honest-verifier *statistical* zero-knowledge *arguments* of knowledge of a committed polynomial with degree at most d, which we denote by $\text{PK}\{\, f : \mathscr{C} = \tilde{g}^{f(\alpha)} \wedge \deg f \leq d \,\}$.

The arguments of knowledge of a committed polynomial are computationally convincing under the SDH assumption; additionally arguing that the committed polynomial has degree at most d requires the polyDH assumption as well.

A full description and security analysis of each new zero-knowledge protocol is included in Appendix C. The communication cost and computation cost for the verifier in each protocol is independent both of $\deg f$ and of the maximum degree n allowed by $\textsc{Poly}_\tau(n)$. In each protocol, the prover issues an announcement that includes a PolyCommit$_{\text{DL}}$ commitment to a polynomial having approximately $\deg f$ non-zero coefficients; as such, the prover's expected cost in each protocol is roughly proportional to the degree of the polynomial under consideration. More precisely, the expected cost for the verifier in each protocol is $\text{ExpCost}_{\tilde{\mathbb{G}}}((2,\tau)) + \text{ExpCost}_{\mathbb{G}_T}((2,\tau))$

plus two pairing evaluations, while the expected cost for the prover is $\mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((\deg f, \tau)) + \mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((2,\tau)) + \mathrm{ExpCost}_{\widetilde{\mathbb{G}}}(\tau)$ plus two pairing evaluations. For communication cost, the prover sends one $\widetilde{\mathbb{G}}$ element, one $\mathbb{G}_{\mathrm{T}}$ element, and two $\mathbb{Z}_p$ elements to the verifier, and the verifier sends either one $\mathbb{Z}_p$ element (in the first protocol) or two $\mathbb{Z}_p$ elements (in the second and third protocols) to the prover.

Astute readers might notice that the *secret* trapdoor exponent α appears to the right of the colon in our proposed Camenisch-Stadler notation for the above protocols. This is indeed an abuse of notation, as α is emphatically not public; nonetheless, we feel that the proposed notation is sufficiently clear. Moreover, we note that the protocols are still zero-knowledge if the verifier happens to know the trapdoor exponent α; only soundness requires that α be kept secret from any potentially dishonest provers.

4.3.2.3 $(n-k, n)$-threshold construction

We now present our construction for trapdoor $(n-k,n)$-threshold mercurial commitments. Trapdoor $(n-k,n)$-threshold mercurial commitments bind the committer $\mathcal{A}$ to an arbitrary subsequence comprising at least $n-k$ components of a sequence $(c_1,\ldots,c_n) \in C^n$. As our ultimate goal is to construct systems for *batch* zero-knowledge arguments of partial knowledge, we take special care to maintain asymptotically low computation and (amortized)[28] communication cost in all steps. Our construction follows from the following fundamental property about polynomials [Gal12; Corollary 1 to Theorem 16.2]:

$$f(i) = 0 \text{ if and only if } x - i \mid f(x).$$

Throughout this section, we assume that the commitment domain C is an interval $C = [0, T-1]$, where $T \in \mathbb{N}^+$ can be any λ_0-bit positive integer.

Public parameters generation. Given a security parameter 1^τ and a maximum sequence length $n < \sqrt{2^\tau}/2$, the trusted initializer invokes the trusted initializer for $\mathrm{PolyCommit}_{\mathrm{DL}}$ to obtain a degree-n common reference string $\textsc{Poly}_\tau(n) = ((\widetilde{\mathbb{G}}, \mathbb{G}_{\mathrm{T}}, p, \tilde{g}, e), (\tilde{g}^{\alpha^j} \mid j \in [1,n]))$. It then precomputes

[28] In particular, the public parameters in our construction comprises $\mathrm{O}(n)$ group elements; however, the commitments $\mathscr{C}$ and decommitments π both have fixed lengths and have a relatively low cost to compute.

1. a $\text{PolyCommit}_{\text{DL}}$ commitment $\mathcal{Z} = \tilde{g}^{z(\alpha)}$ to $z(x) = \prod_{i=1}^{n}(x - i)$, and
2. the Lagrange basis $(\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_n) \in (\widetilde{\mathbb{G}})^{n+1}$ in which each $\tilde{g}_j = \tilde{g}^{\ell_j(\alpha)}$ is a commitment to the Lagrange coefficient $\ell_j(x)$, as described in Equation (4.3) above,

and then it outputs the public parameters

$$\text{PUB}_{\Gamma_{-k}}(C, 1^\tau, n) = ((\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}}, p, \tilde{g}, e), (\tilde{g}^\alpha, \ldots, \tilde{g}^{\alpha^n}), (\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_n), \mathcal{Z}).$$

Note that the threshold $n - k$ is not used by $\text{KeyGen}_{\Gamma_{-k}}$; indeed, the same public parameters work for trapdoor Γ_{-k}-mercurial commitments for any $k \in [0, n-1]$.

Committing. Let $\overline{H} \subseteq [1,n]$ such that $|\overline{H}| \geq n - k$ and let $H = [1,n] \setminus \overline{H}$. A $(n-k, n)$-threshold mercurial commitment to the subsequence $(c_i)_{i \in \overline{H}}$ has three components.

The first component is a $\text{PolyCommit}_{\text{DL}}$ commitment $\mathcal{F}$ to the degree-n polynomial $f \in \mathbb{Z}_p[x]$ defined by

$$f(i) = \begin{cases} \gamma_1 & i = 0 \\ c_i & i \in \overline{H} \\ 0 & i \in H, \end{cases}$$

where $\gamma_1 \in_{\text{R}} \mathbb{Z}_p$.

Observation 4.11. The expected cost for $\mathcal{A}$ to compute $\mathcal{F} = \tilde{g}^{f(\alpha)}$ is $\text{ExpCost}_{\widetilde{\mathbb{G}}}((1,\tau),(n-k,\lambda_0))$ using the Lagrange basis and the fact that $f(i) \in C$ for each $i \in \overline{H}$ and $f(i) = 0$ for each $i \in H$.

The second component is a pair of $\text{PolyCommit}_{\text{DL}}$ commitments $\mathcal{S}$ and $\hat{\mathcal{S}}$ to the polynomials $s, \bar{s} \in \mathbb{Z}_p[x]$ such that

$$s(x) = \gamma_2 \prod_{i \in H} (x - i)$$

and

$$\bar{s}(x) = \gamma_2^{-1} \prod_{i \in \overline{H}} (x - i)$$

for $\gamma_2 \in_R \mathbb{Z}_p^*$. (If $\overline{H} = [1,n]$ and $H = \{\}$, then define $s(x) = \gamma_2$.) Intuitively, the pair of commitments $(\mathcal{S}, \hat{\mathcal{S}})$ binds $\mathcal{A}$ to a subset $\overline{H} \subseteq [1,n]$, and $\mathcal{F}$ binds $\mathcal{A}$ to a particular choice of $c_i \in C$ for each $i \in \overline{H}$.

Observation 4.12. The expected cost for $\mathcal{A}$ to compute $\mathcal{S} = \tilde{g}^{s(\alpha)}$ is $\mathrm{ExpCost}_{\tilde{\mathbb{G}}}((k+1,\tau))$ using the power basis and the fact that $\deg s \leq k$. The expected cost for $\mathcal{A}$ to compute $\hat{\mathcal{S}} = \tilde{g}^{\bar{s}(\alpha)}$ is $\mathrm{ExpCost}_{\tilde{\mathbb{G}}}((k+1,\tau))$ using the Lagrange basis and the fact that $\bar{s}(i) = 0$ for each $i \in \overline{H}$.

The (Γ_{-k}, n)-mercurial commitment is the tuple $\mathscr{C} = (\mathcal{F}, \mathcal{S}, \hat{\mathcal{S}}, k)$ and the committer's auxiliary information is the pair of polynomials $\mathfrak{Y} = (f(x), s(x))$. Note that $k \in \mathbb{N}^+$ indicates the threshold $n - k$ for the given commitment.

Opening a commitment. To open the commitment $\mathscr{C} = (\mathcal{F}, \mathcal{S}, \hat{\mathcal{S}}, k)$ to $(c_1, \ldots, c_n) \in C^n$, $\mathcal{A}$ computes a pair of $\mathrm{PolyCommit}_{\mathrm{DL}}$ commitments $\hat{\mathcal{F}}$ and $\mathcal{H}$ to the polynomials $\bar{f}, h \in \mathbb{Z}_p[x]$ defined by

$$\bar{f}(i) = \begin{cases} -\gamma_1 & i = 0 \\ 0 & i \in \overline{H} \\ c_i & i \in H \end{cases}$$

and

$$h(x) = \frac{\bar{f}(x)}{\bar{s}(x)}.$$

Observe that $\bar{s}(x) \mid \bar{f}(x)$ by construction, which guarantees that $h(x) = \bar{f}(x)/\bar{s}(x)$ is indeed a polynomial (whose degree is at most k).

Observation 4.13. The expected cost for $\mathcal{A}$ to compute $\hat{\mathcal{F}} = \tilde{g}^{\bar{f}(\alpha)}$ is $\mathrm{ExpCost}_{\tilde{\mathbb{G}}}((1,\tau),(k,\lambda_0))$ using the Lagrange basis and the facts that $\bar{f}(i) \in C$ for each $i \in H$ and that $\bar{f}(i) = 0$ for each $i \in \overline{H}$. Likewise, the expected cost for $\mathcal{A}$ to compute $\mathcal{H} = \tilde{g}^{h(\alpha)}$ is $\mathrm{ExpCost}_{\tilde{\mathbb{G}}}((k+1,\tau))$ using the power basis and the fact that $\deg h \leq k$.

To complete the opening, $\mathcal{A}$ engages the recipient in the zero-knowledge statistical argument of knowledge denoted by

$$\mathrm{PK}\{(s(x), h(x)) : \mathcal{S} = \tilde{g}^{s(\alpha)} \wedge \mathcal{H} = \tilde{g}^{h(\alpha)} \wedge \deg s \leq k\}.$$

Observation 4.14. Since both $\deg s \leq k$ and $\deg h \leq k$ by construction, the expected cost for $\mathcal{A}$ in the above protocol depends on k but it does not depend on n. The communication cost and expected verification cost are each independent of both k and n. (See Section 4.3.2.2 for the exact costs.)

The decommitment is $\pi = (\hat{\mathcal{F}}, \mathcal{H})$ and the above argument of knowledge of $(s(x), h(x))$ such that $\mathcal{S} = \tilde{g}^{s(\alpha)}$ and $\mathcal{H} = \tilde{g}^{h(\alpha)}$ with $\deg s \leq k$.

Verifying an opening. To verify the opening of $\mathscr{C} = (\mathcal{F}, \mathcal{S}, \hat{\mathcal{S}}, k)$ to a sequence $(c_1, \ldots, c_n)$ given the decommitment $\pi = (\hat{\mathcal{F}}, \mathcal{H})$ and given that $\mathcal{B}$ accepted in the argument of knowledge wherein $\mathcal{A}$ proved to know $(s(x), h(x))$ such that $\mathcal{S} = \tilde{g}^{s(\alpha)}$ and $\mathcal{H} = \tilde{g}^{h(\alpha)}$ with $\deg s \leq k$, $\mathcal{B}$ interpolates the degree-n polynomial $F \in \mathbb{Z}_p[x]$ defined by

$$F(i) = \begin{cases} 0 & i = 0 \\ c_i & i \in [1, n], \end{cases}$$

and then it outputs 1 if

$$e(\mathcal{S}, \hat{\mathcal{S}}) = e(\mathcal{Z}, \tilde{g}), \tag{4.4}$$

if

$$\tilde{g}^{F(\alpha)} = \mathcal{F} \cdot \hat{\mathcal{F}}, \tag{4.5}$$

and if

$$e(\hat{\mathcal{S}}, \mathcal{H}) = e(\hat{\mathcal{F}}, \tilde{g}); \tag{4.6}$$

otherwise, it outputs 0.

Observation 4.15. The expected cost for $\mathcal{B}$ to compute $\tilde{g}^{F(\alpha)}$ is $\text{ExpCost}_{\tilde{\mathbb{G}}}((n, \lambda_0))$ using the Lagrange basis and the fact that $f(0) = 0$ and $f(i) \in C$ for each $i = 1, \ldots, n$. In addition, $\mathcal{B}$ must verify the above zero-knowledge argument of knowledge and evaluate the four bilinear pairings; these latter two operations both have cost independent of both n and k.

4.3.2.4 Security analysis

We now analyze the security of our construction.

Theorem 4.16. *The construction described in Section 4.3.2.3 is a secure trapdoor $\Gamma_{\cdot k}$-mercurial commitment scheme under the SDH and polyDH assumptions.*

Proof. **Complete:** Completeness follows by inspection of the algorithms.

(Unconditional) $\Gamma_{\cdot k}$-mercurial hiding: We prove that $\mathcal{B}$ wins the $\Gamma_{\cdot k}$-mercurial indistinguishability game (as defined in Definition 23) with probability at most $1/2 + \varepsilon(\tau)$ for some negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ by proving that the distribution from which $\mathcal{A}$ draws $(\mathscr{C}, \pi)$ is statistically independent of the outcome $b \in \{0,1\}$ of $\mathcal{A}$'s random coin flip. It then follows that the commitment-decommitment pair $(\mathscr{C}, \pi)$ reveals nothing to $\mathcal{B}$ about the outcome b, and so any advantage that $\mathcal{B}$ has must result from the negligible knowledge it gains by engaging in the *statistical* zero-knowledge argument of knowledge with $\mathcal{A}$.

Suppose that $\mathcal{B}$ challenges $\mathcal{A}$ with $(c_1, \ldots, c_n) \in C^n$ and a pair of subsets $\overline{H}_0, \overline{H}_1 \subseteq [1,n]$ satisfying $|\overline{H}_0| \geq n-k$ and $|\overline{H}_1| \geq n-k$. Suppose further that $\mathcal{A}$ responds with $\mathscr{C} = (\mathcal{F}, \mathcal{S}, \hat{\mathcal{S}}, k)$ and $\pi = (\hat{\mathcal{F}}, \mathcal{H})$ such that $\mathrm{Ver}(\mathscr{C}, \pi, c_1, \ldots, c_n) = 1$. Let $F \in \mathbb{Z}_p[x]$ denote the degree-n polynomial such that

$$F(x) = \begin{cases} 0 & i = 0 \\ c_i & i \in [1,n], \end{cases}$$

which is known both to $\mathcal{A}$ and to $\mathcal{B}$.

It is easy to verify that the distribution of $(\mathcal{F}, \mathcal{S})$ does not depend on b; in particular, for either outcome $b \in \{0,1\}$, there exists one and only one corresponding choice for $(\gamma_1, \gamma_2) \in \mathbb{Z}_p \times \mathbb{Z}_p^*$ that is consistent both with $(F(x), \overline{H}_b)$ and with $(\mathcal{F}, \mathcal{S})$. But $\mathcal{A}$ chooses $\gamma_1 \in_{\mathrm{R}} \mathbb{Z}_p$ and $\gamma_2 \in_{\mathrm{R}} \mathbb{Z}_p^*$ independently at random, so the pair $(\mathcal{F}, \mathcal{S})$ is equally likely to arise for $b = 0$ and for $b = 1$. Moreover, the pair $(\mathcal{F}, F(x))$ completely determines $\hat{\mathcal{F}}$, the pair $(\mathcal{S}, \mathcal{Z})$ completely determines $\hat{\mathcal{S}}$, and the pair $(\hat{\mathcal{S}}, \hat{\mathcal{F}})$ completely determines $\mathcal{H}$; hence, it follows that the distribution from which $\mathcal{A}$ draws $(\mathscr{C}, \pi)$ is independent of b, as desired.

(Computational) $\Gamma_{\cdot k}$-mercurial binding: Suppose $\mathrm{PUB}_{\Gamma_{\cdot k}}(C, 1^\tau, n) \leftarrow \mathrm{KeyGen}_{\Gamma_{\cdot k}}(C, 1^\tau, n)$ is given and suppose that $(\mathscr{C}, (\pi^{(0)}, c_1^{(0)}, \ldots, c_n^{(0)}), (\pi^{(1)}, c_1^{(1)}, \ldots, c_n^{(1)})) \leftarrow \mathcal{A}(\mathrm{PUB}_{\Gamma_{\cdot k}}(C, 1^\tau, n))$ such that $\mathrm{Ver}(\mathscr{C}, \pi^{(0)}, c_1^{(0)}, \ldots, c_n^{(0)}) = 1$ and $\mathrm{Ver}(\mathscr{C}, \pi^{(1)}, c_1^{(1)}, \ldots, c_n^{(1)}) = 1$. Write $\mathscr{C} = (\mathcal{F}, \mathcal{S}, \hat{\mathcal{S}}, k)$ and, for

each $b \in \{0,1\}$, write $\pi^{(b)} = (\hat{\mathcal{F}}^{(b)}, \mathcal{H}^{(b)})$ and let $F^{(b)} \in \mathbb{Z}_p[x]$ denote the degree-n polynomials defined by

$$F^{(b)}(x) = \begin{cases} 0 & i = 0 \\ c_i^{(b)} & i \in [1,n]. \end{cases}$$

Note that $F^{(b)}(x)$ is known both to $\mathcal{A}$ and to $\mathcal{B}$. Suppose also that $\mathcal{A}$ has successfully argued knowledge of $(s(x), h^{(b)}(x))$ such that $\mathcal{S} = \tilde{g}^{s(\alpha)}$ and $\mathcal{H}^{(b)} = \tilde{g}^{h^{(b)}(\alpha)}$ with $\deg s \leq k$. Finally, we note that both $\mathcal{A}$ and $\mathcal{B}$ are privy to the polynomial $z(x) = \prod_{i=1}^{n}(x - i)$ such that $\mathcal{Z} = \tilde{g}^{z(\alpha)}$.

We will prove that there exists a set $\overline{H} \subseteq [1,n]$ such that $|\overline{H}| \geq n - k$ and such that $x - i \mid F^{(0)}(x) - F^{(1)}(x)$ for each $i \in \overline{H}$, which implies that $F^{(0)}(i) - F^{(1)}(i) = 0$ or, equivalently, that $F^{(0)}(i) = F^{(1)}(i)$, for each $i \in \overline{H}$, as desired.

To begin, define the pair of rational functions $(\dot{s}(x), \ddot{s}(x))$ as

$$\dot{s}(x) = \frac{z(x)}{s(x)},$$

and

$$\ddot{s}(x) = \frac{(F^{(0)}(x) - F^{(1)}(x))}{(h^{(0)}(x) - h^{(1)}(x))}.$$

Note that $\mathcal{A}$ knows the pair $(\dot{s}(x), \ddot{s}(x))$ by our assumptions above.

By Equation (4.4), it follows that $\hat{\mathcal{S}} = \tilde{g}^{\dot{s}(\alpha)}$.

Moreover, from Equation (4.5), it follows both that $\mathcal{F} \cdot \hat{\mathcal{F}}^{(0)} = \tilde{g}^{F^{(0)}(\alpha)}$ and that $\mathcal{F} \cdot \hat{\mathcal{F}}^{(1)} = \tilde{g}^{F^{(1)}(\alpha)}$ so that $\hat{\mathcal{F}}^{(0)}/\hat{\mathcal{F}}^{(1)} = \tilde{g}^{F^{(0)}(\alpha) - F^{(1)}(\alpha)}$. Similarly, from Equation (4.6), it follows that $e(\tilde{g}^{h^{(0)}(\alpha)}, \hat{\mathcal{S}}) = e(\hat{\mathcal{F}}^{(0)}, \tilde{g})$ and that $e(\tilde{g}^{h^{(1)}(\alpha)}, \hat{\mathcal{S}}) = e(\hat{\mathcal{F}}^{(1)}, \tilde{g})$ so that $e(\tilde{g}^{h^{(0)}(\alpha) - h^{(1)}(\alpha)}, \hat{\mathcal{S}}) = e(\hat{\mathcal{F}}^{(0)}/\hat{\mathcal{F}}^{(1)}, \tilde{g})$.

Substituting $\hat{\mathcal{F}}^{(0)}/\hat{\mathcal{F}}^{(1)} = \tilde{g}^{F^{(0)}(\alpha) - F^{(1)}(\alpha)}$ in the latter expression, we find that

$$e(\tilde{g}^{h^{(0)}(\alpha) - h^{(1)}(\alpha)}, \hat{\mathcal{S}}) = e(\tilde{g}^{F^{(0)}(\alpha) - F^{(1)}(\alpha)}, \tilde{g});$$

hence, $\hat{\mathcal{S}} = \tilde{g}^{\ddot{s}(\alpha)}$, and, in particular, we have that $\tilde{g}^{\dot{s}(\alpha)} = \tilde{g}^{\ddot{s}(\alpha)}$.

Now, define the polynomial $E(x) = z(x)\big(h^{(0)}(x) - h^{(1)}(x)\big) - s(x)\big(F^{(0)}(x) - F^{(1)}(x)\big)$. Since

$$\frac{z(\alpha)}{s(\alpha)} = \frac{F^{(0)}(\alpha) - F^{(1)}(\alpha)}{h^{(0)}(\alpha) - h^{(1)}(\alpha)},$$

it follows that $E(\alpha) = 0$. There are two cases to consider.

Case 1 ($E(x) \neq 0$): Here $E(x)$ is a polynomial known to $\mathcal{A}$ and having at most $2n$ roots (counting multiplicity), one of which is the trapdoor exponent α. As $\mathcal{A}$ can efficiently factor $E(x)$, the SDH assumption implies that Case 1 occurs with probability negligible in τ.

Case 2 ($E(x) = 0$): Here $z(x)\big(h^{(0)}(x) - h^{(1)}(x)\big) = s(x)\big(F^{(0)}(x) - F^{(1)}(x)\big)$. Note that both sides of this expression are polynomials known to $\mathcal{A}$. Moreover, note that the $z(x)$ factor on the left-hand side ensures that both polynomials evaluate to 0 on input any $i \in [1,n]$. As $\mathcal{A}$ has successfully argued that $\deg s \leq k$, it follows (with probability overwhelming in τ, under the polyDH assumption) that there is a set $\overline{H} \subseteq [1,n]$ with $|\overline{H}| \geq n-k$ such that $F^{(0)}(i) - F^{(1)}(i) = 0$ for each $i \in \overline{H}$. ∎

Note that the set $\overline{H}$ derived in Case 2 depends only on the polynomials $s(x)$ and $\bar{s}(x)$ to which $\mathcal{S}$ and $\hat{\mathcal{S}}$ commit; in particular, for each $i = 1, \ldots, n$, we must have that either $F^{(0)}(i) - F^{(1)}(i) = 0$ or $s(i) = 0$, always for *the same degree-k polynomial* $s(x)$. Thus, we find not only that the polynomial $F^{(0)}(x) - F^{(1)}(x)$ has at least $n-k$ roots in $[1,n]$, but that it always has *same set of roots* (or some sufficiently large subset thereof) for all feasible openings of $\mathscr{C} = (\mathcal{F}, \mathcal{S}, \hat{\mathcal{S}}, k)$.

The opening and verification algorithms in our construction actually constitute an *interactive protocol*. One can make the zero-knowledge arguments of knowledge non-interactive using the Fiat-Shamir transform, in which case the opening procedure would be non-interactive and the computational Γ_{-k}-mercurial binding property would hold under the SDH and polyDH assumptions in the random oracle model.

4.3.2.5 Cost analysis

Summing the expected computation costs indicated in Observations 4.11–4.15 and Section 4.3.2.2, we find that the aggregate cost for $\mathcal{A}$ to produce and open a (Γ_{-k}, n)-mercurial commitment is less than

$$\begin{aligned}\mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((1,\tau),(n-k,\lambda_0)) + 4\,\mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((k+1,\tau)) \\ + \mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((2,\tau)) + \mathrm{ExpCost}_{\widetilde{\mathbb{G}}}(\tau) \\ + \mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((1,\tau),(k,\lambda_0)) < 4k\tau/\lg\tau + n\lambda_0/\lg n\end{aligned}$$

multiplications in $\widetilde{\mathbb{G}}$, plus two pairing evaluations. Note that this cost is in $\mathrm{O}(k\tau/\lg\tau + n\lambda_0/\lg n)$. Likewise, the cost for $\mathcal{B}$ to verify such an opening is

$$\mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((n,\lambda_0)) + \mathrm{ExpCost}_{\widetilde{\mathbb{G}}}((2,\tau)) < \tau + \tau/\lg\tau + n\lambda_0/\lg n$$

multiplications in $\widetilde{\mathbb{G}}$, plus $\mathrm{ExpCost}_{\mathbb{G}_{\mathrm{T}}}((2,\tau))$ multiplications in $\mathbb{G}_{\mathrm{T}}$ and six pairing evaluations. Hence, for any fixed $k \in \mathbb{N}^+$, the overall computation cost of the above construction is in $\mathrm{O}(n\lambda_0)$ both for $\mathcal{A}$ and for $\mathcal{B}$.

The commitments have constant size, comprising just three $\widetilde{\mathbb{G}}$ elements. An opening comprises three additional $\widetilde{\mathbb{G}}$ elements (two in the decommitment π, and one in the zero-knowledge argument of knowledge), one $\mathbb{G}_{\mathrm{T}}$ element, and two $\mathbb{Z}_p$ elements (beyond the actual values in the opening). In addition, V must challenge P with two $\mathbb{Z}_p$ elements in the zero-knowledge argument of knowledge. The amortized communication overhead is therefore independent of both n and k.

Observation 4.17. Fix a constant $k \in \mathbb{N}^+$. The aggregate computation cost for $\mathcal{A}$ and for $\mathcal{B}$ in the above construction for trapdoor Γ_{-k}-mercurial commitments are each asymptotically lower than those of the naive prover $\hat{\mathrm{P}}$ and the naive verifier $\hat{\mathrm{V}}$ in an n-fold parallelization of Schnorr's protocol. Moreover, the bidirectional communication cost is constant; hence, we can employ the above construction in a system for batch honest-verifier zero-knowledge arguments of Γ_k-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$.

We also point out that $\text{PUB}_{\Gamma_{\text{-}k}}(C, 1^\tau, N) \leftarrow \text{KeyGen}_{\Gamma_{\text{-}k}}(C, 1^\tau, N)$ can be used to commit to sequences with any length $n \in [1, N]$ by setting $c_i = 0$ for each $i > n$. (In this case, the actual sequence length n must be disclosed alongside k in the commitment $\mathscr{C}$.) It is therefore possible to generate a single public parameters suitable for creating $(\Gamma_{\text{-}k}, n)$-mercurial commitments for any length n up to some suitably large bound N.

4.4 Batch Schnorr proofs of partial knowledge

Given our construction for trapdoor $\Gamma_{\text{-}1}$-mercurial commitments, we can easily repair (and simplify) Peng and Bao's RME-based disjunctive Chaum-Pedersen protocol. As before, the common input to (P, V) includes $(\mathbb{G}, q, g, h) \leftarrow \mathcal{G}(1^\tau; 2)$ and $(g_1, h_1, \ldots, g_n, h_n) \in (\mathbb{G})^{2n}$. In addition, the repaired protocol also requires (long-lived) public parameters $\text{PUB}_{\Gamma_{\text{-}1}}(C, 1^\tau, n) \leftarrow \text{KeyGen}_{\Gamma_{\text{-}1}}(C, 1^\tau, n)$ for $(n-1, n)$-threshold mercurial commitments over the challenge domain $C = [0, 2^{\lambda_0} - 1]$.

We note that, as a 2-extractable common-base parallelization of Chaum and Pedersen's protocol, there is no need to have V send random exponents $t = (t_1, \ldots, t_n)$ to P in an opening move, despite Peng and Bao's having V do so in their protocol (see Figure 4.2 on Page 97). Suppose that $(g_i, h_i, x)_{i=1}^n$ is Γ_1-correct and let $j \in [1, n]$ be the index for which P is proving that $g_j = g^x$ and $h_j = h^x$. In the opening move, $\text{P}(x, j)$ selects $r \in_{\text{R}} \mathbb{Z}_q^*$ and a sub-challenge $c_i \in_{\text{R}} [0, 2^{\lambda_0} - 1]$ for each $i \in [1, n] \setminus \{j\}$, and then it uses these values to compute the announcement $A = (A_0, A_1, A_2)$, in which

$$A_1 = g^r \Big(\prod_{i \in [1,n] \setminus \{j\}} g_i^{c_i}\Big),$$

$$A_2 = h^r \Big(\prod_{i \in [1,n] \setminus \{j\}} h_i^{c_i}\Big),$$

and

$$(A_0, \gamma) \leftarrow \text{Com}\big((c_i)_{i \in [1,n] \setminus \{j\}}\big).$$

Upon receiving the announcement, V challenges P with $c \in_{\mathrm{R}} [0, 2^{\lambda_0} - 1]$, from which P computes the missing sub-challenge $c_j = c - \sum_{i \in [1,n] \setminus \{j\}} c_i \bmod 2^{\lambda_0}$, the response $u = r - c_j x$, and the opening $\pi \leftarrow \mathrm{Open}(A_0, \gamma, c_1, \ldots, c_n)$ of A_0 to $(c_1, \ldots, c_n)$. Finally, P sends $v = (u, \pi, c_1, \ldots, c_n)$ to V. To verify the proof, V checks if $\mathrm{Ver}(A_0, \pi, c_1, \ldots, c_n) = 1$, if $c = \sum_{i=1}^{n} c_i \bmod 2^{\lambda_0}$, and if $A_1 = g^u (\prod_{i=1}^{n} g_i^{c_i})$ and $A_2 = h^u (\prod_{i=1}^{n} h_i^{c_i})$ both hold.

The (Γ_{-1}, n)-mercurial commitment A_0 in the announcement binds P to *no fewer* than $n - 1$ of the sub-challenges (with a probability overwhelming in τ, under the SDH and polyDH assumptions), while having the sub-challenges be shares in an additive secret sharing scheme ensures that P^* can choose *no more* than $n-1$ of the sub-challenges before it receives c (with a probability overwhelming in λ_0); thus, the only way for a PPT prover P^* to satisfy both $\mathrm{Ver}(A_0, \pi, c_1, \ldots, c_n) = 1$ and $c = \sum_{i=1}^{n} c_i \bmod 2^{\lambda_0}$ with a probability that is not negligible in $\lambda_0 \in \mathrm{o}(\tau)$ is to choose (and commit to) *exactly* $n - 1$ of the sub-challenges in the opening move. This leaves P^* with zero degrees of freedom to compute its response, thereby thwarting algebraic attacks of the sort presented in Section 4.2.1.

Theorem 4.18. *The repaired Peng-Bao protocol, just described, is a system for batch honest-verifier statistical zero-knowledge arguments of Γ_1-partial knowledge and equality among DLs induced by $\mathcal{G}_{\mathrm{DL}}(1^*)$ (colloquially: a system for batch arguments of knowledge and equality of 1-out-of-n DLs). It is c-simulatable and 2-extractable and, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta = 2^{-\lambda_0}$ and soundness error function $\lambda(\tau) = \max\{1/q, 2^{-\lambda_0}\}$. Arguments in the above protocol are computationally convincing under the SDH and polyDH assumptions for the bilinear group-generating algorithm employed by $\mathrm{KeyGen}_{\Gamma_{-1}}$.*

The proof of Theorem 4.18 is (essentially) a special case of the proof of Theorem 4.26, which we prove in Section 4.4.2 below. The next section considers a batch (k, n)-threshold variant of Schnorr's protocol along the same lines as the repaired Peng-Bao protocol; generalizing that construction to a batch (k, n)-threshold variant of Chaum and Pedersen's protocol is trivial.

4.4.1 A (k, n)-threshold batch Schnorr protocol

By replacing trapdoor Γ_{-1}-mercurial commitments and additive secret sharing with trapdoor Γ_{-k}-mercurial commitments and Shamir's $(n-k+1, n)$-threshold secret sharing, we can generalize the repaired Peng-Bao protocol to a system for honest-verifier statistical zero-knowledge arguments

of Γ_k-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. As explained below, it is necessary in this case to reintroduce the short exponents $t = (t_1, \ldots, t_n)$. Protocol 4.19 gives the full construction applied to Schnorr's protocol. We use $C = \mathbb{Z}_\rho$ for the challenge domain, where ρ is an arbitrary λ_0-bit prime so that $\mathbb{Z}_\rho[x]$ is a ring of polynomials.

Protocol 4.19 (RME-based common-base (k, n)-threshold Schnorr protocol).

Common input: $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$, $(h_1, \ldots, h_n) \in (\mathbb{G})^n$, and $\text{PUB}_{\Gamma_k}(\mathbb{Z}_\rho, 1^\tau, n)$
P's private input: $H \subseteq_k [1, n]$ and $(x_j)_{j=1}^k$ with $h_{H_{(j)}} = g^{x_j}$ for each $j = 1, \ldots, k$

P1: Let $\overline{H} = [1, n] \setminus H$. P chooses a sub-challenge $c_i \in_{\text{R}} \mathbb{Z}_\rho$ for each $i \in \overline{H}$, and then it computes $(A_0, \gamma) \leftarrow \text{Com}((c_i)_{i \in \overline{H}})$. P announces A_0 to V.

V2: V chooses a random exponent $t_i \in_{\text{R}} [0, 2^{\lambda_0} - 1]$ for each $i = 1, \ldots, n$, and then it sends $t = (t_1, \ldots, t_n)$ to P.

P3: P computes $a_i = t_i c_i \bmod \rho$ for each $i \in \overline{H}$, and then it chooses $r \in_{\text{R}} \mathbb{Z}_q^*$ and computes $A_1 = g^r (\prod_{i \in \overline{H}} h_i^{a_i})$. P sends A_1 to V.

V4: V issues a challenge $c \in_{\text{R}} \mathbb{Z}_\rho$ to P.

P5: P uses polynomial interpolation to compute the degree-$(n - k)$ polynomial $f \in \mathbb{Z}_\rho[x]$ passing through $(0, c)$ and through each point in $\{(i, c_i)\}_{i \in \overline{H}}$, and then it computes $a_{H_{(j)}} = t_{H_{(j)}} f(H_{(j)}) \bmod \rho$ for each $j = 1, \ldots, k$, the response $u = r - \sum_{j=1}^k a_{H_{(j)}} x_j$, and the opening $\pi \leftarrow \text{Open}(A_0, \gamma, f(1), \ldots, f(n))$ of A_0. P sends $v = (u, \pi, f)$ to V.

V6: V computes $a_i = t_i f(i) \bmod \rho$ for each $i = 1, \ldots, n$, and then it accepts if $\text{Ver}(A_0, \pi, f(1), \ldots, f(n)) = 1$, if $\deg f \leq n - k$ with $c = f(0)$, and if $A_1 = g^u (\prod_{i=1}^n h_i^{a_i})$; otherwise, it rejects.

Theorem 4.20. *The RME-based common-base Schnorr protocol described in Protocol 4.19 is a system for batch honest-verifier statistical zero-knowledge arguments of Γ_k-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$. It is c-simulatable and, for a fixed soundness parameter $\lambda_0 \in \mathbb{N}^+$, it has absolute soundness error $\delta \leq 2^{-\lambda_0}$. Its arguments are computationally convincing under the SDH and polyDH assumptions for the bilinear group-generating algorithm employed by KeyGen_{Γ_k}.*

We omit a detailed proof of Theorem 4.20 because it is a special case of Theorem 4.26 below. To see why it is necessary to reintroduce the short exponents $t = (t_1, \ldots, t_n)$ when $k > 1$, consider the universal knowledge extractor E_{P^*} for Protocol 4.19. When $k = 1$, such an extractor

would issue a random challenge $c^{(1)} \in_R \mathbb{Z}_\rho$ to obtain the response $v^{(1)} = (u^{(1)}, f^{(1)}, \pi^{(1)})$ from P^*, and then it would rewind P^* and issue a second challenge $c^{(2)} \in_R \mathbb{Z}_\rho \setminus \{c^{(1)}\}$ to obtain a second response $v^{(2)} = (u^{(2)}, f^{(2)}, \pi^{(2)})$. The $(n-1,n)$-threshold mercurial commitment A_0 that P^* outputs in the opening move binds it to a fixed subsequence $(c_i)_{i \in \overline{H}}$ of sub-challenges, where $\overline{H} = [1,n] \setminus \{j\}$ for some $j \in [1,n]$; hence, it follows that $u^{(1)} - u^{(2)} = (f^{(1)}(j) - f^{(2)}(j)) \log_g h_j$ is a linear equation in $k = 1$ unknown and E_{P^*} can easily solve for the desired witness $\log_g h_j = (u^{(1)} - u^{(2)})/(f^{(1)}(j) - f^{(2)}(j))$.

When $k > 1$, the corresponding expression $u^{(1)} - u^{(2)} = \sum_{j=1}^{|H|} (f^{(1)}(H_{(j)}) - f^{(2)}(H_{(j)})) \log_g h_{H_{(j)}}$ is a linear equation in $k > 1$ unknowns and, consequently, E_{P^*} requires a set of at least k linearly independent such equations in order to solve for the sequence of exponents $(x_j)_{j=1}^{|H|}$ in which each $x_j = \log_g h_{H_{(j)}}$. Moreover, since both $\deg f^{(1)} = n - k$ and $\deg f^{(2)} = n - k$, and since A_0 binds P^* to $f^{(1)}(i) = f^{(2)}(i)$ for each $i \in \overline{H}$, it follows that $f^{(1)}(x) - f^{(2)}(x) = \zeta \prod_{i \in \overline{H}} (x - i)$ for some scalar $\zeta \in \mathbb{Z}_\rho^*$. Therefore, if E_{P^*} merely rewinds P^* and issues a third challenge $c^{(3)} \in \mathbb{Z}_\rho$, the resulting expression $u^{(1)} - u^{(3)}$ is *always a scalar multiple of* $u^{(1)} - u^{(2)}$. In particular, the system of equations that such an E_{P^*} obtains always has rank 1 and, consequently, the extractor will always fail. The short exponents $t = (t_1, \ldots, t_n)$ provide a means for E_{P^*} to extract a set of *random* linear equations in the same k unknowns, thus enabling it to extract the desired exponents. The full extractor construction is described in the proof of Theorem 4.26.

4.4.2 Monotone proofs of partial knowledge

The construction for batch proofs of (k,n)-threshold knowledge described in Protocol 4.19 generalizes to a construction for batch proofs of Γ-partial knowledge for any NP-language $\Gamma \subseteq \{0,1\}^*$ inducing access structures described by a family of polynomial-sized monotone Boolean formulas. The only things that must change in Protocol 4.19 are (i) the access structure with respect to which each sub-challenge must be a secret share, and (ii) the binding property of the trapdoor $\mathcal{L}$-mercurial commitment A_0 in P's announcement. The general protocol is described in Protocol 4.21.

Protocol 4.21 (RME-based common-base Γ-partial Schnorr protocol).

Common input: $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$, $(h_1, \ldots, h_n) \in (\mathbb{G})^n$, and $\text{PUB}_{\mathscr{L}}(C(\tau), 1^\tau, n)$

P's private input: $H \subseteq [1,n]$ and $(x_j)_{j \in H}$ with $H \in \Delta_n$ and $h_j = g^{x_j}$ for each $j \in H$

P1: Let $\overline{H} = [1,n] \setminus H$. P chooses a sub-challenge $c_i \in_{\text{R}} C(\tau)$ for each $i \in \overline{H}$, and then it computes $(A_0, \gamma) \leftarrow \text{Com}((c_i)_{i \in \overline{H}})$ and sends it to V.

V2: V chooses $t_i \in_{\text{R}} \mathbb{Z}_\rho$ for each $i = 1, \ldots, n$, and then it sends $t = (t_1, \ldots, t_n)$ to V.

P3: P computes $a_i = t_i c_i \bmod \rho$ for each $i \in \overline{H}$, and then it chooses $r \in_{\text{R}} \mathbb{Z}_q^*$ and computes $A_1 = g^r (\prod_{i \in \overline{H}} h_i^{a_i})$. P sends A_1 to V.

V2: V issues a challenge $c \in_{\text{R}} C(\tau)$ to P.

P3: P computes $(c_1, \ldots, c_n) \leftarrow \mathsf{SS}(c, (c_i)_{i \in \overline{H}})$, and then it computes $a_j = t_j c_j \bmod \rho$ for each $j \in H$, the response $u = r - \sum_{j \in H} a_j x_j$, and the opening $\pi \leftarrow \text{Open}(A_0, \gamma, c_1, \ldots, c_n)$ of A_0 to $(c_1, \ldots, c_n)$. P chooses $\tilde{H} \in_{\text{R}} (\Delta_n^*)^-$ with $|\tilde{H}| \leq |\tilde{H}'|$ for each $\tilde{H}' \in (\Delta_n^*)^-$, and it sends $v = (u, \pi, (c_j)_{j \in \tilde{H}})$ to V.

V4: V computes $(c_1, \ldots, c_n) \leftarrow \mathsf{SS}(c, (c_i)_{i \in \tilde{H}})$ and $a_i = t_i c_i \bmod \rho$ for each $i = 1, \ldots, n$, and it accepts if $\text{Ver}(A_0, \pi, c_1, \ldots, c_n) = 1$ and if $A_1 = g^u (\prod_{i=1}^n h_i^{a_i})$; otherwise, V rejects.

Notes:

1. SS is a semi-smooth secret sharing scheme on $C(\tau)$ with access structure Δ_n^* on $[1,n]$.
2. $\text{PUB}_{\mathscr{L}}(C(\tau), 1^\tau, n) \leftarrow \text{KeyGen}_{\mathscr{L}}(C(\tau), 1^\tau, n)$ denotes public parameters for a secure $\mathscr{L}$-mercurial commitment scheme $(\text{KeyGen}_{\mathscr{L}}, \text{Com}, \text{Open}, \text{Ver})$, where $\mathscr{L}$ is any language inducing an access structure Π_n on $[1,n]$ that satisfies $(\nabla_n^*)^+ \subseteq \Pi_n \subseteq (\nabla_n^*)^+ \cup \Delta_n^*$.

We now present necessary and sufficient conditions for a language $\mathscr{L} \subseteq \{0,1\}^*$ to be such that $\mathscr{L}$-mercurial commitments are suitable to use in Protocol 4.21 in order to implement a system for batch proofs of Γ-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$.

For the (k,n)-threshold protocol presented in Section 4.4.1, each sub-challenge is a secret share of V's challenge with the $(n-k+1,n)$-threshold access structure on $[1,n]$, and A_0 is an $(n-k,n)$-threshold mercurial commitment, which P must ultimately open to the complete sequence of sub-challenges. The access structure of the secret sharing scheme prevents P* from choosing more than $n-k$ sub-challenges, while the $(n-k,n)$-threshold mercurial commitment binds P* to no fewer than $n-k$ sub-challenges; in particular, under this combination of access

structures, the announcement in any accepting interaction always binds P^* to the *largest number of sub-challenges possible*, thus ensuring that V's challenge uniquely determines the sub-challenges in every non-simulated sub-transcript. More generally, if $\Gamma \subseteq \{0,1\}^*$ is an arbitrary monotone language inducing the access structure Δ_n on $[1,n]$ for each $n \in \mathbb{N}^+$, then the commitment A_0 should bind P^* to the sub-challenges indexed by some $\overline{H} \subseteq [1,n]$ such that $\overline{H} \notin \Delta_n^*$ and, moreover, such that $\overline{H}$ is *maximal* among the subsets of $[1,n]$ that are not in Δ_n^*. To make the preceding statement more precise, we require some terminology.

For a given access structure Δ on U, we let $\nabla = \{\overline{H} \subseteq U \mid \overline{H} \notin \Delta\}$ and $\nabla^* = \{\overline{H}^* \subseteq U \mid \overline{H}^* \notin \Delta^*\}$ respectively denote the complements of Δ and Δ^* in the powerset of U. Observe that if Δ is monotone, then ∇ is *anti-monotone*: $\overline{H} \in \nabla$ and $\overline{H}' \subseteq U$ with $\overline{H}' \subseteq \overline{H}$ implies that $\overline{H}' \in \nabla$. We write ∇^+ to denote the set of *maximal subsets* in ∇; that is, ∇^+ denotes the set of $\overline{H} \in \nabla$ for which $\overline{H}' \in \nabla$ and $\overline{H} \subseteq \overline{H}'$ implies $\overline{H}' = \overline{H}$. The subsets in ∇^+ are called *maximal unauthorized subsets* for Δ. Observation 4.22 follows directly from the definition of ∇^+.

Observation 4.22. If Δ is monotone, then $\nabla^+ \cup \Delta$ is monotone.

We likewise write Δ^- to denote the set of *minimal authorized subsets* in Δ; that is, Δ^- denotes the set of $H \in \Delta$ for which $H' \in \Delta$ and $H' \subseteq H$ implies $H' = H$.

Observation 4.23. Let $\mathcal{I}$ be an arbitrary batch predicate over $\mathcal{R}$. If P knows a $(\Gamma, |U|)$-witness for $\mathcal{I}$, then P knows a $(\Gamma, |U|)$-witness for $\mathcal{I}$ whose valid component witnesses are indexed by a minimal authorized subset $H \in \Delta^-$.

Lemma 4.24 (Jackson and Martin, 1994 [JM94; Result 2]). *For any monotone access structure Δ on a finite set U, $(\nabla^*)^+ = \{U \setminus H \mid H \in \Delta^-\}$.*

Observation 4.23 and Lemma 4.24 together imply that if P knows a Γ-witness for a given batch predicate $\mathcal{I}$, then P can always simulate the sub-transcripts indexed by some maximal unauthorized subset for Δ^*. Our next observation follows directly from Observation 4.3 and the definition of ∇^+.

Observation 4.25. Fix a finite set C, a secret sharing scheme SS for C with access structure Δ^*, and a subsequence $(c_i)_{i \in \overline{H}}$ with $\overline{H} \in (\nabla^*)^+$ and with $c_i \in C$ for each $i \in \overline{H}$. For any given secret $c \in C$, there is a *unique* (and efficient) way to complete the sequence $(c_1, \ldots, c_n) \in (C)^n$ as shares of c under SS.

Taken together, Observations 4.23–4.25 suggest that A_0 should provide $\mathscr{L}$-mercurial binding for the language $\mathscr{L}$ inducing $(\nabla_n^*)^+$ on $[1,n]$ for each $n \in \mathbb{N}^+$. Of course, languages inducing certain supersets of the $(\nabla_n^*)^+$ might also be fine. For instance, the language inducing the *monotone closure* of $(\nabla_n^*)^+$ on $[1,n]$ is clearly acceptable; indeed, if Δ_n is the (k,n)-threshold access structure on $[1,n]$, then Lemma 4.24 implies that $(\nabla_n^*)^+$ is just the set of size-$(n-k)$ subsets of $[1,n]$ so that the $(n-k,n)$-threshold access structure we use in Protocol 4.19 is, in fact, the monotone closure of the associated $(\nabla_n^*)^+$. Theorem 4.26 characterizes which supersets of $(\nabla_n^*)^+$ it is acceptable for $\mathscr{L}$ to induce.

Theorem 4.26. *Let $\Gamma \subseteq \{0,1\}^*$ and $\mathscr{L} \subseteq \{0,1\}^*$ be infinite Boolean languages, with Γ monotone, and let Δ_n and Π_n respectively denote the access structures induced by Γ and $\mathscr{L}$ on $[1,n]$. Protocol 4.21 with SS having access structure Δ_n^* and with A_0 being a secure $(\mathscr{L},n)$-mercurial commitment is a system for honest-verifier statistical zero-knowledge arguments of Γ-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ if and only if $(\nabla_n^*)^+ \subseteq \Pi_n \subseteq (\nabla_n^*)^+ \cup \Delta_n^*$. In particular,*

1. (P,V) *is* complete *if and only if* $(\nabla_n^*)^+ \subseteq \Pi_n$,
2. (P,V) *is* statistically c-simulatable *if and only if* $(\nabla_n^*)^+ \subseteq \Pi_n$, *and*
3. (P,V) *is* (computationally) sound *if and only if* $\Pi_n \subseteq (\nabla_n^*)^+ \cup \Delta_n^*$.

Proof. **Complete:** If $(\nabla_n^*)^+ \subseteq \Pi_n$, then completeness follows by Observation 4.23 and inspection of the protocol; inversely, if $(\nabla_n^*)^+ \not\subseteq \Pi_n$, then there exists $\overline{H} \in (\nabla_n^*)^+ \setminus \Pi_n$ for which P cannot compute the necessary $(\mathscr{L},n)$-mercurial commitment A_0 for the sub-challenges indexed by $\overline{H}$. Now, it follows from Lemma 4.24 that $H = [1,n] \setminus \overline{H}$ is a minimal authorized set; hence, this is equivalent to saying that there is some minimal authorized set $H \in \Delta$ for which P cannot prove knowledge of any (Γ,n)-witness whose valid component witnesses are indexed by H.

(Statistically) c-simulatable: Our standard argument establishes that the ensembles of random variables describing real and simulated sub-transcripts having A_0 and π omitted are identical. Let $\tilde{\mathrm{S}}_{\mathrm{V}}$ be the simulator that outputs the above perfect sub-transcripts given the challenge c as input. A simulator for the full transcripts invokes $(A_1, t, c, u, c_1, \ldots, c_n) \leftarrow \tilde{\mathrm{S}}_{\mathrm{V}}(c)$, and then it chooses $\overline{H} \in (\nabla_n^*)^+$ arbitrarily and computes $(A_0, \gamma) \leftarrow \mathrm{Com}((c_i)_{i \in \overline{H}})$ and $\pi \leftarrow \mathrm{Open}(A_0, \gamma, c_1, \ldots, c_n)$. Finally, the simulator chooses $\tilde{H} \in_{\mathrm{R}} (\Delta_n^*)^-$ such that $|\tilde{H}| \leq |\tilde{H}'|$ for all $\tilde{H}' \in (\Delta_n^*)^-$, just like honest P would do. The unconditional $\mathscr{L}$-mercurial hiding property implies that the distribution

from which S_V draws (A_0, π) is statistically indistinguishable from the distribution from which P draws (A_0, π) in the real transcripts; hence, the joint distributions describing the real and simulated transcripts $(A_0, t, A_1, c, u, \pi, (c_i)_{i \in \tilde{H}})$ are themselves statistically indistinguishable.

(Computationally) sound: Suppose that some PPT prover P^* outputs the $(\mathscr{L}, n)$-mercurial commitment A_0 in its opening move. If $\Pi_n \subseteq (\nabla_n^*)^+ \cup \Delta_n^*$, then by the computational $\mathscr{L}$-mercurial binding property, there exists a fixed subset $\overline{H} \in (\nabla_n^*)^+ \cup \Delta_n^*$, and a fixed subsequence of sub-challenges $(c_i)_{i \in \overline{H}}$, such that P^* can only open A_0 to sequences consistent with $(c_i)_{i \in \overline{H}}$, except with probability negligible in τ. Moreover, from Observations 4.3 and 4.25, it follows that, for any such subsequence $(c_i)_{i \in \overline{H}}$ and for any challenge c from V, there is at most one possible way to complete the sequence $(c_1, \ldots, c_n)$ as shares of c in SS. (More precisely, there is exactly one way to complete the sequence of sub-challenges if $\overline{H} \in (\nabla_n^*)^+$, and there is no way to complete the sequence of sub-challenges if $\overline{H} \in \Delta_n^*$.)

Set $H = [1, n] \setminus \overline{H}$ and suppose the extractor E_{P^*} obtains from P^* a pair of accepting transcripts $T_1^{(1)} = (A_0, t^{(1)}, A_1^{(1)}, c_1^{(1)}, u_1^{(1)}, \pi_1^{(1)}, c_{11}^{(1)}, \ldots, c_{1n}^{(1)})$ and $T_2^{(1)} = (A_0, t^{(1)}, A_1^{(1)}, c_2^{(1)}, u_2^{(1)}, \pi_2^{(1)}, c_{21}^{(1)}, \ldots, c_{2n}^{(1)})$ that use the same announcement pair $(A_0, A_1^{(1)})$ and the same sequence of short exponents $t^{(1)} = (t_1^{(1)}, \ldots, t_n^{(1)})$, but distinct challenges $c_1^{(1)} \neq c_2^{(1)}$. The final verification equation implies both that $A_1^{(1)} = g^{u_1^{(1)}} (\prod_{i=1}^n h_i^{a_{1i}^{(1)}})$ and $A_1^{(1)} = g^{u_2^{(1)}} (\prod_{i=1}^n h_i^{a_{2i}^{(1)}})$, where $a_{1i}^{(1)} = t_i^{(1)} c_{1i}$ and $a_{2i}^{(1)} = t_i^{(1)} c_{2i}$ for each $i = 1, \ldots, n$; hence,

$$g^{u_1^{(1)} - u_2^{(1)}} = \left(\prod_{i=1}^{n} h_i^{a_{1i}^{(1)} - a_{2i}^{(1)}}\right).$$

Moreover, as $T_1^{(1)}$ and $T_2^{(1)}$ arose from interactions with the above *PPT* prover P^*, the above argument regarding the computational $\mathscr{L}$-mercurial binding of A_0 guarantees that the latter expression simplifies to

$$g^{u_1^{(1)} - u_2^{(1)}} = \left(\prod_{j \in H} h_j^{a_{1j}^{(1)} - a_{2j}^{(1)}}\right)$$

with a probability overwhelming in τ. Taking logarithms to the base g yields a linear equation

$$u_1^{(1)} - u_2^{(1)} = \sum_{j \in H} (a_{1j}^{(1)} - a_{2j}^{(1)}) \log_g h_j$$

in $|H| = n - |\overline{H}|$ unknowns. Given any linearly independent set of $|H| \leq n$ such equations, a universal knowledge extractor $\mathrm{E}_{\mathrm{P}^*}$ can efficiently solve for $x_j = \log_g h_j$ for the indices $j \in H$. To obtain a second linearly independent such equation, $\mathrm{E}_{\mathrm{P}^*}$ rewinds P^* to Step V2, and then it issues a new sequence of short exponents $t^{(2)} = (t_1^{(2)}, \ldots, t_n^{(2)})$ and extracts another equation of the form $\sum_{j \in H} (a_{1j}^{(2)} - a_{2j}^{(2)}) \log_g h_j$, where $a_{1j}^{(2)} = t_j^{(2)} c_{1j}^{(2)}$ and $a_{2j}^{(2)} = t_j^{(2)} c_{2j}^{(2)}$. As the coefficients in this latter expression are each distributed uniformly at random in $\mathbb{Z}_\rho$, it follows, with a probability overwhelming in λ_0, that this latter equation is linearly independent from the first equation. Moreover, by Chernoff's inequality, it follows that if $\mathrm{E}_{\mathrm{P}^*}$ repeats this process about $|\overline{H}|$ times (or perhaps slightly more), then it will obtain $|\overline{H}|$ linearly independent such equations with probability overwhelming in λ_0.

4.4.2.1 Proofs of partial knowledge from $(n-k, n)$-threshold mercurial commitments

This section considers non-threshold languages $\Gamma \subseteq \{0,1\}^*$ for which an $(n-k, n)$-threshold access structure, with k suitably chosen, satisfies the criteria in Theorem 4.26. In particular, we establish the following generalization of Cramer et al.'s result for non-batch protocols.

Theorem 4.27. *Protocol 4.21, instantiated with our trapdoor $\Gamma_{\text{-}k}$-mercurial commitment scheme (as described in Section 4.3.2.3) and Benaloh and Leichter's secret sharing construction [BL88], is sufficient to construct a system for batch honest-verifier statistical zero-knowledge arguments of Γ-partial knowledge over $\mathcal{R} \leftarrow \mathcal{G}_{\mathrm{DL}}(1^*)$, where Γ can be any NP-language described by a family of polynomial-sized monotone Boolean formulas. Arguments in the resulting protocol are computationally convincing under the SDH and polyDH assumptions for the bilinear group-generating algorithm employed by $\mathit{KeyGen}_{\Gamma_{\text{-}k}}$.*

To facilitate this discussion, we define the shorthand $T(U,k)$ to denote the $(|U| - k, |U|)$-threshold access structure on the finite set U.

Recall that in Cramer et al.'s construction for proofs of Γ-partial knowledge, the "effective" fan-in of the protocol is dictated not by the fan-in n of the common input predicate, but by the size of the Boolean formula chosen to describe the access structure Δ_n induced by Γ on $[1,n]$; in particular, if the variable associated with a given component input appears several times in the chosen formula, then every such occurrence of that variable results in a distinct

announcement-challenge-response triple in the resulting protocol. When we RME-batch the protocol, this results in expressions of the form $(\prod_{i \in U_1} h_i^{a_i^{(1)}}) \cdots (\prod_{i \in U_m} h_i^{a_i^{(m)}})$, wherein the U_i are not necessarily disjoint so that $\sum_{i=1}^{m} |U_i|$ may be larger than n. Fortunately, it is always possible to rearrange such expressions to be of the form $\prod_{i=1}^{n} h_i^{\hat{a}_i}$, where each $\hat{a}_i$ is of the form $\hat{a}_i = \sum_{j=1}^{m} t_i^{(j)} c_i^{(j)}$. Hence, in the batch proof of Γ-partial knowledge, the numbers of sub-challenges c_i and short exponents t_i are both dependent on the size of the formula chosen to describe Δ_n, *but the effective fan-in to the rest of the protocol is not*. In other words, if the access structure Δ_n is equivalent to the composition of several simpler access structures, then we can implement the proof of Γ-partial knowledge by simply "repeating" the common input several times and proving an appropriate composition of the simpler protocols that arise for those simpler access structures. The rest of this section discusses how to implement such compositions.

4.4.2.2 Conjunctions of partial knowledge proofs

Our first set of composition results follow from Theorem 4.32 and deal with languages formed by the conjunction of two or more monotone languages. Let L_1 and L_2 be monotone Boolean languages inducing access structures Δ_1 and Δ_2 on disjoint sets U_1 and U_2, respectively. Define $\Delta_{1\wedge 2} = \Delta_1 \uplus \Delta_2 = \{H_1 \cup H_2 \mid H_1 \in \Delta_1 \wedge H_2 \in \Delta_2\}$, the *Cartesian union* of Δ_1 and Δ_2. Intuitively, $\Delta_{1\wedge 2}$ is the access structure that arises when we wish to prove knowledge of both an L_1-witness and an L_2-witness. We write $L_{1\wedge 2}$ as shorthand for the language inducing $\Delta_{1\wedge 2}$ on $U_1 \cup U_2$.

Observation 4.28. $\Delta_{1\wedge 2}$ is a monotone access structure on $U_1 \cup U_2$.

Before stating our main result of this section, we prove several technical lemmas. In each lemma, we assume that Δ_1 and Δ_2 are monotone access structures on the disjoint sets U_1 and U_2, respectively.

Lemma 4.29. *If $H_1 \in \Delta_1^*$, then $H_1 \cup H_2 \in \Delta_{1\wedge 2}^*$ for all $H_2 \subseteq U_2$.*

Proof. $H_1 \in \Delta_1^*$ implies that $U_1 \setminus H_1 \notin \Delta_1$; hence, $(U_1 \setminus H_1) \cup (U_2 \setminus H_2) \notin \Delta_{1\wedge 2}$. As U_1 and U_2 are disjoint, this simplifies to $(U_1 \cup U_2) \setminus (H_1 \cup H_2) \notin \Delta_{1\wedge 2}$; hence, $H_1 \cup H_2 \in \Delta_{1\wedge 2}^*$. ∎

Lemma 4.30. $\Delta_{1\wedge 2}^- = \Delta_1^- \uplus \Delta_2^-$ *and* $(\nabla_{1\wedge 2}^*)^+ = (\nabla_1^*)^+ \uplus (\nabla_2^*)^+$.

Proof. Suppose $H \in \Delta^-_{1\wedge 2}$ and write $H = H_1 \cup H_2$ with $H_1 \in \Delta_1$ and $H_2 \in \Delta_2$. If $H_1 \notin \Delta^-_1$, then there exists some proper subset $H'_1 \subseteq H_1$ such that $H'_1 \in \Delta_1$ and, consequently, such that $H'_1 \cup H_2 \in \Delta_{1\wedge 2}$. As U_1 and U_2 are disjoint, $H'_1 \cup H_2$ is a proper subset of H, thus contradicting our hypothesis that $H \in \Delta^-_{1\wedge 2}$. A symmetric argument holds if $H_2 \notin \Delta^-_2$; hence, $\Delta^-_{1\wedge 2} \subseteq \Delta^-_1 \uplus \Delta^-_2$.

Conversely, if $H_1 \in \Delta^-_1$ and $H_2 \in \Delta^-_2$, then $H_1 \cup H_2 \in \Delta_{1\wedge 2}$ by definition. If $H_1 \cup H_2 \notin \Delta^-_{1\wedge 2}$, then there exists some proper subset $H \subseteq H_1 \cup H_2$ such that $H \in \Delta_{1\wedge 2}$ and, consequently, such that both $H \cap H_1 \in \Delta_1$ and $H \cap H_2 \in \Delta_2$. But H being a proper subset of $H_1 \cup H_2$ implies that $H \cap H_1$ is a proper subset of H_1, or that $H \cap H_2$ is a proper subset of H_2, or both, thus contradicting our hypothesis that both $H_1 \in \Delta^-_1$ and $H_2 \in \Delta^-_2$; hence, $\Delta^-_1 \uplus \Delta^-_2 \subseteq \Delta^-_{1\wedge 2}$.

This completes the proof that $\Delta^-_{1\wedge 2} = \Delta^-_1 \uplus \Delta^-_2$. The second result, $(\nabla^*_{1\wedge 2})^+ = (\nabla^*_1)^+ \uplus (\nabla^*_2)^+$, now follows from Lemma 4.24 and Observation 4.28. ∎

Lemma 4.31. *For any monotone access structure Δ on a finite set U, if $T(U,k) \subseteq (\nabla^*)^+ \cup \Delta^*$, then $|\overline{H}| = |U| - k$ for every $\overline{H} \in (\nabla^*)^+$.*

Proof. Note that Δ^* and ∇^* are disjoint by construction. Suppose for a contradiction that $\overline{H} \in (\nabla^*)^+$ with $|\overline{H}| > |U| - k$ and let $\overline{H}'$ be any size-$(|U| - k)$ subset of $\overline{H}$. Clearly $\overline{H}' \notin (\nabla^*)^+$; thus, as $T(U,k) \subseteq (\nabla^*)^+ \cup \Delta^*$ by assumption, $\overline{H}' \in \Delta^*$. But Δ^* is monotone; hence $\overline{H}' \in \Delta^*$ implies $\overline{H} \in \Delta^*$, a contradiction. ∎

With the necessary groundwork in place, we now prove the main result of this section.

Theorem 4.32. *If*

$$(\nabla^*_1)^+ \subseteq T(U_1, k_1) \subseteq (\nabla^*_1)^+ \cup \Delta^*_1$$

and

$$(\nabla^*_2)^+ \subseteq T(U_2, k_2) \subseteq (\nabla^*_2)^+ \cup \Delta^*_2,$$

then

$$(\nabla^*_{1\wedge 2})^+ \subseteq T(U_1 \cup U_2, k_1 + k_2) \subseteq (\nabla^*_{1\wedge 2})^+ \cup \Delta^*_{1\wedge 2}.$$

Proof. Let $H_1 \in (\nabla_1^*)^+$ and $H_2 \in (\nabla_2^*)^+$, and set $H = H_1 \cup H_2$. By Lemma 4.31, $|H_1| = |U_1| - k_1$ and $|H_2| = |U_2| - k_2$; therefore, as U_1 and U_2 are disjoint, $|H| = (|U_1| + |U_1|) - (k_1 + k_2)$. Lemma 4.30 states that all such $H \in (\nabla_{1\wedge 2}^*)^+$ can be expressed as such a product of maximal unauthorized subsets; hence, $(\nabla_{1\wedge 2}^*)^+ \subseteq T(U_1 \cup U_2, k_1 + k_2)$.

Now, suppose $H \in T(U_1 \cup U_2, k_1 + k_2)$ and write $H = H_1 \cup H_2$ with $H_1 \in \Delta_1$ and $H_2 \in \Delta_2$. A simple counting argument establishes that at least one of $|H_1| \geq |U_1| - k_1$ or $|H_2| \geq |U_2| - k_2$. If both inequalities hold, then $H_1 \in (\nabla_1^*)^+ \cup \Delta_1^*$ and $H_2 \in (\nabla_2^*)^+ \cup \Delta_2^*$ by assumption; otherwise, one of the inequalities, say H_i, is strict so that $|H_i| > |U_i| - k_i$. In the latter case, Lemma 4.31 implies that $H_i \in \Delta_i^*$. In both cases we therefore have either that (i) both $H_1 \in (\nabla_1^*)^+$ and $H_2 \in (\nabla_2^*)^+$ so that $H \in (\nabla_{1\wedge 2}^*)^+$ follows from Lemma 4.30, or (ii) at least one of $H_1 \in \Delta_1^*$ or $H_2 \in \Delta_2^*$ so that $H \in \Delta_{1\wedge 2}^*$ follows from Lemma 4.29; thus, $T(U_1 \cup U_2, k_1 + k_2) \subseteq (\nabla_{1\wedge 2}^*)^+ \cup \Delta_{1\wedge 2}^*$, as desired. ∎

Corollary 4.33. *If we can implement systems for proofs of L_1-partial and L_2-partial knowledge knowledge, both over $\mathcal{R}$, using Protocol 4.21 instantiated with trapdoor $\Gamma_{\text{-}k_1}$- and $\Gamma_{\text{-}k_2}$-mercurial commitments, respectively, then we can implement a system for proofs $L_{1\wedge 2}$-partial knowledge over $\mathcal{R}$ using Protocol 4.21 instantiated with trapdoor $\Gamma_{\text{-}k_3}$-mercurial commitments, where $k_3 = k_1 + k_2$.*

Corollary 4.34. *Given a language Γ expressed as the conjunction of some number $m \in \mathbb{N}^+$ of (k_i, n_i)-threshold (including OR) operands, we can implement a system for proofs of Γ-partial knowledge for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ using Protocol 4.21 instantiated with $\Gamma_{\text{-}k}$-mercurial commitments, where $k = \sum_{i=1}^{m} k_i$.*

Corollary 4.34 implies that we can construct systems for proofs of Γ-partial knowledge for any language with a polynomial-sized representation in a generalized *conjunctive normal form* (CNF) that permits arbitrary threshold operations in its operands.

4.4.2.3 Disjunctions of partial knowledge proofs

Our next set of results follow from Theorem 4.38 and deal with languages formed by the disjunction of two or more monotone languages. As before, let L_1 and L_2 be monotone Boolean languages inducing access structures Δ_1 and Δ_2 on disjoint sets U_1 and U_2, respectively. Define

$\Delta_{1\vee2} = \{H \subseteq U_1 \cup U_2 \mid H \cap U_1 \in \Delta_1 \vee H \cap U_2 \in \Delta_2\}$. Intuitively, $\Delta_{1\vee2}$ is the access structure that arises when we wish to prove knowledge of either an L_1-witness or an L_2-witness. We write $L_{1\vee2}$ as shorthand for the language inducing $\Delta_{1\vee2}$ on $U_1 \cup U_2$.

Observation 4.35. $\Delta_{1\vee2}$ is a monotone access structure on $U_1 \cup U_2$.

Before stating our main result of this section, we again prove some technical lemmas. In each lemma, we assume that Δ_1 and Δ_2 are monotone access structures on the disjoint sets U_1 and U_2, respectively.

Lemma 4.36. *Let $H_1 \subseteq U_1$ and $H_2 \subseteq U_2$. Then $H_1 \cup H_2 \in \Delta^*_{1\vee2}$ if and only if $H_1 \in \Delta^*_1$ and $H_2 \in \Delta^*_2$.*

Proof. By definition, $H \in \Delta^*_{1\vee2}$ if and only if $U_1 \cup U_2 \setminus H \notin \Delta_{1\vee2}$, which, by the disjointness of U_1 and U_2, is equivalent to $(U_1 \setminus H) \cup (U_2 \setminus H) \notin \Delta_{1\vee2}$. Now, if either $(U_1 \setminus H) \in \Delta_1$ or $(U_2 \setminus H) \in \Delta_2$, then $(U_1 \setminus H) \cup (U_2 \setminus H) \in \Delta_{1\vee2}$ by construction; hence, by taking the contrapositive, we get that $(U_1 \setminus H) \cup (U_2 \setminus H) \notin \Delta_{1\vee2}$ if and only if $(U_1 \setminus H) \notin \Delta_1$ and $(U_2 \setminus H) \notin \Delta_2$, which is equivalent to saying that $U_1 \cap H \in \Delta^*_1$ and $U_2 \cap H \in \Delta^*_2$. ∎

Lemma 4.37. $\Delta^-_{1\vee2} = \Delta^-_1 \cup \Delta^-_2$ *and* $(\nabla^*_{1\vee2})^+ = \{H_1 \cup U_2 \mid H_1 \in (\nabla^*_1)^+\} \cup \{U_1 \cup H_2 \mid H_2 \in (\nabla^*_2)^+\}$.

Proof. If $H_1 \in \Delta^-_1$, then $H_1 \in \Delta^-_{1\vee2}$ as there are no proper subsets of H_1 in Δ_1 by definition and there are no proper subsets of H_1 in Δ_2 by the disjointness of U_1 and U_2. A symmetric argument holds for $H_2 \in \Delta^-_2$; hence, $\Delta^-_1 \cup \Delta^-_2 \subseteq \Delta^-_{1\vee2}$. To prove that $\Delta^-_{1\vee2} \subseteq \Delta^-_1 \cup \Delta^-_2$, we suppose for a contradiction that $H \in \Delta^-_{1\vee2}$ such that $H \notin \Delta^-_1 \cup \Delta^-_2$ and note that, by definition, either $H \cap U_1 \in \Delta_1$, or $H \cap U_2 \in \Delta_2$, or both. Suppose that $H \cap U_i \in \Delta_i$. There are two possibilities: $H \cap U_i = H$ or $H \cap U_i \neq H$. If $H \cap U_i = H$, then $H \in \Delta_i \setminus \Delta^-_i$ by hypothesis; likewise, if $H \cap U_i \neq U_i$, then $H' = H \cap U_i$ is a proper subset of H and, moreover, it is in Δ_i. In both cases, we have show that there exists a proper subset of H in Δ_i and, therefore, in $\Delta_{1\vee2}$, thus contradicting the minimality of H.

This completes the proof that $\Delta^-_{1\wedge2} = \Delta^-_1 \uplus \Delta^-_2$. The second result, $(\nabla^*_{1\vee2})^+ = (\nabla^*_1)^+ \cup (\nabla^*_2)^+$, now follows from Lemma 4.24 and Observation 4.35. ∎

We now prove the main result of this section.

Theorem 4.38. *If*

$$(\nabla_1^*)^+ \subseteq T(U_1, k) \subseteq (\nabla_1^*)^+ \cup \Delta_1^*$$

and

$$(\nabla_2^*)^+ \subseteq T(U_2, k) \subseteq (\nabla_2^*)^+ \cup \Delta_2^*,$$

then

$$(\nabla_{1\vee2}^*)^+ \subseteq T(U_1 \cup U_2, k) \subseteq (\nabla_{1\vee2}^*)^+ \cup \Delta_{1\vee2}^*.$$

Proof. If $H \in (\nabla_{1\vee2}^*)^+$, then, by Lemma 4.37, there exists either $H_1 \in (\nabla_1^*)^+$ such that $H = H_1 \cup U_2$ or $H_2 \in (\nabla_2^*)^+$ such that $H = U_1 \cup H_2$. In the first case, Lemma 4.31 implies that $|H_1| = |U_1| - k$ so that $|H| = (|U_1| - k) + |U_2|$ follows from the disjointness of U_1 and U_2. A symmetric argument establishes that $|H| = (|U_2| - k) + |U_1|$ in the second case; hence, in either case $|H| = |U_1| + |U_2| - k$ so that $H \in T(U_1 \cup U_2, k)$. This establishes that $(\nabla_{1\vee2}^*)^+ \subseteq T(U_1 \cup U_2, k)$.

Now, suppose $H \in T(U_1 \cup U_2, k)$ and set $H = H_1 \cup H_2$ with $H_1 \subseteq U_1$ and $H_2 \subseteq U_2$. A simple counting argument establishes that $|H_1| \geq |U_1| - k$ and $|H_2| \geq |U_2| - k$; hence, $H_1 \in (\nabla_1^*)^+ \cup \Delta_1^*$ and $H_2 \in (\nabla_2^*)^+ \cup \Delta_2^*$ by assumption. If $H_1 \in (\nabla_1^*)^+$, then it follows from Lemma 4.31 that $|H_1| = |U_1| - k$ so that $H_2 = U_2$ and $H \in (\nabla_{1\vee2}^*)^+$. A symmetric argument establishes that $H \in (\nabla_{1\vee2}^*)^+$ when $H_2 \in (\nabla_2^*)^+$. Otherwise, if both $H_1 \in \Delta_1^*$ and $H_2 \in \Delta_2^*$, then it follows from Lemma 4.36 that $H \in \Delta_{1\vee2}^*$; thus, $T(U_1 \cup U_2, k) \subseteq (\nabla_{1\vee2}^*)^+ \cup \Delta_{1\vee2}^*$, as desired. ∎

Corollary 4.39. *If we can construct systems for proofs of L_1-partial and L_2-partial knowledge knowledge, both over $\mathcal{R}$ and both using Protocol 4.21 instantiated with trapdoor $\Gamma_{\text{-}k}$-mercurial commitments, then we can construct a system for proofs of a $L_{1\vee2}$-partial knowledge over $\mathcal{R}$ using Protocol 4.21 also instantiated with trapdoor $\Gamma_{\text{-}k}$-mercurial commitments.*

Corollary 4.40. *Given a language Γ expressed as the disjunction of some number $m \in \mathbb{N}^+$ of $(n_i - k, n_i)$-threshold (including OR) operands, we can implement a system for proofs of Γ-partial knowledge for $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ using Protocol 4.21 instantiated with trapdoor $\Gamma_{\text{-}k}$-mercurial commitments.*

Corollary 4.40 implies that we can construct systems for proofs of Γ-partial knowledge for any language with a polynomial-sized representation in a generalized *disjunctive normal form* (DNF) that permits arbitrary (k, n_i)-threshold operations in its operands for any fixed $k \in \mathbb{N}^+$.

4.4.2.4 Thresholds of partial knowledge proofs

Our final set of results in this section generalizes Theorems 4.32 and 4.38 to the general threshold case. Let $L_1, \ldots, L_m$ be monotone Boolean languages inducing access structures $\Delta_1, \ldots, \Delta_m$ on disjoint sets $U_1, \ldots, U_m$, respectively. Define $\Delta_{(d,m)}$ as the monotone closure of $\{\bigcup_{i \in A} H_i \mid A \subseteq_d [1,m] \wedge H_i \in \Delta_i \text{ for each } i \in A\}$ in $\bigcup_{i=1}^m U_i$. Intuitively, $\Delta_{(d,m)}$ is the access structure that arises when we wish to prove knowledge of d batch witnesses from a set of m such witnesses. We write $L_{(d,m)}$ as shorthand for the language inducing $\Delta_{(d,m)}$ on $U_1 \cup \cdots \cup U_m$.

Observation 4.41. $\Delta_{(d,m)}$ is a monotone access structure on $\bigcup_{i=1}^m U_i$.

Lemma 4.42. $\Delta^-_{(d,m)} = \{\bigcup_{i \in A} H_i \mid A \subseteq_d [1,m] \wedge H_i \in \Delta_i^-\}$ *and* $(\nabla^*_{(d,m)})^+ = \{(\bigcup_{i \in A} H_i) \cup (\bigcup_{i \in [1,m] \setminus A} U_i) \mid A \subseteq_d [1,m] \wedge H_i \in (\nabla_i^*)^+\}$.

The proof of Lemma 4.42 is a direct extension of the proof of Lemma 4.37.

Theorem 4.43. *If*

$$(\nabla_i^*)^+ \subseteq T(U_i, k) \subseteq (\nabla_i^*)^+ \cup \Delta_i^*$$

for each $i = 1, \ldots, m$*, then*

$$(\nabla^*_{(d,m)})^+ \subseteq T(U_1 \cup \cdots \cup U_m, dk) \subseteq (\nabla^*_{(d,m)})^+ \cup \Delta^*_{(d,m)}.$$

The proof of Theorem 4.43 is a direct extension of the proof of Theorem 4.38.

Corollary 4.44. *Given a language* Γ *expressed as a* (d,m)*-threshold operator applied to some number* $m \in \mathbb{N}^+$ *of* (k, n_i)*-threshold operands, we can implement a system for proofs of* Γ*-partial knowledge for* $\mathcal{R} \leftarrow \mathcal{G}_{\text{DL}}(1^*)$ *using Protocol 4.21 instantiated with trapdoor* $\Gamma_{\cdot dk}$*-mercurial commitments.*

Corollaries 4.40 and 4.44 imply that we can use Protocol 4.21 to construct systems for batch proofs of Γ-partial knowledge for any language with a polynomial-sized representation in a *threshold normal form* that permits arbitrary "balanced" thresholds of arbitrary balanced thresholds. We remark that it is also possible to combine "unbalanced" threshold statements by using several distinct trapdoor $\Gamma_{\text{-}k}$-mercurial commitments, with the threshold k varying across commitments.[29] In particular, given an unbalanced threshold statement, we treat each operand as a self-contained "sub-predicate" and construct the appropriate commitment scheme for that sub-predicate only. If some sub-predicate contains an unbalanced threshold, then we apply the same idea recursively until all sub-predicates are expressible using one of the above normal forms. For Boolean formulas with a high "unbalanced threshold depth", this recursive approach can impose a fairly high overhead on the resulting protocol; nonetheless, it enables us to deal with precisely the same class of Boolean formulas as Cramer et al.'s construction for non-batch proofs together with Benaloh and Leichter's secret sharing scheme. Moreover, we note that the overhead imposed by the secret sharing is directly proportional to the cost of the corresponding non-batch protocol; hence, we obtain our desired Theorem 4.27.

4.4.3 Non-monotone proofs of partial knowledge

This section introduces a new methodology for constructing batch zero-knowledge proofs of Γ-partial knowledge for families of linear relations $\mathcal{R}$ when $\Gamma \subseteq \{0,1\}^*$ is a *non-monotone* language. For each $n \in \mathbb{N}^+$, let Δ_n denote the access structure that Γ induces on $[1,n]$. In cases where Γ is monotone, P can prove Γ-partial knowledge for a given batch predicate $\mathcal{I}$ by proving knowledge of witnesses for the component instances in $\mathcal{I}$ indexed by *any* authorized subset $H \in \Delta_n$; indeed, even if P^* actually knows witnesses for the component instances indexed by some strict superset H' of H, the monotonicity of Γ ensures that $H' \in \Delta_n$ and, therefore, that the predicate under consideration is indeed Γ-correct. Things are considerably more complex when Γ is non-monotone as, by definition, Γ being non-monotone implies that there exists some authorized subset $H \in \Delta_n$ having one or more strict supersets in ∇_n. Thus, to prove Γ-partial knowledge in the non-monotone case, not only must P prove knowledge of witnesses for the

[29] Alternatively, one could design a trapdoor $\mathcal{L}$-mercurial commitment scheme with the suitable binding property to handle the full predicate all at once; this latter approach is potentially more efficient.

component instances of $\mathcal{I}$ indexed by an authorized subset $H \in \Delta_n$, but P must also prove that it does *not* know witnesses for all the component instances of $\mathcal{I}$ indexed by *any* unauthorized superset H' of H.

Of course, if a valid witness for some component instance exists, then P cannot prove not to know it; thus, proofs of Γ-partial knowledge over $\mathcal{R} = (R_\tau)_{\tau \in \mathbb{N}^+}$ when Γ is non-monotone are only meaningful when $\mathcal{R}$ is such that, for any given string $s \notin L_{R_\tau}$, there is a means by which P can (efficiently) prove to V that $\mathcal{W}_{R_\tau}(s)$ is empty.[30] For example, it does *not* make sense to consider proofs of Γ-partial knowledge of DLs in prime-order groups when Γ is non-monotone since, in such prime-order groups, the DL always exists, regardless of whether P happens to know it. On the other hand, it *does* make sense to consider proofs of Γ-partial knowledge and *equality* among DLs when Γ is non-monotone: if two DLs are equal, then P can prove so with Chaum and Pedersen's protocol (see Figure 3.1 on Page 56) and, if two DLs are not equal, then P can prove so with the variant of Chaum and Pedersen's protocol depicted in Figure 4.3.

4.4.3.1 Camenisch and Shoup's protocol for inequality of DLs

The protocol depicted in Figure 4.3 is due to Camenisch and Shoup [CS03; §6]. The common input to Camenisch and Shoup's protocol is $(\mathbb{G}, q, g_0, g_1) \leftarrow \mathcal{G}(1^\tau; 2)$ and a pair of group elements $(h_0, h_1) \in (\mathbb{G})^2$. In the protocol, P proves knowledge of an exponent $x \in \mathbb{Z}_q$ such that $\log_{g_0} h_0 = x$ and $\log_{g_1} h_1 \neq x$. We typically assume that $y = \log_{g_1} h_1$ is unknown to P, although the proof remains sound even if this is not the case. The protocol is denoted in Camenisch-Stadler notation by $\mathrm{PK}\{ x : h_0 = g_0^x \wedge h_1 \neq g_1^x \}$.

The protocol works as follows: P first chooses a random exponent $r \in_R \mathbb{Z}_q^*$, and then uses it to compute the auxiliary commitment $A_1 = (g_1^x / h_1)^r$. P then engages V in the protocol denoted by $\mathrm{PK}\{(\gamma_1, \gamma_2) : 1 = g_0^{\gamma_1} h_0^{\gamma_2} \wedge A_1 = g_1^{\gamma_1} h_1^{\gamma_2}\}$, which is implemented using a straightforward 2-DLREP generalization of Chaum and Pedersen's protocol. From the expression $1 = g_0^{\gamma_1} h_0^{\gamma_2}$, we find that $\gamma_1 = -\gamma_2 \log_{g_0} h_0$. Furthermore, by substituting for γ_1 in the expression $A_1 = g_1^{\gamma_1} h_1^{\gamma_2}$,

[30] More precisely, the language L_{R_τ} induced by $R_\tau : S_\tau \times W_\tau$ must be in NP $\cap$ co-NP for each $\tau \in \mathbb{N}^+$, where co-NP refers to the complexity class of languages L_{R_τ} for which the complement $S'_\tau = S_\tau \setminus L_{R_\tau}$ is itself a non-trivial NP-language. In other words, we can view co-NP as the class of languages Γ for which any string not in Γ is associated with one or more short proofs of *non-membership* in Γ. We refer the reader to Arora and Barak [AB09; Section 2.5] for further details about co-NP.

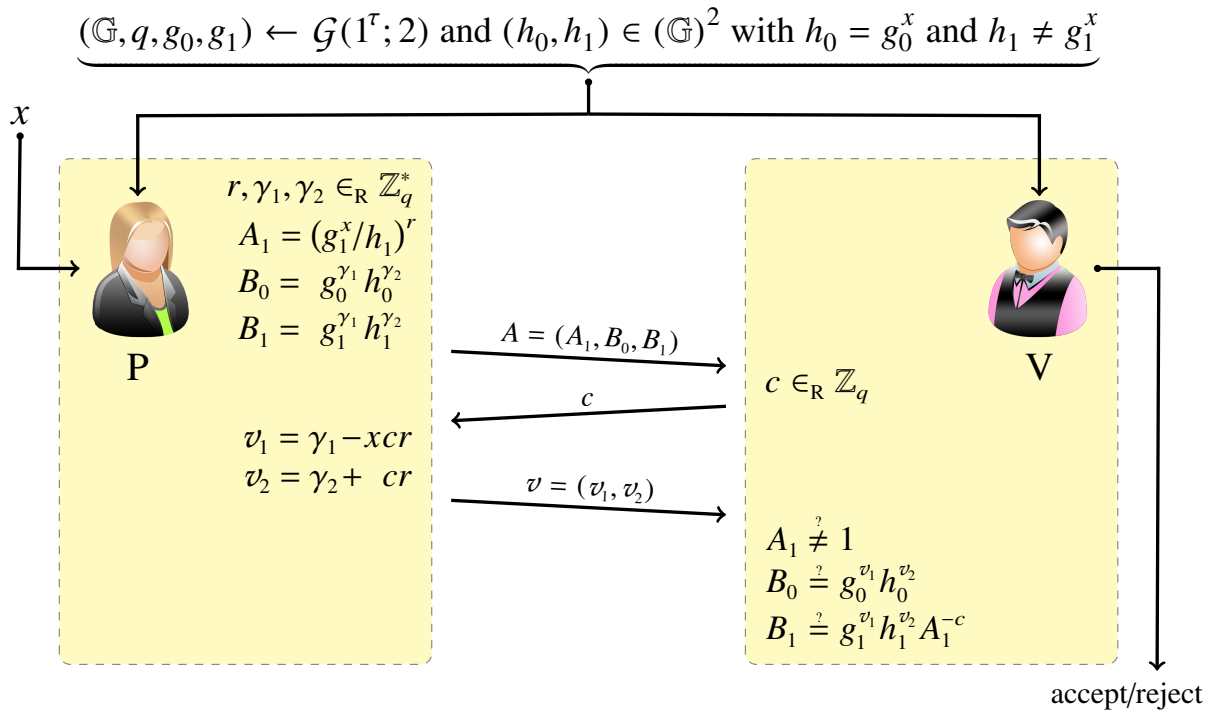

Figure 4.3: A variant of Chaum and Pedersen's protocol for proving the inequality of a pair of DLs, due to Camenisch and Shoup [CS03; §6]. The protocol is denoted by $\mathrm{PK}\{ x : h_0 = g_0^x \wedge h_1 \neq g_1^x \}$.

we find that $A_1 = (g_1^{-\log_{g_0} h_0} h_1)^{\gamma_2}$, from which it follows that $\log_{g_0} h_0 = \log_{g_1} h_1$ if and only if $A_1 = 1$. But V rejects when $A_1 = 1$; hence, if V accepts, then $\log_{g_0} h_0 \neq \log_{g_1} h_1$ with a probability overwhelming in τ. Finally, a universal knowledge extractor for the Chaum-Pedersen sub-protocol can extract the pair (γ_1, γ_2) from P^*, and then use this pair of values to compute the witness $\log_{g_0} h_0 = -\gamma_1 / \gamma_2$; thus, the protocol constitutes a system for proofs of knowledge of $x = \log_{g_0} h_0$.

Theorem 4.45 (Camenisch and Shoup, 2003 [CS03; Theorem 5]). *The protocol depicted in Figure* **??** *is a system for honest-verifier perfect zero-knowledge proofs of knowledge for the inequality of DLs relations induced by* $\mathcal{G}_{\mathrm{DL}}(1^*)$. *Furthermore, it is a c-simulatable and* 2*-extractable sigma protocol and so has constant knowledge error function* $\kappa(\mathbb{G}, q, g_0, g_1, h_0, h_1) = 1/q$.

P's expected cost in Camenisch and Shoup's protocol is less than $\mathrm{ExpCost}^{(2)}_{\mathbb{G}}((2,\tau)) + \mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) < 6\tau$ multiplications in $\mathbb{G}$ and V's expected cost is $\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) + \mathrm{ExpCost}_{\mathbb{G}}((3,\tau)) < 4.5\tau$ multiplications in $\mathbb{G}$. For communication cost, P sends three $\mathbb{G}^*$ elements and two $\mathbb{Z}_q$ elements to V, and V sends one $\mathbb{Z}_q$ element to P.

The RME-based common-exponent Camenisch-Shoup protocol. We now discuss how to parallelize Camenisch and Shoup's protocol into a system for batch honest-verifier *computational* zero-knowledge proofs of knowledge. Our idea is to 'reuse' each of the random exponents that appear in the announcement so that we can employ an RME-based common-exponent parallelization of Chaum and Pedersen's protocol for the remainder of the interaction. The resulting protocol is presented in Protocol 4.46. The common input is $(\mathbb{G}, q, g_0, g_1, \ldots, g_n) \leftarrow \mathcal{G}(1^\tau; n+1)$ and an $(n+1)$-tuple $(h_0, \ldots, h_n) \in (\mathbb{G})^{n+1}$. In the protocol, P proves knowledge of an exponent $x \in \mathbb{Z}_q$ such that $\log_{g_0} h_0 = x$ and $\log_{g_i} h_i \neq x$ for each $i = 1, \ldots, n$. The protocol is denoted in Camenisch-Stadler notation by $\mathrm{PK}\{\, x : h_0 = g_0^x \wedge (\bigwedge_{i=1}^n h_i \neq g_i^x) \,\}$.

We emphasize that each base g_i in our parallelized protocol must be selected *independently at random* from $\mathbb{G}^*$. In particular, to prove that the protocol is computationally zero-knowledge despite P reusing its random exponents in the announcement, it is necessary to assume that $\log_{g_i} g_j$ is not known to V for any $i \neq j$. One common way to enforce this 'lack of knowledge' on V is to have each g_i be output by a random oracle $\mathcal{H}_{\mathbb{G}}$. The common input to the protocol then contains not the sequence of g_i but, rather, a sequence of *preimages* of the g_i with respect to $\mathcal{H}_{\mathbb{G}}$. In Section 5.1, we will encounter an anonymous blacklisting system that uses this approach. Note that our protocol does not require any specific assumptions about how the h_i are selected or about P's (lack of) knowledge.

Protocol 4.46 (RME-based common-exponent Camenisch-Shoup protocol).

Common input: $(\mathbb{G}, q, g_0, \ldots, g_n) \leftarrow \mathcal{G}(1^\tau; n+1)$ and $(h_0, \ldots, h_n) \in (\mathbb{G})^{n+1}$
P's private input: $x \in \mathbb{Z}_q$ such that $h_0 = g_0^x$ and $h_i \neq g_i^x$ for each $i = 1, \ldots, n$

P1: P chooses $r \in_R \mathbb{Z}_q^*$ and then uses it to compute a commitment $A_i = (g_i^x / h_i)^r$ for each $i = 1, \ldots, n$. P announces $A = (A_1, \ldots, A_n)$ to V.

V2: V chooses a random exponent $t_i \in_R [0, 2^{\lambda_0} - 1]$ for each $i = 1, \ldots, n$, and then it sends $t = (t_1, \ldots, t_n)$ to P.

P3: P chooses $(\gamma_1, \gamma_2) \in_R (\mathbb{Z}_q)^2$, and then it computes the commitments $B_0 = g_0^{\gamma_1} h_0^{\gamma_2}$ and $B_1 = (\prod_{i=1}^n g_i^{t_i})^{\gamma_1} (\prod_{i=1}^n h_i^{t_i})^{\gamma_2}$. P announces $B = (B_0, B_1)$ to V.

V4: V issues a challenge $c \in_R [0, 2^{\lambda_0} - 1]$ to P.

P5: P computes the responses $v_1 = \gamma_1 - cxr$ and $v_2 = \gamma_2 + cr$, and then it sends $v = (v_1, v_2)$ to V.

V6: V accepts if $B_0 = g_0^{v_1} h_0^{v_2}$, if $B_1 = (\prod_{i=1}^n g_i^{t_i})^{v_1} (\prod_{i=1}^n h_i^{t_i})^{v_2} (\prod_{i=1}^n A_i^{t_i})^c$, and if $A_i \neq 1$ for each $i = 1, \ldots, n$.

Theorem 4.47. *The RME-based common-exponent variant of Camenisch and Shoup's protocol described in Protocol 4.46 is a system for conjunctive batch honest-verifier* computational *zero-knowledge proofs of knowledge for the inequality of DLs relations induced by* $\mathcal{G}_{\text{DL}}(1^*)$. *It is c-simulatable and 2-extractable and, therefore, has constant knowledge error function* $\kappa(\mathbb{G}, q, g_0, \ldots, g_n, h_0, \ldots, h_n) = 1/q$. *The protocol is computationally honest-verifier zero-knowledge under the decision Diffie-Hellman (DDH) assumption.*[31]

Proof (Sketch). Protocol 4.46 is easily seen to be complete and efficient by inspection; moreover, the above soundness argument for Figure 4.3 generalizes directly to prove that Protocol 4.46 is sound.[32] As the parallelized Chaum-Pedersen sub-protocol is itself a system for batch honest-verifier *perfect zero-knowledge* proofs of knowledge, all we must show is that a PPT simulator S_V for V can efficiently sample $A = (A_1, \ldots, A_n)$ from an ensemble of random variables that is

[31] Briefly, the *decision Diffie-Hellman (DDH) problem* in $\mathbb{G}$ is: Given as input a 4-tuple $(g_0, g_1, h_0, h_1) \in (\mathbb{G}^*)^2 \times (\mathbb{G})^2$, determine whether $\log_{g_0} h_0 = \log_{g_1} h_1$. The DDH assumption for a given group-generating algorithm $\mathcal{G}$ posits that no PPT algorithm can solve the DDH problem in groups $(\mathbb{G}, q, g_1, g_2) \leftarrow \mathcal{G}(1^\tau; 2)$ with advantage non-negligible in τ. A formal definition for the DDH assumption is provided in Appendix A.

[32] The latter requires that the RME-based parallelization of Chaum and Pedersen's protocol is sound. We proved a single-base analog of this result in Theorem 3.17; the proof for the above 2-DLREP version is nearly identical to that proof.

computationally indistinguishable from the ensemble describing P's first announcement in a real interaction transcript. Lemma 4.48 establishes that S_V can indeed sample according to such an ensemble, thus completing the proof Theorem 4.47. ∎

Lemma 4.48. *Fix a group-generating algorithm $\mathcal{G}$ and a positive integer-valued function $n(\tau) \in \mathrm{poly}(\tau)$, and, for each $\tau \in \mathbb{N}^+$, let $(\mathbb{G}, q, g) \leftarrow \mathcal{G}_{\mathrm{DL}}(1^\tau)$. Define the following two ensembles of random variables:*

$$\{X_\tau\}_{\tau\in\mathbb{N}^+} = \{g_0, \ldots, g_{n(\tau)}, H_0, \ldots, H_{n(\tau)}, g_0^x, (g_1^x/H_1)^r, \ldots, (g_{n(\tau)}^x/H_{n(\tau)})^r\}_{\tau\in\mathbb{N}^+},$$

where $g_i \in_R \mathbb{G}^*$ and $H_i \in \mathbb{G}$ for $i = 0, \ldots, n(\tau)$, and where $x \in_R \mathbb{Z}_q^*$, and

$$\{Y_\tau\}_{\tau\in\mathbb{N}^+} = \{g_0, \ldots, g_{n(\tau)}, H_0, \ldots, H_{n(\tau)}, g_0^x, (g_1^y/H_1)^r, \ldots, (g_{n(\tau)}^y/H_{n(\tau)})^r\}_{\tau\in\mathbb{N}^+},$$

where $g_i \in_R \mathbb{G}^*$ and $H_i \in \mathbb{G}$ for $i = 0, \ldots, n(\tau)$, and where $x, y \in_R \mathbb{Z}_q^*$. The ensembles $\{X_\tau\}_{\tau\in\mathbb{N}^+}$ and $\{Y_\tau\}_{\tau\in\mathbb{N}^+}$ are computationally indistinguishable under the DDH assumption.

Proof (Sketch). Suppose D is a PPT algorithm that distinguishes between $\{X_\tau\}_{\tau\in\mathbb{N}^+}$ and $\{Y_\tau\}_{\tau\in\mathbb{N}^+}$ with advantage non-negligible in τ. We exhibit a PPT algorithm $\mathcal{A}$ that, given oracle access to D, solves the DDH problem in $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ with advantage non-negligible in τ.

$\mathcal{A}$ works as follows: On input $(\mathbb{G}, q, g_0, g_1) \leftarrow \mathcal{G}(1^\tau; 2)$ and $(h_0, h_1) \in (\mathbb{G})^2$, choose $s_j \in_R \mathbb{Z}_q^*$ for each $j = 0, \ldots, n(\tau)$, and then set $g_0' = g_0^{s_0}$ and $h_0' = h_0^{s_0}$ and, for each $j = 1, \ldots, n(\tau)$, set $g_j' = g_1^{s_j}$ and $h_j' = h_1^{s_j}$. Next, choose $r \in_R \mathbb{Z}_q^*$ and, for each $j = 1, \ldots, n(\tau)$, choose $H_i \in \mathbb{G}$ arbitrarily. Now, compute the tuple

$$(g_0', \ldots, g_{n(\tau)}', H_0, \ldots, H_{n(\tau)}, h_0', (h_1'/H_1)^r, \ldots, (h_{n(\tau)}'/H_{n(\tau)})^r),$$

which, as is easily seen by inspection, is distributed according to X_τ if $\log_{g_0} h_0 = \log_{g_1} h_1$ and is distributed according to Y_τ if $\log_{g_0} h_0 \neq \log_{g_1} h_1$. $\mathcal{A}$ then passes this tuple to D for a verdict. ∎

P's expected cost in Protocol 4.46 is no more than

$$(n+2)\,\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) + 2\,\mathrm{ExpCost}_{\mathbb{G}}((n,\lambda_0)) < (n+2)(2\tau + \lambda_0)$$

multiplications in $\mathbb{G}$, and V's expected cost is

$$\begin{aligned}\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) + 2\,\mathrm{ExpCost}_{\mathbb{G}}((n,\lambda_0)) \\ + \mathrm{ExpCost}_{\mathbb{G}}((2,\tau),(1,\lambda_0)) < 4\tau + (2n+3)\lambda_0/2\end{aligned}$$

multiplications in $\mathbb{G}$. For communication cost, P sends $n+2$ elements from $\mathbb{G}^*$ and two elements from $\mathbb{Z}_q$ to V, and V sends $n+1$ λ_0-bit integers to P.

4.4.3.2 Proving equality among exactly k-out-of-n DLs

We now present a new protocol that combines Protocol 4.19 with Protocol 4.46 to implement a system for batch proofs of knowledge and equality of exactly k-out-of-n DLs. The common input to P and V is $(\mathbb{G}, q, g_0, \ldots, g_n) \leftarrow \mathcal{G}(1^\tau; n+1)$ and an $(n+1)$-tuple $(h_0, \ldots, h_n) \in (\mathbb{G})^{n+1}$, and the private input to P is an exponent $x \in \mathbb{Z}_q^*$ and a subset $H \subseteq_k [1,n]$ such that $h_0 = g_0^x$ and $h_j = g_j^x$ if and only if $j \in H$. The protocol is denoted in Camenisch-Stadler notation by $\mathrm{PK}\{(x,H) : h_0 = g_0^x \wedge H \subseteq_k [1,n] \wedge (\bigwedge_{j\in H} h_j = g_j^x) \wedge (\bigwedge_{i\in[1,n]\setminus H} h_i \neq g_i^x)\}$.

The idea is for P to prove knowledge of $x = \log_{g_0} h_0$ while simultaneously proving that x is (i) equal to k-out-of-n of the other DLs (a monotone statement), and (ii) not equal to $(n-k)$-out-of-n of the other DLs (another monotone statement), thus establishing that x must be equal to *exactly* k-out-of-n of the other DLs (a non-monotone statement).

To begin, we focus on the second (inequality) statement. In the first step of the conjunctive protocol, in which $k = 0$, P computes commitments of the form $A_i = (g_i^x/h_i)^r$ for each $i = 1, \ldots, n$. We already observed that $A_i = 1$ if and only if $h_i = g_i^x$; thus, having P reveal the resulting $A = (A_1, \ldots, A_n)$ to V would also reveal the subset H to V; on the other hand, it is necessary for V to check that $A_i \neq 1$ for each $i \in [1,n] \setminus H$. Our solution to this dilemma is to let $A_i = (g_i^x/h_i)^r$ as usual for the $i \in [1,n] \setminus H$, and then, for each $j \in H$, to let $A_j = g_0^{r_j}$ for a uniform random choice of $r_j \in_{\mathrm{R}} \mathbb{Z}_q^*$. Intuitively, this prevents V from learning H by merely searching for the subset of 1s in the announcement $A = (A_1, \ldots, A_n)$.

Given the announcement $A = (A_1, \ldots, A_n)$ constructed above, we now consider the standard RME-based verification equation: $B_1 = (\prod_{i=1}^n g_i^{t_i})^{v_1} (\prod_{i=1}^n h_i^{t_i})^{v_2} (\prod_{i=1}^n A_i^{t_i})^c$ with each $t_i \in_{\mathrm{R}} \mathbb{Z}_\rho$. In particular, we note that

$$\begin{aligned}\prod_{i=1}^n A_i^{t_i} &= (\prod_{j\in H} A_j^{t_j})(\prod_{i\in\overline{H}} A_i^{t_i}) \\ &= (\prod_{j\in H} g_0^{r_j t_j})(\prod_{i\in\overline{H}} (g_i^x / h_i)^{r t_i}) \\ &= g_0^{\sum_{j\in H} r_j t_j} (\prod_{i\in\overline{H}} g_i^{t_i})^{xr} (\prod_{i\in\overline{H}} h_i^{t_i})^{-r}.\end{aligned}$$

Now, using the standard responses $v_1 = \gamma_1 - cxr$ and $v_2 = \gamma_2 + cr$, we additionally note that $g_j^{v_1} h_j^{v_2} = g_j^{\gamma_1} h_j^{\gamma_2}$ if and only if $j \in H$; hence, we can rewrite

$$(\prod_{i=1}^n g_i^{t_i})^{v_1} (\prod_{i=1}^n h_i^{t_i})^{v_2} = \Big((\prod_{i=1}^n g_i^{t_i})^{\gamma_1} (\prod_{i=1}^n h_i^{t_i})^{\gamma_2}\Big)\Big((\prod_{i\in\overline{H}} g_i^{t_i})^{-cxr} (\prod_{i\in\overline{H}} h_i^{t_i})^{cr}\Big), \tag{4.7}$$

where the index i in the latter two products ranges only over the (secret) subset $\overline{H} = [1,n] \setminus H$; furthermore, as the pair (v_1, v_2) additionally satisfies $g_0^{v_1} h_0^{v_2} = g_0^{\gamma_1} h_0^{\gamma_2}$, Lemma 3.5 yields the following result.

Lemma 4.49. *If $H \neq \{i \in [1,n] \mid h_i = g_i^x\}$, then Equation (4.7) holds with probability at most $1/\rho$.*

However, if $H = \{i \in [1,n] \mid h_i = g_i^x\}$, then Equation (4.7) always holds, and we find that

$$(\prod_{i=1}^n g_i^{t_i})^{v_1} (\prod_{i=1}^n h_i^{t_i})^{v_2} (\prod_{i=1}^n A_i^{t_i})^c = g_0^{\zeta} (\prod_{i=1}^n g_i^{t_i})^{\gamma_1} (\prod_{i=1}^n h_i^{t_i})^{\gamma_2},$$

where $\zeta = \sum_{j\in H} r_j t_j$ is a random linear equation in $|H|$ unknowns, which is induced by the subsequence of short exponents $(t_j)_{j\in H}$ indexed by H. Hence, all that remains is a way for P to prove knowledge of $H \subseteq_k [1,n]$, and $r_j \in \mathbb{Z}_q^*$ for each $j \in H$, such that $\zeta = \sum_{j\in H} t_j r_j$ in the above expression. We can implement this latter proof using an RME-based common-base (k,n)-threshold variant of Schnorr's protocol with common input $A = (A_1, \ldots, A_n)$. Protocol 4.50 assembles these ideas into a complete protocol.

Protocol 4.50 (RME-based exactly-k-out-of-n batch Camenisch-Shoup protocol).

Common input: $(g_0,\ldots,g_n) \leftarrow \mathcal{G}(1^\tau; n+1)$ and $(h_0,\ldots,h_n) \in (\mathbb{G})^{n+1}$
P's private input: $x \in \mathbb{Z}_q$ and $H \subseteq_k [1,n]$ with $h_0 = g_0^x$ and $h_j = g_j^x$ if and only if $j \in H$

P1: Let $\overline{H} = [1,n] \setminus H$. P chooses a sub-challenge $c_i \in_{\mathrm{R}} \mathbb{Z}_\rho$ for each $i \in \overline{H}$, and then it computes $(A_0, \gamma) \leftarrow \mathrm{Com}((c_i)_{i\in\overline{H}})$. Next, P chooses $r \in_{\mathrm{R}} \mathbb{Z}_q^*$ and, for each $j \in H$, it chooses $r_j \in_{\mathrm{R}} \mathbb{Z}_q^*$. Finally, for each $i = 1,\ldots,n$, P computes the announcement

$$A_i = \begin{cases} (g_i^x/h_i)^r & i \in \overline{H} \\ g_0^{r_i} & i \in H \end{cases}$$

P announces $A = (A_0, A_1, \ldots, A_n)$ to V.

V2: V chooses a random exponent $t_i \in_{\mathrm{R}} [0, 2^{\lambda_0} - 1]$ for each $i = 1,\ldots,n$, and then it sends $t = (t_1,\ldots,t_n)$ to P.

P3: For each $i \in \overline{H}$, P computes $a_i = c_i t_i \bmod \rho$, and then it chooses $(\gamma_0,\gamma_1,\gamma_2) \in_{\mathrm{R}} (\mathbb{Z}_q^*)^3$, and computes $B_0 = g_0^{\gamma_1} h_0^{\gamma_2}$ and $B_1 = g_0^{\gamma_0} (\prod_{i\in\overline{H}} g_i^{a_i})^{\gamma_1} (\prod_{i\in\overline{H}} h_i^{a_i})^{\gamma_2}$. P sends $B = (B_0, B_1)$ to V.

V4: V issues a challenge $c \in_{\mathrm{R}} \mathbb{Z}_\rho$ to P.

P5: P uses polynomial interpolation to compute the degree-$(n-k)$ polynomial $f \in \mathbb{Z}_\rho[x]$ passing through $(0,c)$ and through each point in $\{(i,c_i)\}_{i\in\overline{H}}$, and then it computes $a_j = t_j f(j) \bmod \rho$ for each $j \in H$, the responses $v_0 = \gamma_0 - c\sum_{j\in H} r_j a_j$, $v_1 = \gamma_1 - cxr$, and $v_2 = \gamma_2 + cr$, and the opening $\pi \leftarrow \mathrm{Open}(A_0, \gamma, f(1),\ldots,f(n))$ of A_0. P sends $v = (v_0, v_1, v_2, \pi, f)$ to V.

V6: V constructs the exponents $a_i = t_i f(i) \bmod \rho$ for each $i = 1,\ldots,n$. V accepts if $\mathrm{Ver}(A_0, \pi, f(1),\ldots,f(n)) = 1$, if $\deg f \le n-k$ with $c = f(0)$, if both $B_0 = g_0^{v_1} h_0^{v_2}$ and $B_1 = g_0^{v_0} (\prod_{i=1}^n g_i^{a_i})^{v_1} (\prod_{i=1}^n h_i^{a_i})^{v_2} (\prod_{i=1}^n A_i^{a_i})^c$, and if $A_i \neq 1$ for each $i = 1,\ldots,n$, and it rejects otherwise.

Theorem 4.51. *Let $\Gamma_{=k}$ be the language of finite bit strings with Hamming weight equal to $k \in \mathbb{N}^+$. The variant of Camenisch and Shoup's protocol described in Protocol 4.50 is a system for batch honest-verifier* computational *zero-knowledge proofs of $\Gamma_{=k}$-partial knowledge for the equality of DLs relations induced by $\mathcal{G}_{\mathrm{DL}}(1^*)$ (colloquially: a system for batch arguments of knowledge and equality of exactly-k-out-of-n DLs). The protocol is honest-verifier computational zero-*

knowledge under the DDH assumption for $\mathcal{G}$, and its arguments are computationally convincing under the SDH and polyDH assumptions for the bilinear group-generating algorithm employed by $KeyGen_{\Gamma_k}$.

P's expected cost in Protocol 4.50 is no more than

$$(n-k+2)\,\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) + k\,\mathrm{ExpCost}_{\mathbb{G}}(\tau) + 2\,\mathrm{ExpCost}_{\mathbb{G}}((n-k,\lambda_0)) < (n-k+2)(2\tau+\lambda_0) + k\lambda_0$$

multiplications in $\mathbb{G}$, plus about $2k\tau + n\lambda_0/2$ multiplications in $\widetilde{\mathbb{G}}$ and two pairing evaluations for the Γ_k-threshold mercurial commitment. V's expected cost is

$$\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) + 3\,\mathrm{ExpCost}_{\mathbb{G}}((n,\lambda_0)) + \mathrm{ExpCost}_{\mathbb{G}}((2,\tau),(1,\lambda_0)) < 4\tau + (3n+7)\lambda_0/2$$

multiplications in $\mathbb{G}$, plus about $(3\tau + n\lambda_0)/2$ multiplications in $\widetilde{\mathbb{G}}$, $3\tau/2$ multiplications in $\mathbb{G}_T$, and six pairing evaluations, for the Γ_k-threshold mercurial commitment. For communication cost, P sends $n+2$ elements from $\mathbb{G}^*$, five elements from $\widetilde{\mathbb{G}}$, three elements from $\mathbb{Z}_q$, and $n-k$ λ_0-bit integers to V, and V sends $n+1$ λ_0-bit integers to P.

4.4.3.3 Proving equality among at most k-out-of-n DLs

We can easily generalize Protocol 4.50 to prove knowledge and equality of *at most* k-out-of-n DLs. Again, the common input to P and V is $(\mathbb{G}, q, g_0, \ldots, g_n) \leftarrow \mathcal{G}(1^\tau; n+1)$ and an $(n+1)$-tuple $(h_0, \ldots, h_n) \in (\mathbb{G})^{n+1}$, and the private input to P is an exponent $x \in \mathbb{Z}_q^*$ and a subset $H \subseteq [1,n]$ with $|H| \leq k$ such that $h_0 = g_0^x$ and $h_j = g_j^x$ if and only if $j \in H$. The protocol is denoted in Camenisch-Stadler notation by $\mathrm{PK}\{(x,H) : h_0 = g_0^x \wedge H \subseteq [1,n] \wedge |H| \leq k \wedge (\bigwedge_{j\in H} h_j = g_j^x) \wedge (\bigwedge_{i\in[1,n]\setminus H} h_i = g_i^x)\}$. The idea is not to change the protocol, but, rather, to change the *input* to the protocol.

In particular, given a (n,τ)-predicate $\mathcal{I} = (\mathbb{G}, q, g_0, \ldots, g_n, h_0, \ldots, h_n)$ as common input, we permit P to "manufacture" an additional k "synthetic" component instances $(g_{n+i}, h_{n+i}) \in (\mathbb{G}^* \times \mathbb{G})$, for $i = 1, \ldots, k$, and then to append them to $\mathcal{I}$. The augmented input is therefore an

$(n+k,\tau)$-predicate $(\mathbb{G}, q, g_0, \ldots, g_{n+k}, h_0, \ldots, h_{n+k})$. By choosing the k synthetic inputs so that exactly k-out-of-$(n+k)$ DLs in the augmented batch predicate are equal to $x = \log_{g_0} h_0$, P can complete the proof exactly as in Protocol 4.50 without leaking any information about $|H|$ beyond that $|H| \leq k$. As the threshold k is always less than or equal to n, it follows that the augmented input predicate has fan-in at most twice that of the original input; hence, the resulting protocol is still asymptotically efficient.

Theorem 4.52. *Let $\Gamma_{\leq k}$ be the language of finite bit strings with Hamming weight less than or equal to $k \in \mathbb{N}^+$. The variant of Protocol 4.50 just described is a system for batch honest-verifier* computational *zero-knowledge proofs of $\Gamma_{\leq k}$-partial knowledge for the equality of DLs relations induced by $\mathcal{G}_{\text{DL}}(1^*)$ (colloquially: a system for batch arguments of knowledge and equality of at-most-k-out-of-n DLs). The protocol is honest-verifier computational zero-knowledge under the DDH assumption for $\mathcal{G}$, and its arguments are computationally convincing under the SDH and polyDH assumptions for the bilinear group-generating algorithm employed by $KeyGen_{\Gamma_{\cdot k}}$.*

4.4.3.4 Proving equality among between-k_1-and-k_2-out-of-n DLs

More generally, P can prove knowledge and equality of at least k_1-out-of-n and at most k_2-out-of-n DLs using the approach described in Section 4.4.3.3. All that needs to change is the number of inputs that we permit P to synthesize; in particular, to prove knowledge and equality of "between-k_1-and-k_2-out-of-n" DLs, it suffices to have P synthesize $k_2 - k_1$ inputs and then run Protocol 4.50 with threshold k_2. It is easy to confirm that P can issue an accepting response in such a protocol if and only if between k_1 and k_2 instances in the original predicate satisfy $\log_{g_0} h_0 = \log_{g_i} h_i$.

Theorem 4.53. *Let $\Gamma_{(k_1,k_2)}$ be the language of finite bit strings with Hamming weight at least $k_1 \in \mathbb{N}^+$ and at most $k_2 \in \mathbb{N}^+$. The variant of Protocol 4.50 just described is a system for batch honest-verifier* computational *zero-knowledge proofs of $\Gamma_{(k_1,k_2)}$-partial knowledge for the equality of DLs relations induced by $\mathcal{G}_{\text{DL}}(1^*)$ (colloquially: a system for batch arguments of knowledge and equality of between-k_1-and-k_2-out-of-n DLs). The protocol is honest-verifier computational zero-knowledge under the DDH assumption for $\mathcal{G}$, and its arguments are computationally convincing under the SDH and polyDH assumptions for the bilinear group-generating algorithm employed by $KeyGen_{\Gamma_{k_2}}$.*

4.4.3.5 Additional non-monotone access structures

Protocol 4.50 naturally generalizes to a system for batch proofs of Γ-partial knowledge over several other non-monotone languages Γ. For instance, given an arbitrary monotone access structure Δ_n that is suitable for use in Protocol 4.21, the "obvious" generalization of Protocol 4.50 using Δ_n yields a system for proofs of Γ-partial knowledge for the language Γ inducing Δ_n^- on $[1,n]$. By additionally employing the synthetic inputs idea described in Sections 4.4.3.3 and 4.4.3.4, it is possible to transform such a protocol into a system for batch proofs partial knowledge with respect to some additional non-monotone languages, although the precise class of languages that this approach can support is not clear. We leave further investigation along these lines as an exciting area for future research.

4.5 Chapter summary

This chapter discussed batch proofs of *partial* knowledge for linear relations. We revisited Peng and Bao's disjunctive batch variant of Chaum and Pedersen's protocol for proving knowledge of and equality among of 1-out-of-n DL pairs. Our analysis uncovered a critical flaw in that protocol and we devised a practical, lattice-based attack to exploit it. To defend against such attacks, we proposed trapdoor $\mathscr{L}$-mercurial commitments, a new kind of cryptographic commitments that generalize trapdoor n-mercurial commitments, and provided an efficient construction for the special case of $(n-k,n)$-threshold mercurial commitments. We then used these new commitments to generalize Peng and Bao's protocol to a system for batch honest-verifier zero-knowledge proofs of knowledge for any language inducing monotone access structures described by a given family of polynomial-sized Boolean formulas. Finally, we extended our approach for proofs of partial knowledge over monotone access structures to work over certain *non-monotone* access structures.

Chapter 5

Example applications and constructions

The introduction listed several constructions for privacy-enhancing technologies (PETs), each of which leverages one or more high-fan-in zero-knowledge proofs of knowledge regarding DLs in prime-order groups. As such, those constructions can each be sped up using the batch zero-knowledge proof techniques introduced so far in this dissertation. Readers interested in learning more about the role of batch zero-knowledge proof techniques in the construction of PETs are encouraged to peruse the author's papers listed in the List of Related Publications. For instance, the author's work on *priced symmetric private information retrieval* (PSPIR) [HOG11, HHG13] demonstrates how batch zero-knowledge proofs of knowledge can yield PET constructions that are both conceptually simpler and more efficient than competing approaches, while simultaneously making it easier to implement novel additional functionality atop the basic construction. Likewise, the technical report that introduced $(n - k, n)$-threshold mercurial commitments discusses how batch zero-knowledge proofs of knowledge can lead to more efficient protocols for *coercion-resistant Internet voting* [CH11].

In this chapter, we describe in detail applications of our new batch techniques to three constructions for *anonymous blacklisting and reputation systems*.

5.1 Anonymous blacklisting and reputation systems

Anonymous communications systems provide Internet users with a means to access services over the public Internet while concealing their identities and usage patterns from prying eyes. The most popular such system in deployment is Tor [DMS04], a worldwide-distributed overlay network comprising some 5,500 volunteer-operated relays that forward encrypted traffic on behalf of its users [Torb]. Using layered encryption with multiple levels of indirection, Tor seeks to defend its users against *traffic analysis attacks* [BMS01], wherein an attacker leverages its privileged position on the network to surreptitiously monitor the actions of certain users.

People from all walks of life leverage the anonymity provided by Tor in order to circumvent online censorship, to research taboo and unpopular subjects, to report human rights violations, to organize politically, and to speak their minds without fear of retaliation. Indeed, Tor currently helps to protect the privacy of nearly one million privacy-conscious Internet users located in hundreds of countries around the world each day [Torc]. Not only is this a win for online privacy and free speech, but it is also a potential boon for many online communities that might benefit from added diversity in their respective user populations. Compelling examples of such online communities include collaborative encyclopedias like Wikipedia[32] and community-driven review sites like Yelp[33].

Yet reality is rarely so simple. The providers of such online services must ultimately weigh the expected benefits (both to themselves and to their user communities) of more inclusivity against the risks posed by abusive users, especially those who would hide behind the veil of anonymity to skirt accountability for their actions. A number of popular services — notably including Wikipedia, Yelp, Slashdot[34], Cloudflare[35], Craigslist[36], 4chan[37], Vimeo [38], GitHub [39], and some IRC networks [Tora] — presently block or limit participation by anonymous users, despite the implied loss of diversity and the broader implications for free speech and the open exchange of knowledge and ideas.

[32] https://en.wikipedia.org/
[33] https://www.yelp.com/
[34] https://slashdot.org/
[35] https://www.cloudflare.com/
[36] https://www.craigslist.org/
[37] https://www.4chan.org/
[38] https://vimeo.com/
[39] https://github.com/

In response, the cryptographic and privacy research communities have proposed several *anonymous blacklisting* designs, which seek to provide mechanisms through which service providers (SPs) may hold anonymous users accountable for their individual actions *without threatening those users' anonymity*. SPs can thereby protect their user communities from abuse by the occasional "naughty" anonymous user without inflicting collateral damage on all the "nice" users. An early proposal called Nymble [JKTS07] solved the anonymous blacklisting problem both elegantly and efficiently; however, Nymble and its progeny [LH10, LH11, HHG10] rely on powerful trusted third parties (TTPs) that can deanonymize (or at least link) users' connections undetectably and at will. Subsequent designs [TAKS10, ATK11, BL12] have made clever use of zero-knowledge proofs of knowledge to eliminate the need for TTPs, thus solving the trust problem at a cost of much computation and communication overhead, both for the users and for the SPs.

5.1.1 Blacklistable anonymous credentials

One such TTP-free anonymous blacklisting design is Tsang, Au, Kapadia, and Smith's *Blacklistable Anonymous Credentials* (BLAC) system [TAKS07]. In BLAC, a semi-trusted *group manager* (GM) registers each new user into the system by issuing it an anonymous credential $C(x)$ that encodes as an attribute a secret identity key x, which is unique to, and known only by, that particular user.[41] The user holding $C(x)$ authenticates to an SP by first choosing a random generator g_0 from a given prime-order group $\mathbb{G}$, and then using it to produce a *ticket* $\mathcal{T} = (g_0, g_0^x)$ together with a zero-knowledge proof asserting that

1. the user holds a valid credential $C(x)$ from the GM,
2. the exponent x in $\mathcal{T} = (g_0, g_0^x)$ is the same as the secret identity key x in $C(x)$, and
3. no ticket associated with any past abusive session uses the identity key x.

To facilitate the third part of the proof, the SP maintains a public *blacklist* of tickets associated with any as-of-yet-unforgiven abusive sessions.

The SP grants the user access (and stores $\mathcal{T}$ for future reference) if and only if it accepts in the above zero-knowledge proof of knowledge. If the SP later deems the user's actions during the session to have been abusive, then it can place $\mathcal{T}$ on its blacklist, thus curtailing any further

[41] We emphasize that the GM learns nothing about x in this exchange; indeed, although the users and the SPs must trust the GM if the system is to provide availability and accountability, the GM is only semi-trusted in the sense that the users need not trust it to maintain their anonymity.

abuse by that user. Note that the SP here does not learn the secret identity key x in $\mathcal{T}$, nor does it learn anything to help it link $\mathcal{T}$ with other tickets that use the same identity key; hence, the user holding $C(x)$ remains completely anonymous yet, going forward, it is unable to authenticate (until the SP removes $\mathcal{T}$ from its blacklist).

The notion of "abuse" in this model is entirely subjective: each SP must define, identify, and penalize abusive behaviour in a way that is appropriate within the context of its user community and the services it provides. For instance, the SPs comprising an IRC network may collectively define "abuse" to include acts of hate speech, cyberbullying, circumventing bans, spamming, or copyright infringement. Any ticket associated with an abusive act can then be placed on the blacklist at the server operators' discretion, and left there for a duration commensurate both with the severity of the abuse and with the perceived likelihood that its perpetrator will re-offend. Each server that comprises the IRC network must require all anonymous users to authenticate, and they must refuse connection requests from those users that fail to prove they are not on the blacklist.[42]

5.1.2 Variants of BLAC

BLAC's all-or-nothing approach to revocation may be overly punitive in some settings. The anonymous blacklisting literature includes two additional variants of BLAC that seek to address this shortcoming. The first variant does so with a *d-strikes-out revocation policy* [TAKS10], wherein each anonymous user may continue to authenticate until it has accumulated d or more tickets on the blacklist (after which further authentications are not possible). The second variant supports *reputation-based blacklisting* [AKS12], wherein SPs can assign scores (both positive and negative) to the anonymous actions of users, and each user can subsequently authenticate only if the aggregate score associated with all of its scored tickets exceeds some minimum threshold

[42] An alternative to such *subjective revocation* is *objective* (or *contract-based*) *revocation* [HG11b; §IV.A], introduced by Schwartz, Brumley, and McCune and exemplified by their trusted hardware-based RECAP protocol [SBM10]. In the objective revocation model, users and SPs enter into mutually binding *contracts* that stipulate unambiguously the SPs' terms of service. An SP can then revoke a given anonymous user's authentication privileges if and only if that user *provably* violates the terms set forth in its contract with the SP. Unfortunately, some perfectly reasonable terms of service are far too nebulous to specify and enforce without some degree of human subjectivity. For example, consider a contract that forbids hoaxes on Wikipedia or one that forbids disingenuous reviews on Yelp — in both examples, identifying contract violations seems to necessitate a human in the loop.

value. (More generally, SPs may categorize positive and negative scores according to the nature of the associated behaviours and require each authenticating user to prove a statement pertaining to its aggregate scores *across all categories* of scored behaviours.) We herein refer to the first variant as 'd-BLAC' and to the second variant as 'BLACR'. We refer to the original system simply as 'BLAC' or, occasionally, as 'vanilla BLAC' to emphasize when a remark applies to BLAC but not to d-BLAC or to BLACR.

5.1.3 The scalability problem

BLAC and its variants provide very strong privacy guarantees under a standard cryptographic assumption (namely, the DDH assumption); however, the BLAC approach to anonymous blacklisting suffers from some scalability problems. In all three of the above BLAC variants, the bottleneck operation is the portion of the zero-knowledge proof of knowledge in which the user demonstrates that its own tickets on an SP's blacklist (if it has any) do not meet that SP's criteria for revocation. The fan-in of this proof scales as the total blacklist length, which can introduce substantial delays and consume considerable bandwidth and computation capacity for large SPs that must cater to millions of anonymous users.

Prior work [TAKS07, TAKS10, AKS12] essentially regards the zero-knowledge proofs as black boxes, to be instantiated using the appropriate naive variants of Schnorr's protocol. Such an approach is prohibitively expensive even for moderate-sized blacklists (say, those containing a few hundred tickets), which has contributed to the common conception [AK12, ATK11, HG11b] that — despite being both novel and elegant — BLAC's approach to anonymous blacklisting is impractical at Internet scale.

This section outlines how to significantly reduce both the communication and computation overhead of BLAC, d-BLAC, and BLACR using the batch zero-knowledge proof techniques presented in the earlier chapters. In particular, we observe that each of BLAC, d-BLAC, and BLACR may be efficiently implemented using the techniques we developed in Section 4.4.3

5.1.4 Threat model and design goals

Our own contributions to BLAC and its variants are contained entirely within the black boxes implementing their respective bottleneck zero-knowledge proofs of knowledge. Nonetheless, we shall find it useful to provide some additional detail on the high-level design, if only for completeness. (In fact, some aspects of the design are relevant in our security analyses.) We claim no originality here; the BLAC design is due entirely to Tsang et al. [TAKS10] and any differences in our presentation of it are purely cosmetic.

The basic setting is exactly as described above: A community of anonymous users wish to use the services offered by one or more participating SPs and the SPs, in turn, will only service those users whom the SPs can hold individually accountable for their respective actions. The SPs do this by each maintaining a *blacklist* of tickets from (and possibly other meta-data about) past abusive sessions. The SPs then require each authenticating user to prove that it is not responsible for "too many" of those abuses. (The precise definition of "too many" abuses, of course, varies from vanilla BLAC to d-BLAC to BLACR.) A semi-trusted (and possibly distributed) GM facilitates all of this.

Initialization. The GM runs the *initialization protocol* once to set up the system. On input a security parameter 1^τ, it outputs (i) $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\tau)$ for some group-generating algorithm $\mathcal{G}$ with respect to which the DDH assumption holds, (ii) a random oracle $\mathcal{H}_{\mathbb{G}}$ mapping every finite bit string $t \in \{0,1\}^*$ to a uniform random generator $\mathcal{H}_{\mathbb{G}}(t)$ in $\mathbb{G}^*$, and (iii) a public-private key pair (pk, sk) for anonymous credentials [CL02]. In an actual implementation, $\mathcal{H}_{\mathbb{G}}$ would be a cryptographically secure hash function and $\mathbb{G}$ would be an elliptic curve group that can be efficiently "hashed into" [Ica09]. Each of the remaining algorithms takes the tuple $(\mathbb{G}, q, g, \mathcal{H}_{\mathbb{G}}, pk)$ as an *implicit* input.

Registration. Each user runs the interactive *registration protocol* once with the GM to enroll in the system. Upon successful completion of the protocol, the user obtains an anonymous credential $C(x)$ under the GM's public key pk. The credential $C(x)$ encodes a secret, uniform random *identity key* $x \in_{\mathrm{R}} \mathbb{Z}_q^*$, which serves as a sort of secret "pseudonym" for the user. We emphasize that the GM learns *zero information* about x during this interaction and it therefore has no computational advantage in, for example, linking tickets (g_0, g_0^x) to the particular registration

that issued $C(x)$. The choice of anonymous credential system is immaterial to the following protocols, provided credentials in it (i) hide the secret identity key x unconditionally, (ii) are fully anonymous (that is, distinct showings of the same credential are mutually unlinkable), and (iii) admit *efficient* honest-verifier perfect zero-knowledge proofs of knowledge of the identity key x. (Note that *unconditional* hiding and *perfect* zero-knowledge are not strictly required, but they do help to simplify the security analysis.)

The authors of BLAC and its derivatives suggest using *BBS+ signatures* [ASM06; §4.2] for the above anonymous credentials. BBS+ signatures satisfy each of the above criteria and are computationally binding under the SDH assumption. As our new protocols do not affect registration, we do not discuss it any further. Interested readers can consult the relevant sections of Tsang et al.'s paper for the full details [TAKS10; §4.1.2 and §5.3].

Trust in the GM. Despite its learning nothing about the user's secret identity key x during registration, both the users and the SPs must trust the GM to some extent. For instance, SPs must trust the GM to issue at most one credential to any given user, lest some users obtain several credentials with which to launch *Sybil attacks* [Dou02] against the SPs. A reliable GM must consequently collect (and retain, in some form) *personally identifiable information* (PII) about each enrolled user; those users, in turn, must trust the GM to act responsibly with that PII.[43]

Authentication. The *authentication protocol* is an interactive protocol that an anonymous user runs with a participating SP in order to initiate an anonymous session. The user's private input is its secret identity key x. The common input consists of (i) the user's (freshly re-randomized) credential $C(x)$, (ii) a user-selected nonce $t \in_R \{0,1\}^\tau$, (iii) a *canonical name* $s \in \{0,1\}^*$ for the SP, (iv) the SP's current *blacklist* β, (v) the SP's *revocation policy* P, and (vi) a *soundness parameter* $\lambda_0 \in \mathbb{N}^+$. One can think of the revocation policy as a Boolean-valued function that, on input the SP's current blacklist β and the authenticating user's secret identity x, outputs 1 if the entries on β with tickets encoding x meet the SP's revocation criteria and outputs 0 otherwise. The SP will accept the authentication only if the user convinces it that $P(\beta, x) = 0$. The output of the authentication protocol is a return value $b \in \{0, 1, \perp\}$ and an *authentication transcript* containing the *ticket* $\mathcal{T} = (t, \mathcal{H}_{\mathbb{G}}(t\|s)^x)$. A return value of 0 indicates that the SP rejected the authentication and a return value of 1 indicates that the SP accepted the authentication. A return

[43] The author's survey of anonymous blacklisting systems discusses this point in more detail [HG11b; §V].

value of $\perp$ indicates that the user aborted the protocol prematurely (perhaps because it discovered during the interaction that $P(\boldsymbol{\beta}, x) = 1$). The SP should output $b = 1$ with probability at most $2^{-\lambda_0}$ when $P(\boldsymbol{\beta}, x) = 1$.

Blacklist management. Blacklist management involves three protocols that SPs use to manage their respective blacklists. The *extraction protocol* takes as input an authentication transcript T and it outputs the associated *ticket* $\mathcal{T}$. The *revocation protocol* takes as input a blacklist $\boldsymbol{\beta}$ and a ticket $\mathcal{T}$ (and, in the case of reputation-based blacklisting, an associated score $\varsigma \in \mathbb{Z}$), and it outputs a new blacklist $\boldsymbol{\beta}^+$ that contains every entry from $\boldsymbol{\beta}$ plus a new entry for ticket $\mathcal{T}$ (with score ς). The *pardon protocol* takes as input a blacklist $\boldsymbol{\beta}$ and a ticket $\mathcal{T}$, and it outputs a new blacklist $\boldsymbol{\beta}^-$ that contains every entry from $\boldsymbol{\beta}$ whose ticket is not equal to $\mathcal{T}$.

5.1.5 Security definitions

We sketch (informal) definitions for the necessary security and privacy properties of a *secure BLAC construction* below; interested readers should consult Tsang et al. [TAKS10] for the formal versions. The informal definitions suffice for our purposes: since we only replace zero-knowledge protocols inside of black boxes with more efficient protocols providing the same security guarantees, the existing system-level security proofs for BLAC and d-BLAC [TAKS10; §7.2] and for BLACR [AKS12; Appendix A] also prove that our batch protocols each yield secure BLAC constructions.

1. *Completeness:* If the GM and a given SP are both honest, and if a given user's entries on that SP's blacklist do not meet its revocation criteria, then the user can successfully authenticate to the SP.
2. *Misauthentication resistance:* A user can authenticate to an honest SP only if that user holds a valid credential $C(x)$ that was issued by the GM.
3. *Blacklistability:* A coalition of dishonest SPs and users holding credentials $C(x_1), \ldots, C(x_k)$ can successfully authenticate to an honest SP having blacklist $\boldsymbol{\beta}$ only if $P(\boldsymbol{\beta}, x_i) = 0$ for some $i \in [1, k]$.

4. *Anonymity:* No coalition of dishonest SPs, users, and the GM can distinguish authentication transcripts associated with the same honest user from those associated with different honest users. Moreover, no such coalition can link any given authentication transcript with the registration in which the GM issued the associated credential.
5. *Non-frameability:* No coalition of dishonest SPs, users, and the GM can prevent an honest user from successfully authenticating with an honest SP.

5.1.6 Batch BLAC constructions

We now discuss how to improve the efficiency of each of the above three BLAC variants using the batch zero-knowledge proof techniques presented in the preceding chapters.

5.1.6.1 Batch vanilla BLAC

The simplest of the three authentication protocols is, of course, the one arising in vanilla BLAC. In the vanilla BLAC authentication protocol, the user outputs a ticket $\mathcal{T} = (t, \mathcal{H}_{\mathbb{G}}(t\|s)^x)$ and engages the SP in a zero-knowledge proof asserting that (i) it holds a valid credential $C(x)$ from the GM, (ii) the exponent x in $\mathcal{T}$ is the same as the secret identity key x in $C(x)$, and (iii) no ticket on the SP's blacklist $\boldsymbol{\beta}$ also uses the exponent x. Note that the cost of the first two steps does not depend on the contents of the blacklist and, moreover, the implementation details for those steps are dependent on the particular choice of anonymous credential system; thus, we focus our attention on the more costly (and credential-agnostic) portion of the protocol.

Suppose the SP's blacklist is $\boldsymbol{\beta} = ((t_1, \mathcal{H}_{\mathbb{G}}(t_1\|s)^{x_1}), \ldots, (t_n, \mathcal{H}_{\mathbb{G}}(t_n\|s)^{x_n}))$. We assume that the nonces $t_i \in \{0,1\}^\tau$ occurring in $\boldsymbol{\beta}$ are pairwise distinct so that, with probability overwhelming in τ, each $\mathcal{H}_{\mathbb{G}}(t_i\|s) \in_{\mathrm{R}} \mathbb{G}^*$ is a distinct, uniform random generator of $\mathbb{G}$. (If $\mathcal{H}_{\mathbb{G}}(t_i\|s) = \mathcal{H}_{\mathbb{G}}(t_j\|s)$ for some $i \neq j$, then the user should abort.) For ease of exposition, we rewrite each ticket $(t_i, \mathcal{H}_{\mathbb{G}}(t_i\|s)^{x_i})$ in the form $\mathcal{T} = (g_i, h_i)$, where $g_i = \mathcal{H}_{\mathbb{G}}(t_i\|s)$ and $h_i = g_i^{x_i}$. Then, by our assumptions on the t_i and $\mathcal{H}_{\mathbb{G}}$, Lemma 4.48 applies and we can implement this proof using Protocol 4.46 with common input $\mathcal{I} = ((g_0, h_0), (g_1, h_1), \ldots, (g_n, h_n))$.

In fact, we can do slightly better than Lemma 4.48 suggests: since the first component of $\mathcal{I}$ is output by the user as part of the authentication protocol, and since the user completes the first two steps of that protocol using a *perfect zero-knowledge* proof of knowledge, it follows that

a simulator for the full authentication protocol can choose $t_0 \in_R \{0,1\}^\tau$ and a "fake" identity key $y \in_R \mathbb{Z}_q^*$ arbitrarily, and then output the "simulated" ticket (g_0, h_0), where $g_0 = \mathcal{H}(t_0 \| s)$ and $h_0 = g_0^y$, together with a *perfectly simulated proof* that y is the secret identity key in $C(x)$. (That the latter proof of knowledge is indeed perfectly simulatable for an arbitrary choice of $y \in \mathbb{Z}_q^*$ follows from the fact that $C(x)$ is unconditionally hiding.) Now, because such a simulator knows the exponent y used to compute (g_0, h_0), it can follow Protocol 4.46 *honestly* to complete a perfect simulation for the entire authentication protocol. We summarize the results just sketched in a theorem.

Theorem 5.1. *Using Protocol 4.46 to implement the zero-knowledge proof that $P(\boldsymbol{\beta}, x) = 0$ in vanilla BLAC's authentication protocol yields a secure BLAC construction under the DDH assumption for $\mathcal{G}$ and the SDH assumption for the group-generating algorithm used for BBS+ signatures.*

Comparison with the original vanilla BLAC protocol. The creators of BLAC suggested instantiating the above protocol using a naive parallelization of Camenisch and Shoup's protocol. In such a naive parallelization, the user's expected cost is

$$n \operatorname{ExpCost}_{\mathbb{G}}^{(3)}((2,\tau)) + \operatorname{ExpCost}_{\mathbb{G}}^{(2)}((2,\tau)) < 2(3n+2)\tau$$

multiplications in $\mathbb{G}$, and the SP's expected cost is

$$\operatorname{ExpCost}_{\mathbb{G}}((2,\tau)) + n \operatorname{ExpCost}_{\mathbb{G}}((3,\tau)) < (3n+2)\tau$$

multiplications in $\mathbb{G}$. For communication cost, the user sends $3n + 2$ elements from $\mathbb{G}^*$ and $2n$ elements from $\mathbb{Z}_q$ to the SP, and the SP sends one $\mathbb{Z}_q$ element to the user.

Referring to the cost analysis for Protocol 4.46 at the end of Section 4.4.3.1, we observe that our own protocol reduces these costs substantially. For instance, setting $\tau = 256$ and $\lambda_0 = 40$ and letting n grow large, we find that the expected cost for the user in Protocol 4.46 approaches $\approx 36\%$

of its expected cost in the naive protocol; likewise, the expected cost for the SP in Protocol 4.46 approaches $\approx 6\%$ that of its expected cost in the naive protocol. Similarly, the communication overhead in Protocol 4.46 approaches $\approx 25\%$ that of the naive protocol.[44]

Most of the user's remaining cost in Protocol 4.46 arises from computing the first announcement $A = (A_1, \ldots, A_n)$. We note that, given an up-to-date copy of the SP's blacklist $\boldsymbol{\beta}$, the user can readily *precompute* these values [TAKS10; §7.1]; therefore, we should also consider the user's expected *online* cost when all (or essentially all) of the announcement $A = (A_1, \ldots, A_n)$ has been precomputed. Indeed, excluding the cost of computing $A = (A_1, \ldots, A_n)$, we find that the expected online cost for the user reduces to just

$$2\,\mathrm{ExpCost}_{\mathbb{G}}((2,\tau)) + 2\,\mathrm{ExpCost}_{\mathbb{G}}((n,\lambda_0)) < 4\tau + (n+2)\lambda_0$$

multiplications in $\mathbb{G}$. If we let $\tau = 256$ and $\lambda_0 = 40$, as before, then the expected online cost for the user in Protocol 4.46 is only $\approx 2.6\%$ of its cost in the naive protocol.

Moreover, we note that once it has computed $A_i = (g_i^x / h_i)^r$ to use in one protocol run, the user can choose a new blinding factor $r' \in_{\mathrm{R}} \mathbb{Z}_q^*$ and *reblind* each A_i as $A_i^{r'} = (g_i^x / h_i)^{r \cdot r'}$ to use in a subsequent protocol run. Such reblinding requires just $n\,\mathrm{ExpCost}_{\mathbb{G}}(\tau) \leq 3\tau/2$ multiplications in $\mathbb{G}$, which is a little over half of what is required to compute a new sequence of A_i from scratch.

5.1.6.2 Batch *d*-BLAC

Next we discuss the authentication protocol arising in d-BLAC. Similar to in the vanilla BLAC authentication protocol, the user in the d-BLAC authentication protocol outputs a ticket $\mathcal{T} = (t, \mathcal{H}_{\mathbb{G}}(t \| s)^x)$, and then engages the SP in a zero-knowledge proof that (i) it holds a valid credential $C(x)$ from the GM, (ii) the exponent x in $\mathcal{T}$ is the same as the secret identity key x

[44] To arrive at this latter estimate, we assume that (i) $\mathbb{G}$ is an elliptic curve group whose elements are about $2\tau = 512$ bits long, and (ii) implementations use the Fiat-Shamir transform (see Section 2.2.5) to make Protocol 4.46 non-interactive in the random oracle model. The latter assumption implies that the SP need not send a challenge or any short exponents to the user; indeed, the user and the SP will each compute the challenge and all such short exponents locally using a cryptographically secure hash function. By using point compression for the elements of $\mathbb{G}$, the communication savings approach $\approx 20\%$ that of the naive protocol, at the cost of some additional computation overhead for point decompression.

in $C(x)$, and (iii) at most $d - 1$ tickets on the SP's blacklist $\boldsymbol{\beta}$ also use the exponent x. Again, the cost of the first two steps does not depend on the contents of the blacklist and we focus our attention on the more costly (and credential-agnostic) portion of the protocol.

Theorem 5.2. *Using the variant of Protocol 4.50 described in Section 4.4.3.3 to implement the zero-knowledge proof that $P(\boldsymbol{\beta}, x) = 0$ in d-BLAC's authentication protocol yields a secure d-BLAC construction under the DDH assumption for $\mathcal{G}$, the SDH and polyDH assumptions for the bilinear group-generating algorithm used for $(n - k, n)$-threshold mercurial commitments, and the SDH assumption for the group-generating algorithm used for BBS+ signatures.*

Comparison with the original d-BLAC protocol. In the naive instantiation for the above protocol, as proposed by the creators of d-BLAC [TAKS10; §6.3], the expected cost for the user is

$$n \operatorname{ExpCost}_{\mathbb{G}}((2,\tau)) + d \operatorname{ExpCost}_{\mathbb{G}}((3,\tau)) + (n-d) \operatorname{ExpCost}_{\mathbb{G}}((2,\tau)) < (4n+d)\tau$$

multiplications in $\mathbb{G}$, and the expected cost for the SP is

$$n \operatorname{ExpCost}_{\mathbb{G}}((2,\tau)) + n \operatorname{ExpCost}_{\mathbb{G}}((3,\tau)) < 5n\tau$$

multiplications in $\mathbb{G}$. For communication cost, the user sends $2n$ elements from $\mathbb{G}^*$ and $3n - d$ elements of $\mathbb{Z}_q$ to the SP. The protocol is non-interactive using the Fiat-Shamir transform; thus, the SP sends nothing to the user.

Referring to the cost analysis for Protocol 4.50, we again find that our own instantiation reduces the cost substantially. For instance, setting $\tau = 256$ and $\lambda_0 = 40$ and letting n grow large, we find that the expected *total* and *online* costs for the user in the new protocol respectively approach $\approx 51\%$ and $\approx 2\%$ of its expected total and online costs in the naive protocol; likewise, the expected cost for the SP in the new protocol approaches $\approx 5\%$ that of its expected cost in the naive protocol. Similarly, the communication overhead in Protocol 4.46 approaches $\approx 25\%$ that of the naive protocol.

5.1.6.3 Batch BLACR

The last protocol that we address is the authentication protocol arising in BLACR. Similar to in vanilla BLAC and d-BLAC, the user in the BLACR authentication protocol outputs a ticket $\mathcal{T} = (t, \mathcal{H}_{\mathbb{G}}(t\|s)^x)$ and engages the SP in a zero-knowledge proof that (i) it holds a valid credential $C(x)$ from the GM, and (ii) the exponent x in $\mathcal{T}$ is the same as the secret identity key x in $C(x)$. The third part of the proof, however, is quite different: given the SP's blacklist $\boldsymbol{\beta} = ((g_1, h_1, \varsigma_1), \ldots, (g_n, h_n, \varsigma_n))$ in which $g_i = \mathcal{H}_{\mathbb{G}}(t_i\|s)$, $h_i = g_i^{x_i}$, and $\varsigma_i \in \mathbb{Z}$ for each $i = 1, \ldots, n$, the user must prove that $\sum_{i \in H} \varsigma_i > K$ for some constant integer K (as specified by the SP's revocation policy P), where $H = \{i \in [1,n] \mid h_i = g_i^x\}$. Again, the first two steps do not depend on the contents of the blacklist and we focus our attention on the more costly (and credential-agnostic) portion of the protocol.

The construction follows almost directly from the results in Sections 4.1.3 and 4.4.3. Let g and h be arbitrary generators of $\mathbb{G}$, and let $\overline{H} = [1,n] \setminus H$. The user chooses $r \in_{\mathrm{R}} \mathbb{Z}_q^*$ and $r_i \in_{\mathrm{R}} \mathbb{Z}_q^*$ for each $i = 1, \ldots, n$, and, for each $i \in H$, it chooses $s_i \in_{\mathrm{R}} \mathbb{Z}_q^*$. The user then uses these values to compute

$$A_i = \begin{cases} (g_i^x / h_i)^r & \text{if } i \in \overline{H} \\ g^{s_i} & \text{if } i \in H, \end{cases}$$

as in Protocol 4.50, and

$$D_i = \begin{cases} g^{r_i} & \text{if } i \in \overline{H} \\ g^{r_i} h^{\varsigma_i} & \text{if } i \in H, \end{cases}$$

for each $i = 1, \ldots, n$. The user then sends $A = (A_1, D_1, \ldots, A_n, D_n)$ to the SP and engages the SP in a proof of knowledge of $(\gamma_1, \gamma_2, r_1, \ldots, r_n)$ and $(s_i)_{i \in H}$ such that $1 = g_0^{\gamma_1} h_0^{\gamma_2}$ and, for each $i = 1, \ldots, n$, either $A_i = g_i^{\gamma_1} h_i^{\gamma_2}$ and $D_i = g^{r_i}$ or $A_i = g^{s_i}$ and $D_i / h^{\varsigma_i} = g^{r_i}$.

Observation 5.3. If the SP accepts in the above proof of knowledge, then, with probability overwhelming in τ, $\prod_{i=1}^n D_i = g^{\sum_{i=1}^n r_i} h^{\sum_{i \in H} \varsigma_i}$ and is a commitment to the user's (correctly computed) aggregate score $\sum_{i \in H} \varsigma_i$.

From here, the user can employ a standard *range proof* protocol [Bou00, CCas08, CLZ12] to prove that $\sum_{i \in H} \varsigma_i$ exceeds the necessary threshold K.

To instantiate the above protocol as a system for batch arguments of knowledge, we apply RME parallelization to the $n = 1$ case. As the $n = 1$ case is a just a disjunctive proof, it follows from Theorem 4.32 that we can implement this protocol using an $(n, 2n)$-threshold mercurial commitment. In particular, we associate each index $i \in [1, n]$ with a pair of sub-challenges (c_i, c_{n+i}) such that each $c_i + c_{n+i} = c \bmod \rho$. (As usual, the modulus ρ here denotes an arbitrary λ_0-bit integer.) Because the $(n, 2n)$-threshold mercurial commitment forces the user to commit to at least n challenges (with a probability overwhelming in τ), and the secret sharing ensures the user commits to at most one challenge from each pair (c_i, c_{n+i}) (with a probability overwhelming in λ_0), it follows, with a probability overwhelming in $\lambda_0 \in o(\tau)$, that the user must commit to either c_i or c_{n+i}, but not both, for each $i = 1, \ldots, n$. The full protocol follows.

Protocol 5.4 (RME-based BLACR authentication protocol).

Common input: $(\mathbb{G}, q, g, h, g_0, \ldots, g_n) \leftarrow \mathcal{G}(1^\tau; n+3)$, $(h_0, \ldots, h_n) \in (\mathbb{G})^{n+1}$, and $(\varsigma_1, \ldots, \varsigma_n) \in (\mathbb{Z})^n$

P's private input: $x \in \mathbb{Z}_q$ and $H \subseteq [1,n]$ with $h_0 = g_0^x$ and $h_j = g_j^x$ if and only if $j \in H$

P1: Let $\overline{H} = [1,n] \setminus H$ and let $\tilde{H} = \{\, i \in [1,2n] \mid i \in \overline{H} \vee (i-n) \in H \,\}$. P chooses a sub-challenge $c_i \in_{\mathrm{R}} \mathbb{Z}_\rho$ for each $i \in \tilde{H}$, and then it computes $(A_0, \gamma) \leftarrow \mathrm{Com}((c_i)_{i \in \tilde{H}})$. P chooses $r \in_{\mathrm{R}} \mathbb{Z}_q^*$, $r_i \in_{\mathrm{R}} \mathbb{Z}_q^*$ for each $i = 1, \ldots, n$, and $s_i \in_{\mathrm{R}} \mathbb{Z}_q^*$ for each $i \in H$, and then it computes the announcements

$$A_i = \begin{cases} (g_i^x / h_i)^r & i \in \overline{H} \\ g^{s_i} & i \in H \end{cases} \qquad \text{and} \qquad D_i = \begin{cases} g^{r_i} & i \in \overline{H} \\ g^{r_i} h^{\varsigma_i} & i \in H \end{cases}$$

for each $i = 1, \ldots, n$. P announces $A = (A_0, A_1, D_1, \ldots, A_n, D_n)$ to the SP.

V2: V chooses a random exponent $t_i \in_{\mathrm{R}} \mathbb{Z}_\rho$ for each $i = 1, \ldots, n$, and then it sends $t = (t_1, \ldots, t_n)$ to P.

P3: For each $i \in \overline{H}$ and for each $j \in H$, P computes $a_i = t_i c_i \bmod \rho$ and $b_j = t_j c_{n+j} \bmod \rho$, and then it chooses $(\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4) \in_{\mathrm{R}} (\mathbb{Z}_q^*)^5$, and computes $B_0 = g_0^{\gamma_1} h_0^{\gamma_2}$, $B_1 = g^{\gamma_0} (\prod_{i \in \overline{H}} g_i^{a_i})^{\gamma_1} (\prod_{i \in \overline{H}} h_i^{a_i})^{\gamma_2}$, $B_2 = g^{\gamma_3} (\prod_{i \in \overline{H}} (D_i / h^{\varsigma_i})^{a_i})$, and $B_3 = g^{\gamma_4} (\prod_{j \in H} D_i^{b_i})$. P sends $B = (B_0, B_1, B_2, B_3)$ to V.

V4: V issues a challenge $c \in_{\mathrm{R}} \mathbb{Z}_\rho$ to P.

P5: For each $i \in \overline{H}$, P computes $c_{n+i} = c - c_i \bmod \rho$ and $b_i = t_i c_{n+i} \bmod \rho$, and, for each $j \in H$, it computes $c_j = c - c_{n+j} \bmod \rho$ and $a_j = t_j c_j \bmod \rho$. P then computes the responses $v_0 = \gamma_0 - c \sum_{j \in H} a_j s_j$, $v_1 = \gamma_1 - cxr$, $v_2 = \gamma_2 + cr$, $v_3 = \gamma_3 - \sum_{j \in H} a_j r_j$, and $v_4 = \gamma_4 - \sum_{i \in \overline{H}} b_i r_i$, and the opening $\pi \leftarrow \mathrm{Open}(A_0, \gamma, c_1, \ldots, c_{2n})$ of A_0. P sends $v = (v_0, v_1, v_2, v_3, v_4, \pi, c_1, \ldots, c_n)$ to V.

V6: V computes $a_i = t_i c_i \bmod \rho$ and $b_i = t_i c_{n+i} \bmod \rho$ for each $i = 1, \ldots, n$. V accepts if $\mathrm{Ver}(A_0, \pi, c_1, \ldots, c_n, c - c_1, \ldots, c - c_n) = 1$, if each of $B_0 = g_0^{v_1} h_0^{v_2}$, $B_1 = g^{v_0} (\prod_{i=1}^n g_i^{a_i})^{v_1} (\prod_{i=1}^n h_i^{a_i})^{v_2} (\prod_{i=1}^n A_i^{a_i})^c$, $B_2 = g^{v_3} (\prod_{i=1}^n (D_i / h^{\varsigma_i})^{a_i})$, and $B_3 = g^{v_4} (\prod_{i=1}^n D_i^{b_i})$ hold, and if $A_i \neq 1$ for each $i = 1, \ldots, n$, and it rejects otherwise.

Theorem 5.5. *Using Protocol 5.4 to implement the zero-knowledge proof that* $P(\beta, x) = 0$ *in BLACR's authentication protocol yields a secure BLACR construction under the DDH assumption for* $\mathcal{G}$*, the SDH and polyDH assumptions for the bilinear group-generating algorithm used for* $(n, 2n)$*-threshold mercurial commitments, and the SDH assumption for the group-generating algorithm used for BBS+.*

5.2 Chapter summary

This chapter examined how to leverage the batch zero-knowledge proof techniques introduced in this dissertation to speed up three constructions for *anonymous blacklisting and reputation systems*. Interested readers can find additional details on these constructions in a WPES 2013 paper [HG13c] by the author and Ian Goldberg. (Applications of batch zero-knowledge proof techniques to a different anonymous blacklisting construction can be found in an IEEE S&P 2011 paper [HG11a], also by the author and Ian Goldberg.) The new batch protocols result in considerable cost savings compared to the "schoolbook" protocols proposed by Tsang et al. [TAKS10] for BLAC and d-BLAC, and by Au et al. [AKS12] for BLACR and, thereby, significantly improve the practicality of the BLAC approach to anonymous blacklisting.

Chapter 6

Conclusion

This dissertation examined batch zero-knowledge proof and verification techniques, with a particular emphasis on batch zero-knowledge proof systems for discrete logarithms in prime-order groups. The overarching goal was to convince the reader that such batch techniques have the potential to substantially reduce the communication and computation overhead imposed by the zero-knowledge protocols naturally arising in constructions for privacy-enhancing technologies (PETs). Given the extensive list of PET constructions for which the cost of "high fan-in" zero-knowledge proofs remains a primary obstacle to adoption, the author believes that the results presented in this dissertation provide strong support for such a thesis.

The author hopes these results will motivate and provide a solid theoretical basis for the development of additional batch zero-knowledge proof techniques, and, ultimately, have a direct impact to privacy by eliminating or reducing barriers to the adoption of new and existing PETs.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach, First Edition*. Cambridge University Press, New York, NY, USA (April 2009). *(Two citations on pages 11 and 134.)*

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of USENIX Security 2008*, pages 335–348, San Jose, CA, USA (August 2008). *(One citation on page 2.)*

[AK12] Man Ho Au and Apu Kapadia. PERM: Practical reputation-based blacklisting without TTPs. In *Proceedings of CCS 2012*, pages 929–940, Raleigh, NC, USA (October 2012). *(One citation on page 149.)*

[AKS12] Man Ho Au, Apu Kapadia, and Willy Susilo. BLACR: TTP-free blacklistable anonymous credentials with reputation. In *Proceedings of NDSS 2012*, San Diego, CA, USA (February 2012). *(Seven citations on pages 2×2, 5, 148, 149, 152, and 160.)*

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In *Proceedings of SCN 2006*, volume 4116 of *LNCS*, pages 111–125, Maiori, Italy (September 2006). *(One citation on page 151.)*

[ATK11] Man Ho Au, Patrick P. Tsang, and Apu Kapadia. PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. *ACM Transactions on Information and System Security (TISSEC)*, 14(4):Article No. 29 (December 2011). *(Four citations on pages 2×2, 147, and 149.)*

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Proceedings of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238, Interlaken, Switzerland (May 2004). *(One citation on page 179.)*

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177 (April 2008). *(Two citations on pages 108 and 179.)*

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark (May 2005). *(One citation on page 179.)*

[BC89] Jurjen N. Bos and Matthijs J. Coster. Addition chain heuristics. In *Proceedings of CRYPTO 1989*, volume 435 of *LNCS*, pages 400–407, Santa Barbara, CA, USA (August 1989). *(One citation on page 26.)*

[BDD07] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Proceedings of ACISP 2007*, volume 4586 of *LNCS*, pages 400–415, Townsville, Australia (July 2007). *(One citation on page 59.)*

[Ber02] Daniel J. Bernstein. Pippenger's exponentiation algorithm. To be incorporated into the author's *High-speed cryptography* book, (January 2002). *(Three citations on pages 26×2 and 44.)*

[BGMW92] Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley, and David Bruce Wilson. Fast exponentiation with precomputation (extended abstract). In *Proceedings of EUROCRYPT 1992*, volume 658 of *LNCS*, pages 200–207, Balatonfüred, Hungary (May 1992). *(Three citations on pages 26×2 and 27.)*

[BGR98a] Mihir Bellare, Juan A. Garay, and Tal Rabin. Batch verification with applications to cryptography and checking. In *Proceedings of LATIN 1998*, volume 1380 of *LNCS*, pages 170–191, Campinas, Brazil (April 1998). *(One citation on page 37.)*

[BGR98b] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Proceedings of EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 236–250, Espoo, Finland (June 1998). *(Nine citations on pages 4, 25, 37, 40, 41, 43, 44, 53, and 86.)*

[BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Proceedings of CRYPTO 1988*, volume 403 of *LNCS*, pages 27–35, Santa Barbara, CA, USA (August 1988). *(Two citations on pages 93 and 126.)*

[BL12] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 9(3):345–360 (May–June 2012). *(One citation on page 147.)*

[BMS01] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Proceedings of Information Hiding 2001*, volume 2137 of *LNCS*, pages 245–257, Pittsburgh, PA, USA (April 2001). *(One citation on page 146.)*

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of STOC 1988*, pages 1–10, Chicago, IL, USA (May 1988). *(One citation on page 2.)*

[Bon98] Dan Boneh. The decision diffie-hellman problem. In *Proceedings of ANTS III (1998)*, volume 1423 of *LNCS*, pages 48–63, Portland, OR, USA (June 1998). *(One citation on page 24.)*

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Proceedings of EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444, Bruges, Belgium (May 2000). *(One citation on page 157.)*

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of CCS 1993*, pages 62–73, Fairfax, VA, USA (November 1993). *(One citation on page 21.)*

[Bra39] Alfred Brauer. On addition chains. *Bulletin of the American Mathematical Society*, 45(10):736–739 (October 1939). *(One citation on page 26.)*

[Bra00] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy, First Edition*. MIT Press, Cambridge, MA, USA (August 2000). *(Five citations on pages 24, 25, and 35×3.)*

[Bra06] Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5(4):201–216 (October 2006). *(One citation on page 2.)*

[BT04] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review (SIREV)*, 46(3):501–517 (September 2004). *(One citation on page 95.)*

[Cac99] Christian Cachin. Efficient private bidding and auctions with an oblivious third party. In *Proceedings of CCS 1999*, pages 120–127, Singapore (November 1999). *(One citation on page 2.)*

[CCas08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In *Proceedings of ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 234–252, Melbourne, Australia (December 2008). *(One citation on page 157.)*

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of STOC 1988*, pages 11–19, Chicago, IL, USA (May 1988). *(One citation on page 2.)*

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of CRYPTO 1994*, volume 839 of *LNCS*, pages 174–187, Santa Barbara, CA, USA (August 1994). *(Seven citations on pages 88, 89, 90, 91, 93, 94, and 96.)*

[CFM08] Dario Catalano, Dario Fiore, and Mariagrazia Messina. Zero-knowledge sets with short proofs. In *Proceedings of EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 433–450, Istanbul, Turkey (April 2008). *(One citation on page 102.)*

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594 (July 2004). *(One citation on page 22.)*

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *Proceedings of FOCS 1985*, pages 383–395, Portland, OR, USA (October 1985). *(Two citations on pages 6 and 109.)*

[CH11] Jeremy Clark and Urs Hengartner. Selections: Internet voting with over-the-shoulder coercion-resistance. In *Proceedings of FC 2011*, volume 7035 of *LNCS*, pages 47–61, Gros Islet, St. Lucia (February 2011). *(Three citations on pages 2×2 and 145.)*

[CHL+05] Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *Proceedings of EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 422–439, Aarhus, Denmark (May 2005). *(One citation on page 101.)*

[CHL+13] Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to zero-knowledge sets. *Journal of Cryptology*, 26(2):251–279 (April 2013). *(One citation on page 88.)*

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of SCN 2002*, volume 2576 of *LNCS*, pages 268–289, Amalfi, Italy (September 2002). *(One citation on page 150.)*

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Proceedings of CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96, Santa Barbara, CA, USA (August 2006). *(One citation on page 21.)*

[CLZ12] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A non-interactive range proof with constant communication. In *Proceedings of FC 2012*, volume 7397 of *LNCS*, pages 179–199, Kralendijk, Bonaire (March 2012). *(One citation on page 157.)*

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Proceedings of CRYPTO 1992*, volume 740 of *LNCS*, pages 89–105, Santa Barbara, CA, USA (August 1992). *(Two citations on pages 55 and 56.)*

[Cra96] Ronald Cramer. *Modular Design of Secure Yet Practical Cryptographic Protocols*. PhD thesis, CWI and University of Amsterdam, Amsterdam, Netherlands (November 1996). *(Two citations on pages 30 and 31.)*

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings of CRYPTO 1997*, volume 1294 of *LNCS*, pages 410–424, Santa Barbara, CA, USA (August 1997). *(One citation on page 19.)*

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proceedings of CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144, Santa Barbara, CA, USA (August 2003). *(Three citations on pages 134 and 135².)*

[CY08] Koji Chida and Go Yamamoto. Batch processing for proofs of partial knowledge and its applications. *IEICE Transactions*, 91-A(1):150–159 (January 2008). *(One citation on page 97.)*

[Dam11] Ivan Damgård. On σ-protocols. Lecture notes for CPT 2011, University of Aarhus BRICS, Aarhus, Denmark (March 2011). *(Four citations on pages 19, 31, 69, and 91.)*

[DGOW95] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs dishonest verifier in public coin zero-knowledge proofs. In *Proceedings of CRYPTO 1995*, volume 963 of *LNCS*, pages 325–338, Santa Barbara, CA, USA (August 1995). *(One citation on page 19.)*

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654 (November 1976). *(One citation on page 23.)*

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proceedings of Usenix Security 2004*, San Diego, CA, USA (August 2004). *(One citation on page 146.)*

[Dou02] John R. Douceur. The Sybil attack. In *Proceedings of IPTPS 2002*, volume 2429 of *LNCS*, pages 251–260, Cambridge, MA, USA (March 2002). *(One citation on page 151.)*

[Erd61] Paul Erdös. Remarks on number theory III: On addition chains. *Acta Arithmetica*, 6(1):77–81 (1960–1961). *(Two citations on pages 26 and 50.)*

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of FOCS 1987*, pages 427–437, Los Angeles, CA, USA (October 1987). *(One citation on page 2.)*

[For09] Lance Fortnow. The status of the P versus NP problem. *Communications of the ACM (CACM)*, 52(9):78–86 (September 2009). *(One citation on page 10.)*

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA (August 1986). *(Three citations on pages 19, 21, and 34.)*

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of STOC 1990*, pages 416–426, Baltimore, MD, USA (May 1990). *(One citation on page 190.)*

[Gal12] Joseph A. Gallian. *Contemporary Abstract Algebra, Eighth Edition*. Brooks Cole, Boston, MA, USA (July 2012). *(Three citations on pages 7, 107, and 110.)*

[GLSY04] Rosario Gennaro, Darren Leigh, Ravi Sundaram, and William S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In *Proceedings of ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 276–292, Jeju Island, South Korea (December 2004). *(Eight citations on pages 59, 66, 80, 81, 82, 84, 85, and 86.)*

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems (extended abstract). In *Proceedings of STOC 1985*, pages 291–304, Providence, RI, USA (May 1985). *(One citation on page 2.)*

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge and a methodology of cryptographic protocol design. In *Proceedings of CRYPTO 1986*, volume 263 of *LNCS*, pages 171–185, Santa Barbara, CA, USA (August 1986). *(One citation on page 2.)*

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of STOC 1987*, pages 218–229, New York, NY, USA (May 1987). *(One citation on page 2.)*

[Gol01] Oded Goldreich. *The Foundations of Cryptography – Volume 1, Basic Techniques*. Cambridge University Press, New York, NY, USA (June 2001). *(Three citations on pages 1, 10, and 40.)*

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121 (September 2008). *(One citation on page 106.)*

[Gro04] Jens Groth. *Honest Verifier Zero-Knowledge Arguments Applied*. PhD thesis, University of Aarhus BRICS DS-04-3, Aarhus, Denmark (October 2004). *(One citation on page 19.)*

[Gro10] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579 (October 2010). *(Two citations on pages 2 and 59.)*

[GSV98] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of STOC 1998*, pages 399–408, Dallas, TX, USA (May 1998). *(One citation on page 19.)*

[HG11a] Ryan Henry and Ian Goldberg. Extending Nymble-like systems. In *Proceedings of IEEE S&P 2011*, pages 523–537, Berkeley, CA, USA (May 2011). *(One citation on page 160.)*

[HG11b] Ryan Henry and Ian Goldberg. Formalizing anonymous blacklisting systems. In *Proceedings of IEEE S&P 2011*, pages 81–95, Berkeley, CA, USA (May 2011). *(Three citations on pages 148, 149, and 151.)*

[HG12] Ryan Henry and Ian Goldberg. All-but-k mercurial commitments and their applications. Technical Report CACR 2012-26, University of Waterloo, Waterloo, ON, Canada (November 2012). *(Three citations on pages 5, 6, and 102.)*

[HG13a] Ryan Henry and Ian Goldberg. Batch proofs of partial knowledge. In *Proceedings of ACNS 2013*, volume 7954 of *LNCS*, pages 502–517, Banff, AB, Canada (June 2013). *(Three citations on pages 4[x2] and 60.)*

[HG13b] Ryan Henry and Ian Goldberg. Batch proofs of partial knowledge. Technical Report CACR 2013-08, University of Waterloo, Waterloo, ON, Canada (March 2013). *(One citation on page 4.)*

[HG13c] Ryan Henry and Ian Goldberg. Thinking inside the BLAC box: Smarter protocols for faster anonymous blacklisting. In *Proceedings of WPES 2013*, pages 71–81, Berlin, Germany (November 2013). *(Five citations on pages 2×2, 5×2, and 160.)*

[HHG10] Ryan Henry, Kevin Henry, and Ian Goldberg. Making a nymbler Nymble using VERBS. In *Proceedings of PETS 2010*, volume 6205 of *LNCS*, pages 111–129, Berlin, Germany (July 2010). *(One citation on page 147.)*

[HHG13] Ryan Henry, Yizhou Huang, and Ian Goldberg. One (block) size fits all: PIR and SPIR with variable-length records via multi-block queries. In *Proceedings of NDSS 2013*, San Diego, CA, USA (February 2013). *(Two citations on pages 2 and 145.)*

[HOG11] Ryan Henry, Femi Olumofin, and Ian Goldberg. Practical PIR for electronic commerce. In *Proceedings of CCS 2011*, pages 677–690, Chicago, IL, USA (October 2011). *(Four citations on pages 2×2, 59, and 145.)*

[Ica09] Thomas Icart. How to hash into elliptic curves. In *Proceedings of CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316, Santa Barbara, CA, USA (August 2009). *(One citation on page 150.)*

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of WPES 2005*, pages 61–70, Alexandria, VA, USA (November 2005). *(Two citations on page 2.)*

[JKTS07] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Nymble: Anonymous IP-address blocking. In *Proceedings of PETS 2007*, volume 4776 of *LNCS*, pages 113–133, Ottawa, ON, Canada (June 2007). *(One citation on page 147.)*

[JM94] Wen-Ai Jackson and Keith M. Martin. Geometric secret sharing schemes and their duals. *Designs, Codes and Cryptography*, 4(1):83–95 (January 1994). *(Two citations on pages 92 and 123.)*

[JMV05] David Jao, Stephen D. Miller, and Ramarathnam Venkatesan. Do all elliptic curves of the same order have the same difficulty of discrete log? In *Proceedings of ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 21–40, Chennai, India (December 2005). *(One citation on page 23.)*

[KK09] Murat Kantarcioglu and Onur Kardes. Privacy-preserving data mining in the malicious model. *International Journal of Information and Computer Security (IJICS)*, 2(4):353–375 (January 2009). *(Two citations on page 2.)*

[Knu76] Donald E. Knuth. Big Omicron and big Omega and big Theta. *SIGACT News*, 8(2):18–24 (April–June 1976). *(One citation on page 8.)*

[KZG10a] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Proceedings of ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194, Singapore (December 2010). *(Seven citations on pages 5, 105, 106, 107, 108×2, and 180.)*

[KZG10b] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Polynomial commitments. Technical Report CACR 2010-10, University of Waterloo, Waterloo, ON, Canada (December 2010). *(One citation on page 185.)*

[LH10] Zi Lin and Nicholas Hopper. Jack: Scalable accumulator-based Nymble system. In *Proceedings of WPES 2010*, pages 53–62, Chicago, IL, USA (October 2010). *(Two citations on pages 2 and 147.)*

[LH11] Peter Lofgren and Nicholas Hopper. BNymble: More anonymous blacklisting at almost no cost (a short paper). In *Proceedings of FC 2011*, volume 7035 of *LNCS*, pages 268–275, Gros Islet, St. Lucia (February 2011). *(One citation on page 147.)*

[Lim00] Chae Hoon Lim. Efficient multi-exponentiation and application to batch verification of digital signatures. Technical report, Sejong University, Seoul, South Korea (August 2000). *(Three citations on pages 26 and 27×2.)*

[LL94] Chae Hoon Lim and Pil Joong Lee. More flexible exponentiation with precomputation. In *Proceedings of CRYPTO 1994*, volume 839 of *LNCS*, pages 95–107, Santa Barbara, CA, USA (August 1994). *(Two citations on pages 26 and 27.)*

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534 (December 1982). *(Two citations on pages 100 and 183.)*

[LY10] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *Proceedings of TCC 2010*, volume 5978 of *LNCS*, pages 499–517, Zürich, Switzerland (February 2010). *(One citation on page 4.)*

[MN96] David M'Raïhi and David Naccache. Batch exponentiation: A fast DLP-based signature generation strategy. In *Proceedings of CCS 1996*, pages 58–61, New Delhi, India (March 1996). *(One citation on page 26.)*

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, New York, NY, USA (October 1996). Fifth printing (August 2001). *(Two citations on pages 25 and 183.)*

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of CCS 2011*, pages 116–125, Philadelphia, PA, USA (November 2001). *(One citation on page 2.)*

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Proceedings of PKC 2001*, volume 1992 of *LNCS*, pages 104–118, Jeju Island, South Korea (February 2001). *(One citation on page 179.)*

[PB08] Kun Peng and Feng Bao. Batch ZK proof and verification of OR logic. In *Proceedings of INSCRYPT 2008*, volume 5487 of *LNCS*, pages 144–156, Beijing, China (December 2008). *(Five citations on pages 4, 97×2, 98, and 99.)*

[PB10] Kun Peng and Feng Bao. Batch range proof for practical small ranges. In *Proceedings of AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 114–130, Stellenbosch, South Africa (May 2010). *(One citation on page 59.)*

[PBD07] Kun Peng, Colin Boyd, and Ed Dawson. Batch zero-knowledge proof and verification and its applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(2):Article No. 6 (May 2007). *(Eight citations on pages 59, 66, 70, 73×2, 84, 85, and 86.)*

[Pen12] Kun Peng. Attack against a batch zero-knowledge proof system. *IET Information Security*, 6(1):1–5 (March 2012). *(One citation on page 97.)*

[Pip76] Nicholas Pippenger. On the evaluation of powers and related problems (preliminary version). In *Proceedings of FOCS 1976*, pages 258–263, Houston, TX, USA (October 1976). *(Two citations on pages 26 and 44.)*

[Pip80] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal on Computing (SICOMP)*, 9(2):230–250 (May 1980). *(Two citations on page 27.)*

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Proceedings of EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 387–398, Saragossa, Spain (May 1996). *(One citation on page 22.)*

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of STOC 1989*, pages 73–85, Seattle, WA, USA (May 1989). *(One citation on page 2.)*

[Rog06] Phillip Rogaway. Formalizing human ignorance: Collision-resistant hash functions without the keys. In *Proceedings of VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 211–228, Hanoi, Vietnam (September 2006). *(One citation on page 20.)*

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Proceedings of FSE 2004*, volume 3017 of *LNCS*, pages 371–388, Delhi, India (February 2004). *(Two citations on page 20.)*

[SBM10] Edward J. Schwartz, David Brumley, and Jonathan M. McCune. Contractual anonymity. In *Proceedings of NDSS 2010*, San Diego, CA, USA (March 2010). *(One citation on page 148.)*

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of CRYPTO 1989*, volume 435 of *LNCS*, pages 239–252, Santa Barbara, CA, USA (August 1989). *(Five citations on pages 4, 28, 30, 34, and 189.)*

[SD92] Jörg Sauerbrey and Andreas Dietel. Resource requirements for the application of addition chains in modulo exponentiation. In *Proceedings of EUROCRYPT 1992*, volume 658 of *LNCS*, pages 174–182, Balatonfüred, Hungary (May 1992). *(One citation on page 26.)*

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613 (1979). *(One citation on page 94.)*

[SJF08] José Luis Subirats, José M. Jerez, and Leonardo Franco. A new decomposition algorithm for threshold synthesis and generalization of boolean functions. *IEEE Transactions on Circuits and Systems – Part I: Regular Papers*, 55-I(10):3188–3196 (November 2008). *(One citation on page 93.)*

[SJM91] Gustavus J. Simmons, Wen-Ai Jackson, and Keith M. Martin. The geometry of shared secret schemes. *Bulletin of the Institute of Combinatorics and its Application (ICA)*, 1:71–88 (1991). *(One citation on page 92.)*

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth. In *Proceedings of EUROCRYPT 1995*, volume 921 of *LNCS*, pages 393–403, Saint-Malo, France (May 1995). *(Two citations on pages 2 and 59.)*

[SR94] Kai-Yeung Siu and Vwani P. Roychowdhury. On optimal depth threshold circuits for multiplication and related problems. *SIAM Journal on Discrete Mathematics*, 7(2):284–292 (1994). *(One citation on page 93.)*

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *Proceedings of EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 190–199, Saragossa, Spain (May 1996). *(One citation on page 2.)*

[Str64] Ernst G. Straus. Additions chains of vectors (problem 5125). *American Mathematical Monthly*, 71(7):806–808 (August–September 1964). *(Two citations on pages 26 and 27.)*

[TAKS07] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable Anonymous Credentials: Blocking misbehaving users without TTPs. In *Proceedings of CCS 2007*, pages 72–81, Alexandria, VA, USA (October 2007). *(Three citations on pages 5, 147, and 149.)*

[TAKS10] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. *ACM Transactions on Information and Systems Security (TISSEC)*, 13(4):Article No. 39 (December 2010). *(14 citations on pages 2×2, 5×2, 147, 148, 149, 150, 151, 152×2, 155, 156, and 160.)*

[Tora] Tor Project, Inc. doc/BlockingIrc – Tor Bug Tracker & Wiki. Retrieved: 2014-04-05. *(One citation on page 146.)*

[Torb] Tor Project, Inc. Tor metrics portal: Network. Retrieved: 2014-04-05 from https://metrics.torproject.org/users.html?graph=···. *(One citation on page 146.)*

[Torc] Tor Project, Inc. Tor metrics portal: Users. Retrieved: 2014-04-05 from https://metrics.torproject.org/users.html?graph=···. *(One citation on page 146.)*

[TWC+04] Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable linkable threshold ring signatures. In *Proceedings of INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 384–398, Chennai, India (December 2004). *(Two citations on page 2.)*

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM Journal on Computing (SICOMP)*, 36(4):1160–1214 (March 2006). *(One citation on page 19.)*

[Yao76] Andrew Chi-Chih Yao. On the evaluation of powers. *SIAM Journal on Computing (SICOMP)*, 5(1):100–103 (March 1976). *(Two citations on page 26.)*

[ZS09] Gregory M. Zaverucha and Douglas R. Stinson. Group testing and batch verification. In *Proceedings of ICITS 2009*, volume 5973 of *LNCS*, pages 140–157, Shizuoka, Japan (December 2009). *(Two citations on page 39.)*

APPENDICES

Appendix A

Intractability assumptions

This appendix introduces four computational intractability assumptions that are used in the main text. Two of the four assumptions require the following notion of a *bilinear group-generating algorithm*.

Definition 25. A *bilinear group-generating algorithm* $\widetilde{\mathcal{G}}$ is a PPT algorithm that, on input 1^τ, outputs a description of a pair of isomorphic multiplicative groups $(\widetilde{\mathbb{G}}, \mathbb{G}_{\mathrm{T}})$, their τ-bit prime order p, a fixed generator $\tilde{g} \in \widetilde{\mathbb{G}}^*$, and a bilinear pairing function $e\colon \widetilde{\mathbb{G}} \times \widetilde{\mathbb{G}} \to \mathbb{G}_{\mathrm{T}}$.

A.1 The computational Diffie-Hellman problem

The *computational Diffie-Hellman (CDH) problem* in $\mathbb{G}$ is:

> **CDH problem:** Given as input a 3-tuple $(g_1, g_2, h_1) \in (\mathbb{G}^*)^2 \times \mathbb{G}$, output $h_2 \in \mathbb{G}$ such that $\log_{g_1} h_1 = \log_{g_2} h_2$.

An algorithm $\mathcal{A}$ solves the CDH problem in $\mathbb{G}$ with advantage ε if

$$\Pr\left[g_2^x \leftarrow \mathcal{A}(g_1, g_2, g_1^x) \mid g_1, g_2 \in_{\mathrm{R}} \mathbb{G}^* \wedge x \in_{\mathrm{R}} \mathbb{Z}_q\right] = \varepsilon,$$

where the probability is over the random choices of $(g_1, g_2) \in (\mathbb{G}^*)^2$ and $x \in \mathbb{Z}_q$ and the random bits consumed by $\mathcal{A}$. We say the CDH problem is (t, ε)-intractable in $\mathbb{G}$ if no t-time probabilistic algorithm solves the CDH problem in $\mathbb{G}$ with advantage greater than ε.

Definition 26. The *CDH assumption* holds for group-generating algorithm $\mathcal{G}$ if there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $(\mathbb{G}, q, g_1, g_2) \leftarrow \mathcal{G}(1^\tau; 2)$, then $\mathcal{A}$ solves the CDH problem in $\mathbb{G}$ with advantage at most $\varepsilon(\tau)$.

We cast the following trivial observation about Definition 26.

Observation A.1. If the CDH assumption holds for $\mathcal{G}$, then the DL assumption also holds for $\mathcal{G}$.

A.2 The decision Diffie-Hellman problem

The *decision Diffie-Hellman (DDH) problem* in $\mathbb{G}$ is the "decision problem" variant of the CDH problem in $\mathbb{G}$:

> **DDH problem:** Given as input a 4-tuple $(g_1, g_2, h_1, h_2) \in (\mathbb{G}^*)^2 \times (\mathbb{G})^2$, determine whether $\log_{g_1} h_1 = \log_{g_2} h_2$.

An algorithm $\mathcal{A}$ solves the DDH problem in $\mathbb{G}$ with advantage ε if

$$\Big|\Pr[1 \leftarrow \mathcal{A}(g_1, g_2, h_1, h_2) \mid g_1, g_2 \in_{\mathrm{R}} \mathbb{G}^* \wedge h_1, h_2 \in_{\mathrm{R}} \mathbb{G} \wedge \log_{g_1} h_1 \neq \log_{g_2} h_2] - \Pr[1 \leftarrow \mathcal{A}(g_1, g_2, g_1^x, g_2^x) \mid g_1, g_2 \in_{\mathrm{R}} \mathbb{G}^* \wedge x \in_{\mathrm{R}} \mathbb{Z}_q]\Big| = \varepsilon,$$

where the probability is over the random choices of $(g_1, g_2) \in (\mathbb{G}^*)^2$, $(h_1, h_2) \in (\mathbb{G})^2$, and $x \in \mathbb{Z}_q$ and the random bits consumed by $\mathcal{A}$. We can reinterpret the DDH problem as inducing an NP-relation with witness $x = \log_{g_1} h_1$. We say the DDH problem is (t, ε)-intractable in $\mathbb{G}$ if no t-time probabilistic algorithm solves the DDH problem in $\mathbb{G}$ with advantage greater than ε.

Definition 27. The *DDH assumption* holds for group-generating algorithm $\mathcal{G}$ if there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $(\mathbb{G}, q, g_1, g_2) \leftarrow \mathcal{G}(1^\tau; 2)$, then $\mathcal{A}$ solves the DDH problem in $\mathbb{G}$ with advantage at most $\varepsilon(\tau)$.

We cast a trivial observation about Definition 27.

Observation A.2. If the DDH assumption holds for $\mathcal{G}$, then the CDH assumption also holds for $\mathcal{G}$.

The converse of Observation A.2 is not believed to be true. For instance, consider the DDH and CDH problems with respect to $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_T, p, \tilde{g}, e) \leftarrow \widetilde{\mathcal{G}}(1^\tau)$ for some bilinear group generating algorithm $\widetilde{\mathcal{G}}$; in this case, there is an efficient algorithm — namely, the pairing algorithm e — with which $\mathcal{A}$ can solve the DDH problem in $\widetilde{\mathbb{G}}$, yet the CDH problem in $\widetilde{\mathbb{G}}$ may nonetheless be intractable. (Such groups $\widetilde{\mathbb{G}}$ in which the DDH problem is easy while the CDH problem is hard are sometimes called *gap-DH groups* [OP01].)

A.3 The strong Diffie-Hellman assumption

For this section, let $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_T, p, \tilde{g}, e) \leftarrow \widetilde{\mathcal{G}}(1^\tau)$ for a fixed bilinear group-generating algorithm $\widetilde{\mathcal{G}}$.

The *n-strong Diffie-Hellman (n-SDH) problem* in $\widetilde{\mathbb{G}}$ is [BB08; §3]:

> ***n*-SDH problem:** Given as input an $(n+1)$-tuple $(\tilde{g}, \tilde{g}^\alpha, \ldots, \tilde{g}^{\alpha^n}) \in (\widetilde{\mathbb{G}}^*)^{n+1}$, output $(i, \tilde{g}^{(\alpha-i)^{-1}})$ for any $i \in \mathbb{Z}_p \setminus \{\alpha\}$.

An algorithm $\mathcal{A}$ solves the n-SDH problem in $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_T, p, \tilde{g}, e)$ with advantage ε if

$$\Pr\left[(i, \tilde{g}^{(\alpha-i)^{-1}}) \leftarrow \mathcal{A}(\tilde{g}, \tilde{g}^\alpha, \ldots, \tilde{g}^{\alpha^n}) \;\middle|\; \alpha \in_R \mathbb{Z}_p^*\right] = \varepsilon,$$

where the probability is over the random choice of $\alpha \in \mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$. The n-SDH problem is (t, ε)-intractable in $\widetilde{\mathbb{G}}$ if no t-time probabilistic algorithm solves the n-SDH problem in $\widetilde{\mathbb{G}}$ with advantage greater than ε.

Definition 28. The *SDH assumption* holds for bilinear group-generating algorithm $\widetilde{\mathcal{G}}$ if, for every positive integer-valued function $n(\tau) \in \mathit{poly}(\tau)$, there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_T, p, \tilde{g}, e) \leftarrow \widetilde{\mathcal{G}}(1^\tau)$, then $\mathcal{A}$ solves the $n(\tau)$-SDH problem in $\widetilde{\mathbb{G}}$ with advantage at most $\varepsilon(\tau)$.

One can view the n-SDH problem as a generalization of the related n-Diffie-Hellman inversion (n-DHI) problem [BB04; Appendix A]. (The n-DHI problem is: given $(\tilde{g}, \tilde{g}^\alpha, \ldots, \tilde{g}^{\alpha^n})$ for $\alpha \in_R \mathbb{Z}_p^*$, output $\tilde{g}^{\alpha^{-1}}$ or, equivalently [BBG05], output $\tilde{g}^{\alpha^{n+1}}$.)

A.4 The polynomial Diffie-Hellman assumption

For this section, let $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_{\mathrm{T}}, p, \tilde{g}, e) \leftarrow \widetilde{\mathcal{G}}(1^\tau)$ for a fixed bilinear group-generating algorithm $\widetilde{\mathcal{G}}$.

The *n-polynomial Diffie-Hellman (n-polyDH) problem* in $\widetilde{\mathbb{G}}$ is [KZG10a; Definition 2]:

> ***n*-polyDH problem:** Given as input an $(n+1)$-tuple $(\tilde{g}, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^n}) \in (\widetilde{\mathbb{G}}^*)^{n+1}$, output $(f, \tilde{g}^{f(\alpha)})$ for any $f \in \mathbb{Z}_p[x]$ such that $\sqrt{p} > \deg f > n$.

An algorithm $\mathcal{A}$ solves the n-polyDH problem in $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_{\mathrm{T}}, p, \tilde{g}, e)$ with advantage ε if

$$\Pr\left[(f, \tilde{g}^{f(\alpha)}) \leftarrow \mathcal{A}(\tilde{g}, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^n}) \;\middle|\; \alpha \in_{\mathrm{R}} \mathbb{Z}_p^* \wedge (\sqrt{p} > \deg f) \wedge (\deg f > n)\right] = \varepsilon,$$

where the probability is over the random choice of $\alpha \in \mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$. The n-polyDH problem is (t, ε)-intractable in $\widetilde{\mathbb{G}}$ if no t-time probabilistic algorithm has advantage ε in solving the n-polyDH problem in $\widetilde{\mathbb{G}}$.

Definition 29. The *polyDH assumption* holds for bilinear group-generating algorithm $\widetilde{\mathcal{G}}$ if, for every positive integer-valued function $n(\tau) \in \mathit{poly}(\tau)$, there exists a negligible function $\varepsilon\colon \mathbb{N} \to \mathbb{R}^+$ such that, for all PPT algorithms $\mathcal{A}$ and for all $\tau \in \mathbb{N}^+$, if $(\widetilde{\mathbb{G}}, \widetilde{\mathbb{G}}_{\mathrm{T}}, p, \tilde{g}, e) \leftarrow \widetilde{\mathcal{G}}(1^\tau)$, then $\mathcal{A}$ solves the $n(\tau)$-polyDH problem in $\widetilde{\mathbb{G}}$ with advantage at most $\varepsilon(\tau)$.

The n-polyDH problem also generalizes the n-DHI problem: whereas the n-DHI problem can be viewed as the problem of outputting $(x^{n+1}, \tilde{g}^{\alpha^{n+1}})$ given $(\tilde{g}, \tilde{g}^{\alpha}, \ldots, \tilde{g}^{\alpha^n})$, the polyDH problem is to output such a pair in which x^{n+1} can be replaced by an *arbitrary* polynomial of degree at least $n+1$ and at most $\sqrt{p}$.

Appendix B

Attacking Peng and Bao's protocol

Note from the author: *I observed the weakness in Peng and Bao's protocol and the attack at a high level; however, the details around solving for sufficiently small solutions by encoding the problem as a short vector search problem is due to Ian Goldberg and he wrote the first draft of the text in this appendix. As such, I can only claim partial credit for the contents of this appendix.*

Details of the attack

This appendix provides additional details on how to implement the attack on Peng and Bao's protocol outlined in Section 4.2. We seek solutions $c_j \in [0, 2^{\lambda_0} - 1]$ for each $j \in H$ to satisfy the system

$$0 \equiv \sum_{j \in H} c_j \, \gamma_j \pmod{q} \tag{4.1a}$$

and

$$\bar{c} \equiv \sum_{j \in H} c_j \pmod{2^{\lambda_0}}, \tag{4.2a}$$

where $H \subseteq_k [1,n]$, $\bar{c} \in [0, 2^{\lambda_0} - 1]$, and $\gamma_j = t_j(x_j - u_j) \bmod q$ are each given. We first shift the range of each equation to center the desired solution about 0 by setting

$$d_j = c_j - 2^{\lambda_0}/2$$

for each $j \in H$,

$$\bar{d} = \bar{c} - k(2^{\lambda_0}/2),$$

and

$$K = -(2^{\lambda_0}/2) \sum_{j \in H} \gamma_j \bmod q.$$

We now seek solutions $d_j \in [-2^{\lambda_0 - 1}, 2^{\lambda_0 - 1} - 1]$ to satisfy the system

$$K \equiv \sum_{j \in H} d_j \, \gamma_j \pmod{q}, \tag{4.1b}$$

and

$$\bar{d} \equiv \sum_{j \in H} d_j \pmod{2^{\lambda_0}}. \tag{4.2b}$$

To find such solutions, we consider the $(k+3)$-dimensional integer lattice M:

$$M = \left[\begin{array}{c|ccc|c|c} X & & & & -KY & -\bar{d}Y \\ \hline & 1 & & & \gamma_1 Y & Y \\ & & 1 & & \gamma_2 Y & Y \\ & & & \ddots & \vdots & \vdots \\ & & & 1 & \gamma_k Y & Y \\ \hline & & & & qY & \\ & & & & & 2^{\lambda_0} Y \end{array}\right],$$

where X and Y will be integers suitably chosen below.

If $\vec{v} = \langle X, d_1, d_2, \ldots, d_k, 0, 0\rangle$ is a vector in M (that is, an integer linear combination of the rows of M), then $\{d_j \mid j \in [1, n]\}$ forms a required solution to Equation (4.1b) and Equation (4.2b) since, letting M_j denote the j^{th} row of M (counting from 0), we observe that

$$\vec{v} = M_0 + d_1 M_1 + \cdots + d_k M_k + aM_{k+1} + bM_{k+2}$$

for some integers a and b. We thus use the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [LLL82] to produce a reduced basis M' for M, which should enable us to find a vector of the above form.

Let B denote the set of rows in the reduced basis M' for which the first entry is nonzero and the last two entries are both zero. Because the rows of M' span the rows of M, the greatest common divisor of the leading component of each row of M' is certainly X. Our hope is that, restricting our consideration to just the rows in B, we find that the greatest common divisor of the leading elements is still X; if so, we can use the extended Euclidean algorithm [MvOV96; Algorithm 2.107] to find a linear combination of the rows in B whose leading entry is X. This will be our row $\vec{v}$, which contains our desired result.

We next very roughly compute the expected size of our solution. The determinant of M (and thus also of M') is $D = XY^2 q 2^{\lambda_0}$, and its dimension is $k + 3$. By choosing

$$X = Y = \lceil \sqrt[k]{q2^{\lambda_0}} \rceil,$$

we get that

$$\begin{aligned} D^{(k+3)^{-1}} &\approx \sqrt[k]{q2^{\lambda_0}} \\ &\approx X. \end{aligned}$$

We thus heuristically expect LLL to give us basis vectors whose entries are around the size of X, or not too much larger. This means the size of the coefficients produced by the extended Euclidean algorithm step will also be small, as they will be about the size of the ratio of the leading elements of the basis vectors and X. Therefore, the size of our resulting d_j, $j \in H$, should also be about

the size of X, or just a little larger. Let $k_0 = (\lg q)/\lambda_0 + 2$. When $k \geq k_0$,

$$\begin{aligned} X \approx \sqrt[k]{q2^{\lambda_0}} &\leq 2^{\lambda_0}/(2^{\lambda_0/k_0}), \\ &= 2^{\lambda_0(1-k_0^{-1})} \end{aligned}$$

and we expect our solution to be in the required range.

We tested the attack using the implementation of LLL in version 6.1.1 of the mathematical software package Sage[44]. When $q = 2^{256} - 189$ (the largest 256-bit prime), and $\lambda_0 = 40$, we find that setting $k = 8$ yields a solution to Equation (4.1b) and Equation (4.2b) in the appropriate range almost every time (we observed only one failure in 10,000 trials). Solving the problem took about 12 ms per trial, on average. When we tried $k = 7$, the attack never succeeded (which is what we expect, as for $k < 8$ we do not expect that a solution even *exists* in the required range, by the counting argument in Section 4.2).

[44] http://www.sagemath.org/

Appendix C

Zero-knowledge protocols for $\text{PolyCommit}_{\text{DL}}$

This appendix presents three new systems for honest-verifier zero-knowledge proofs and arguments of knowledge regarding committed polynomials. Given any positive integer-valued function $n(\tau) \in \mathit{poly}(\tau)$ satisfying $n(\tau) < \sqrt{2^\tau}/2$ for each $\tau \in \mathbb{N}^+$, we call an interactive protocol (P, V) a system for proofs (or arguments) of knowledge regarding $\text{PolyCommit}_{\text{DL}}$ if the appropriate soundness, simulatability, and extractability criteria hold asymptotically with respect to $\textsc{Poly}_\tau(n(\tau)) \leftarrow \text{PK-Init}(1^\tau, n(\tau))$ as τ tends to infinity.

C.1 Proving knowledge of a point on a committed polynomial

In an appendix to the extended version of their paper on polynomial commitments [KZG10b; Appendix E], Kate, Zaverucha, and Goldberg proposed a system for honest-verifier zero-knowledge proofs of knowledge of a point on a committed polynomial. More precisely, they proposed a system for honest-verifier zero-knowledge proofs of knowledge of an evaluation witness $w_i \in \widetilde{\mathbb{G}}$ and a *non-zero* scalar $\gamma_i \in \mathbb{Z}_p^*$ such that $e(\mathscr{C}/\tilde{g}^{\gamma_i}, \tilde{g}) = e(w_i, \tilde{g}^\alpha/\tilde{g}^i)$, where the commitment $\mathscr{C} \in \widetilde{\mathbb{G}}$, input $i \in \mathbb{Z}_p \setminus \{\alpha\}$, and the $\text{PolyCommit}_{\text{DL}}$ public parameters $\textsc{Poly}_\tau(n) = ((\widetilde{\mathbb{G}}, \mathbb{G}_\mathrm{T}, p, \tilde{g}, e), (\tilde{g}^{\alpha^j} \mid j \in [1, n]))$ are each part of the common input.

In Section C.2 below, we propose a system for honest-verifier zero-knowledge arguments of knowledge of a polynomial $f \in \mathbb{Z}_p[x]$ such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$. The latter protocol follows from the observation that, if P^* knows the witness-evaluation pair (w_i, γ_i) for a random, *verifier-selected* index $i \in_{\mathrm{R}} \mathbb{Z}_p$, then, with probability overwhelming in τ, P knows a polynomial $f \in \mathbb{Z}_p[x]$ such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$. This is because, under the polyDH assumption, all committed polynomials have degree at most n; hence, knowledge of any $n + 1$ points on a committed polynomial is sufficient to interpolate it.

Of course, we cannot simply let V^* challenge P to prove knowledge of an arbitrary point on f using Kate et al.'s protocol, since doing so might leak information about f's roots. If honest V selects its challenges $i \in_{\mathrm{R}} \mathbb{Z}_p$ uniformly at random, then with probability overwhelming in τ, we have that $f(i) = \gamma_i$ is non-zero and the resulting protocol is indeed honest-verifier statistical zero-knowledge. However, if dishonest V^* conjectures that $f(i) = 0$ for some particular input i (which may or may not be feasible to do, depending on the application), then it can use P as an oracle to confirm or reject this hypothesis. We could dismiss such attacks as being outside of the threat model; after all, absent the attack, we must still assume an honest verifier to prove the zero-knowledge property. Nevertheless, we feel that the simplicity and devastating power of this attack makes Kate et al.'s protocol unpalatable as the subprotocol in our argument of knowledge of a committed polynomial.

We therefore propose our own protocol, which is slightly more efficient than Kate et al.'s protocol and, importantly, does not require that the evaluation $f(i) = \gamma_i$ is non-zero. As a trade off, our protocol *does* require that $\mathscr{C}/\tilde{g}^{f(i)} \neq 1$. On the one hand, if f is non-constant, then this is true with probability overwhelming in τ, regardless of the strategy cheating V^* employs to select $i \in \mathbb{Z}_p \setminus \{\alpha\}$; on the other hand, however, if f is constant, then $\mathscr{C}/\tilde{g}^{f(i)}$ is *always* equal to 1. Hence, the new protocol is suitable for proving knowledge of points on a committed, *non-constant* polynomial. The protocol is depicted in Figure C.1. The idea is for P to choose a random exponent $r \in \mathbb{Z}_p^*$ with which to *blind* the witness as $w_i' = w_i^{1/r}$, and then engage V in the protocol denoted in Camenisch-Stadler notation by $\mathrm{PK}\{(r, \gamma_i) : e(\mathscr{C}, \tilde{g}) = e(w_i', \tilde{g}^{\alpha}/\tilde{g}^{i})^{r} \cdot e(\tilde{g}, \tilde{g})^{\gamma_i}\}$. We can implement the latter interaction using a slightly modified instance of Brands' system for honest-verifier zero-knowledge proofs of knowledge of a 2-DLREP in $\overline{\mathbb{G}}$ (see Figure 2.6 on page 35).

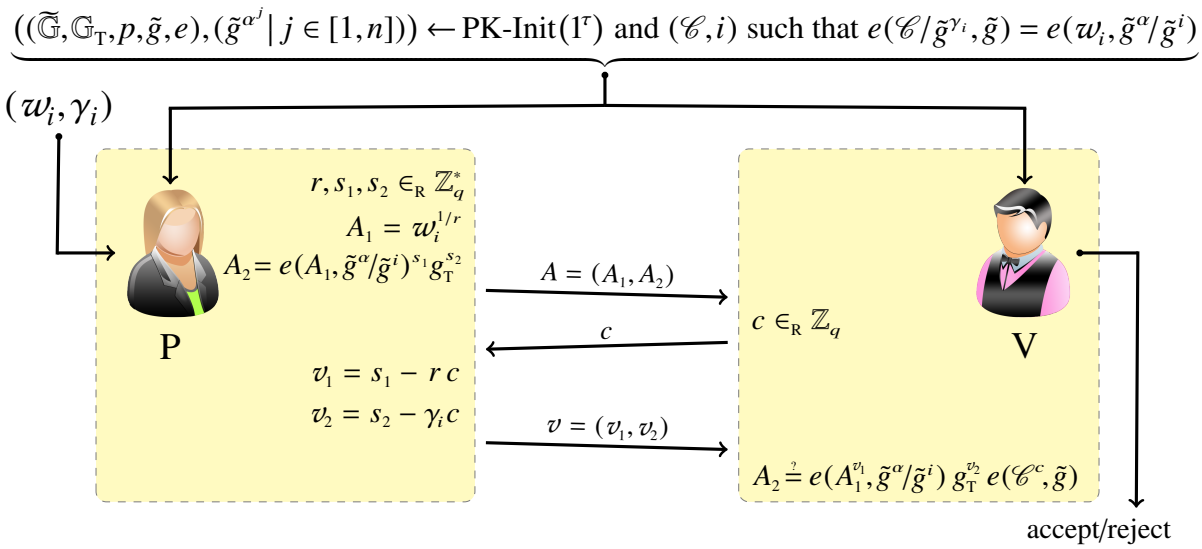

Figure C.1: A system for honest-verifier statistical zero-knowledge proofs of knowledge of a witness-evaluation pair for the $\text{PolyCommit}_{\text{DL}}$ commitment $\mathscr{C}$. In the verification equation, g_T is used as shorthand for $e(\tilde{g},\tilde{g})$. The protocol is c-simulatable and 2-extractable. It is denoted in Camenisch-Stadler notation by $\text{PK}\{(w_i,\gamma_i) : e(\mathscr{C}/\tilde{g}^{\gamma_i},\tilde{g}) = e(w_i,\tilde{g}^\alpha/\tilde{g}^i)\}$.

Theorem C.1. *The protocol depicted in Figure C.1 is a system for honest-verifier statistical zero-knowledge proofs of knowledge for the language of witness-evaluation pairs for a given non-constant, PolyCommit_{DL}-committed polynomial. Furthermore, it is a c-simulatable and 2-extractable sigma protocol and so has knowledge error $\kappa(\text{Poly}_\tau(n),\mathscr{C},i) = 1/(p-1)$.*

We denote the new protocol by $\text{PK}\{(w_i,\gamma_i) : e(\mathscr{C}/\tilde{g}^{\gamma_i},\tilde{g}) = e(w_i,\tilde{g}^\alpha/\tilde{g}^i)\}$. As should be clear from this notation, an accepting proof does *not* convince V that P^* knows a polynomial-evaluation pair (f,γ_i) such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$ and $f(i) = \gamma_i$; rather, it proves that P^* knows a witness-evaluation pair (w_i,γ_i) that, together with the public values $(\mathscr{C},i)$ and $\text{Poly}_\tau(n)$, satisfies V's the appropriate $\text{PolyCommit}_{\text{DL}}$ verification equation. In particular, P^* only requires the pair (w_i,γ_i), and not (f,γ_i), in its private auxiliary input.

Note that the protocol is sound even if f is constant; however, in this case, we have that w_i and, consequently, w_i', are both equal to 1 and the zero-knowledge property fails to hold. To extend the protocol from a system for non-constant polynomials to a system for *non-zero* polyno-

mials, we could implement the protocol denoted in Camenisch-Stadler notation by $\mathrm{PK}\{(w_i, \gamma_i) : e(\mathscr{C}/\tilde{g}^{\gamma_i}, \tilde{g}) = e(w_i, \tilde{g}^{\alpha}/\tilde{g}^{i}) \vee \mathscr{C} = \tilde{g}^{\gamma_i}\}$, using the protocol depicted in Figure C.1 and Schnorr's protocol. We emphasize that if $\mathscr{C}$ is a $\mathrm{PolyCommit}_{\mathrm{DL}}$ commitment opening to the zero polynomial, then $\mathscr{C} = 1$; conversely, if $\mathscr{C} = 1$, then, with probability overwhelming in τ, $\mathscr{C}$ opens to the zero polynomial. In particular, the new protocol is suitable for proving knowledge of *any* polynomial that can be "usefully" committed to in $\mathrm{PolyCommit}_{\mathrm{DL}}$.

C.2 Proving knowledge of a committed polynomial

Consider a variant of the protocol depicted in Figure C.1 that starts with an additional step in which V arbitrarily chooses the index $i \in_{\mathrm{R}} \mathbb{Z}_p \setminus \{\alpha\}$ with respect to which P must prove knowledge of the witness-evaluation pair (w_i, γ_i). Under the SDH and polyDH assumptions, a PPT prover P^* can open a given commitment $\mathscr{C}$ to evaluations consistent with at most one polynomial and, moreover, the degree of this polynomial can be at most n. We now prove that this variant yields a system of honest-verifier statistical zero-knowledge arguments of knowledge of a non-zero polynomial f such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$, with the arguments being computationally convincing under the SDH and polyDH assumptions. Completeness of the augmented protocol follows immediately from the completeness of sub-protocol and the fact that f is non-zero. The augmented protocol is clearly honest-verifier statistical zero-knowledge: a simulator for V merely chooses $i \in_{\mathrm{R}} \mathbb{Z}_p \setminus \{\alpha\}$, then invokes the simulator for V in the sub-protocol to produce a perfect simulation whenever $\deg f = 1$ or $f(i) \neq f(\alpha)$. Now, if a PPT prover P^* does *not* know a (non-zero) polynomial f such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$, then the probability that it knows an (w_i, γ_i) pair to satisfy verifier V's challenge index $i \in_{\mathrm{R}} \mathbb{Z}_p \setminus \{\alpha\}$ is at most about n/p under the SDH and polyDH assumptions. The restriction $n < \sqrt{p}$ implies that $n/p < 1/\sqrt{p}$ so that the augmented protocol's knowledge error is negligible in τ. Thus, if P^* convinces V with non-negligible probability in τ, then a knowledge extractor for the augmented protocol can invoke the knowledge extractor for the sub-protocol an expected polynomial number of times to extract $\deg f + 1$ distinct points on f, and then use polynomial interpolation to compute f from these extracted points. Note that the DL assumption, which is implied by both the SDH and polyDH assumptions, ensures that each extracted point is on the *same* polynomial, which is crucial for the extractor's expected running time to be polynomial in $n \in \mathit{poly}(\tau)$. We denote the augmented protocol by $\mathrm{PK}\{ f : \mathscr{C} = \tilde{g}^{f(\alpha)} \}$.

C.3 Proving that a committed polynomial has degree $d < n$

We now describe a simple extension to the above system for arguments of knowledge of a committed polynomial, which proves that the degree of that polynomial is at most $d < n$. Soundness of this protocol requires that the $\text{PolyCommit}_{\text{DL}}$ reference string PK bounds the degree of committed polynomials by $n < \sqrt{p}/2$ (which, in practice, is not a restriction on n since $\sqrt{p}/2$ is still super-polynomial in τ).

Under the polyDH assumption, it is infeasible for a non-trapdoor PPT P^* to output $(f', \tilde{g}^{f'(\alpha)})$ for any polynomial $f' \in \mathbb{Z}_p[x]$ with $n < \deg f' < \sqrt{p}$. In particular, if P^* knows a polynomial f such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$ and $d < \deg f \leq n$, then it can exhibit a commitment $\mathscr{C}_d$ to $f_d(x) = x^{n-d} f(x)$ with at most negligible probability in τ. The restriction $n < \sqrt{p}/2$ is necessary to ensure that $\deg f_d = \deg f + (n - d)$ does not exceed $\sqrt{p}$. Protocol C.2 is a noninteractive *trapdoor perfect zero-knowledge* argument (i.e., it is zero-knowledge with respect a *trapdoor verifier* who knows α) that exploits the above observation to prove that the committed polynomial $f(x) = \sum_{j=0}^{n} a_j x^j$ has degree at most $d < n$.

Protocol C.2 (Proof that a committed polynomial has degree at most d).

Common input:	$((\widetilde{\mathbb{G}}, \mathbb{G}_{\text{T}}, p, \tilde{g}, e), (\tilde{g}^{\alpha^j} \mid j \in [1,n])) \leftarrow \text{PK-Init}(1^\tau, n(\tau))$ and $\mathscr{C} \in \widetilde{\mathbb{G}}$
P's private input:	$f = \sum_{j=1}^{d} a_j x^j \in \mathbb{Z}_p[x]$ such that $\mathscr{C} = \tilde{g}^{f(\alpha)}$

P1: Compute $\mathscr{C}_d = \prod_{j=0}^{d} (\tilde{g}^{\alpha^{n-d+j}})^{a_j} = \tilde{g}^{f'(\alpha)}$ using the appropriate values from $\text{Poly}_\tau(n)$, and then send $\mathscr{C}_d$ to V.

PV2: Engage in the protocol denoted by $\text{PK}\{ f : \mathscr{C} = \tilde{g}^{f(\alpha)} \}$.

V3: Output 1 if V accepts in Step 2 and if $e(\mathscr{C}, \tilde{g}^{\alpha^{n-d}}) = e(\mathscr{C}_d, \tilde{g})$, and output 0 otherwise.

We denote Protocol C.2 $\text{PK}\{ f : \mathscr{C} = \tilde{g}^{f(\alpha)} \wedge \deg f \leq d \}$. It is straightforward to convert Protocol C.2 from a noninteractive trapdoor zero-knowledge argument into a (non-trapdoor, interactive) honest-verifier zero-knowledge argument by, for example, having P choose $r \in_{\text{R}} \mathbb{Z}_p^*$ and blind $\mathscr{C}'$ as $\mathscr{C}'' = (\mathscr{C}')^{1/r}$. P can then use Schnorr's proof of knowledge of a discrete logarithm [Sch89] in the target group $\mathbb{G}_{\text{T}}$ to prove knowledge of r such that $e(\mathscr{C}'', \tilde{g}) = e(\mathscr{C}, \tilde{g}^{\alpha^{n-d}})^r$. Fortunately, the simpler trapdoor zero-knowledge variant in Protocol C.2 suffices for the security of our all-but-k constructions. (In particular, our constructions only require that the argument be

witness indistinguishable [FS90; Definition 3.1], which is the case because $\text{PolyCommit}_{\text{DL}}$ hides committed polynomials unconditionally; i.e., even with respect to a computationally unbounded adversary that can compute the trapdoor.)