Batch Proofs of Partial Knowledge

Ryan Henry
Cheriton School of Computer Science
University of Waterloo
Waterloo ON Canada N2L 3G1
rhenry@cs.uwaterloo.ca

Ian Goldberg
Cheriton School of Computer Science
University of Waterloo
Waterloo ON Canada N2L 3G1
iang@cs.uwaterloo.ca

ABSTRACT

We present a practical attack on the soundness of Peng and Bao's 'batch zero-knowledge proof and verification' protocol for proving knowledge and equality of one-out-of-n pairs of discrete logarithms. Fixing the protocol seems to require a commitment scheme with a nonstandard, mercurial-esque binding property: the prover commits to just $n-1$ values, but later opens the commitment to n values without revealing which one out of the n values was not part of the original commitment. With this requirement as a motivator, we propose and formally define *all-but-k commitment schemes*, and give a concrete construction based on polynomial commitments. We use the special case of "all-but-one" commitments to fix the above zero-knowledge protocol and then we describe a variant of the protocol that uses the more general all-but-k commitments to implement a batch zero-knowledge proof of knowledge and equality of k-out-of-n pairs of discrete logarithms, for arbitrary (public) $k \in [1,n]$. This latter protocol is asymptotically efficient, and it naturally yields batch "OR" proofs (one-out-of-n) and batch "AND" proofs (n-out-of-n) as two special cases; for all intermediate $1 < k < n$, it is entirely novel.

Keywords

Batch proof and verification, zero-knowledge proofs, cryptanalysis, commitment schemes, proofs of partial knowledge, interactive protocols, lattice-based attacks

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 is already convinced that the proposition is true, but the verifier does not trust the prover and is skeptical about her claim. What differentiates a zero-knowledge proof from an ordinary proof is the amount of information that the prover reveals to the verifier: in a zero-knowledge proof, the prover convinces the verifier without revealing *any* information beyond the veracity of the proposition itself. Zero-knowledge proofs have played a central role in cryptography ever since Goldwasser et al. introduced them in 1985 [12]; indeed, they are the basis for a large number of cryptographic protocols in the literature. Alas, the computation and communication costs of conventional zero-knowledge techniques do not scale well for problems with large "fan-in" and few usable, large-scale zero-knowledge-based systems exist in practice.

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 standard ways to prove these two statements can be prohibitively expensive when n is large: the prover and verifier each compute $\Theta(n)$ full-length exponentiations, and the prover sends $\Theta(n)$ group elements to the verifier. (The verifier only needs to send $\Theta(1)$ group elements to the prover.) In 1998, Bellare et al. [2, 3] suggested using *batch verification* to reduce the verifier's cost. Their "small exponents" batch verification test [3, §3.3] only requires the verifier to compute $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications (for soundness parameter $\lambda \in \mathbb{N}$). Nearly a decade later, Peng et al. [20] proposed a *batch proof* protocol for the first proposition above (i.e., a batch proof of *complete* knowledge), which is based on the same idea as Bellare et al.'s small-exponent batch testing. In Peng et al.'s protocol, the prover and the verifier each compute $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications, 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 [18] proposed a similar batch proof for the second proposition above (i.e., a batch proof of *partial* knowledge), which combines the idea of small-exponent batch testing with Cramer et al.'s proofs of partial knowledge [9]. In this latter protocol, the prover and verifier each compute $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications, and they each send $\Theta(1)$ group elements and $\Theta(\lambda n)$ additional bits. A handful of other papers in the literature use similar batch proof techniques to create protocols with similarly low asymptotic complexities [7, 19].

This paper proposes a new batch proof protocol that generalizes the aforementioned protocols by Peng et al. [20] and by Peng and Bao [18]. Our protocol uses a new type of commitment with similarities to *mercurial vector commitments* [16]. Briefly, a mercurial commitment is a special type of commitment with a modified binding property. A committer can either *hard commit* to a value or *soft commit* to no specific value; the two types of commitment look indistinguishable but they have different binding properties. If the committer hard commits to a value, then he can later *soft open* or *hard open* the commitment to that same value (and only that same value). If the committer instead soft commits, then he can later soft open the commitment to *any value* of his choosing, but he can never hard open it. (The security guarantee is that a soft opening cannot disagree with a hard opening of the same commitment, even though soft openings are otherwise non-binding.) Therefore, soft opening a commitment to a value x effectively proves that "if the commitment can be hard

opened at all, then it hard opens to x". A mercurial vector commitment is just a commitment scheme that commits to an entire vector of values in such a way that each individual component of the vector is committed to mercurially.

Our contributions.

The main contribution of this paper is a novel batch zero-knowledge proof of knowledge and equality of k-out-of-n pairs of discrete logarithms. Our new protocol is asymptotically efficient: to get soundness that is overwhelming in a soundness parameter λ, the prover and the verifier each need to compute just $\Theta(1)$ full-length exponentiations and $\Theta(\lambda n)$ multiplications; similarly, they each send and receive just $\Theta(1)$ group elements and $\Theta(\lambda n)$ additional bits. The protocol naturally yields batch "OR" proofs (one-out-of-n) and batch "AND" proofs (n-out-of-n) as special cases; for all intermediate $1 < k < n$, it is entirely novel in that no other protocol in the literature proves such statements using a sublinear number of full-length exponentiations and group element transfers. The secondary contributions leading up to our new protocol are threefold:

1. 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 logarithms.
2. We propose and formally define a new type of cryptographic commitments called *all-but-k* commitments. Our new commitments have similarities with mercurial vector commitments [16], with the key conceptual difference being that the revelation protocol for all-but-k commitments explicitly reveals an upper bound on the number of soft-committed values in the opening. We provide a concrete construction of all-but-k commitments that is based on Kate et al.'s $\mathbf{PolyCommit}_{\text{DL}}$ polynomial commitment scheme [13]. To facilitate this construction, we introduce new zero-knowledge protocols for 1) proving knowledge of a polynomial committed to by a $\mathbf{PolyCommit}_{\text{DL}}$ commitment, and 2) proving that a polynomial committed to by a $\mathbf{PolyCommit}_{\text{DL}}$ commitment has degree at most k.
3. We suggest a new formal definition for *batch zero-knowledge proofs* and *batch zero-knowledge proofs of knowledge*. Our definition adds a 'batching' property to the standard notion of zero-knowledge to capture the unique asymptotic behaviour that differentiates existing batch zero-knowledge proofs from conventional zero-knowledge proofs. We prove that our new protocol is a batch zero-knowledge proof of knowledge and equality of k-out-of-n pairs of discrete logarithms according to our new definition.

Outline.

The rest of the paper proceeds as follows: §2 begins with a description of our notation and formal definitions for the cryptographic assumptions we make in the rest of the paper. The first half of §3 introduces a 'batch zero-knowledge proof' protocol due to Peng and Bao [18], while the latter half of that section presents a practical attack on the protocol's soundness. We observe that to protect against this attack, it seems necessary to devise a new type of commitment scheme in which commitments have an unusual binding property. We refer to these new commitments as *all-but-one commitments*; in §4, we formally define a generalization of all-but-one-commitments called *all-but-k commitments*, and we describe a concrete construction for all-but-k commitments based on polynomial commitments. We return to batch zero-knowledge proofs in §5: we propose our new batch zero-knowledge definitions, use all-but-one commitments to repair Peng and Bao's protocol, and then extend the repaired protocol using all-but-k commitments and a variant of Cramer et al.'s method for zero-knowledge proofs of partial knowledge [9]. Our new protocol is a novel batch zero-knowledge proof of knowledge of a size-k subset $S \subseteq [1,n]$ of indices and a corresponding set $x_S = \{x_i \in \mathbb{Z}_q \mid i \in S\}$ of exponents such that $x_i = \log_g g_i = \log_h h_i$ for all $i \in S$. We prove the security of this latter protocol (which generalizes the repaired Peng-Bao proof) with respect to our new batch zero-knowledge definition. Finally, §6 concludes the paper with a brief summary of the highlights.

2. MATHEMATICAL PRELIMINARIES

Throughout, $\mathbb{G}$ will denote a cyclic group of prime order p with two fixed generators g, h. From §4 onward, we will also consider a group $\widetilde{\mathbb{G}}$, with generator $\widetilde{g}$ and prime order q, satisfying certain hardness assumptions, and an admissible bilinear pairing $e : \widetilde{\mathbb{G}} \times \widetilde{\mathbb{G}} \to \mathbb{G}_{\text{T}}$. (For ease of presentation, we assume a symmetric bilinear pairing; however, generalizing our constructions to use asymmetric pairings is not difficult.) A formal statement of each hardness assumption follows in Definitions 1 through 3; in these definitions — and throughout the paper — the term *negligible* describes a function $\varepsilon : \mathbb{N} \to \mathbb{R}$ with the property that, for every real number $c > 0$, there exists a positive integer τ_0 such that $\varepsilon(\tau) < 1/\tau^c$ for all $\tau > \tau_0$. The first assumption that we use is the so-called discrete logarithm assumption; by now, this assumption is standard fare in the cryptographic literature.

DEFINITION 1 (DISCRETE LOG ASSUMPTION; [17, §3.6]). *Let $\mathscr{G}$ be a PPT algorithm that, on input $\tau \in \mathbb{N}$, outputs a group $\widetilde{\mathbb{G}}$, its τ-bit order q, and a generator $g \in \widetilde{\mathbb{G}}$. The* ***discrete logarithm (DL) assumption*** *holds in the sequence of groups output by $\mathscr{G}$ if there exists a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$ such that, for every PPT adversary $\mathscr{A}$, we have $\Pr\left[x \leftarrow \mathscr{A}(g, g^x, q) \mid (\widetilde{\mathbb{G}}, q, g) \leftarrow \mathscr{G}(\tau)\right] \leq \varepsilon(\tau)$ for $x \in_{\text{R}} \mathbb{Z}_q^*$.*

The second and third assumptions that we use both regard the hardness of related variants of the well-known Diffie-Hellman problem [17, §3.7]. Many cryptographers have used and studied the second assumption — called the n-strong Diffie-Hellman (n-SDH) assumption — since its introduction by Boneh and Boyen in 2004 [5]. The third assumption, on the other hand, is less common than the DL and n-SDH assumptions; it concerns a problem similar to the more standard n-Diffie-Hellman inversion (n-DHI) problem [4] called the n-polynomial Diffie-Hellman (n-polyDH) problem. Au et al. made implicit use of the latter assumption in their compact e-cash scheme [1], and Kate et al. identified the assumption and explicitly defined it in subsequent work [13, 14].

DEFINITION 2 (n-STRONG DH ASSUMPTION; [6, §3]). *Let $\mathscr{G}$ be a PPT algorithm that, on input $\tau \in \mathbb{N}$, outputs a group $\widetilde{\mathbb{G}}$, its τ-bit order q, and a generator $g \in \widetilde{\mathbb{G}}$. Let $n \in \mathbb{N}$ and let $\alpha \in_{\text{R}} \mathbb{Z}_q^*$ be fixed but secret. The* ***n-strong Diffie-Hellman (n-SDH) assumption*** *holds in the sequence of groups output by $\mathscr{G}$ if there exists a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$ such that, for every PPT adversary $\mathscr{A}$, we have $\Pr\left[(c, g^{1/(\alpha+c)}) \leftarrow \mathscr{A}(g, g^\alpha, \ldots, g^{\alpha^n}, q) \mid (\widetilde{\mathbb{G}}, q, g) \leftarrow \mathscr{G}(\tau)\right] \leq \varepsilon(\tau)$ for any integer c.*

DEFINITION 3 (n-POLYNOMIAL DH ASSUMPTION; [13]). *Let $\mathscr{G}$ be a PPT algorithm that, on input $\tau \in \mathbb{N}$, outputs a group $\widetilde{\mathbb{G}}$, its τ-bit order q, and a generator $g \in \widetilde{\mathbb{G}}$. Let $n \in \mathbb{N}$ and let $\alpha \in_R \mathbb{Z}_q^*$ be fixed but secret. The **n-polynomial Diffie-Hellman (n-polyDH) assumption** holds in the sequence of groups output by $\mathscr{G}$ if there exists a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$ such that, for every PPT adversary $\mathscr{A}$, we have $\Pr\left[(f, g^{f(\alpha)}) \leftarrow \mathscr{A}(g, g^{\alpha}, \ldots, g^{\alpha^n}, q) \mid (\widetilde{\mathbb{G}}, q, g) \leftarrow \mathscr{G}(\tau)\right] \leq \varepsilon(\tau)$ for any $f \in \mathbb{Z}_q[x]$ with $\sqrt{q} > \deg f > n$.*

3. BATCH ZERO-KNOWLEDGE PROOF AND VERIFICATION

The following protocol is due to Peng and Bao [18, §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 verification [3, §3.3] into both the proof and verification phases of an otherwise standard sigma proof of knowledge and equality of one-out-of-n pairs of discrete logarithms. Both the prover P and verifier V know the same two generators $g, h \in \mathbb{G}$ and a set of n pairs of group elements $\{(g_i, h_i) \mid i \in [1,n]\}$, but only P knows an index $j \in [1,n]$ and exponent $x_j \in \mathbb{Z}_p$ such that $x_j = \log_g g_j = \log_h h_j$. 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 $H = [1,n] - \{j\}$ for the (j, x_j) that honest P is proving knowledge of. We also define a *soundness parameter* $\lambda \in \mathbb{N}$, which tunes the cost versus soundness trade off in small exponent batching.

- **V1:** Choose $t_i \in_R [0, 2^{\lambda} - 1]$ for $i \in [1,n]$. Send $(t_1, \ldots, t_n)$ to P.
- **P2:** Receive $(t_1, \ldots, t_n)$ from V. Choose $c_i \in_R [0, 2^{\lambda} - 1]$ for $i \in H$ and $r \in_R \mathbb{Z}_p$. 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 $c \in_R [0, 2^{\lambda} - 1]$ 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 $(v, c_1, \ldots, c_n)$ to V.
- **V5:** Receive $(v, c_1, \ldots, c_n)$ from P. Return "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 $c \stackrel{?}{=} \sum_{i=1}^n c_i \bmod 2^{\lambda}$.

Some remarks about this protocol are in order. Rather unexpectedly, we observe that V speaks before P does; what he sends to P in Step V1 is just the list of short exponents for small-exponent batch testing. The purpose of Step P2 is ostensibly to force P to commit to the index j (such that $H = [1,n] - \{j\}$) and to $\{c_i \mid i \in H\}$; if so, then when V chooses $c \in_R [0, 2^{\lambda} - 1]$ in Step V3, it is equivalent to choosing $c_j \in_R [0, 2^{\lambda} - 1]$ for P in Step P4, which is exactly what we want for good soundness. It is trivial to verify that the protocol is *complete*; honest P always convinces honest V. Theorem 1 in Peng and Bao's paper [18] 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 equation holds even though $\log_g g_j \neq \log_h h_j$ for some $j \in [1,n]$, given the following two implicit assumptions: 1) P committed to $H = [1,n] - \{j\}$ and $\{c_i \mid i \in H\}$ in Step P2, and 2) V chose c — and thus c_j, by virtue of assumption 1 — uniformly at random from $[0, 2^{\lambda} - 1]$ 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 P to using $H = [1,n] - \{j\}$; hence, the first implicit assumption in Peng and Bao's soundness proof is not guaranteed to hold.* It turns out that dishonest P can exploit this observation to pass the verification equation even when the claimed equality of logarithms does not hold. We give a high-level description of the attack below; the interested reader can find details on how to carry out the attack in Appendix A.

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 $x_j \not\equiv y_j \bmod p$. Partition the interval $[1,n]$ into two sets H and S, where S is the set of indices for which P knows the above tuple and H is the set of indices for which she does not. (Note that in some settings, $H = \emptyset$ is a perfectly reasonable assumption.) In Step P2, P computes (a, b) using this new H so that when she receives c from V in Step P4 she has $|S| - 1$ degrees of freedom to compute her response. In particular, the missing $\{c_j \mid j \in S\}$ that she must incorporate into her response are just solutions of the following system of two linear equations in $k = |S|$ unknowns:

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

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

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

The following counting argument suggests that such "suitably small" solutions are indeed plentiful whenever $k \cdot \lambda$ is "sufficiently large" compared to $\lg p$.[1] Let $\mathbf{X}$ be some instance of the above system induced by some real interaction between P and honest V. We heuristically expect the distribution of solutions of $\mathbf{X}$ to be uniform among all possible $\langle c_{j_1}, \ldots, c_{j_k} \rangle \in (\mathbb{Z}_p)^k$; in particular, we expect that the proportion of solutions that are "suitably small" is about $(2^{\lambda}/p)^k$. Now, only p^{k-1} of the $\langle c_{j_1}, \ldots, c_{j_k} \rangle \in (\mathbb{Z}_p)^k$ can satisfy Equation (1), and only about $p^{k-1}/2^{\lambda}$ of these can satisfy Equation (2) as well. This leads to the conclusion that $\mathbf{X}$ has around $\dfrac{p^{k-1}}{2^{\lambda}} \cdot \left(\dfrac{2^{\lambda}}{p}\right)^k = \dfrac{(2^{\lambda})^{k-1}}{p}$ suitably small solutions. Appendix A discusses how P can find one of these suitably small solutions by solving a short vector search problem in a particular lattice of dimension $k+3$. When k is reasonably small (such as when λ is large), P can use a standard basis reduction algorithm [15] to quickly find a suitably small solution. 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.

Our attack uses the fact that P can wait until *after* she sees c in Step P4 to choose $k > 1$ of the c_i. If V somehow forces P to commit to $\{c_i \mid i \in H\}$ for $H = [1,n] - \{j\}$ in Step P2, then Peng and Bao's upper bound of $1/2^{\lambda}$ for the protocol's soundness error holds. The most direct fix therefore seems to require a

[1] Recall that $k = |S|$ is 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 just make suitably small solutions more numerous and easier to find.

commitment scheme with a nonstandard, mercurial-esque binding property: the prover commits to just $n-1$ values, but later opens the commitment to n values without revealing which one of the n values was not part of the original commitment. We call this type of commitment an *all-but-one commitment*; the most natural generalization of all-but-one commitments is of course *all-but-k commitments*, which we formally introduce in the next section.

4. ALL-BUT-K COMMITMENTS

An all-but-k commitment scheme is a type of cryptographic commitment scheme that enables a prover P to commit to a collection of n' group elements (called *hard-committed values*) in a single commitment, and later open the commitment to $n = n' + k$ group elements: the original n' hard-committed values plus k additional *soft-committed values* that P chooses after forming the commitment. As part of the interactive "revealing" protocol, P proves an upper bound on k so that the verifier V knows that there are at most k soft-committed values in the opening. If there exists some polynomial upper bound $n_0 = n_0(\tau) \in \mathbb{N}$, for security parameter $\tau \in \mathbb{N}$, on the number of group elements n that P can commit to in a single commitment, then the scheme is n_0-*bounded* (or just *bounded*); otherwise, it is *unbounded*. We consider two basic varieties of (bounded or unbounded) all-but-k commitments:

All-but-k multiset commitments: An all-but-k *multiset commitment* commits to an *unordered multiset* of group elements (and preserves multiplicity).

All-but-k vector commitments: An all-but-k *vector commitment* commits to an *ordered list* of group elements; in particular, each group element is committed to along with its intended position in the final opening.[2]

The *binding* property for all-but-k commitments states that 1) P cannot open a commitment to any collection not containing all n' hard-committed values, 2) P cannot open a commitment under the pretense that k is smaller than the true number of soft-committed values, and, for vector commitments only, 3) all hard-committed values maintain their originally specified position in the final opening. The *hiding* property for all-but-k commitments states that, after P opens a commitment, V cannot distinguish the hard-committed values from the soft-committed values with nonnegligible advantage. Note that this implies the more conventional notion of hiding (wherein the requirement is roughly that a commitment alone does not help V deduce any committed value), since if that notion fails, then V can distinguish between hard- and soft-committed values by first deducing the hard-committed values from the commitment.

Let $\{H,S\}$ be a partitioning of $[1,n]$ and define $n' = |H|$ and $k = |S|$; here, H denotes the set of *hard-committed indices* and S denotes the set of *soft-committed indices*.

Formal definition.

An **all-but-k commitment scheme** is a suite of three PPT algorithms:

1. **AllButK-Initialize($\tau[,n_0]$):** Generate the public system parameters PK, which includes efficient descriptions of any necessary algebraic structures and the set $\mathscr{V}$ of committable values. The resulting construction should provide at least τ bits of security. If an upper bound n_0 is specified, then the scheme is n_0-*bounded* and the system parameters are suitable for committing to all-but-k from a set of n values for any $n \in [1,n_0]$ and $k \in [0,n-1]$.
2. **AllButK-Commit$_{\text{PK}}(H,\mathscr{H})$:** If $H \subseteq [1,n_0]$ and $\mathscr{H}$ is a multiset of elements $c_i \in \mathscr{V}$, then output a commitment $\mathscr{C}$ to $\mathscr{H}$, and some auxiliary decommitment information $d = \{\mathscr{H}, H, \ldots\}$; otherwise, output $\perp$. (In the case of a vector commitment scheme, $\mathscr{C}$ commits to (i,c_i) pairs rather than just $\{c_i \mid i \in H\}$.)
3. **AllButK-Reveal$_{\text{PK}}(\mathscr{C},\mathscr{O},k[,d])$:** This is an interactive protocol; both P and V input a commitment $\mathscr{C}$, a multiset $\mathscr{O}$, and an upper bound k on $|S|$, but only P inputs the decommit information d (indeed, this value is private to P). At the end of the protocol, V outputs a boolean value: "true" if the interaction with P convinced V that $\mathscr{O}$ is a valid opening of $\mathscr{C}$; otherwise, "false".

Alternatively, **AllButK-Reveal$_{\text{PK}}$** can be made noninteractive and split into two separate algorithms called **AllButK-Open$_{\text{PK}}(\mathscr{C},\mathscr{O},d)$** and **AllButK-Verify$_{\text{PK}}(\mathscr{C},\mathscr{O},w)$**, where w is a *witness element* output by **AllButK-Open$_{\text{PK}}$** to prove that $\mathscr{O}$ is a valid opening of $\mathscr{C}$.[3]

DEFINITION 4. *Let* PK $\leftarrow$ **AllButK-Initialize**$(\tau[,n_0])$. *A triple* (**AllButK-Initialize**, **AllButK-Commit$_{\text{PK}}$**, **AllButK-Reveal$_{\text{PK}}$**) *of PPT algorithms is a* ***secure (bounded) all-but-k commitment scheme*** *if it satisfies the following properties. (If the scheme is unbounded, then assume that $n_0 = \infty$.)*

1. ***Correctness:*** *For all possible choices of $n \in [1,n_0]$, $H \subseteq [1,n]$, and $S = [1,n] - H$, and for all multisets $\mathscr{H} = \{c_i \in \mathscr{V} \mid i \in H\}$ and $\mathscr{S} = \{c_j \in \mathscr{V} \mid j \in S\}$, we have that* **AllButK-Reveal$_{\text{PK}}(\mathscr{C},(\mathscr{H} \cup \mathscr{S}),k[,d])$** *between honest P and honest V causes V to output "true" for all $k \geq |S|$ and $(\mathscr{C},d) \leftarrow$ **AllButK-Commit$_{\text{PK}}(H,\mathscr{H})$**.*
2. ***Binding:*** *Fix $m \in \mathbb{N}$. For every PPT adversary $\mathscr{A}$ and commitment $\mathscr{C}$, there exists a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$ such that*

$$\Pr\left[\begin{array}{c} \{(C_i,k_i,d_i)\}_{i=1}^m \leftarrow \mathscr{A}(\text{PK},\mathscr{C},d) \ \wedge \\ \wedge_{i=1}^m \left(\textbf{AllButK-Reveal}_{\text{PK}}(\mathscr{C},C_i,k_i[,d_i]) = \text{“true”}\right) \\ \left| \ \left|\bigcap_{i=1}^m C_i\right| < \max_i\{|C_i| - k_i\} \right. \end{array}\right] \leq \varepsilon(\tau).$$

   Intuitively, if $\mathscr{A}$ opens the same commitment m times, then the number of common values (counting multiplicity) across all openings must be at least as high as maximum number of hard-committed values claimed in any one opening.
3. ***Hiding:*** *For all adversaries $\mathscr{A}$, there exists a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$ such that $\mathscr{A}$ wins the following game with probability at most $\frac{1}{2} + \varepsilon(\tau)$. If $\mathscr{A}$ is restricted to running PPT algorithms, then the hiding is computational; otherwise, it is unconditional.*
   (a) *$\mathscr{A}$ chooses and sends $\mathscr{O}$, $\mathscr{H}_0 \subseteq \mathscr{O}$, $\mathscr{H}_1 \subseteq \mathscr{O}$, and $k \geq |\mathscr{O}| - \min\{|\mathscr{H}_0|,|\mathscr{H}_1|\}$ to the challenger C.*
   (b) *C chooses $b \in_{\text{R}} \{0,1\}$ and computes $(\mathscr{C},d) \leftarrow$ **AllButK-Commit$_{\text{PK}}(H,\mathscr{H})$** and sends $\mathscr{C}$ to $\mathscr{A}$.*
   (c) *$\mathscr{A}$ and C engage in* **AllButK-Reveal$_{\text{PK}}(\mathscr{C},\mathscr{O},k[,d])$**.
   (d) *$\mathscr{A}$ outputs b' and wins if and only if $b = b'$.*

[2] One can convert all-but-k multiset commitments into all-but-k vector commitments by prepending a unique, fixed-length index $i \in [1,n]$ to each hard- and soft-committed value; this reduces the number of "committable" values by a factor of $2^{\lceil \lg n \rceil}$.

[3] The construction we give in this paper has an interactive revelation protocol, but we note that it can easily be made noninteractive using the Fiat-Shamir heuristic [10] (in which case security holds in the random oracle model).

4.1 Polynomial commitments

A *polynomial commitment scheme* is a cryptographic commitment scheme that lets P commit to a polynomial $f \in \mathbb{Z}_q[x]$ using a single group element. P can open the commitment either to f, or to $f(A)$ for any $A \subseteq \mathbb{Z}_q$. In this section, we briefly recall Kate et al.'s $\textbf{PolyCommit}_{\text{DL}}$ polynomial commitments [13] and give efficient, novel zero-knowledge protocols for 1) proving knowledge of a polynomial committed to by a $\textbf{PolyCommit}_{\text{DL}}$ commitment, and 2) proving that a polynomial committed to by a $\textbf{PolyCommit}_{\text{DL}}$ commitment has degree at most k. We use $\textbf{PolyCommit}_{\text{DL}}$ commitments and these new zero-knowledge proofs to construct all-but-k commitments in the next subsection.

The public parameters for a degree-n_0 $\textbf{PolyCommit}_{\text{DL}}$ scheme are an ordered list $\text{PK} = \langle \widetilde{\mathbb{G}}, q, \widetilde{g}^{\alpha^i} \mid i \in [0, n_0] \rangle$; typically, a trusted (or distributed) initializer selects $\alpha \in_{\text{R}} \mathbb{Z}_q$, generates and publishes the public parameters, and then securely discards α. Alternatively, one can regard α as a *trapdoor key* for the commitment scheme: with knowledge of the trapdoor key α, it is trivial to open any commitment to an arbitrary set of evaluation points.

To form a commitment to $f = \sum_{i=0}^{m} f_i x^i \in \mathbb{Z}_q[x]$ of degree $m \leq n_0$, P computes $\mathscr{C}_f = \prod_{i=0}^{m} \left(\widetilde{g}^{\alpha^i}\right)^{f_i} = \widetilde{g}^{f(\alpha)}$ using values from PK. V can of course confirm that $\mathscr{C}_f$ really commits to f by redoing the calculation. More interestingly, however, is that P can prove that f satisfies $y = f(i)$ for some public (i, y) pair without leaking additional information about f as follows.

P1: Write $f(x) = Q(x)(x-i) + f(i)$ using the polynomial remainder theorem. Compute a *witness* $w = \widetilde{g}^{Q(\alpha)}$ that $f(i) = y$ via $\prod_{i=0}^{m} \left(\widetilde{g}^{\alpha^i}\right)^{Q_i}$, where $Q(x) = \sum_{i=0}^{m} Q_i x^i$. Send w to V.

V2: Return "true" if and only if $e(\mathscr{C}_f / \widetilde{g}^y, \widetilde{g}) \stackrel{?}{=} e(w, \widetilde{g}^{\alpha} / \widetilde{g}^i)$.

$\textbf{PolyCommit}_{\text{DL}}$ commitments are unconditionally hiding when V knows at most $m-1$ evaluations of a committed degree-m polynomial, and computationally hiding under the DL assumption when V knows exactly m evaluations. Clearly, there is no hiding when V knows $m+1$ or more evaluations, since these evaluations completely determine f. Binding is computational under the n-polyDH assumption [13]. A trapdoor prover of course violates n-polyDH, which is how it can open commitments to arbitrary evaluations.

Proving knowledge of a committed polynomial.

Kate et al. describe a protocol for zero-knowledge proofs of knowledge of an evaluation of a committed polynomial (at a public input) [14]. In their protocol, P and V share a common polynomial commitment $\mathscr{C}$ and input i, but only P knows $f(i) \in \mathbb{Z}_q^*$ and a witness w such that $e(\mathscr{C} / \widetilde{g}^{f(i)}, \widetilde{g}) = e(w, \widetilde{g}^{\alpha} / \widetilde{g}^i)$. The protocol lets P prove knowledge of $f(i)$ without revealing anything about it (or w) to V. We point out that if P can compute proofs of knowledge for *arbitrary* evaluations of a committed polynomial, then this implies that P knows the committed polynomial itself (since knowing just $\deg f + 1$ evaluations of f is sufficient for P to interpolate and determine f). This observation suggests the following, extremely simple protocol for zero-knowledge proofs of knowledge of a committed polynomial: V chooses an evaluation point $a \in_{\text{R}} \mathbb{Z}_q$ and sends it to P. P receives a from V and engages in Kate et al.'s zero-knowledge proof of knowledge of the evaluation of f at $x = a$ with V [14, Appendix E]. If P can cause V to accept with a probability that is nonnegligibly greater than $(\deg f)/q$ (which is itself a negligible function in $\lg q$), then any knowledge extractor for Kate et al.'s protocol can compute f by extracting $\deg f + 1$ random evaluations of f from P and then interpolating.

Proving that a committed polynomial has degree at most k.

Fix a polynomial $f \in \mathbb{Z}_q[x]$ and an alleged upper bound k for $\deg f$. Suppose P and V both know the polynomial commitment $\mathscr{C} = \widetilde{g}^{f(\alpha)}$ to f, but only P knows f itself. P can convince V that $\deg f \leq k$ using the following procedure.

P1: Let $f_k(x) = x^{(n_0-k)} \cdot f(x)$. Write $f = b_k x^k + \cdots + b_1 x + b_0$ (some leading coefficients might be zero). Compute $\mathscr{C}' = \prod_{i=0}^{k} \left(\widetilde{g}^{\alpha^{i+(n_0-k)}}\right)^{b_i} = \widetilde{g}^{f_k(\alpha)}$ and send $(\mathscr{C}', k)$ to V.

V2: Receive $(\mathscr{C}', k)$ from P. Return "true" if and only if $e(\mathscr{C}, \widetilde{g}^{\alpha^{(n_0-k)}}) \stackrel{?}{=} e(\mathscr{C}', \widetilde{g})$.

LEMMA 1. *The above procedure is a noninteractive proof that* $\deg f \leq k$*: it is* ***complete***, ***computationally sound*** *under the n-polyDH assumption, and (perfectly)* ***trapdoor simulatable***.

PROOF. Both completeness and soundness follow almost immediately from the observation that $\deg f_k \leq n_0$ if and only if $\deg f \leq k$. In particular, if $\deg f \leq k$, then P can compute $\mathscr{C}'$ as above using PK; honest V will always accept since $e(\mathscr{C}', \widetilde{g}) = e(\mathscr{C}^{\alpha^{(n_0-k)}}, \widetilde{g}) = e(\mathscr{C}, \widetilde{g}^{\alpha^{(n_0-k)}})$ and $\widetilde{g}^{\alpha^{(n_0-k)}} \in \text{PK}$. Conversely, if $\deg f > k$, then the *strong correctness* property of $\textbf{PolyCommit}_{\text{DL}}$ commitments [13, Theorem 3.5] implies that, under the n-polyDH assumption, no (non-trapdoor) PPT prover can output an accepting $\mathscr{C}'$ with nonnegligible probability. One easily sees that the proof is trapdoor simulatable: a trapdoor simulator S for V just uses α to compute $\mathscr{C}' = \mathscr{C}^{\alpha^{(n_0-k)}}$ directly from $\mathscr{C}$ (i.e., without knowing f). For any given instance of the protocol induced by $(\mathscr{C}, k)$, S and V always output the same transcript $(\mathscr{C}, \mathscr{C}', k)$ and the trapdoor simulation is perfect. □

Since hiding of $\textbf{PolyCommit}_{\text{DL}}$ commitments holds with respect to trapdoor adversaries [14], Lemma 1 proves that the above procedure leaks no information about the committed polynomial f. The procedure is *not*, however, zero-knowledge with respect to non-trapdoor verifiers, since V learns $\mathscr{C}' = \mathscr{C}^{\alpha^{(n_0-k)}}$. (The trapdoor verifier can already compute this value given $(\mathscr{C}, k)$, and hence learns nothing from $\mathscr{C}'$.) It turns out that we can modify the above proof into an honest-verifier interactive zero-knowledge proof that $\deg f \leq k$. The key idea is to use the above proof, but to *blind* $\mathscr{C}'$ as $\widetilde{g}^{r_1 \cdot f_k(\alpha)}$ so that V no longer learns anything about $\widetilde{g}^{f_k(\alpha)}$. To make the verification equation work, P also sends $\left(\widetilde{g}^{\alpha^{(n_0-k)}}\right)^{r_1}$ to V, together with a standard Schnorr proof of knowledge of the discrete logarithm [21] of this latter value with respect to $\widetilde{g}^{\alpha^{(n_0-k)}}$.

P1: Let $f_k(x) = x^{(n_0-k)} \cdot f(x)$ and choose $r_1, r_2 \in_{\text{R}} \mathbb{Z}_q^*$. Compute $\mathscr{C}' = \widetilde{g}^{r_1 \cdot f_k(\alpha)}$, $R_1 = \left(\widetilde{g}^{\alpha^{(n_0-k)}}\right)^{r_1}$, and $R_2 = \left(\widetilde{g}^{\alpha^{(n_0-k)}}\right)^{r_2}$, then send $(\mathscr{C}', R_1, R_2, k)$ to V.

V2: Receive $(\mathscr{C}', R_1, R_2, k)$ from P. Choose $c \in_{\text{R}} \mathbb{Z}_q$ and send it to P.

P3: Receive c from V. Compute $v = r_2 - c\, r_1 \bmod q$ and send it to V.

V4: Receive v from P. Return "true" if and only if $\left(\widetilde{g}^{\alpha^{(n_0-k)}}\right)^{v} {R_1}^{c} \stackrel{?}{=} R_2$ and $e(\mathscr{C}, R_1) \stackrel{?}{=} e(\mathscr{C}', \widetilde{g})$.

LEMMA 2. *The above procedure is an honest-verifier perfect zero-knowledge proof that* $\deg f \leq k$*: it is* ***complete***, ***computationally sound*** *under the n-polyDH assumption, and (perfectly)* ***simulatable***.

PROOF. It is helpful to think of the above protocol as two independent subprotocols. The first subprotocol is (R_1, R_2, c, v) and the first verification equation; this is just a standard Schnorr proof of knowledge of discrete logarithm r_1 of R_1 with respect to $\widetilde{g}^{\alpha^{(n_0-k)}}$. The second subprotocol is $(R_1, \mathscr{C}', k)$ and the second verification equation; this is just the above noninteractive proof that $\deg \bar{f} \leq k$ for $\bar{f} = r_1 \cdot f$. Completeness follows because each subprotocol is complete. Soundness follows because the Schnorr subprotocol proves that P knows r_1, from which she can easily compute $(\mathscr{C}')^{1/r_1} = \widetilde{g}^{f_k(\alpha)}$; this shows that if P can cheat in this protocol she can also cheat in the earlier noninteractive protocol, contradicting Lemma 1. A simulator S for honest V works as follows:

1. S chooses $r \in \mathbb{Z}_q$, then computes $\mathscr{C}' = \mathscr{C}^r$ and $R_1 = \widetilde{g}^r$,
2. S simulates a Schnorr proof of knowledge of r' such that $R_1 = \left(\widetilde{g}^{\alpha^{(n_0-k)}}\right)^{r'}$; let the simulated transcript from this Schnorr proof be (R_1, R_2, c, v), and
3. S outputs the simulated transcript $(\mathscr{C}, \mathscr{C}', R_1, k), (R_1, R_2, c, v)$.

It is easy to see that the distribution of $(\mathscr{C}, \mathscr{C}', R_1, k)$ that S outputs is identical to the distribution that arises when honest V interacts with honest P on input $(\mathscr{C}, k)$. The Schnorr protocol itself is honest-verifier perfectly simulatable; hence, the distribution of (R_1, R_2, c, v) that S outputs, conditioned on $R_1 = R_1^* \in \mathbb{G}$, is identical to the distribution that arises when honest V interacts with honest P. Since the first argument implies that each $R_1^* \in \mathbb{G}$ arises with equal probability in real and simulated transcripts, this proves that the joint probability distribution $(\mathscr{C}, \mathscr{C}', R_1, k), (R_1, R_2, c, v)$ is also identical in both the real case and the simulated case; i.e., S simulates the honest verifier perfectly. □

4.2 Bounded all-but-k commitments from polynomial commitments

In this subsection, we describe a bounded all-but-k multiset commitment scheme based on **PolyCommit**$_{\text{DL}}$ commitments. This construction encodes commitment values into the roots of committed polynomials to implement all-but-k multiset commitments.

Bounded all-but-k multiset commitments from roots of polynomials.

Our bounded all-but-k multiset commitments encode both hard- and soft-committed values as the roots of certain polynomials: $f_H(x) = \prod_{i \in H}(x - c_i)$, $f_S(x) = \prod_{j \in S}(x - c_j)$, and $f_O(x) = \prod_{i \in [1,n]}(x - c_i)$. P commits to the set of hard-committed values in **AllButK-Commit**$_{\text{PK}}$ by creating a **PolyCommit**$_{\text{DL}}$ commitment to a random, nonzero multiple of f_H. To open the commitment in **AllButK-Open**$_{\text{PK}}$, P commits to a random, nonzero multiple of f_S and then proves in $\mathbb{G}_{\text{T}}$ that the product of the two committed polynomials is a scalar multiple of f_O. P and V conclude the interaction by having P prove in zero-knowledge that she knows f_S and f_H, and that $\deg f_S \leq k$.

1. **AllButK-Initialize**(τ, n_0)**:** Generate a group $\widetilde{\mathbb{G}}$ and generator $\widetilde{g} \in \widetilde{\mathbb{G}}$ such that $\widetilde{\mathbb{G}}$ is a pairing-friendly elliptic curve of order $q \approx 2^{2\tau}$ in which the n_0-SDH and n_0-polyDH problems are both hard. Choose $\alpha \in_{\text{R}} \mathbb{Z}_q^*$, then output $\text{PK} = \langle \widetilde{\mathbb{G}}, q, \widetilde{g}^{\alpha^i} \mid i \in [0, n_0] \rangle$ and securely delete α.
2. **AllButK-Commit**$_{\text{PK}}(H, \mathscr{H})$**:** If $H \not\subseteq [1, n_0]$, $|\mathscr{H}| \neq |H|$, or $\mathscr{H} \not\subseteq \mathbb{Z}_q$, then output $\perp$. Choose $r_1 \in_{\text{R}} \mathbb{Z}_q^*$, then compute $f_H(x) = \prod_{c_i \in \mathscr{H}}(x - c_i)$ and output $\mathscr{C} = \widetilde{g}^{r_1 \cdot f_H(\alpha)}$ and $d = \{\mathscr{H}, H, r_1\}$.
3. **AllButK-Reveal**$_{\text{PK}}(\mathscr{C}, \mathscr{O}, k\,[, d])$**:**
   - **3 a. P:** Compute $\mathscr{S} = \mathscr{O} - \mathscr{H}$ and $f_S(x) = \prod_{c_j \in \mathscr{S}}(x - c_j)$. Choose $r_2 \in_{\text{R}} \mathbb{Z}_q^*$ and send $G = \widetilde{g}^{r_2 \cdot f_S(\alpha)}$ to V. (Note that $\mathscr{H} \in d$.)
   - **3 b. V:** Receive G from P. Compute $f_O(x) = \prod_{c_i \in \mathscr{O}}(x - c_i)$.
   - **3 c. P and V:** Engage in an interactive zero-knowledge proof of knowledge of the polynomials committed to in $\mathscr{C}$ and G, and of the discrete logarithm of $e(\mathscr{C}, G)$ with respect to $e(g^{f_O(\alpha)}, \widetilde{g})$, followed by a zero-knowledge proof that $f_S(x)$ has degree at most k. V outputs "true" if and only if all of the zero-knowledge proofs succeed; otherwise, V outputs "false".

THEOREM 1. *The above triple* (**AllButK-Initialize**, **AllButK-Commit**$_{\text{PK}}$, **AllButK-Reveal**$_{\text{PK}}$) *of algorithms constitutes a secure n_0-bounded all-but-k multiset commitment scheme. Hiding is unconditional, and binding holds under the n_0-polyDH and n_0-SDH assumptions.*

PROOF.

1. **Correctness:** $e(\mathscr{C}, G) = e(\widetilde{g}^{r_1 \cdot f_H(\alpha)}, \widetilde{g}^{r_2 \cdot f_S(\alpha)}) = e(\widetilde{g}^{(f_H f_S)(\alpha)}, \widetilde{g})^{r_1 r_2} = e(\widetilde{g}^{f_O(\alpha)}, \widetilde{g})^{r_1 r_2}$; thus, P knows $r_1 r_2$ and can prove knowledge of it in the first zero-knowledge proof in **AllButK-Reveal**$_{\text{PK}}$. If $\deg f_S \leq k$, then P can always make V accept the zero-knowledge proofs in **AllButK-Reveal**$_{\text{PK}}$; hence, honest P always makes honest V output "true".
2. **Binding:** In **AllButK-Reveal**$_{\text{PK}}$, P proves knowledge of $f_1, f_2 \in \mathbb{Z}_q[x]$ and $k \in \mathbb{N}$ such that $\mathscr{C} = \widetilde{g}^{f_1(\alpha)}$, $G = \widetilde{g}^{f_2(\alpha)}$, and $\deg f_2 \leq k$. P also proves knowledge of $r' \in \mathbb{Z}_q$ such that $e(\mathscr{C}, G) = e(\widetilde{g}^{f_O(\alpha)}, \widetilde{g})^{r'}$. From f_1, f_2, and f_O, P can easily compute the following rational function:
$$r(x) = f_1(x) f_2(x) / f_O(x).$$
Moreover, we observe that $r(x) - r'$ is zero when $x = \alpha$. If $r(x)$ is non-constant (which happens if and only if $f_1(x) f_2(x)$ is not a multiple of $f_O(x)$), then P can solve for α by finding the roots of the polynomial $F(x) = f_1(x) f_2(x) - r' f_O(x)$, thereby contradicting both the n_0-SDH and n_0-polyDH assumptions. Therefore, we conclude that $r(x) = r' \in \mathbb{Z}_q$ is constant and, furthermore, that $r' \cdot f_O = f_1 f_2$. Moreover, we must have that $f_1 \mid f_O'$ for any opening $\mathscr{O}'$ of $\mathscr{C}$, since otherwise, in some other invocation of **AllButK-Reveal**$_{\text{PK}}$, P must prove knowledge of $f_1' \in \mathbb{Z}_q[x]$ such that $\mathscr{C} = \widetilde{g}^{f_1'(\alpha)}$ and $f_1 \neq f_1'$, which contradicts the binding property of **PolyCommit**$_{\text{DL}}$ commitments. Note that $f_1 \mid f_O'$ implies that $\mathscr{H} \subseteq \mathscr{O}'$ (where $\mathscr{H} = f_1^{-1}(0)$ is the set of roots of f_1), which concludes the proof.
3. **Hiding:** The **PolyCommit**$_{\text{DL}}$ commitments $\mathscr{C}$ and G are both unconditionally hiding, and the proof that $\mathscr{O}$ is a scalar multiple of the product of the polynomials committed to in $\mathscr{C}$ and G is zero-knowledge and therefore leaks no information about $\mathscr{H}$ and $\mathscr{S}$.

□

5. BATCH ZERO-KNOWLEDGE PROOF AND VERIFICATION, REVISITED

Defining batch zero-knowledge proofs and batch zero-knowledge proofs of knowledge.

We propose the following formal definition of a batch zero-knowledge proof. The key difference between our batch zero-knowledge proof definition and the standard zero-knowledge proof definition is the relationship between the *soundness* and the *asymptotic cost* of the protocol. We also restrict the prover to be PPT, since the protocol we propose uses computationally binding all-but-k commitments. (Some authors prefer to use the name "zero-knowledge arguments" for protocols that are only zero-knowledge with respect to PPT provers [8].) Peng et al. [20, §4] suggested a different formal definition for batch zero-knowledge proofs; however, their definition does not explicitly address the unique asymptotic properties that differentiate existing batch zero-knowledge protocols from regular zero-knowledge protocols.

DEFINITION 5. *Fix a soundness parameter $\lambda \in \mathbb{N}$ and a security parameter $\tau \in \mathbb{N}$. An interactive protocol between a prover P and a verifier V is a **batch zero-knowledge proof** for proposition $p(x)$ if it satisfies each of the following four criteria.*

1. ***Completeness:*** *If $p(x)$ is true, then honest V interacting with honest P always accepts the proof.*
2. ***Soundness:*** *If $p(x)$ is false, then honest V rejects the proof with probability at least $1-\varepsilon(\lambda)$, for some negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$.*
3. ***Zero-knowledge:*** *For every possibly deviating PPT verifier V, there exists a PPT algorithm S called a **simulator** for V that outputs simulated "interaction transcripts" from a distribution that is "indistinguishable" from the distribution of real interaction transcripts that arise from protocol runs between that V and honest P. If the distributions are computationally indistinguishable to a PPT adversary, then the proof is computational zero-knowledge; if they are statistically indistinguishable, then the proof is statistical zero-knowledge; finally, if they are identical, then the proof is perfect zero-knowledge (see Goldreich et al. [11] for formal definitions of the various notions of indistinguishability).*
4. ***Batching:*** *For $p(x)$ with fan-in $n > 1$ (i.e., for propositions $p(x)$ about batches of $n > 1$ predicates), the computation and communication costs are in $\widetilde{O}(\tau + \lambda n)$; intuitively, this property implies that P and V each only need to compute $O(1)$ full-length exponentiations and $O(\lambda n)$ multiplications, and they only need to send $O(1)$ group elements and $O(\lambda n)$ additional bits to one another.*

*If it is only possible to construct a simulator for honest V, but each of the other properties hold, then we call the protocol **honest-verifier batch zero-knowledge**. A protocol for proposition $p \in$ NP is called an **(honest-verifier) batch zero-knowledge proof of knowledge** if it satisfies the below knowledge extractability criterion (which replaces the above soundness criterion).*

2′. ***Knowledge extractability:*** *For every possibly deviating PPT prover P, there exists a PPT algorithm E called a **knowledge extractor** for P such that, if V accepts proofs from P with probability ρ, then given black box access to P, E outputs an NP-witness for $p(x)$ with probability at least $\frac{1}{g(\lambda)}(\rho - \varepsilon(\lambda))^c$ for some polynomial g, constant $c > 0$, and negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}$.*

Repairing Peng and Bao's protocol.

Fixing Peng and Bao's protocol with an all-but-one commitment scheme is straightforward. Let PK $\leftarrow$ **AllButK-Initialize**(τ, n_0) for some suitable parameters $\tau, n_0 \in \mathbb{N}$ in a secure all-but-k commitment scheme.[4] The setup is similar to before: Both P and V know PK, the same two generators $g, h \in \mathbb{G}$, and a set of $n \le n_0$ pairs of group elements $\{(g_i, h_i) \mid i \in [1,n]\}$, but only P knows an index $j \in [1,n]$ and exponent $x_j \in \mathbb{Z}_q$ such that $x_j = \log_g g_j = \log_h h_j$. 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. As before, $\lambda \in \mathbb{N}$ is the soundness parameter and $H = [1,n] - \{j\}$. (Note that in addition to sharing $2n$ elements of $\mathbb{G}$ as in Peng and Bao's setting, P and V must now share an additional n_0 elements of $\widetilde{\mathbb{G}}$.)

P1: Choose $c_i \in_R [0, 2^\lambda - 1]$ for $i \in H$. Compute $(\mathscr{C}, d) \leftarrow$ **AllButK-Commit**$_{\text{PK}}(H, \{c_i \mid i \in H\})$ and send it to V.

V2: Receive $\mathscr{C}$ from P. Choose $t_i \in_R [0, 2^\lambda - 1]$ for $i \in [1,n]$. Send $(t_1, \ldots, t_n)$ to P.

P3: Receive $(t_1, \ldots, t_n)$ from V. Choose $r \in_R \mathbb{Z}_p$. 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.

V4: Receive (a, b) from P. Choose $c \in_R [0, 2^\lambda - 1]$ and send it to P.

P5: 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 $(v, c_1, \ldots, c_n)$ to V.

V6: Receive $(v, c_1, \ldots, c_n)$ from P. If each of $a \stackrel{?}{=} g^v \prod_{i=1}^n g_i^{c_i t_i}$, $b \stackrel{?}{=} h^v \prod_{i=1}^n h_i^{c_i t_i}$ and $c \stackrel{?}{=} \sum_{i=1}^n c_i \bmod 2^\lambda$ is true, and $0 \le c_i < 2^\lambda$ for all $i \in [1,n]$, then proceed to step B7; otherwise, output "false" and do not proceed.

B7: P and V engage in **AllButK-Reveal**$_{\text{PK}}(\mathscr{C}, \langle c_1, \ldots, c_n \rangle, 1\,[, d])$. V outputs "true" if and only if he also output true from **AllButK-Reveal**$_{\text{PK}}$; otherwise, V outputs "false".

THEOREM 2. *Let n, and n_0 be fixed positive integers with $n \le n_0$. If (**AllButK-Initialize**, **AllButK-Commit**$_{\text{PK}}$, **AllButK-Reveal**$_{\text{PK}}$) is a secure n_0-bounded all-but-one commitment scheme with unconditional hiding, then the above protocol is an honest-verifier batch zero-knowledge proof of knowledge of index $j \in [1,n]$ and exponent $x_j \in \mathbb{Z}_p$ such that $x_j = \log_g g_j = \log_h h_j$.*

We omit the proof of Theorem 2, since Appendix B contains the proof of the analogous theorem for the protocol in the next subsection, which generalizes the above protocol.

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

We now present our batch zero-knowledge proof of knowledge and equality of k-out-of-n pairs of discrete logarithms. Let PK $\leftarrow$ **AllButK-Initialize**(τ, n_0) for some suitable parameters $\tau, n_0 \in \mathbb{N}$ in a secure all-but-k commitment scheme. In the protocol, we use $\mathbf{V}^{n \times k}$ to denote an $n \times k$ rectangular *Vandermonde matrix*:

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

[4] The commitment $\mathscr{C}$ can either be a multiset commitment or a vector commitment; the choice has negligible impact on soundness (i.e., at most a factor of n).

Both P and V know the same two generators $g, h \in \mathbb{G}$ and a set of $n \in [1, n_0]$ pairs of group elements $\{(g_i, h_i) \mid i \in [1,n]\}$, but only P knows a size-k subset $S \subseteq [1,n]$ of indices and a corresponding set $x_S = \{x_i \in \mathbb{Z}_p \mid i \in S\}$ of exponents such that $x_i = \log_g g_i = \log_h h_i$ for all $i \in S$. 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. Let $\lambda \in \mathbb{N}$ be the soundness parameter, and let $H = [1,n] - S$.

P1: Choose $c_i \in_R [0, 2^\lambda - 1]$ for $i \in H$. Compute $(\mathscr{C}, d) \leftarrow$ **AllButK-Commit**$_{PK}(H, \{c_i \mid i \in H\})$ and send it to V.

V2: Receive $\mathscr{C}$ from P. Choose $t_i \in_R [0, 2^\lambda - 1]$ for $i \in [1,n]$. Send $\langle t_1, \ldots, t_n \rangle$ to P.

P3: Receive $\langle t_1, \ldots, t_n \rangle$ from V. Choose $r \in_R \mathbb{Z}_p$. 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.

V4: Receive (a, b) from P. Choose $T_i \in_R [0, 2^\lambda - 1]$ for $i \in [1,k]$. Send $\langle T_1, \ldots, T_k \rangle$ to P.

P5: Receive $\langle T_1, \ldots, T_k \rangle$ from V. Let $\vec{c} = \langle c_1, \ldots, c_n \rangle$ and solve for $\{c_j \in [0, 2^\lambda] \mid j \in S\}$ to satisfy

$$\vec{c} \cdot \mathbf{V}^{n \times k} \equiv \langle T_1, \ldots, T_k \rangle \bmod 2^\lambda, \tag{3}$$

then compute $v = r - \sum_{i \in S} x_i c_i t_i \bmod p$. Send $(v, c_1, \ldots, c_n)$ to V.

V6: Receive $(v, c_1, \ldots, c_n)$ from P. If each of $a \stackrel{?}{=} g^v \prod_{i=1}^n g_i^{c_i t_i}$, $b \stackrel{?}{=} h^v \prod_{i=1}^n h_i^{c_i t_i}$ and $\vec{c} \cdot \mathbf{V}^{n \times k} \stackrel{?}{\equiv} \langle T_1, \ldots, T_k \rangle \bmod 2^\lambda$ is true, and $0 \le c_i < 2^\lambda$ for all $i \in [1,n]$, then proceed to step B7; otherwise, output "false" and do not proceed.

B7: P and V engage in **AllButK-Reveal**$_{PK}(\mathscr{C}, \langle c_1, \ldots, c_n \rangle, k, [, d])$. V outputs "true" if and only if he also output true from **AllButK-Reveal**$_{PK}$; otherwise, V outputs "false".

THEOREM 3. *Let k, n, and n_0 be fixed positive integers with $k \le n \le n_0$. If (***AllButK-Initialize***, **AllButK-Commit**$_{PK}$, **AllButK-Reveal**$_{PK}$*) is a secure n_0-bounded all-but-k scheme with unconditional hiding, then the above protocol is an honest-verifier batch zero-knowledge proof of knowledge of a size-k subset $S \subseteq [1,n]$ of indices and a corresponding set $x_S = \{x_i \in \mathbb{Z}_p \mid i \in S\}$ of exponents such that $x_i = \log_g g_i = \log_h h_i$ for all $i \in S$.*

The proof of Theorem 3 is in Appendix B.

6. CONCLUSION

We described a practical and devastating attack on the soundness of Peng and Bao's batch zero-knowledge proof of knowledge and equality of one-out-of-n pairs of discrete logarithms. Using our attack, a malicious prover can convince an unsuspecting verifier to accept a proof even though the claimed equality of logarithms does not hold. The vulnerability that we exploit comes about because Peng and Bao's protocol uses a type of "commitment" with insufficient binding. We noted that to fix the protocol we need a new type of commitment scheme with a rather unusual binding property; to fit the bill, we proposed and formally defined all-but-k commitments. We presented a concrete all-but-k commitment construction, and then used the special case of all-but-one commitments to repair Peng and Bao's proof. We also presented a new protocol that generalizes and significantly enhances Peng and Bao's repaired protocol using all-but-k commitments to obtain batch zero-knowledge proofs of knowledge and equality of k-out-of-n pairs of discrete logarithms. Appendix B proves the security of our protocol with respect to a new batch zero-knowledge proof definition, which we propose to capture the unique asymptotic behaviour that differentiates batch zero-knowledge proofs from conventional zero-knowledge proofs.

Acknowledgements..

Funding for this research was provided in part by NSERC, Mprime NCE (formerly MITACS) and the Ontario Research Fund. The first author is supported by an NSERC Vanier Canada Graduate Scholarship and a Cheriton Graduate Scholarship.

References

[1] Man Ho Au, Qianhong Wu, Willy Susilo, and Yi Mu. Compact e-cash from bounded accumulator. In *Proceedings of CT-RSA 2007*, volume 4377 of *LNCS*, pages 178–195, San Francisco, CA, February 2007.

[2] 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.

[3] 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.

[4] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Proceedings of EUROCRYPT 2004*, pages 223–238, Interlaken, Switzerland, May 2004.

[5] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Proceedings of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2004.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[7] 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.

[8] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Science*, 37(2):156–189, October 1988.

[9] 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, August 1994.

[10] 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, August 1986.

[11] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991.

[12] 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, May 1985.

[13] 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.

[14] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Polynomial commitments. Tech. Report CACR 2010-10, University of Waterloo, 2010.

[15] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[16] 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, Zurich, Switzerland, February 2010.

[17] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[18] 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.

[19] 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.

[20] Kun Peng, Colin Boyd, and Ed Dawson. Batch zero-knowledge proof and verification and its applications. *ACM Transactions on Information and System Security*, 10(2), May 2007.

[21] 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, August 1989.

[22] William A. Stein et al. *Sage Mathematics Software (Version 4.8)*. The Sage Development Team, 2012. `http://www.sagemath.org/`.

[23] Ernst Gabor Straus. Addition chains of vectors (problem 5125). *American Mathematical Monthly*, 71(7):806–808, September 1964.

APPENDIX

A. ATTACKING SOUNDNESS IN PENG AND BAO'S PROTOCOL

We wish to find $c_j \in [0, 2^\lambda - 1]$ for $j \in S$ to satisfy

$$0 \equiv \sum_{j \in S} \alpha_j c_j \bmod p \text{ and} \tag{1 a}$$

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

where $c' \in [0, 2^\lambda - 1]$ and $\alpha_j = t_j (x_j - y_j) \bmod p$ are given.

First, we shift the range of the desired solution so that it is centered about 0: we set

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

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

$$K \equiv \sum_{j \in S} \alpha_j d_j \bmod p \text{ and} \tag{1 b}$$

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

Consider the $(k+3)$-dimensional integer lattice M:

$$M = \begin{bmatrix} X & & & & & -K \cdot Y & -d' \cdot Y \\ & 1 & & & & \alpha_1 \cdot Y & Y \\ & & 1 & & & \alpha_2 \cdot Y & Y \\ & & & \ddots & & \vdots & \vdots \\ & & & & 1 & \alpha_k \cdot Y & Y \\ & & & & & p \cdot Y & \\ & & & & & & 2^\lambda \cdot Y \end{bmatrix},$$

where X and Y will be integers suitably chosen below. Note that if $v = [X \;\; d_1 \;\; d_2 \;\; \ldots \;\; d_k \;\; 0 \;\; 0]$ is a vector in this lattice (an integer linear combination of the rows of M), then it must be the case that $\{d_j \mid j \in [1,n]\}$ forms a required solution to equations 1 b and 2 b. This is because v must equal $1 \cdot M_0 + d_1 \cdot M_1 + \cdots + d_k \cdot M_k + a \cdot M_{k+1} + b \cdot M_{k+2}$ for some integers a and b, where M_j is the j^{th} row of M (counting from 0).

We use LLL [15] to produce a reduced basis M' for this lattice in order to find a vector of the above form. We keep only those rows of M' for which the first entry is nonzero and the last two entries are zero. Call this set of rows B. We know that the rows of M' span the rows of M, so the GCD of the leading elements of all of the rows of M' is certainly X. We now hope that the GCD of the leading elements of just the rows in B is also X. If it is, we 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 v, which contains our result.

We next very roughly compute the expected size of our solution. The determinant of M (and thus also of M') is $D = XY^2 p 2^\lambda$, and its dimension is $k+3$. By choosing $X = Y = \left\lceil \sqrt{\lceil k \rceil p 2^\lambda} \right\rceil$, we get that $D^{\frac{1}{k+3}} \approx \sqrt{\lceil k \rceil p 2^\lambda} \approx X$. 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 be about the size of X, or just a little larger. Let $k_0 = \frac{\lg p}{\lambda} + 2$. When $k \geq k_0$, $X \approx \sqrt{[k]p}2^{\lambda} \leq 2^{\lambda}/2^{\frac{\lambda}{k_0}}$, and we expect our solution to be in the required range.

We tested this using the implementation of LLL in the mathematical software package Sage [22]. When $p = 2^{160} - 47$ (the largest 160-bit prime), and $\lambda = 40$, we find that when we set $k = 6$, we can find a solution to the above equations in the appropriate range almost every time (we observed only two failures in over 6000 trials). Solving the problem took an average of about 25 ms per trial. If k is only 5, we still succeed about 52% of the time (in 1000 trials). When k is less than 5, we do not expect that a solution even *exists* in the required range, by the counting argument in §3.

B. PROOF OF THEOREM 3

THEOREM 4 (RESTATEMENT OF THEOREM 3). *Let k, n, and n_0 be fixed positive integers with $k \leq n \leq n_0$. If* (**AllButK--Initialize**, **AllButK-Commit**$_{\text{PK}}$, **AllButK-Reveal**$_{\text{PK}}$) *is a secure n_0-bounded all-but-k scheme with unconditional hiding, then the protocol at the end of §5 is an honest-verifier batch zero-knowledge proof of knowledge of a size-k subset $S \subseteq [1,n]$ of indices and a corresponding set $x_S = \{x_i \in \mathbb{Z}_p \mid i \in S\}$ of exponents such that $x_i = \log_g g_i = \log_h h_i$ for all $i \in S$.*

We prove Theorem 3 in four parts below, which correspond to the four properties in Definition 5.

Proof of completeness in Theorem 3.

Suppose that P actually knows a valid (S, x_S) pair to satisfy the claim. Since $0 < n \leq n_0$, P can always use PK to compute the commitment $\mathscr{C}$ in Step P1. To solve the system of linear equations in Step P5 with nonnegligible probability, P must hard commit to *at most* $n-k$ values from $\{c_i \mid i \in [1,n]\}$ in Step P1; on the other hand, P can pass **AllButK-Reveal**$_{\text{PK}}$ in Step B7 with nonnegligible probability if and only if she hard commits to *at least* $n-k$ values in Step P1 (so that she can prove, in **AllButK-Reveal**$_{\text{PK}}$, that she soft committed to at most k values). Therefore, if P hard commits to *exactly* $n-k$ of the $\{c_i \mid i \in [1,n]\}$ in Step P1, then she can satisfy the last two verification checks in Step V6. Moreover, if the $n-k$ hard-committed values that P commits to in Step P1 are in fact $\{c_i \mid i \in H\}$, then honest V always accepts the first two verification checks because $a = g^r \prod_{i \in H} g_i^{c_i t_i} = g^r \prod_{i \in S} g_i^{-c_i t_i} \prod_{i=1}^n g_i^{c_i t_i} = g^v \prod_{i=1}^n g_i^{c_i t_i}$, and similarly for b. □

Proof of knowledge extractability in Theorem 3.

To help us prove knowledge extractability, we first prove the following lemma and its corollary.

LEMMA 3. *Let $\lambda \in \mathbb{N}$ be a soundness parameter with $\lambda < \lg p$ and let $g_i, h_i \in \mathbb{G}$ be fixed for all $i \in [1,n]$. If $\log_g g_j \neq \log_h h_j$ for some $j \in [1,n]$ and $t_i \in_{\text{R}} [0, 2^{\lambda}-1]$ for $i \in [1,n] - \{j\}$, then there exists at most one solution $t_j \in [0, 2^{\lambda}-1]$ such that $\log_g \prod_{i=1}^n g_i^{t_i} = \log_h \prod_{i=1}^n h_i^{t_i}$.*

PROOF. Let $H = [1,n] - \{j\}$ and suppose, by way of contradiction, that there exist two distinct solutions $t_j, t_j' \in [0, 2^{\lambda}-1]$ for which

$$\log_g\Big(g_j^{t_j} \prod_{i \in H} g_i^{t_i}\Big) = \log_h\Big(h_j^{t_j} \prod_{i \in H} h_i^{t_i}\Big), \text{ and} \tag{4}$$

$$\log_g\Big(g_j^{t_j'} \prod_{i \in H}^{n} g_i^{t_i}\Big) = \log_h\Big(h_j^{t_j'} \prod_{i \in H} h_i^{t_i}\Big). \tag{5}$$

Subtracting (5) from (4) yields $\log_g g_j^{t_j - t_j'} = \log_h h_j^{t_j - t_j'}$. Because $t_j \not\equiv t_j' \bmod 2^{\lambda}$ and $t_j, t_j' < 2^{\lambda} < p$, $t_j - t_j' \not\equiv 0 \bmod p$ and we have $\log_g g_j = \log_h h_j$, which is a contradiction. □

COROLLARY 1. *Let $\lambda \in \mathbb{N}$ be a soundness parameter with $\lambda < \lg p$ and let $g_i, h_i \in \mathbb{G}$ be fixed for all $i \in [1,n]$. If $\log_g g_j \neq \log_h h_j$ for some $j \in [1,n]$, then*

$$\Pr\left[\log_g \prod_{i=1}^n g_i^{t_i} \stackrel{?}{=} \log_h \prod_{i=1}^n h_i^{t_i} \mid t_i \in_{\text{R}} [0, 2^{\lambda}-1],\ \forall i \in [1,n]\right] \leq \frac{1}{2^{\lambda}}.$$

PROOF. Lemma 3 implies that at most $\left(2^{\lambda}\right)^{n-1}$ out of $\left(2^{\lambda}\right)^n$ possible assignments of $t_1, \ldots, t_n$ in $[0, 2^{\lambda}-1]$ can satisfy $\log_g \prod_{i=1}^n g_i^{t_i} = \log_h \prod_{i=1}^n h_i^{t_i}$ when $\log_g g_j \neq \log_h h_j$ for some $j \in [1,n]$. Thus, the probability that a randomly selected assignment is a satisfying assignment is at most $\left(2^{\lambda}\right)^{n-1} / \left(2^{\lambda}\right)^n = 1/2^{\lambda}$. □

Using Corollary 1, we construct a knowledge extractor E for P that succeeds in extracting S and $\{x_j \mid j \in S\}$ with negligible knowledge error (in λ).

Extractor construction: E runs the protocol with P until it receives the first set of response $(v^{(1)}, c_1^{(1)}, \ldots, c_n^{(1)})$ from P in Step P5. At this point, E rewinds P to Step V4 and sends a new set of challenges $(T_1^{(2)}, \ldots, T_k^{(2)})$, to get a second response $(v^{(2)}, c_1^{(2)}, \ldots, c_n^{(2)})$; E repeats this rewinding step m times for some positive integer $m \geq k$. By the binding property of all-but-k commitments, with overwhelming probability in λ, there exists some size-n' subset of indices $H \subset [1,n]$, where $n' \geq n-k$, such that $c_i^{(j_1)} = c_i^{(j_2)}$ for all $i \in H$ and $j_1, j_2 \in [1, m+1]$. Moreover, because each $(v^{(j_i)}, c_1^{(j_i)}, \ldots, c_n^{(j_i)})$ satisfies a different, random system of k linear equations over $\mathbb{Z}_{(s^{\lambda})}$ in Step P5, we must also have with overwhelming probability in λ that $n' = n-k$, and that any given pair of responses from P disagrees in each of the k soft-committed values. Thus, E can extract S from any pair of responses with overwhelming probability.

For any pair of indices $j_1, j_2 \in [1, m+1]$, we have that $a = g^{v^{(j_1)}} \prod_{i=1}^n g_i^{t_i c_i^{(j_1)}} = g^{v^{(j_2)}} \prod_{i=1}^n g_i^{t_i c_i^{(j_2)}}$, so that

$$\begin{aligned} 1 = a/a &= g^{v^{(j_1)} - v^{(j_2)}} \prod_{i=1}^n g_i^{t_i (c_i^{(j_1)} - c_i^{(j_2)})} \\ &= g^{v^{(j_1)} - v^{(j_2)}} \prod_{i \in S}^n g_i^{t_i (c_i^{(j_1)} - c_i^{(j_2)})}. \end{aligned}$$

Therefore, $v^{(j_1)} - v^{(j_2)} = \log_g \prod_{i \in S} g_i^{t_i (c_i^{(j_1)} - c_i^{(j_2)})}$ and $v^{(j_1)} - v^{(j_2)} = \log_h \prod_{i \in S} h_i^{t_i (c_i^{(j_1)} - c_i^{(j_2)})}$. By Corollary 1, this only happens with negligible probability unless $\log_g g_i = \log_h h_i = x_i$ for all $i \in S$. In this case, we have that $v^{(j_1)} - v^{(j_2)}$ is a linear equation in k unknowns, namely $v^{(j_1)} - v^{(j_2)} = \sum_{i \in S} t_i (c_i^{(j_1)} - c_i^{(j_2)}) x_i \bmod p$. Given only polynomially many queries to P, E can obtain k linearly independent such equations and solve for $\{x_i = \log_g g_i \mid i \in S\}$ with overwhelming probability. □

Proof of zero-knowledge in Theorem 3.
We construct a simulator S for the honest verifier.

1. Choose $\{c_i, t_i \in_{\mathrm{R}} [0, 2^\lambda - 1] \mid i \in [1,n]\}$ and $r, v \in \mathbb{Z}_p$.
2. Compute $\mathscr{C} = g^r$, $a = \prod_{i=1}^n g_i^{c_i t_i} / g^v$, $b = \prod_{i=1}^n h_i^{c_i t_i} / h^v$ and $\vec{T} = \vec{c} \cdot \mathbf{V}^{n \times k} \bmod 2^\lambda$, where $\vec{T} = \langle T_i \mid i \in [1,k] \rangle$ and $\vec{c} = \langle c_i \mid i \in [1,n] \rangle$.
3. Output $(\mathscr{C}, \vec{t}, a, b, \vec{T}, \vec{c}, v)$ as the simulated transcript, where $\vec{t} = \langle t_i \mid i \in [1,n] \rangle$.

It is straightforward to verify that the distribution of these simulated transcripts is identical to the distribution for real transcripts. The commitment $\mathscr{C}$, small exponents $\vec{t}$, and response v are all distributed independently and uniformly at random from their respective domains in the real and simulated transcript distributions. Moreover, v completely determines a and b together with $\vec{t}$, $\vec{c}$, and $\{(g_i, h_i) \mid i \in [1,n]\}$; thus, if the distributions of $\vec{c}$ and $\vec{T}$ are identical in both distributions, then the joint distribution is also identical. It is easy to see that this is indeed the case, since choosing all but k elements of $\vec{c}$ at random and computing the remaining k to satisfy $\vec{c} \cdot \mathbf{V}^{n \times k} = \vec{T} \bmod 2^\lambda$ is equivalent to choosing all of $\vec{c}$ at random and then computing $\vec{T} = \vec{c} \cdot \mathbf{V}^{n \times k} \bmod 2^\lambda$ (since, given $\vec{T}$, there exists a *unique* solution modulo 2^λ for the missing c_i, and vice-versa). □

Proof of batching in Theorem 3.
We count the number of full-length exponentiations, multiplications (that mostly take the form of 'short exponent' exponentiations, which we count as $\approx 3/2\,\lambda$ multiplications), group element transfers, and additional bit transfers of the protocol itself, not including the all-but-k commitments. In the protocol, P computes $2 \in \Theta(1)$ exponentiations and $\approx (6\lambda + 2)(n-k) + 2k \in \Theta(\lambda\, n)$ multiplications and sends $4 \in \Theta(1)$ group elements and $\lambda\, n \in \Theta(\lambda\, n)$ additional bits to V, while V computes $2 \in \Theta(1)$ full-length exponentiations and $\approx (6\lambda + 2)\, n \in \Theta(\lambda\, n)$ multiplications and sends $0 \in O(1)$ group elements and $\lambda\,(n+k) \in \Theta(\lambda\, n)$ additional bits to P.

It is straightforward to verify that our all-but-k commitment scheme adds a constant number of exponentiations, group element transfers, and multiexponentiations. In 1964, Straus proved that a multiexponentiation with exponents modulo p is computable using $O(\lg p)$ work [23]; thus, the overhead from all-but-k commitments is just $O(1)$ full-length exponentiations and $O(1)$ group element transfers. □