Batch Proofs of Partial Knowledge

Ryan Henry and Ian Goldberg

Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1

{rhenry, iang}@cs.uwaterloo.ca

Abstract. This paper examines "batch zero-knowledge" protocols for communication- and computation-efficient proofs of propositions composed of many simple predicates. We focus specifically on batch protocols that use Cramer, Damgård, and Schoenmakers' *proofs of partial knowledge* framework (Crypto 1994) to prove propositions that may be true even when some of their input predicates are false. Our main result is a novel system for batch zero-knowledge arguments of knowledge and equality of k-out-of-n discrete logarithms. Along the way, we propose the first general definition for batch zero-knowledge proofs and we revisit Peng and Bao's batch zero-knowledge proofs of knowledge and equality of one-out-of-n discrete logarithms (Inscrypt 2008). Our analysis of the latter protocol uncovers a critical flaw in the security proof, and we present a practical lattice-based attack to exploit it.

Keywords: Batch proof and verification, zero-knowledge, cryptanalysis, lattice-based attacks, efficiency.

1 Introduction

An interactive zero-knowledge proof is a conversation between two mutually distrusting parties—a *prover* and a *verifier*—in which the prover tries to convince the verifier that some proposition is true. The prover holds evidence (e.g., an NP witness) but is unwilling to reveal it to the verifier; the verifier, conversely, is skeptical of the prover and needs convincing. What makes a zero-knowledge proof special, therefore, is how much extra information the verifier learns: in a zero-knowledge proof, the verifier learns *nothing* beyond the veracity of the prover's claim. Zero-knowledge proofs have had profound implications for cryptography since Goldwasser, Micali, and Rackoff introduced them back in 1985 [21]; indeed, they are integral to many cryptographic protocols in the literature ranging from end-to-end verifiable voting schemes [12, 25], through to anonymous blacklisting and reputation systems [2, 3, 34], protocols for priced symmetric private information retrieval [24], threshold ring signatures [35], verifiable mix networks [22, 31], and cryptographic auctions [9], among others.

Alas, zero-knowledge does not come free. Each application above gives rise to at least one proposition whose "fan-in" scales with a critical system parameter (e.g., the global user count [12, 25, 35] or the size of a database [2, 3, 24, 34]).

Take for example a prover and a verifier that share generators g, h and a set of n pairs of group elements $\{(g_i, h_i) = (g^{x_i}, h^{y_i}) \mid i \in [1, n]\}$ from some suitably chosen group $\mathbb{G}$. The prover wishes to prove a proposition about this entire *batch* of predicates, such as "For each $i \in [1, n]$, I know $x_i \in \mathbb{Z}_{|\mathbb{G}|}$ such that $\log_g g_i = \log_h h_i = x_i$" or "For some $i \in [1, n]$, I know $x_i \in \mathbb{Z}_{|\mathbb{G}|}$ such that $\log_g g_i = \log_h h_i = x_i$". The first ("AND") proposition naturally arises, e.g., in universally verifiable shuffling protocols for mix networks [31, §2.3], while the second ("OR") proposition arises, e.g., in coercion-resistant Internet voting when each voter must prove that she appears on the election roster [12, §3.5]. In both cases, the standard zero-knowledge protocols scale linearly with the number of inputs n: the prover and verifier each compute $\Theta(n)$ full-length exponentiations in $\mathbb{G}$, the verifier sends $\Theta(1)$ group elements to the prover, and the prover sends $\Theta(n)$ group elements to the verifier. In 1998, Bellare, Garay, and Rabin [4, 5] suggested *batch verification* techniques to reduce verification costs. Their "small exponents" batch verification test [5, §3.3] reduces the verifier's computation cost to just $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications in $\mathbb{G}$. The quantity λ is a *soundness parameter*; perhaps $\lambda = 40$ or 60 in practice.

Inspired by Bellare et al.'s small-exponent batch test, Peng, Boyd, and Dawson [30, §4.1] proposed a four-round *batch proof* of (complete) knowledge for the above "AND" proposition. The prover and verifier each compute just $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications in $\mathbb{G}$, the prover sends $\Theta(1)$ group elements to the verifier, and the verifier sends $\Theta(1)$ group elements and $\Theta(\lambda n)$ additional bits to the prover. More recently, Peng and Bao [28, §5.1] proposed a four-round batch proof of *partial knowledge* for the above "OR" proposition, which blends small-exponent batch testing with a special case of Cramer, Damgård, and Schoenmakers' proofs of partial knowledge [13]. The prover and verifier each compute just $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications in $\mathbb{G}$, and they each send and receive just $\Theta(1)$ group elements and $\Theta(\lambda n)$ additional bits. A handful of other papers [8, 16, 22, 24, 29, 31] propose similar "batch proofs" with similar "sublinear" costs.

Our contributions.

1. We propose a novel system for batch zero-knowledge arguments of knowledge and equality of k-out-of-n discrete logarithms for fixed $k \in [1, n]$. As special cases, we obtain batch "AND" proofs (n-out-of-n) and batch "OR" proofs (one-out-of-n). Our protocol has similar costs to the protocols of Peng et al. [30, §4.1] and of Peng and Bao [28, §5.1].
2. We present a practical, lattice-based attack on the soundness of Peng and Bao's protocol for batch zero-knowledge proofs of knowledge and equality of one-out-of-n pairs of discrete logatarithms. We provide a fix that uses *all-but-k mercurial commitments* [23], a variant of mercurial vector commitments [11] with a strengthened binding property.
3. We propose formal definitions for *batch zero-knowledge* proofs and proofs of knowledge. Prior treatments of batch proofs have been informal and ad hoc. Our new definitions address this with a *conciseness* property describing

the asymptotic performance of a "batch" protocol relative to one formed by sequential composition of corresponding single-instance protocols.

Outline. We examine Peng and Bao's [28, §5.1] batch "OR" protocol in §2 and describe a practical attack on its soundness (the full details of which are in Appendix A). In §3, we discuss using *all-but-k mercurial commitments* [23] to repair Peng and Bao's protocol. In §4, we propose our formal definitions for batch zero-knowledge proofs and batch zero-knowledge proofs of knowledge. Our new batch protocol follows in §5 (and provide a security proof in Appendix B). We list some potential applications in §6 and conclude in §7.

2 Peng and Bao's batch proof protocol

The following protocol is due to Peng and Bao [28, §5.1]; they call it "batch ZK proof and verification of 1-out-of-n equality of logarithms". The protocol incorporates Bellare et al.'s small-exponent batch testing [5, §3.3] into both the proof and verification phases of an otherwise standard sigma protocol for proving knowledge and equality of one-out-of-n pairs of discrete logarithms. Both the prover P and verifier V know the same two generators g, h of an order-p group $\mathbb{G}$ and a set of n pairs of group elements $\big\{(g_i, h_i) \mid i \in [1, n]\big\}$, but only P knows an index $j \in [1, n]$ and exponent $x_j \in \mathbb{Z}_p^*$ such that $\log_g g_j = \log_h h_j = x_j$. The goal of the protocol is for P to convince V that she knows such a (j, x_j) pair without revealing any additional information. For ease of notation below, we define $H = [1, n] \setminus \{j\}$ for the (j, x_j) pair that honest P is proving knowledge of. We also introduce a *soundness parameter* $\lambda \in \mathbb{N}$, which tunes the cost versus soundness trade off in small-exponent batch testing.

Protocol 1. (Peng & Bao's Batch Proof of Partial Knowledge [28, §5.1]).

V1: Choose $t_i \in_{\mathrm{R}} \big[0, 2^\lambda - 1\big]$ for each $i \in [1, n]$. Send $(t_1, \ldots, t_n)$ to P.

P2: Receive $(t_1, \ldots, t_n)$ from V. Choose $r \in_{\mathrm{R}} \mathbb{Z}_p^*$ and $c_i \in_{\mathrm{R}} \big[0, 2^\lambda - 1\big]$ for each $i \in H$. Compute $a = g^r \prod_{i \in H} g_i^{c_i t_i}$ and $b = h^r \prod_{i \in H} h_i^{c_i t_i}$. Send (a, b) to V.

V3: Receive (a, b) from P. Choose $c \in_{\mathrm{R}} \big[0, 2^\lambda - 1\big]$ and send it to P.

P4: Receive c from V. Compute $c_j = c - \sum_{i \in H} c_i \bmod 2^\lambda$ and $v = r - t_j c_j x_j \bmod p$. Send $(c_1, \ldots, c_n, v)$ to V.

V5: Receive $(c_1, \ldots, c_n, v)$ from P. Output "true" if and only if $a \stackrel{?}{=} g^v \prod_{i=1}^n g_i^{c_i t_i}$, $b \stackrel{?}{=} h^v \prod_{i=1}^n h_i^{c_i t_i}$, and $c \stackrel{?}{\equiv} \sum_{i=1}^n c_i \pmod{2^\lambda}$, otherwise output "false".

Some remarks about Protocol 1 are in order. Perhaps surprisingly, we observe that V speaks before P does. What V sends to P in Step V1 is a list of short exponents for small-exponent batch testing. Step P2 ostensibly forces P to commit to an index j (such that $H = [1, n] \setminus \{j\}$) and to $\{c_i \mid i \in H\}$; if so, then V choosing $c \in_{\mathrm{R}} \big[0, 2^\lambda - 1\big]$ in Step V3 is equivalent to V choosing the missing $c_j \in_{\mathrm{R}} \big[0, 2^\lambda - 1\big]$ for P in Step P4, which is exactly what we want for good soundness. It is trivial to verify that the protocol is *complete*; i.e., that honest P always convinces honest V. Theorem 1 in Peng and Bao's paper [28] states that "Soundness in [the above

protocol] only fails with an overwhelmingly small probability [in the soundness parameter λ]." Their soundness proof works by computing an upper bound of $1/2^\lambda$ on the probability that the verification equations hold if $\log_g g_j \neq \log_h h_j$, given the following two implicit assumptions: 1) P committed to $H = [1, n] \setminus \{j\}$ and to $\{c_i \mid i \in H\}$ in Step P2, and 2) V chose c and (hence, c_j) uniformly at random from $\left[0, 2^\lambda - 1\right]$ in Step V3. *However, it is easy to see that the pair (a, b) of "commitments" that P computes and sends to V in Step P2 does not bind her to using $H = [1, n] \setminus \{j\}$; hence, the first implicit assumption in Peng and Bao's soundness proof is not guaranteed to hold when P is dishonest.* Dishonest P can exploit this observation to pass the verification equations even when the claimed equality of logarithms is false.

We give a high-level description of the attack below; interested readers can find further details in Appendix A.

Overview of the attack. Suppose that P knows several (x_j, y_j) pairs such that $(g_j, h_j) = (g^{x_j}, h^{y_j})$ but $x_j \not\equiv y_j \pmod p$ for any of the known pairs. Partition the interval $[1, n]$ into two sets H and S, where S is a subset of indices for which P knows the above pair of discrete logarithms and H is a superset of indices for which she does not. (Note that in some reasonable settings P may know *every* such pair.) In Step P2, P computes (a, b) using this new H so that when V sends c to P in Step V3, P has $|S| - 1$ extra degrees of freedom to compute her response in Step P4. In particular, to find the missing $\{c_j \mid j \in S\}$ she solves the following system of two linear equations in $k = |S|$ unknowns:

$$0 \equiv \sum_{j \in S} c_j t_j \left(x_j - y_j\right) \pmod p, \text{ and} \tag{1}$$

$$c' \equiv \sum_{j \in S} c_j \pmod{2^\lambda}, \tag{2}$$

where $c' = c - \sum_{i \in H} c_i \bmod 2^\lambda$. Equation (1) implies $\sum_{j \in S} c_j t_j x_j \equiv \sum_{j \in S} c_j t_j y_j \pmod p$; hence, if P sets $v = r - \sum_{j \in S} c_j t_j x_j \bmod p$ in Step P4, then $(c_1, \ldots, c_n, v)$ will satisfy each verification equation in Step V5. Of course, if P just naively solves the above system of equations and obtains a solution $\{c_j \mid j \in S\}$ containing $c_{j'} \geq 2^\lambda$ for some $j' \in S$, then V may notice that P is cheating. Therefore, what P really wants to do is find a solution to the above system subject to the additional restriction that $0 \leq c_j < 2^\lambda$ for all $j \in S$.

A counting argument suggests that such "suitably small" solutions are plentiful whenever $k \cdot \lambda$ is "sufficiently large" compared to $\lg p$.[1] If $\mathbf{X}$ is an instance of the above system induced by some real interaction between P and honest V, then we heuristically expect the distribution of solutions of $\mathbf{X}$ to be uniform among all possible $\langle c_{j_1}, \ldots, c_{j_k} \rangle \in \left(\mathbb{Z}_p\right)^k$; in particular, we expect the proportion of solutions that are suitably small to be about $\left(2^\lambda/p\right)^k$. Now, only p^{k-1} of

[1] Recall that $k = |S|$ is a lower bound on the number of exponent pairs that P knows and that λ is the soundness parameter. Larger values of λ are *supposed* to result in better soundness; however, what we find is just the opposite: larger values of λ only make suitably small solutions more numerous and easier for P to find.

the $\langle c_{j_1}, \ldots, c_{j_k} \rangle \in \left(\mathbb{Z}_p\right)^k$ can satisfy Equation (1), and of these only about $p^{k-1}/2^\lambda$ can simultaneously satisfy Equation (2). This leads us to conclude that $\mathbf{X}$ has around $(p^{k-1}/2^\lambda)\,(2^\lambda/p)^{\,k} = \left(2^\lambda\right)^{k-1}/p$ suitably small solutions. Appendix A 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 Lenstra-Lenstra-Lovász (LLL) [27], to find a suitably small solution quickly. For example, setting $\lambda = 40$ and letting $\lg p \approx 160$, P only needs to know about $k = 5$ exponent pairs to find a suitably small solution, on average.

3 All-but-k mercurial commitments

Our attack on Protocol 1 is possible because P can wait until *after* she sees the challenge c in Step P4 to choose $k > 1$ of the c_i. If the "commitment" in Step P2 actually bound P to using $H = [1, n] \setminus \{j\}$ and $\{c_i \mid i \in H\}$, then Peng and Bao's upper bound of $1/2^\lambda$ on the protocol's soundness error would hold. For a direct fix, we therefore desire a special commitment that will (i) force P to commit to all but one component of $\langle c_1, \ldots, c_n \rangle$ in Step P2 and (ii) let P specify an arbitrary value for the missing component—*without betraying its position*—when she opens the commitment in Step P4. This, informally, is the binding and hiding guarantees that all-but-k mercurial commitments [23] provide when $k = 1$. More generally, an all-but-k mercurial commitment allows P to commit to an *arbitrary subset* of $n - k$ components from a length-n vector so that she is bound to these $n - k$ components but is still free to choose the k unspecified components prior to opening. V does not learn which components P chose before committing and which she chose after committing; V does, however, learn the total number 'k' of non-committed components in the opening.

We refer the reader to Henry and Goldberg's paper [23] for a more comprehensive exposition of all-but-k mercurial commitments, including formal statements of the security properties. For our own purposes, we use an abridged notation that abstracts away certain technical details.

Informal definition. An *all-but-k mercurial commitment scheme* is a 4-tuple of probabilistic polynomial-time (PPT) algorithms (ABK-Init, ABK-Commit, ABK-Open, ABK-Verify) that work as follows:

- ABK-Init outputs a common reference string PK for use in the other protocols.
- $\text{ABK-Commit}_{\text{PK}}$ outputs commitments to vectors in which some subset of components is as-of-yet unspecified.
- $\text{ABK-Open}_{\text{PK}}$ opens such commitments to fully specified vectors, explicitly revealing the number of components k not bound by the commitment.
- $\text{ABK-Verify}_{\text{PK}}$ verifies the output of $\text{ABK-Open}_{\text{PK}}$, including the validity of k.

Repairing Peng and Bao's protocol. Given secure all-but-k mercurial commitments, it is straightforward to protect Protocol 1 from attacks like the one in §2. In Step P2, P commits to $\langle c_i \mid i \in H \rangle$ for $H = [1, n] \setminus \{j\}$. After V sends

c to P in Step V3, P computes the missing c_j as usual, then opens the above commitment to $\langle c_1, \ldots, c_n \rangle$ as part of Step P4, proving as she does so that she chose only one of the c_j after committing in Step P2. Constructing a simulator and extractor for this augmented protocol is simple (and we give explicit simulator and extractor constructions for the generalized version in Appendix B); furthermore, the augmented protocol is still intuitively a "batch" protocol provided the all-but-k scheme satisfies certain efficiency requirements. In §5, we let the parameter k vary and thereby generalize the repaired Peng-Bao protocol to a system for batch zero-knowledge arguments of knowledge and equality of k-out-of-n discrete logarithms for any $k \in [1, n]$. Our protocol (including the special case just outlined) appears to be the first such batch protocol for $k \neq n$.

4 Defining batch zero-knowledge proofs

Several papers (many of which we listed in the introduction [8, 16, 22, 24, 29, 31]) propose protocols that implement what their respective authors refer to as "batch zero-knowledge proofs (of knowledge)". Regrettably, the community has no agreed upon definition of what constitutes a "batch" zero-knowledge proof. Prior works, consequently, justify the terminology using ad hoc arguments that contrast the communication cost (counted in terms of group elements transfers) and computation cost (counted in terms of full-length exponentiations) of their protocols with those of the most "obvious" protocols to implement proofs of the same propositions. (Peng et al. did suggest one definition for batch zero-knowledge proofs [30, Definition 1]; however, their definition fails to address asymptotic communication and computation costs, which we believe to be the key property differentiating the abovementioned "batch" proofs from their "non-batch" counterparts.) We therefore offer our own, very general definition for batch zero-knowledge proofs (of knowledge). We model our new definition after the standard zero-knowledge definitions (specifically, [19, Definition 3] and [6, Definition 3.1]), but add a new parameterized *conciseness* criterion that places asymptotic restrictions on how the communication and the computation costs of the interaction scale with respect to the size of the proposition under consideration. In particular, our conciseness criterion characterizes the asymptotic relationship between the number of predicates under consideration, the *soundness (or knowledge) error* of the proof, and the communication and computation cost of the resulting interaction.

Formal model. We model our prover P and verifier V as a pair of interactive functions and consider the interaction $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ that occurs when both functions take $x = \langle x_1, \ldots, x_n \rangle$ as common input, P takes $y = \langle y_1, \ldots, y_n \rangle$ as private input, and V takes string z as private auxiliary input. In general, some (possibly trivial) subset of (x_i, y_i) pairs satisfy a given witness relation R and z encodes arbitrary prior knowledge of V, such as a set of transcripts from earlier interactions with P. (The *transcript* of an interaction $\big(\mathrm{P}_x(\cdot), \mathrm{V}_x(z)\big)$, which is denoted by $\mathrm{tr}_{\mathrm{P,V}}(x, z)$, is the string-valued random variable that records V's inputs and all correspondence with P up to the end of an interaction.)

We let φ_R be the function that maps pairs of n-tuples (x, y) as above to n-bit strings in which the i^{th} bit is 1 if and only if $(x_i, y_i) \in R$. Intuitively, we are interested in interactions $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ that implement zero-knowledge proofs of the proposition $p(x, y)$ induced by R and a given language L in the following sense: $p(x, y)$ is true if and only if $\varphi_R(x, y) \in L$. Note that the pair (L, R) uniquely determines the proposition p, and vice versa. For ease of notation below, we define the language of n-tuples induced by (L, R) as $L_R = \big\{x \mid \exists\, y \text{ for which } \varphi_R(x, y) \in L\big\}$. We parameterize the lengths of the x_i and y_i in a given interaction by τ; in particular, we assume throughout that the x_i are all τ bits long and the y_i are all $poly(\tau)$ bits long, where $poly(\cdot)$ is some fixed polynomial. Let $T = \big\{(\{0,1\}^\tau)^n \mid \tau, n \in \mathbb{N}_{\geq 1}\big\}$ denote set of n-tuples of fixed-length strings.

Example. For propositions $p(x, y)$ asserting knowledge and equality of discrete logarithms, as in the actual protocols we consider in this paper, $R = \big\{(x_i, y_i) = \big((g_i, h_i),\, y_i\big) \in \mathbb{G}^2 \times \mathbb{Z}_p \;\big|\; \log_g g_i = \log_h h_i = y_i\big\}$. If $p(x, y)$ is the "AND" proposition, then L is the language of strings comprised entirely of 1s; if $p(x, y)$ is the "OR" proposition, then L is the language of strings with nonzero Hamming weight. For our own k-out-of-n proofs, L is the language of strings with Hamming weight at least k. Note that in general P is proving *partial knowledge* of witnesses for R, with the strings in L reflecting which subsets of witnesses P might actually know.

We now recall the standard notions of a *simulator* for verifier V, which we use to formalize what it means for $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ to be "zero-knowledge", and of a *knowledge extractor* for P, which we use to formalize what it means for $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ to be a "proof of knowledge".

Definition 1. A probabilistic function S_{V^*} is a *simulator* for verifier-language pair (V^*, L_R) if the probability ensembles $\big\{\mathrm{tr}_{\mathrm{P},\mathrm{V}^*}(x, z)\big\}_{x \in L_R}$ and $\big\{S_{\mathrm{V}^*}(x, z)\big\}_{x \in L_R}$ are (computationally, statistically, or perfectly [18, Definitions 3,4]) indistinguishable, where $S_{\mathrm{V}^*}(x, z)$ is the string-valued random variable describing the output of S_{V^*} on input (x, z).

An *oracle machine* for P^* is a function E^{P^*} that is endowed with *rewinding black box oracle access* to P^*. In other words, E^{P^*} is able to 1) submit arbitrary challenges to P^* and get truthful responses in a single time step, and 2) "rewind" P^* to a previous state to get several responses for the same input and random coin flips but different challenges. (Note that E^{P^*} is generally *not* privy to P^*'s inputs or internal state.)

Definition 2. Let $\kappa\colon T \to [0, 1]$ and let $q(x)$ denote the probability that V outputs "true" in $\big(\mathrm{P}^*_x(y), \mathrm{V}_x(z)\big)$. An oracle machine E^{P^*} is a *knowledge extractor* (with knowledge error $\kappa(\cdot)$) for the prover-language pair (P^*, L_R) if there exists a positive polynomial $g(\cdot)$ such that, for all n-tuples $x \in L_R$, if $q(x) \geq \kappa(x)$ then, with probability at least $\frac{q(x) - \kappa(x)}{g(|x|)}$, $E^{\mathrm{P}^*}(x)$ outputs an n-tuple y' for which $\varphi_R(x, y') \in L$.

Given the above definitions, we formally define what it means for a pair of interactive functions to implement a system of *batch zero-knowledge proofs* or

a system of *batch zero-knowledge proofs of knowledge* for the language-relation pair (L, R). What sets our Definition 3 apart from the standard zero-knowledge definitions is that we include an explicit *conciseness* condition, which characterizes the cost of proving $\varphi_R\big((x_1, \ldots, x_n), (y_1, \ldots, y_n)\big) \in L$ in terms of the cost of proving $(x_1, y_1) \in R$.

For example, consider an interactive protocol A between P and V in which, on common input $x_0 \in \{0,1\}^\tau$, P convinces V that there exists (or, perhaps, that it "knows") some y_0 such that $(x_0, y_0) \in R$. Let $a_0(\tau)$ and $a_1(\tau)$ respectively denote the computation cost (for both P and V) and the bidirectional communication cost of A. Now, consider a second interactive protocol B between P and V in which, on common input $x \in (\{0,1\}^\tau)^n$, P convinces V that there exists (or it "knows") some y such that $\varphi_R(x, y) \in L$. Let $b_0(\tau, n)$ and $b_1(\tau, n)$ respectively denote the computation cost (for both P and V) and the bidirectional communication cost of B. For a fixed pair of constants $\alpha, \beta \in [0, 1]$, we say that B is (α, β)-*concise* if there exists a constant $\delta > 0$ such that, for all $\epsilon > 0$, we have

$$b_0(\tau, n) \in O\Big(n^\alpha a_0(\tau) + n^{\beta+\epsilon}\tilde{a}_0(\tau)\Big) \text{ for some } \tilde{a}_0(\tau) \in o\Big(a_0(\tau)^{1-\delta}\Big),$$

and

$$b_1(\tau, n) \in O\Big(n^\alpha a_1(\tau) + n^{\beta+\epsilon}\tilde{a}_1(\tau)\Big) \text{ for some } \tilde{a}_1(\tau) \in o\Big(a_1(\tau)^{1-\delta}\Big).$$

(That is, the computation cost and communication cost of B grow no faster than n^α times the corresponding cost of A plus at most about n^β times some function that grows at least polynomially slower than the corresponding cost of A as τ grows large. The ϵ and δ factors are present so that we may ignore the contribution of polylogarithmic terms.) Recalling that $\alpha, \beta \in [0, 1]$, we call B a *batch proof (of knowledge)* for (L, R) if it is (α, β)-concise for any $\alpha < 1$, or, very roughly, if the cost of the protocol grows slower than n times the cost of A as we let both n and τ tend to infinity.

Example.

1. Consider Peng et al.'s protocol for proofs of complete knowledge [30, §4.1], our repaired version of Peng and Bao's protocol in §3 for proofs of partial knowledge, and the forthcoming protocol in §5 for proofs of partial knowledge. In each, we find that $a_0(\tau) \in \Omega\big(\tau^2 \lg^2 \tau\big)$ and $a_1(\tau) \in \Omega\big(\tau\big)$ while, for all $\epsilon > 0$ and for any soundness parameter $\lambda \in \mathbb{N}$, we find that $b_0(\tau, n) \in O\big(a_0(\tau) + n^{1+\epsilon}\lambda\,\tau \lg^2 \tau\big)$ and $b_1(\tau, n) \in O\big(a_1(\tau) + n\,\lambda\big)$.[2] For any fixed $\delta < 1/2$, we have that $\tilde{a}_0(\tau) = \lambda\,\tau \lg^2 \tau \in o\big(a_0(\tau)^{1-\delta}\big)$ and $\tilde{a}_1(\tau) = \lambda \in o\big(a_1(\tau)^{1-\delta}\big)$, and therefore each of these protocols is $(0, 1)$-concise; moreover, since $\alpha = 0 < 1$, each protocol satisfies our conciseness criterion for a batch proof of knowledge.

2. Brands, Demuynck, and De Decker describe a protocol [8, §3.4] to prove that a given commitment commits to a different value than every other commitment on a list. As in the previous example, we find that $a_0(\tau) \in \Omega\big(\tau^2 \lg^2 \tau\big)$ and $a_1(\tau) \in \Omega\big(\tau\big)$ but, in this case, we have $b_0(\tau, n) \in O\big(n^{1/2}\tau^2 \lg^2 \tau\big)$ and $b_1(\tau, n) \in O\big(n^{1/2}\tau\big)$. Thus, letting $\delta = 1/2$ and letting $\tilde{a}_0(\tau)$ and $\tilde{a}_1(\tau)$ be arbitrary constant functions,

[2] We assume here that multiplication in $\mathbb{G}$ requires $O\big(\tau \lg \tau \lg\lg \tau\big) \in O\big(\tau \lg^2 \tau\big)$ bit operations using the Fast Fourier Transform (FFT) [15].

we see that Brands et al.'s protocol is $(\frac{1}{2}, 0)$-concise; moreover, since $\alpha = \frac{1}{2} < 1$, it satisfies our conciseness criterion for a batch proof.

Definition 3. (System of batch zero-knowledge proofs (of knowledge)). Let $\Lambda\colon \mathbb{N} \to \mathbb{N}$ be a nondecreasing *soundness function* and let $\alpha, \beta \in [0,1]$ be constants such that $\alpha < 1$. An interactive protocol $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ is a *system of* (α, β, Λ)*-batch zero-knowledge proofs* for the language-relation pair (L, R) if there exists a negligible function $\varepsilon_0\colon \mathbb{N} \to \mathbb{R}$ for which $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ satisfies each of the following four conditions.

1. **Complete:** For any $n \in \mathbb{N}$ and pair (x, y) such that $\varphi_{R}(x, y) \in L$, if y is input to honest P and x is input to P and honest V, then V outputs "true".
2. **(Unconditionally) sound:** For every (possibly malicious) prover P^*, $\tau \in \mathbb{N}$, $n \in \mathbb{N}$, and $x \in (\{0,1\}^\tau)^n \setminus L_{R}$, if P^* and honest V receive x as common input then, with probability at least $1 - \varepsilon_0\big(\Lambda(\tau)\big)$, V outputs "false".
3. **(General) zero-knowledge:** For every (possibly malicious) PPT verifier V^*, there exists a PPT simulator S_{V^*} for (V^*, L_{R}).
4. **(α, β)-concise:** If $a_0(\tau)$ and $a_1(\tau)$ respectively denote the computation and communication cost of $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ when n is fixed as 1, then there exists some constant $\delta > 0$ and functions $\tilde{a}_0(\tau) \in o\big(a_0(\tau)^{1-\delta}\big)$ and $\tilde{a}_1(\tau) \in o\big(a_1(\tau)^{1-\delta}\big)$ such that, for every $\epsilon > 0$, we have that
   a. for every (possibly malicious) PPT verifier V^*, $\tau \in \mathbb{N}$, and pair (x, y) such that $\varphi_{R}(x, y) \in L$, if y is input to honest P and x is input to P and V^*, then P runs in $O\big(n^\alpha a_0(\tau) + n^{\beta+\epsilon}\tilde{a}_0(\tau)\big)$ time and sends $O\big(n^\alpha a_1(\tau) + n^{\beta+\epsilon}\tilde{a}_1(\tau)\big)$ bits to V; and
   b. for every (possibly malicious) prover P^*, $\tau \in \mathbb{N}$, and n-tuple x, if x is input to P^* and honest V, then V runs in $O\big(n^\alpha a_0(\tau) + n^{\beta+\epsilon}\tilde{a}_0(\tau)\big)$ time and sends $O\big(n^\alpha a_1(\tau) + n^{\beta+\epsilon}\tilde{a}_1(\tau)\big)$ bits to P^*.

If $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ additionally satisfies the following condition, then it is a system of (α, β, Λ)-batch zero-knowledge proofs *of knowledge* for (L, R).

5. **(Unconditionally) knowledge extractable:** There exists an oracle machine E and function $\kappa\colon T \to [0,1]$ such that, for every (possibly malicious) prover P^*, E^{P^*} is an expected PPT knowledge extractor for (P^*, L_{R}) with knowledge error $\kappa(\cdot) \le \varepsilon_0\big(\Lambda(\tau)\big)$.

We also consider the following two (standard) relaxations of the above definition. First, if $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ satisfies Conditions 1, 2, 4 (and 5) as stated above, but it only satisfies the weaker Condition 3b as stated below (instead of Condition 3), then it is a system of *honest-verifier* (α, β, Λ)-batch zero-knowledge proofs (of knowledge) for (L, R).

3b. (Honest-verifier) zero-knowledge: There exists a PPT simulator S_{V} for (V, L_{R}), where V is the *honest verifier*.

If there exists a negligible function $\varepsilon_1\colon \mathbb{N} \to \mathbb{R}$ for which $\big(\mathrm{P}_x(y), \mathrm{V}_x(z)\big)$ satisfies Conditions 1, 3[b], and 4 as stated above, but only satisfies the weaker Conditions

2b (and 5b) as stated below, then it is a computationally convincing system of [honest-verifier] (α, β, Λ)-batch zero-knowledge *arguments* (of knowledge) for (L, R).

2b. (Computationally) sound: For every (possibly malicious) PPT prover P^*, there exists a constant τ_0 such that, for every $\tau > \tau_0$ and $n \in \mathbb{N}$, if P^* and honest V receive $x \in (\{0,1\}^\tau)^n \setminus L_R$ as common input then, with probability at least $1 - \varepsilon_0\big(\Lambda(\tau)\big) - \varepsilon_1\big(\tau\big)$, V outputs "false".

5b. (Computationally) knowledge extractable: There exists an oracle machine E^{P^*} and function $\kappa\colon T \to [0,1]$ such that, for every (possibly malicious) PPT prover P^*, there exists a constant τ_0 such that, for every $\tau > \tau_0$ and $n \in \mathbb{N}$, E^{P^*} is an expected PPT knowledge extractor for (P^*, L_R) with knowledge error $\kappa(\cdot) \leq \varepsilon_0\big(\Lambda(\tau)\big) + \varepsilon_1\big(\tau\big)$.

5 Batch proof of knowledge and equality of k-out-of-n pairs of discrete logarithms

Our new protocol draws inspiration from the repaired version of Peng and Bao's protocol outlined in §3, but it improves on that protocol by letting k vary in the all-but-k mercurial commitments, which allows us to prove a more general class of propositions. More precisely, the new protocol generalizes from a system for proofs of knowledge and equality of one-out-of-n pairs of discrete logarithms to a system for arguments of knowledge and equality of k-out-of-n pairs of discrete logarithms for any $k \in [1, n]$. We defer a formal security analysis of the new protocol to Appendix B, wherein we prove that, for any fixed k and soundness parameter $\lambda \in \mathbb{N}$, it is a system for honest-verifier $(0, 1, \min\{\tau, \lambda\})$-batch zero-knowledge proofs of knowledge (in the sense of Definition 3). The latter analysis uses efficiency characteristics of the underlying construction for all-but-k mercurial commitments [23]. Using Henry and Goldberg's all-but-k mercurial commitment scheme [23], which is computationally binding under the n-Strong Diffie-Hellman assumption [7, §3], yields a system of honest-verifier batch zero-knowledge arguments. We note that in this particular instantiation, the prover is assumed to be computationally bounded not in the bit-length τ but in the security parameter for the all-but-k mercurial commitments. Standard tricks from the literature [20] can relax the honest-verifier assumption at only a small cost to efficiency. Swapping in unconditionally binding all-but-k mercurial commitments would yield a system of proofs rather than arguments.

Table 1 compares the cost of our protocol and those arising from a naive application of Cramer et al.'s framework [13], Peng and Bao's protocol [28], and Peng et al.'s protocol [30]. The latter three protocols are all systems for *proofs* of knowledge; ours is a system for *arguments* of knowledge. Observe that Peng et al.'s protocol is both sound and a batch protocol, *but it only handles the simple $k = n$ case*, and that Peng and Bao's protocol is a batch protocol and handles the interesting $k = 1$ case, *but it is not sound*. Cramer et al.'s framework is sound and handles every $k \in [1, n]$, *but it is not a batch protocol*.

Protocol	Communication	Computation	Concise	Batch?	k-out-of-n	Sound?
Cramer et al. [13]	$\Theta(\tau\, n)$	$\Theta(\tau\, n)$	$(1, 0)$	✗	$k \in [1, n]$	✓
Peng-Bao [28]	$\Theta(\tau + \lambda\, n)$	$\Theta(\tau + \lambda\, n)$	$(0, 1)$	✓	$k = 1$	✗
Peng et al. [30]	$\Theta(\tau + \lambda\, n)$	$\Theta(\tau + \lambda\, n)$	$(0, 1)$	✓	$k = n$	✓
This work	$\Theta(\tau + \lambda\, n)$	$\Theta(\tau + \lambda\, n \lg n)$	$(0, 1)$	✓	$k \in [1, n]$	✓

Table 1.

This table compares the communication cost (in bits) and the computation cost (in τ-bit multiplications) for four different protocols that each implement honest-verifier zero-knowledge proofs of knowledge and equality of k-out-of-n pairs of discrete logarithms for some k in a group with τ-bit order. The "**Concise**" column indicates the conciseness of the protocol (in the sense of Definition 3); the "**Batch?**" column indicates if the protocol satisfies our definition of a batch proof; the "**k-out-of-n**" column lists values of k that the protocol supports; the "**Sound?**" column indicates if the protocol achieves overwhelming soundness in the soundness parameter λ. Note that $\lambda = \tau$ in the protocol by Cramer et al.; for the other protocols, typically $\lambda \ll \tau$ and λ is fixed as the smallest value yielding a palatable soundness error.

5.1 The protocol

Suppose that $\mathbf{ABK} = (\text{ABK-Init}, \text{ABK-Commit}, \text{ABK-Open}, \text{ABK-Verify})$ is a secure all-but-k mercurial commitment scheme. Fix a soundness parameter $\lambda \in \mathbb{N}$ and use ABK-Init to generate a common reference string PK. Protocol 2 implements a system for batch zero-knowledge proofs or arguments of knowledge and equality of k-out-of-n pairs of discrete logarithms for any pair of nonnegative integers (k, n) with $k \leq n$ and $n \leq n_0$. In the protocol, we use $\mathbf{V}_q^{n \times k}$ to denote the column-wise $n \times k$ *rectangular Vandermonde matrix* with entries reduced modulo q:

$$\mathbf{V}_q^{n \times k} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & \cdots & 2^k \\ 1 & 3 & 3^2 & \cdots & 3^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & n & n^2 & \cdots & n^k \end{bmatrix} \bmod q.$$

Note that because q is prime with $n < q$ and $k \leq n$, every subset of k rows of $\mathbf{V}_q^{n \times k}$ has full rank and thus forms a non-singular (i.e., invertible) square matrix modulo q. If desired, one could replace $\mathbf{V}_q^{n \times k}$ with any other matrix that has this property when all arithmetic is modulo q.

The setting for the new protocol is similar to before. Both P and V know the same two generators g, h of an order-p group $\mathbb{G}$, the above-generated all-but-k reference string PK, and a set of $n \in [1, n_0]$ pairs of group elements $\big\{(g_i, h_i) \mid i \in [1, n]\big\}$, but only P knows a size-k subset $S \subseteq [1, n]$ of indices and corresponding set $x_S = \{x_j \in \mathbb{Z}_p \mid j \in S\}$ of exponents such that $\log_g g_j = \log_h h_j = x_j$ for all $j \in S$. The goal of the protocol is for P to convince V that she knows such a (S, x_S) pair without revealing any additional information. For ease of notation below, we let $H = [1, n] \setminus S$ for the (S, x_S) pair that honest P is proving knowledge of.

Intuitively, our k-out-of-n proof replaces the all-but-one mercurial commitment from the repaired Peng-Bao proof with an all-but-k mercurial commitment. P

commits to $\{c_i \mid i \in H\}$ in Step P2, thus assuring V that she can choose at most $k = |S|$ of the c_i after V sends the challenge in Step V3. Rather than challenge P to produce c_i that sum to c modulo q, V challenges P to produce c_i that obey a system of k non-degenerate linear constraints induced by $\mathbf{V}_q^{n \times k}$ and the k free components in $\langle c_1, \ldots, c_n \rangle$. V verifies the constraints in Step V5 by checking if $\langle c_1, \ldots, c_n \rangle \cdot \mathbf{V}_q^{n \times k} \stackrel{?}{\equiv} \langle c, 0, \ldots, 0 \rangle \pmod{q}$; the all-but-$k$ commitment ensures that P chose all but k of the c_i before she received c. This assures V that c uniquely determined a size-k subset of the c_i, although he learns no information about which subset. From here, P essentially uses Peng et al.'s batch "AND" proof for the size-k subset she is proving knowledge of, and "simulates" the proof for the remaining $n - k$ predicates, as in a standard proof of partial knowledge.

Protocol 2. (Generalized batch proof of partial knowledge).

V1: Choose $t_i \in_{\mathrm{R}} \left[0, 2^\lambda - 1\right]$ for each $i \in [1, n]$. Send $(t_1, \ldots, t_n)$ to P.

P2: Receive $(t_1, \ldots, t_n)$ from V. Choose $r \in_{\mathrm{R}} \mathbb{Z}_p^*$ and $c_i \in_{\mathrm{R}} [0, q-1]$ for $i \in H$. Compute $a = g^r \prod_{i \in H} g_i^{c_i\, t_i}$, $b = h^r \prod_{i \in H} h_i^{c_i\, t_i}$, and $\mathcal{C} \leftarrow \text{ABK-Commit}_{\text{PK}}(\langle c_1, \ldots, c_n \rangle)$. Send $(a, b, \mathcal{C})$ to V.

V3: Receive $(a, b, \mathcal{C})$ from P. Choose $c \in_{\mathrm{R}} [0, q-1]$ and send it to P.

P4: Receive c from V. Solve for $\vec{c} = \langle c_1, \ldots, c_n \rangle \in \mathbb{Z}_q^n$ such that $\vec{c} \cdot \mathbf{V}_q^{n \times k} \equiv \langle c, 0, \ldots, 0 \rangle \pmod{q}$, then compute $v = r - \sum_{j \in S} c_j\, t_j\, x_j \bmod p$ and $\omega \leftarrow \text{ABK-Open}_{\text{PK}}(\mathcal{C}, k, \vec{c})$. Send $(\vec{c}, v, \omega)$ to V.

V5: Receive $(\vec{c}, v, \omega)$ from P. Output "true" if and only if $a \stackrel{?}{=} g^v \prod_{i=1}^n g_i^{c_i\, t_i}$, $b \stackrel{?}{=} h^v \prod_{i=1}^n h_i^{c_i\, t_i}$, $\vec{c} \cdot \mathbf{V}_q^{n \times k} \stackrel{?}{\equiv} \langle c, 0, \ldots, 0 \rangle \pmod{q}$, and $\text{ABK-Verify}_{\text{PK}}(\mathcal{C}, \vec{c}, k, \omega) \stackrel{?}{=}$ "true"; otherwise, output "false".

As before, some remarks about this protocol are in order. Protocol 2 follows the same basic recipe as Protocol 1, with V starting the conversation in Step V1 by sending to P a list of short exponents for small-exponent batching. In fact, one easily sees by inspection that the repaired version of Protocol 1 is just the special case of Protocol 2 with k fixed to one. (The only difference being that the former protocol uses $q = 2^\lambda - 1$ since it does not require a prime q to guarantee linearly independent constraints.) Completeness holds trivially by inspection and constructing a simulator for honest V is equally straightforward. In Appendix B, we prove that using Henry and Goldberg's all-but-k mercurial commitment construction [23] in our protocol yields a system for honest-verifier $(0, 1, \min\{\lg p, \lambda\})$-batch zero-knowledge arguments of knowledge of a size-k subset $S \subseteq [1, n]$ of indices and corresponding set $x_S = \{x_j \in \mathbb{Z}_p \mid j \in S\}$ of exponents such that $\log_g g_j = \log_h h_j = x_j$ for all $j \in S$.

6 Applications

In the introduction, we listed the following example applications in which the need to prove propositions about large batches of predicates naturally arise: cryptographic voting [12,25], anonymous blacklisting and reputation systems [2,3], priced symmetric private information retrieval [24], threshold ring signatures [35],

verifiable mix networks [22, 31], and cryptographic auctions [9]. We now briefly discuss how our new protocol can directly speed up and extend three such constructions from the literature.

Symmetric private information retrieval. Henry, Olumofin, and Goldberg [24] describe a symmetric variant of Goldberg's information-theoretic private information retrieval protocol [17] that achieves data privacy by having each client commit to her query using polynomial commitments [26] and then exhibit a zero-knowledge proof that the committed query is "well formed". The final step in their proof—which dominates the computation cost of their enhancements and contributes considerable communication overhead to the protocol—is a proof of equality of one-out-of-r pairs of discrete logarithms, where r is the number of records in the database. The authors suggest small-exponent batch testing to speed up the verification at the database servers; however, playing the role of prover in that interaction still accounts for a significant fraction of a client's per-query computational expenditure. Simply swapping in our protocol leads to significant reductions in both the computation overhead and the communication overhead of their protocol.

Cryptographic voting systems. The *JCJ protocol* of Juels, Catalano, and Jakobsson [25] underlies a number of protocols for coercion-resistant, receipt-free verifiable Internet voting [1, 12, 36]; indeed, Spycher et al. [32] opine that "[JCJ is] the only known protocol for remote e-voting that offers individual verifiability and receipt-freeness simultaneously under somewhat acceptable trust assumptions". The bottleneck operation in JCJ is its *vote authorization* phase, which eliminates fake votes and duplicate votes prior to tallying. The computational cost of both steps grows quadratically in the number of votes cast: JCJ detects fake votes by having voters attach zero-knowledge proofs (made non-interactive via the *Fiat-Shamir heuristic* [14]) that they are on the registered voters roster, and it detects duplicate votes by employing a pairwise *plaintext equivalence test* on each vote. Several papers suggest strategies that can detect duplicate votes in linear time [1, 32, 36]; however, eliminating fake votes in linear time appears to necessitate a weakening the protocol's security guarantees. In particular, some existing schemes have voters prove membership within some smaller *anonymity set* rather than the entire roster [12, 32]. Our batch proof of partial knowledge may help to reduce the cost of this step and thereby allow for significantly larger anonymity sets; for smaller elections, it might even make the quadratic algorithm practical.

A second cryptographic operation that frequently arises in end-to-end verifiable voting systems is "proofs of re-encryption" of El Gamal ciphertexts: that is, given two sets of pairs $\{(g^{y_i}, m_i\, h^{y_i}) \mid i \in [1,n]\}$ and $\{(g^{y'_i}, m'_i\, h^{y'_i}) \mid i \in [1,n]\}$ of El Gamal ciphertexts encrypted under public key $h = g^x$, prove that $m_i = m'_i$ for all $i \in S$ (where S is a subset of indices suitably defined by the application). Such proofs work by considering the quotients $(m_i\, h^{y_i})/(m'_i\, h^{y'_i}) = (m_i/m'_i)\, h^{y_i - y'_i}$ and $g^{y_i}/g^{y'_i} = g^{y_i - y'_i}$ and noting that $\log_h (m_i/m'_i)\, h^{y_i - y'_i} = \log_g g^{y_i - y'_i}$ if and only if $m_i = m'_i$. Thus, batch proofs of knowledge and equality for k-out-of-n pairs of

discrete logarithms imply batch proofs of re-encryption of k-out-of-n El Gamal ciphertexts.

Anonymous blacklisting. Tsang et al. proposed *Blacklistable Anonymous Credentials* (BLAC) [34] to help online service providers revoke access from individual misbehaving users of an anonymous communications system. An out-of-band registration authority issues each BLAC user a single unconditionally hiding multi-show credential that encodes a secret pseudorandom identity u for that user. When user U with identity u wishes to request service from a service provider S, she chooses $a_0 \in_{\mathrm{R}} \mathbb{G}$ and prepares a token $\Gamma = (a_0, b_0) = (a_0, a_0^u)$ plus a noninteractive proof that the u encoded in Γ is the same as the u in her credential. S will use Γ to identify U for the session. If U misbehaves, S can append Γ to its blacklist $\mathcal{B} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$, indicating that U may not authenticate with S at this time. To complete the service request, U therefore proves that none of the tokens on $\mathcal{B}$ use the u from her credential. She does this with Camenisch and Shoup's protocol [10, §6] for honest-verifier zero-knowledge proofs of knowledge of a discrete logarithm that is not equal to some other (possibly unknown) discrete logarithm. The latter protocol works as follows: U chooses $r \in_{\mathrm{R}} \mathbb{Z}_q^*$, then for each $i \in [1, n]$, she computes and sends $A_i = \left(a_i^u / b_i\right)^{r_i}$ to S. S rejects if $A_i = 1$ for any $i \in [1, n]$ since, for honestly computed A_i, this happens if and only if $b_i = a_i^u$. U completes the proof by engaging S in a proof of knowledge of $(\alpha, \beta) = (ru, -r)$ such that $1 = a_0^\alpha b_0^\beta$ and $A_i = a_i^\alpha b_i^\beta$ for all $i \in [1, n]$.

The linear cost of the above proof is a bottleneck in BLAC. Bellare et al.'s small-exponent batch testing [5] can help: let S choose $t_1, \ldots, t_n \in_{\mathrm{R}} [0, q-1]$, then set $a = \prod_{i=1}^n a_i^{t_i}$, set $b = \prod_{i=1}^n b_i^{t_i}$, and set $A = \prod_{i=1}^n A_i^{t_i} = a^{ru} b^{-r}$, and have U prove knowledge of (α, β) such that $1 = a_0^\alpha b_0^\beta$ and $A = a^\alpha b^\beta$. Note that U must send each A_i separately so S can check individually if $A_i = 1$.

We can slightly modify the above protocol and combine it with Protocol 2 to get a novel batch protocol for BLAC with a d-strikes-out policy [34]. Our d-strikes-out BLAC protocol appears to be secure under a Diffie-Hellman type assumption in the random oracle model, though we leave a careful security analysis to future work. Suppose that U wishes to prove that she has $k < d$ tokens on $\mathcal{B} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$. She sets $S = \{i \mid b_i = a_i^u\}$ and $H = [1, n] \setminus S$ and chooses $r \in_{\mathrm{R}} \mathbb{Z}_q^*$, then for each $i \in H$, she computes $A_i = \left(a_i^u / b_i\right)^r$ and, for each $i \in S$, she chooses $s_i \in_{\mathrm{R}} \mathbb{Z}_q^*$ and computes $A_i = a_0^{s_i}$. U sends $(A_1, \ldots, A_n)$ to S, then uses Protocol 2 to prove knowledge of a set $S \subseteq [1, n]$ with $|S| = k$ and of $s_i = \log_{a_0} A_i$ for each $i \in S$. Recall that U's first announcement in Protocol 2 is $R = a_0^w \prod_{i \in H} A_i^{c_i\, t_i}$; after the protocol, S is convinced not only that U knows $S \subseteq [1, n]$, $|S| = k$, and $s_i = \log_{a_0} A_i$ for each $i \in S$, but also that U knows $w \in \mathbb{Z}_q^*$ and $H = [1, n] \setminus S$ such that $R = a_0^w \prod_{i \in H} A_i^{c_i\, t_i}$. Thus, U completes the proof by proving knowledge of (α, β, γ) such that $R = a_0^\gamma a^\alpha b^\beta$ and $1 = a_0^\alpha b_0^\beta$ where, as before, $a = \prod_{i=1}^n a_i^{t_i}$ and $b = \prod_{i=1}^n b_i^{t_i}$. Note that the product for R ranges over powers of a_i for $i \in H \cup \{0\}$ and that H is a part of U's private knowledge, wheres the products for a and b are over powers of a_i for the publicly known interval $[1, n]$. The additional terms involving powers of a_i for $i \in S$ cancel out in the verifi-

cation equation since $\left(\prod_{i\in S} a_i^{t_i}\right)^{\alpha}\left(\prod_{i\in S} b_i^{t_i}\right)^{\beta} = \left(\prod_{i\in S} a_i^{t_i}\right)^{u\,r}\left(\prod_{i\in S} a_i^{u\,t_i}\right)^{-r} = 1$. A couple of remarks about the above d-strikes-out protocol are needed.

Remark 1: The protocol reveals that *U is on the blacklist exactly k times*—all that it *should* reveal is that U is on the blacklist *at most* $d-1$ *times*. One way to accomplish this is for S to let U append $d-1$ ephemeral tokens to $\mathcal{B}$ at the start of the protocol, thus yielding $\mathcal{B}'$. U selects these tokens so that she appears on $\mathcal{B}'$ exactly $d-1$ times, which is always possible if she is on $\mathcal{B}$ itself at most $d-1$ times.

Remark 2: For security, we need to assume that S cannot feasibly deduce any information about H and S from $(A_1, \ldots, A_n)$. This assumption is only plausible if we know that each a_i appearing on $\mathcal{B}$ is selected in such a way that $\log_{a_i} a_j$ is unknown to S whenever $i \neq j$. (For example, if S knows $\sigma_{ij} = \log_{a_i} a_j = \log_{b_i} b_j$, then she can check if $A_i^{\sigma_{ij}} = A_j$ to learn whether $\log_{a_0} b_0 = \log_{a_i} b_i$.) This can be enforced in practice by, for example, having the a_i be output by a cryptographic hash function modeled as a random oracle.

7 Conclusion

We have examined "batch zero-knowledge" protocols for communication- and computation-efficient proofs of propositions composed of many simple predicates. Our primary contribution is a novel system for batch zero-knowledge arguments of knowledge and equality of k-out-of-n discrete logarithms for fixed $k \in [1, n]$. We also suggested the first general definitions for *batch zero-knowledge proofs and arguments (of knowledge)*. Our new definitions introduce a *conciseness* property that describes the asymptotic performance of a protocol relative to one formed by sequential composition of single-instance protocols. Our new argument system came about when we analyzed and uncovered a critical flaw in the security proof for Peng and Bao's [28] batch proofs of knowledge and equality of one-out-of-n discrete logarithms. A malicious prover can exploit the flaw to cause unsuspecting verifiers to accept proofs when the claimed equality of logarithms is false. Fortunately, we showed that the flaw is not fatal: we sketched a fix based on *all-but-k* mercurial commitments with $k = 1$ and then generalized to our main result by varying k in the repaired protocol. In addition, we illustrated the usefulness of our new protocol by sketching some example applications where its adoption could result in noteworthy speedups.

Acknowledgements. We thank Jalaj Uphadyay and Colleen Swanson for helpful discussions and the anonymous reviewers for their comments. The first author is supported by a GO-Bell Graduate Scholarship and by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Vanier Canada Graduate Scholarship. The second author is supported by the Ontario Research Fund (ORF) and an NSERC Discovery Grant.

References

1. Araujo, R., Foulle, S., Traoré, J.: A practical and secure coercion-resistant scheme for remote elections. In: Frontiers of Electronic Voting. Volume 07311 of Dagstuhl Seminar Proceedings., Schloss Dagstuhl, Germany (July 2007)
2. Au, M.H., Tsang, P.P., Kapadia, A.: PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. ACM Transactions on Information and System Security **14**(4) (December 2011) pp. Article No. 29
3. Au, M.H., Kapadia, A., Susilo, W.: BLACR: TTP-free blacklistable anonymous credentials with reputation. In: Proceedings of NDSS 2012, San Diego, CA, USA (February 2012)
4. Bellare, M., Garay, J.A., Rabin, T.: Batch verification with applications to cryptography and checking. In: Proceedings of LATIN 1998. Volume 1380 of LNCS., Campinas, Brazil (April 1998) pp. 170–191
5. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Proceedings of EUROCRYPT 1998. Volume 1403 of LNCS., Espoo, Finland (June 1998) pp. 236–250
6. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Proceedings of CRYPTO 1992. Volume 740 of LNCS., Santa Barbara, CA, USA (August 1992) pp. 390–420
7. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology **21**(2) (April 2008) pp. 149–177
8. Brands, S., Demuynck, L., De Decker, B.: A practical system for globally revoking the unlinkable pseudonyms of unknown users. In: Proceedings of ACISP 2007. Volume 4586 of LNCS., Townsville, Australia (July 2007) pp. 400–415
9. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: Proceedings of CCS 1999, Singapore (November 1999) pp. 120–127
10. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Proceedings of CRYPTO 2003. Volume 2729 of LNCS., Santa Barbara, CA, USA (August 2003) pp. 126–144
11. Catalano, D., Fiore, D., Messina, M.: Zero-knowledge sets with short proofs. In: Proceedings of EUROCRYPT 2008. Volume 4965 of LNCS., Istanbul, Turkey (April 2008) pp. 433–450
12. Clark, J., Hengartner, U.: Selections: Internet voting with over-the-shoulder coercion-resistance. In: Proceedings of FC 2011. Volume 7035 of LNCS., Gros Islet, St. Lucia (February 2011) pp. 47–61
13. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Proceedings of CRYPTO 1994. Volume 839 of LNCS., Santa Barbara, CA, USA (August 1994) pp. 174–187
14. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Proceedings of CRYPTO 1986. Volume 263 of LNCS., Santa Barbara, CA, USA (August 1986) pp. 186–194
15. Fürer, M.: Faster integer multiplication. SIAM Journal on Computing **39**(3) (2009) pp. 979–1005
16. Gennaro, R., Leigh, D., Sundaram, R., Yerazunis, W.S.: Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In: Proceedings of ASIACRYPT 2004. Volume 3329 of LNCS., Jeju Island, Korea (December 2004) pp. 276–292

17. Goldberg, I.: Improving the robustness of private information retrieval. In: Proceedings of IEEE S&P 2007, Oakland, CA, USA (May 2007) pp. 131–148
18. Goldreich, O.: A note on computational indistinguishability. Information Processing Letters **34**(6) (1990) pp. 277–281
19. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or languages in NP have zero-knowledge proof systems. Journal of the ACM **38**(3) (July 1991) pp. 691–729
20. Goldreich, O., Sahai, A., Vadhan, S.P.: Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In: Proceedings of STOC 1998, Dallas, TX, USA (May 1998) pp. 399–408
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems (extended abstract). In: Proceedings of STOC 1985, Providence, RI, USA (May 1985) pp. 291–304
22. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Journal of Cryptology **23**(4) (October 2010) pp. 546–579
23. Henry, R., Goldberg, I.: All-but-k mercurial commitments and their applications. Tech. Report CACR 2012-26, University of Waterloo, Waterloo, ON, Canada (November 2012)
24. Henry, R., Olumofin, F.G., Goldberg, I.: Practical PIR for electronic commerce. In: Proceedings of CCS 2011, Chicago, IL, USA (October 2011) pp. 677–690
25. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of WPES 2005, Alexandria, VA, USA (November 2005) pp. 61–70
26. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Proceedings of ASIACRYPT 2010. Volume 6477 of LNCS., Singapore (December 2010) pp. 177–194
27. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4) (December 1982) 515–534
28. Peng, K., Bao, F.: Batch ZK proof and verification of OR logic. In: Proceedings of Inscrypt 2008. Volume 5487 of LNCS., Beijing, China (December 2008) pp. 144–156
29. Peng, K., Bao, F.: Batch range proof for practical small ranges. In: Proceedings of AFRICACRYPT 2010. Volume 6055 of LNCS., Stellenbosch, South Africa (May 2010) pp. 114–130
30. Peng, K., Boyd, C., Dawson, E.: Batch zero-knowledge proof and verification and its applications. ACM Transactions on Information and System Security **10**(2) (May 2007) pp. Article No. 6
31. Sako, K., Kilian, J.: 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., Saint-Malo, France (May 1995) pp. 393–403
32. Spycher, O., Koenig, R.E., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: Proceedings of FC 2011. Volume 7035 of LNCS., Gros Islet, St. Lucia (February 2011) pp. 182–189
33. Stein, W.A., et al.: Sage Mathematics Software (Version 4.8). The Sage Development Team. (January 2012) `http://www.sagemath.org/`.
34. Tsang, P.P., Au, M.H., Kapadia, A., Smith, S.W.: BLAC: Revoking repeatedly misbehaving anonymous users without relying on TTPs. ACM Transactions on Information and Systems Security **13**(4) (December 2010) pp. Article No. 39
35. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Proceedings of INDOCRYPT 2004. Volume 3348 of LNCS., Chennai, India (December 2004) pp. 384–398

36. Weber, S.G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: Proceedings of ARES 2007, Vienna, Austria (April 2007) pp. 908–916

A Attacking Peng and Bao's protocol

This appendix provides details on how to implement the attack on Protocol 1 outlined in §2. We seek solutions $c_j \in \left[0, 2^\lambda - 1\right]$ for each $j \in S$ to satisfy the system

$$0 \equiv \sum_{j \in S} c_j \, \gamma_j \pmod{p}, \text{ and} \tag{1a}$$

$$c' \equiv \sum_{j \in S} c_j \pmod{2^\lambda}, \tag{2a}$$

where $c' \in \left[0, 2^\lambda - 1\right]$ and $\gamma_j = t_j(x_j - y_j) \bmod p$ are both given. We first shift the range of each equation to center the desired solution about 0 by setting

$$\begin{aligned} d_j &= c_j - 2^{\lambda-1} \text{ for each } j \in S, \\ d' &= c' - k\,2^{\lambda-1}, \text{ and} \\ K &= -2^{\lambda-1} \sum_{j \in S} \gamma_j \bmod p. \end{aligned}$$

We now seek solutions $d_j \in \left[-2^{\lambda-1}, 2^{\lambda-1}\right)$ to satisfy the system

$$K \equiv \sum_{j \in S} d_j \, \gamma_j \pmod{p}, \text{ and} \tag{1b}$$

$$d' \equiv \sum_{j \in S} d_j \pmod{2^\lambda}. \tag{2b}$$

To find such solutions, we consider the $(k+3)$-dimensional integer lattice M:

$$M = \begin{bmatrix} X & & & & -K\,Y & -d'\,Y \\ & 1 & & & \gamma_1 Y & Y \\ & & 1 & & \gamma_2 Y & Y \\ & & & \ddots & \vdots & \vdots \\ & & & 1 & \gamma_k Y & Y \\ & & & & p\,Y & \\ & & & & & 2^\lambda\,Y \end{bmatrix},$$

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 (i.e., an integer linear combination of the rows of M), then $\{d_j \mid j \in [1, n]\}$ forms a required solution to Equations (1b) and (2b) since, letting M_j denote the j^{th} row of M (counting from 0), we observe that $\vec{v} = M_0 + d_1 M_1 + \cdots + d_k M_k + a M_{k+1} + b M_{k+2}$ for some integers a and b. We thus use the LLL lattice basis reduction algorithm [27] 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 (GCD) 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 GCD of the leading elements is still X; if so, we can use the extended Euclidean algorithm 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 $D = XY^2p2^\lambda$, and its dimension is $k+3$. By choosing $X = Y = \lceil \sqrt[k]{p2^\lambda} \rceil$, we get that $D^{1/(k+3)} \approx \sqrt[k]{p2^\lambda} \approx X$. 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 should also be about the size of X, or just a little larger. Let $k_0 = (\lg p)/\lambda + 2$. When $k \geq k_0$, $X \approx \sqrt[k]{p2^\lambda} \leq 2^\lambda/2^{\lambda/k_0}$, and we expect our solution to be in the required range.

We tested our attack using the implementation of LLL in the mathematical software package Sage [33]. When $p = 2^{160} - 47$ (the largest 160-bit prime), and $\lambda = 40$, we find that setting $k = 6$ yields a solution to Equations (1b) and (2b) in the appropriate range almost every time (we observed only two failures in over 6000 trials). Solving the problem took about 25 ms per trial, on average. When we tried $k = 5$, the attack still succeeded about 52% of the time (out of 1000 trials). With k less than 5, we do not expect that a solution even *exists* in the required range, by the counting argument in §2.

B Proof of security for Protocol 2

Let a cyclic group $\mathbb{G}$ of order p with two fixed generators g and h, a set of tuples $\big\{(g_i, h_i) \in \mathbb{G}^2 \mid i \in [1, n]\big\}$, and a secure all-but-$k$ mercurial commitment scheme $\mathbf{ABK} = (\text{ABK-Init}, \text{ABK-Commit}, \text{ABK-Open}, \text{ABK-Verify})$ all be given. For a fixed soundness parameter $\lambda \in \mathbb{N}$, let q be an arbitrary λ-bit prime and use ABK-Init to generate a common reference string PK for vectors over $\mathbb{Z}_q$ of length up to $n_0 \in \mathbb{N}$. We require that **ABK** satisfies the following *efficiency* requirements:

Logarithmic length: The lengths of the outputs of $\text{ABK-Commit}_{\text{PK}}$ and $\text{ABK-Open}_{\text{PK}}$ are each in $O\big(\lg n_0\big)$ bits.

(Essentially) linear work: For a fixed k and for all $n \geq k$, the running time of $\text{ABK-Commit}_{\text{PK}}$, $\text{ABK-Open}_{\text{PK}}$, and $\text{ABK-Verify}_{\text{PK}}$ when the prover commits to all-but-k components of a length-n vector are each in $O(n\,\lambda\,\lg^2 n_0)$.

Henry and Goldberg's scheme [23] satisfies both requirements. It is unconditionally hiding and computationally binding under the n_0-SDH assumption [23, Definition 2].

Let R be the set of pairs $\big((g_i, h_i), y_i\big) \in \mathbb{G}^2 \times \mathbb{Z}_p$ for which $\log_g g_i = \log_h h_i = y_i$ and let $L^{(k)}_{n_0}$ be the language of binary strings with length at most n_0 and Hamming weight at least k.

Theorem 1. *For any fixed soundness parameter $\lambda \in \mathbb{N}$, Protocol 2 is a system of honest-verifier $(0, 1, \min\{\lg p, \lambda\})$-batch zero-knowledge proofs or arguments of knowledge for $(L_{n_0}^{(k)}, R)$. If* ***ABK*** *is computationally binding, the system is computationally convincing; otherwise, it is unconditionally convincing.*

Proof. We prove each of the five requirements of Definition 3 individually. Let $S = \{j \in [1, n] \mid (x_j, y_j) \in R\}$ and $H = [1, n] \setminus S$.

Claim 1: *Protocol 2 is* complete.

Suppose P and V invoke Protocol 2 on common input an n-tuple x for which $n \leq n_0$. If P knows an n-tuple y such that $\varphi_R(x, y) \in L_{n_0}^{(k)}$, then P can always choose and commit to $\{c_i \mid i \in H\}$ in Step P2 using $\mathrm{PK}(n_0)$. The all-but-k binding property lets P choose $\{c_j \mid j \in S\}$ arbitrarily after V sends the challenge $\vec{K}$ in Step V3, which provides her with sufficiently many degrees of freedom to ensure that $\vec{c}$ satisfies the induced system of k linearly independent constraints in Step P4. Thus, P can always satisfy the first and second verification equations. Now, a straightforward computation shows that if P does indeed know $\{y_j \in \mathbb{Z}_p \mid j \in S\}$ such that $\log_g g_j = \log_h h_j = y_j$ for every $j \in S$, then her response v will satisfy the third and fourth verification equation. Thus, honest V will always output "true" upon interacting with honest P. $\diamond$

Claim 2: *Protocol 2 is* sound.

We first prove the following technical lemma and its corollary (both of which assume that the problem setup from Theorem 1 is still in place).

Lemma 1. *Suppose that $q < p$ and fix $t_1, \ldots, t_n \in \mathbb{Z}_q^*$ and $c_i \in \mathbb{Z}_q^*$ for all $i \in H$. If there exists some $j \in S$ for which $\log_g g_j \neq \log_h h_j$ and $t_j \neq 0$, then there are at most q^{k-1} assignments for the $c_j \in \mathbb{Z}_q^*$, $j \in S$, for which $\log_g\bigl(\prod_{i=1}^n g_i^{c_i t_i}\bigr) = \sum_{i=1}^n \bigl(c_i t_i \log_g g_i\bigr)$ is equal to $\log_h\bigl(\prod_{i=1}^n h_i^{c_i t_i}\bigr) = \sum_{i=1}^n \bigl(c_i t_i \log_h h_i\bigr)$.*

Proof. There are q^{k-1} ways to choose the $c_i \in \mathbb{Z}_q^*$ for $i \in S \setminus \{j\}$. Let $J = [1, n] \setminus \{j\}$ and suppose, by way of contradiction, that for a given choice of $c_i \in \mathbb{Z}_q^*$ for $i \in S \setminus \{j\}$, there exist distinct $c_j, c_j' \in \mathbb{Z}_q^*$ for which

$$\log_g\Bigl(g_j^{c_j t_j} \prod_{i \in J} g_i^{c_i t_i}\Bigr) = \log_h\Bigl(h_j^{c_j t_j} \prod_{i \in J} h_i^{c_i t_i}\Bigr), \tag{3}$$

and

$$\log_g\Bigl(g_j^{c_j' t_j} \prod_{i \in J} g_i^{c_i t_i}\Bigr) = \log_h\Bigl(h_j^{c_j' t_j} \prod_{i \in J} h_i^{c_i t_i}\Bigr). \tag{4}$$

Subtracting (4) from (3) yields $(c_j - c_j')t_j \log_g g_j = (c_j - c_j')t_j \log_h h_j$. Now, since c_j and c_j' are distinct with $0 \leq c_j < q < p$ and $0 \leq c_j' < q < p$, we have that $c_j - c_j' \not\equiv 0 \pmod{p}$; thus, $t_j \log_g g_j = t_j \log_h h_j$ and, in particular, $\log_g g_j = \log_h h_j$, which is a contradiction. Hence, for each of the q^{k-1} possible assignments of $c_i \in \mathbb{Z}_q^*$ for $i \in S \setminus \{j\}$, there is at most one $c_j \in \mathbb{Z}_q^*$ to satisfy $\log_g\Bigl(\prod_{i=1}^n g_i^{c_i t_i}\Bigr) = \log_h\Bigl(\prod_{i=1}^n h_i^{c_i t_i}\Bigr)$.

Corollary 1. *Suppose that $q < p$ and fix $t_1, \ldots, t_n \in \mathbb{Z}_q^*$ and $c_i \in \mathbb{Z}_q^*$ for all $i \in H$. If there is some $j \in S$ for which $\log_g g_j \neq \log_h h_j$, then*

$$\Pr\left[\log_g\Big(\prod_{i=1}^{n} g_i^{c_i t_i}\Big) = \log_h\Big(\prod_{i=1}^{n} h_i^{c_i t_i}\Big)\right] \leq \frac{2}{q},$$

where the probability is taken over random choices of $c_i \in_{\mathrm{R}} \mathbb{Z}_q^*$ for $i \in S$.

Proof. By the Lemma, if $t_j \neq 0$ then at most q^{k-1} out of q^k possible assignments of $c_i \in \mathbb{Z}_q^*$ for $i \in S$ can satisfy $\log_g\Big(\prod_{i=1}^{n} g_i^{c_i t_i}\Big) = \log_h\Big(\prod_{i=1}^{n} h_i^{c_i t_i}\Big)$. Now, the probability that $t_j = 0$ is just $1/q$; thus, the probability that a randomly selected assignment is a satisfying assignment is at most $(q^{k-1}/q^k) + (1/q) = 2/q$. □

Given the above lemma and its corollary, proving soundness is straightforward. We first note that the all-but-k binding property of $\mathcal{C}$ ensures that P must have committed to $\{c_i \mid i \in H\}$ in Step P2, i.e., prior to V sending $\vec{K}$ in Step V3. Now, the rows of $\mathbf{V}_q^{n \times k}$ corresponding to the k indices in S form a nonsingular matrix modulo q; hence, there is a one-to-one correspondence between V's choices for $\vec{K}$ and the resulting $\{c_j \mid j \in S\}$ that P will compute in Step P4. In particular, V choosing $\vec{K} \in_{\mathrm{R}} (\mathbb{Z}_q^*)^k$ is equivalent to choosing $c_j \in_{\mathrm{R}} \mathbb{Z}_q^*$ for each $j \in S$. Now, suppose that P can successfully respond to two distinct challenges $\vec{K}$ and $\vec{K}'$ with responses $(v, c_1, \ldots, c_n)$ and $(v', c_1', \ldots, c_n')$; that is,

$$a = g^v \prod_{i=1}^{n} g_i^{c_i t_i} = g^{v'} \prod_{i=1}^{n} g_i^{c_i' t_i}$$

and

$$b = h^v \prod_{i=1}^{n} h_i^{c_i t_i} = h^{v'} \prod_{i=1}^{n} h_i^{c_i' t_i}.$$

Now, the all-but-k binding property ensures that $c_i = c_i'$ for all $i \in H$ so that

$$\begin{aligned} 1 = a/a &= g^{v-v'} \prod_{i=1}^{n} g_i^{t_i(c_i - c_i')} \\ &= g^{v-v'} \prod_{i \in S} g_i^{t_i(c_i - c_i')} \end{aligned}$$

and

$$\begin{aligned} 1 = b/b &= h^{v-v'} \prod_{i=1}^{n} h_i^{t_i(c_i - c_i')} \\ &= h^{v-v'} \prod_{i \in S} h_i^{t_i(c_i - c_i')}. \end{aligned}$$

In particular, $v' - v = \log_g\big(\prod_{i=1}^{n} g_i^{t_i(c_i - c_i')}\big)$ and $v' - v = \log_h\big(\prod_{i=1}^{n} h_i^{t_i(c_i - c_i')}\big)$; by Corollary 1, such an event happens with probability at most $2/q$, which is a negligible function of λ. If **ABK** is computationally binding, then we must

also allow for some negligible probability (in the security parameter) that the all-but-k binding property fails. $\diamond$

Claim 3: *Protocol 2 is* honest-verifier zero-knowledge.

We exhibit a simulator S for the honest verifier. On input an n-tuple x and auxiliary string z, S does the following:

1. Choose $t_1, \ldots, t_n \in_{\mathrm{R}} \mathbb{Z}_q^*$, $c_1, \ldots, c_n \in_{\mathrm{R}} \mathbb{Z}_q^*$ and $v \in_{\mathrm{R}} \mathbb{Z}_p$.
Let $\vec{c} = \langle c_1, \ldots, c_n \rangle$.

2. Compute each of the following:
$\mathcal{C} \leftarrow$ ABK-Commit$(\vec{c})$,
$a = \bigl(\prod_{i=1}^n g_i^{c_i t_i}\bigr)/g^v$,
$b = \bigl(\prod_{i=1}^n h_i^{c_i t_i}\bigr)/h^v$,
$\vec{K} = \vec{c} \cdot \mathbf{V}_q^{n \times k} \bmod q$, and
$\omega \leftarrow$ ABK-Open$(\mathcal{C}, \vec{c}, k)$.

3. Output $(x, z, \mathcal{C}, \vec{t}, a, b, \vec{K}, \vec{c}, \omega, v)$.

It is easy to see that the distributions $\bigl\{\mathrm{S}(x,z)\bigr\}_{x \in L_R}$ and $\bigl\{\mathrm{tr}_{\mathrm{P,V}}(x,z)\bigr\}_{x \in L_R}$ are identical when **ABK** is unconditionally hiding and computationally indistinguishable when **ABK** is computationally hiding. In both distributions, the short exponents $t_1, \ldots, t_n \in \mathbb{Z}_q^*$ and response $v \in \mathbb{Z}_p$ are uniformly distributed in their respective domains. $(\mathcal{C}, \omega)$ is uniform in its domain when **ABK** is unconditionally hiding; the distributions for $(\mathcal{C}, \omega)$ in $\bigl\{\mathrm{S}(x,z)\bigr\}_{x \in L_R}$ and $\bigl\{\mathrm{tr}_{\mathrm{P,V}}(x,z)\bigr\}_{x \in L_R}$ are computationally indistinguishable when **ABK** is computationally hiding. Moreover, in both distributions we note that v in combination with $t_1, \ldots, t_n$, $\vec{c}$, and $x = \bigl\{(g_i, h_i) \mid i \in [1,n]\bigr\}$ completely determines (a, b); thus, if we can show that the distributions of $c_1, \ldots, c_n \in \mathbb{Z}_q^*$ and $\vec{K} \in (\mathbb{Z}_q^*)^k$ are the same in $\bigl\{\mathrm{S}(x,z)\bigr\}_{x \in L_R}$ and $\bigl\{\mathrm{tr}_{\mathrm{P,V}}(x,z)\bigr\}_{x \in L_R}$, then it will follow that the $\bigl\{\mathrm{S}(x,z)\bigr\}_{x \in L_R}$ and $\bigl\{\mathrm{tr}_{\mathrm{P,V}}(x,z)\bigr\}_{x \in L_R}$ are indistinguishable. This is indeed the case, since by the nonsingularity of the submatrix of $\mathbf{V}_q^{n \times k}$ corresponding to the k rows in S, choosing $\vec{K} \in_{\mathrm{R}} (\mathbb{Z}_q^*)^k$ and all but k of the $c_i \in_{\mathrm{R}} \mathbb{Z}_q^*$ then computing the remaining $c_i \in \mathbb{Z}_q^*$ to satisfy $\vec{c} \cdot \mathbf{V}_q^{n \times k} = \vec{K} \bmod q$ is equivalent to choosing $\vec{c} \in_{\mathrm{R}} (\mathbb{Z}_q^*)^n$ and computing $\vec{K} = \vec{c} \cdot \mathbf{V}_q^{n \times k} \bmod q$. $\diamond$

Claim 4: *Protocol 2 is* $(0,1)$-concise.

Let $a_0(\tau)$ and $a_1(\tau)$ respectively denote the computation cost and communication cost of Protocol 2 when $n = 1$. By inspection, we see that $a_0(\tau) \in \Omega\bigl(\tau^2 \lg^2 \tau\bigr)$ and $a_1 \in \Omega\bigl(\tau\bigr)$.

Computation. Set $\tilde{a}_0(\tau) = \lambda\, \tau \lg^2 \tau$ and note that $\tilde{a}_0 \in o\bigl(a_0(\tau)^{1-\delta}\bigr)$ for any positive $\delta < 1/2$. We count the computation cost (in bit operations) for each step in the protocol.

Step P2: P computes $(a, b, \mathcal{C})$, which is $O(k \lg n + \lambda\, n \lg n)$ bit operations for $\mathcal{C}$ and $O\bigl(\tau^2 \lg^2 \tau + (n-k)\lambda\, \tau \lg^2 \tau\bigr)$ bit operations for (a, b).

Step P4: P computes $(v, c_{j_1}, \ldots, c_{j_k}, \omega)$, which is $O\bigl(k\, \tau \lg^2 \tau\bigr)$ bit operations for v, $O\bigl(\lambda^2 k\, n\bigr)$ bit operations for $c_{j_1}, \ldots, c_{j_k}$, and $O\bigl(k \lg n + \lambda\, n \lg n\bigr)$ bit operations for ω.

Step V5: V computes the verification equations, which is $O\big(\lambda^2\, k\, n\big)$ bit operations for the first equation, $O\big(k\, \lg n + \lambda\, n\, \lg n\big)$ bit operations for the second equation, and $O\big(\tau^2 \lg^2 \tau + (n-k)\, \lambda\, \tau\, \lg^2 \tau\big)$ for the third and fourth equations.

Summing up all of the costs, we see that the total computation cost for P and V is in $O\big(a_0(\tau) + n^{1+\epsilon}\, \tilde{a}_0(\tau)\big)$ for any $\epsilon > 0$.

Communication. Now, set $\tilde{a}_1(\tau) = \lambda$ and note that $\tilde{a}_0 \in o\big(a_1(\tau)^{1-\delta}\big)$ for any positive $\delta \leq 1$. We count the communication cost (in bits) for each step.

Step V1: V sends $t_1, \ldots, t_n$, which is $n\, \lambda$ bits.

Step P2: P sends $(a, b, \mathcal{C})$, which is $O\big(\tau\big)$ bits for (a, b) and $O\big(\lg n\big)$ bits for $\mathcal{C}$.

Step V3: V sends $\vec{K}$, which is $k\, \lambda$ bits.

Step P4: P sends $(v, \vec{c}, \omega)$, which is $O\big(\tau\big)$ bits for v, $n\, \lambda$ bits for $\vec{c}$, and $O\big(\lg n\big)$ bits for ω.

Summing up all of the costs, we see that the total bidirectional communication cost is in $O\big(a_1(\tau) + n^{1+\epsilon}\, \tilde{a}_1(\tau)\big)$ for any $\epsilon \geq 0$.

This proves that Protocol 2 is $(0, 1)$-concise. ◇

Claim 5: *Protocol 2 is* knowledge extractable.

We exhibit a knowledge extractor E^{P^*} for prover P^*. E^{P^*} initializes $i \leftarrow 1$ and runs the protocol with P^* to Step V3, then does the following:

1. Send challenge $\vec{K}^{(1)} \in_{\mathrm{R}} (\mathbb{Z}_q^*)^k$ in Step V3

2. Receive response $(v^{(1)}, \vec{c}^{(1)}, \omega^{(1)})$ in Step P4

3. Do

  Rewind P^* to Step V3

  $i \leftarrow i + 1$

  Send challenge $\vec{K}^{(i)} \in_{\mathrm{R}} (\mathbb{Z}_q^*)^k$ in Step V3

  Get response $(v^{(i)}, \vec{c}^{(i)}, \omega^{(i)})$ in Step P4

  Set $V_{i-1} = v^{(1)} - v^{(i)}$

Until $\mathrm{rank}\big\{V_1, \ldots, V_i\big\} = k$

4. Solve $\{V_1, \ldots, V_i\}$ for $\big\{x_j \in \mathbb{Z}_p \mid j \in S\big\}$.

With overwhelming probability in $\lg q$, E^{P^*} will halt after looping polynomially many times. Note that, because soundness is overwhelming in $\Lambda(\tau)$, we also have with overwhelming probability that each of the V_i is of the form $V_i = v^{(1)} - v^{(i+1)} = \sum_{j \in S} t_j (c_j^{(1)} - c_j^{(i+1)}) x_j$, which is why E^{P^*} can solve for $\big\{x_j \in \mathbb{Z}_p \mid j \in S\big\}$ in Step 4. ◇

This concludes the proof of Theorem 1.