How to Prove Megabytes (Per Second)

Yaron Gvili

Cryptomnium LLC
cryptomniumllc@gmail.com

Abstract. We propose the first provably secure zero-knowledge (ZK) argument of knowledge (AoK) protocol running at close to 1 megabyte per second (MBps) on commodity hardware – about an order of magnitude faster than relevant current protocols. It is a post-quantum, (efficient-prover) honest-verifier (HV) statistical zero-knowledge (SZK) sigma protocol in the standard model under a hardness assumption on ideal lattices. We further propose an overhead-efficient low-latency amortization yielding a witness indistinguishable (WI) and witness hiding (WH) AoK protocol running at > 100 MBps. Both protocols have absolute soundness slack 1, or zero for small completeness error, and an argument size growing linearly, where amortization has slope 2 and latency 1 microsecond. Non-interactive (NI), non-HV, resettable ZK (rZK) and resettable WI (rWI) variations of the protocols are obtained using standard transforms. Choices of parameters with concrete security $\geq 2^{100}$ against known attacks are given along with experimental results showing practicality.

Keywords: zero-knowledge, witness indistinguishable, witness hiding, argument of knowledge, lattice-based hashing, verifiable secret sharing, large secrets

1 Introduction

A zero-knowledge proof of knowledge (ZK-PoK) is a fundamental cryptographic primitive. It is a protocol that allows a verifier to be convinced by a prover that the latter knows a secret satisfying some (public) NP-statement yet without the verifier learning any information beyond this. This work focuses on arguments (computationally-sound proofs [45]) of knowledge (AoK), where the prover is computationally bound (i.e. a probabilistic polynomial time machine), and specifically on AoK of a secret (AoKoS) where the NP-statement is that a secret x maps to a public value y via a hard-to-invert (or one-way) function $h(\cdot)$. An AoKoS is useful in constructing larger secure protocols, in particular UC-secure ones [4]. While ZK-PoK/AoK protocols provide strong security guarantees, making them potent tools in designing cryptographic protocols, slow speed and often high demand for computational resources of existing such protocols make them unacceptable for handling large secrets in practice. Our goal in this work is protocols that are practical for large secrets in terms of speed and demand for computational resources.

The study of ZK-AoKoS for large secrets is motivated by privacy-preservation applications. These often involve a commitment to a document, in which a party makes a ZK-AoK of a secret document that maps via a cryptographic hash function to a public value. As the amount and variety of personally identifying information being collected increases [48], techniques for practical ZK-AoK for large secrets would become increasingly useful in this context.

1.1 Overview of Our Techniques

Key to our techniques is fitting known and new ingredients to obtain high speed AoKs. The first ingredient is SWIFFT, a fast lattice-based hash function due to Lyubashevsky et al [44], used to identify secrets. It admits a high-performance vectorized implementation fit to its structure and is a much better starting point for high speed than e.g. large-prime modular arithmetic. The security of SWIFFT comes from the difficulty of finding a pre-image in a certain small sub-domain given an image. However, this applies to random secrets and not to e.g. a class of documents. This is addressed by a second ingredient presented, a statistically hiding (and computationally binding) commitment fit to SWIFFT that may be viewed as randomizing. As shown in detail later, SWIFFT also offers concrete security and can be extended to large secrets. However, how to make a ZK-AoK of a pre-image in the sub-domain is not obvious. Existing range proof methods can be slow e.g. due to per-bit encryption. Circuit-based ZK-AoK methods are not helpful as they require arguing correct execution for a complex (e.g. Boolean or arithmetic) circuit to represent the SWIFFT function; hence, any speed gain from using the SWIFFT function would be dwarfed by the much slower speed of such a method. Established verifiable secret sharing methods are not helpful either since they do not work for a secret in the sub-domain needed for SWIFFT. This is addressed by a third key ingredient we introduce: embeddable-splits; it yields a verifiable secret sharing scheme that can be used with SWIFFT while maintaining high speed. Critically, the structure of embeddable-splits (for verifiable secret sharing) is preserved under the SWIFFT mapping for the same small sub-domain. This enables fast arguments that work directly with SWIFFT, not with a circuit for it. However, a naïve protocol using only these ingredients (and repetition) is shown to leak secrets. This is addressed by a fourth ingredient, a partially cheating protocol that eliminates the leak while still yielding high speed. The basic protocol introduced makes use of these ingredients (and repetition) to achieve close to 1 MBps on commodity hardware experimented with. Next, an amortized protocol introduced uses a modified partially cheating protocol to achieve a much higher rate of operation, effectively recovering the slowdown due to repetitions in the basic protocol, yet is shown to have a somewhat weaker security property, WI/WH, as compared to ZK. Finally, a number of standard transformations, which apply to our protocols due to their simple structure, are used to derive additional protocol variations.

1.2 Related Work

Zero-knowledge (ZK) was introduced by Goldwasser, Micali, and Rackoff in a seminal work [58] close to 30 years ago. A huge body of literature on zero-knowledge has accumulated since and many research directions have branched, e.g. zero-knowledge types, complexity classes, and protocols. We briefly evaluate several such directions toward our rate goal, based on experimental results we are aware of and on estimates otherwise, and provide justifications for our approach.
Direct ZK-PoKoS. An early direct method for ZK-PoKoS is Schnorr's protocol [60]. This protocol is simple to implement, can run on limited hardware, and does not require repetitions for amplifying the statistical security. However, it is relatively slow and its security proof requires a hardness assumption. In this protocol, $x \in \mathbb{Z}_q$ for some large prime q, $h(x) := g^x$, and g is a generator of order q of a group G. For asymptotic security, an assumption is required that the discrete logarithm problem (DLP) in G is hard, and for concrete security q must be sufficiently large. For RSA, where $G \subset \mathbb{Z}_n^*$ and q has about 2048 bits, the protocol running time is about 8 milliseconds on commodity hardware, putting its single-thread rate at about 30 kilobytes per second (KBps). We note that 2048 bits is the minimum recommended by NIST as of 2015 for an acceptable level of security for RSA secret keys, defined over $\mathbb{Z}_n^*$, for personal identification verification [53] (though it is expected [40] to remain secure only until 2030). We further note that our rate estimate is in line with Crypto++ benchmarks [24] and that a low number of bits is in favor of the protocol for reporting its rate. For elliptic curves (EC), where $G = \mathbb{F}_q$ and q has about 256 bits, the speed advantage of EC over RSA in the protocol is offset by a smaller domain of secrets, yielding a rate of similar order of magnitude. We note that 256 bits is the minimum recommended by NIST as of 2015 for an acceptable level of security for EC for personal identification verification [53]. These yield a rate of tens of KBps which does not meet our goal. Moreover, Schnorr's protocol does not directly extend to large secrets, i.e. it cannot be directly used to prove knowledge of a large secret for a given small-sized public value.

A more recent set of direct protocols is the NI statistical ZK-PoKoS (SZK-PoKoS) protocols based on lattice problems [51]. As our basic protocol, these protocols are secure in the standard model, use a lattice-based one-way function (OWF), and require repetition for amplifying the statistical security. Moreover, these protocol have a theoretical advantage of being a PoK rather than AoK as our protocols. However, they involve a relatively complicated procedure [32] of sampling discrete Gaussian distributions over lattices (the use of discrete Gaussian sampling in lattice-based cryptography was introduced in [56]), whereas our protocols use simple sampling of uniformly distributed bit strings, and they do not offer amortized variations, whereas our protocols do. Therefore, though we are not aware of experimental results for these protocols, we expect a lower rate for these protocols than for ours. Moreover, they provide proof systems for promise problems [27] and a reduction to a lattice problem with some approximation factor, whereas our protocols provide a concrete security guarantee for decision problems. Hence, we believe these protocols do not meet our goal.

Garbled circuits. Yao's seminal work [62] introduced the technique of garbled circuits. Using this technique, an NP-statement (to be proven) is captured in a Boolean circuit that computes the relation corresponding to the language of the statement. The gate-structure of the circuit is known to both the prover and the verifier. The prover garbles the truth table of each gate and sends it to the verifier. Then, the verifier chooses to either use the circuit, by challenging the prover to run the protocol to evaluate the garbled circuit, or to check the circuit, by challenging the prover to reveal the truth tables for validation. Evaluation is more computationally demanding, often due to the use of oblivious transfers. Garbled circuit based protocols often involve a cut-and-choose [17] protocol, in which the prover sends commitments for multiple garbled circuits and the verifier chooses which of these to use and which to check. The verifier may then take the output value as a (possibly majority) vote among the output values of each circuit used or abort if that fails. In this type of protocols, typical OWFs such as SHA-256 involve a large Boolean circuit where each bit is represented by a much larger garbled value. Various optimization techniques for garbled circuits [47, 5, 39, 7] have been devised, yet these still require significant running time. A recent result [35] for a highly optimized implementation reports a measurement of 0.74 milliseconds per garbling of an AES circuit (having thousands of non-XOR-gates and taking > 100 kilobytes to garble). Hence, the rate for 128-bit AES is about 20 KBps, which does not meet our rate goal. Moreover, since AES is not contracting, this rate does not extend to large secrets: more complex circuitry is needed to prove knowledge of a large secret for a given small-sized public value.

PCP and ZK-SNARK. Probabilistically Checkable Proofs (PCPs) are interactive proof systems in which for a given proof, generated by the prover, the verifier has oracle-access to a limited number of bits chosen using limited randomness. Zero-Knowledge Succinct Non-interactive ARguments of Knowledge (ZK-SNARKs) are ZK-AoK in which the complexity of the argument size is small, possibly constant. PCP and ZK-SNARK protocols are similar for our purposes here since in both types of protocols the bulk of the work is done by the prover while the verifier is light-weight. The large prover effort required by these protocols make them inappropriate for our goal, as elaborated next.

The PCP theorem [2] is an important result in complexity theory stating that any NP language has a PCP system with logarithmic bound on the randomness and a constant bound on the number of bits accessed. A large body of literature followed on the connections between PCP and hardness of approximation as well as cryptography [3]. Protocols with both a short PCP, e.g. [12], and a long PCP, e.g. [36], have been proposed. Such protocols use a PCP for a Boolean or arithmetic circuit for the NP-statement. However, they require at least a linear and often super-linear number of slow (e.g. encryption) cryptographic prover operations. Hence we believe these techniques are not appropriate for our goal.

ZK-SNARKs have received significant attention recently [34, 41, 31, 42, 14, 10, 8, 9, 26, 11, 20, 49]. Techniques presented in these papers build on Quadratic Span Programs (QSP), Square Span Programs (SSP), or Quadratic Arithmetic Programs (QAP), in which an NP-statement is formulated, and some only require from the verifier a constant number of cryptographic operations. A recent one [43] further requires just a linear number of cryptographic operations from the prover. Experimental results reported in [10] for ZK-SNARKs with constant-time verification are about 100 microseconds *per gate* for the prover and about 7 milliseconds for verification regardless of the number of gates; proving a 1 million gate arithmetic circuit with a 1 thousand bit input took about 100 seconds [10].

However, these PCP and ZK-SNARK techniques do not meet our rate goal since the size of a circuit, QSP, SSP, or QAP required for capturing a secure one-way function is fairly large and the (often pairing-based) cryptographic operations involved are fairly slow. For example, the prover running time reported in [10] for the ZK-SNARK of the same 1 million gate circuit yields a rate of about a few *bytes* per second; a similar rate can be estimated based on the reported per-gate costs for proving a 128-bit AES circuit. While the ZK-SNARK verifier running time is about 7 milliseconds for any circuit size, we will see later that our amortized protocol verified about 1 MB within this duration.

MPC in the head. Secure Multi-Party Computation (MPC) is used for Secure Function Evaluation (SFE) where a number of parties compute a common function that inputs a secret held by each of the parties, without revealing any information about secrets of honest parties even when up to t dishonest parties collude – a property referred to as t-privacy. MPC-in-the-head [37] is a technique for transforming an MPC protocol to a ZK-PoK protocol that roughly works as follows. First, the ZK-PoK prover simulates the views of all parties in the MPC protocol, including their send- and receive-tapes, and provides a commitment to each of these views as well as for the final output of each view. Then, the ZK-PoK verifier challenges the prover to reveal up to t views as well as outputs of all views. Finally, the ZK-PoK verifier checks that each commitment was opened correctly, and that the revealed outputs evaluate to the revealed final output, and that each pair of revealed views has consistent sent- and received-messages. Due to t-privacy, the revealed views do not provide any information about the secrets, while their consistency contributes to reducing the soundness error.

A number of ZK-PoK constructions based on MPC-in-the-head have followed. Bendlin and Damgård [13] used an MPC protocol that involves packed secret sharing [30], a generalization of Shamir's secret sharing [61] where secret values are assigned to more than one interpolation point, and hyper-invertible matrices [6]. For a security parameter n, their protocol requires interpolating a high-degree (at least $2n$) polynomial over a large field (of $\Theta(n)$ bits) to prove knowledge of a secret in the field. Moreover, the protocol requires super-linear complexity (a loose bound $O(n^4 \log q)$ is given in the paper). Damgård and López-Alt [25] described a comparatively improved MPC protocol where the amortized complexity is linear in the lattice dimension, the message is in $\mathbb{Z}_p$, and the gap between the size of the message and randomness that an honest prover chooses and the size in which an accepting verifier is convinced is zero. However, this protocol similarly requires interpolating a high-degree polynomial due to the use of packed secret sharing. It also makes use of Lagrange's four square theorem, for proving that a secret is small, that adds to the overhead (as demonstrated in our discussion of [54]). Therefore, while we are not aware of experimental results for these protocols, we believe they do not meet our goal.

Recently, ZKBoo [33] was described as a protocol for practically efficient zero-knowledge arguments especially tailored for Boolean circuits. Common to ZKBoo and our protocols is the linear argument size. However, the ZKBoo prover and verifier reportedly take a few milliseconds each for SHA-1 with an input size of 64 bytes. This yields a rate of a few KBps which does not meet our goal.

Generic ZK-PoK and ZK-AoK. Recent techniques for generic ZK-PoK and ZK-AoK obtained a rate significantly lower than our goal. Ranellucci et al [55] provide a generic ZK-PoK based on a XOR-commitment scheme that proved knowledge of an AES key mapping a plaintext to a ciphertext in less than 4 seconds with statistical security 2^{-40}. For a typical AES Boolean circuit with 35,000 gates, this yields a rate of about 10,000 gates per second. We estimate Rabin et al [54] achieved a higher rate. They provide a fast ZK-AoK based on a secret sharing scheme for secrets in a ring. Their method is similar to ours in its use of split representations of secrets. They target financial transaction applications and report a 2 millisecond running time for proving a 100 bidder Vickrey auction. Given that 3 addition and 4 multiplication gates are required for proving one inequality (of 99 in the Vickrey auction) using Lagrange's four square theorem, we estimate this translates to a proving rate of at most about 500,000 arithmetic gates per second. As typically the OWF involved require about 1000 (bitwise and arithmetic) gates for a small secret (e.g., SHA-1 took hundreds of gates [33] while AES took thousands [54]), we expect the proving rate to be at most about 500 proofs per second. For SHA-1, with an input size of 64 bytes, this yields a rate of at most about 30 KBps.

ZK-PoK for a small integer vector secret. We elaborate on this direction and compare with our work in Section 1.3.

1.3 Our Contributions

A new (basic) protocol for a statistical ZK-AoK (SZK-AoK) of a small integer vector secret is presented. This protocol is based on a new technique for verifiable secret sharing scheme that is introduced. This technique has a number of theoretical and practical advantages over previous techniques for verifiable secret sharing for a small integer vector secret. An amortized protocol that is a statistically witness hiding (SWH) and statistically witness indistinguishable (SWI) AoK of a small integer vector is presented as well. This protocol runs at a rate about two orders of magnitude faster than that of the basic protocol. While WI/WH is a weaker property than ZK, it is acceptable to many practical applications [17]. We then show a rZK (reps. RWI) variation of the former (resp. latter) protocol secure in a hybrid model. The variations are relatively simple to describe and to implement. Next, we provide a choice of protocol parameters and argue on its concrete security. Finally, experimental results for the basic protocol show that the prover (resp. verifier) ran at a rate of about $4/5$ (resp. $2^1/2$) MBps on commodity hardware, while the amortized one ran about two orders of magnitude faster yet. Thus, the protocols are practical for large secrets as defined here. We proceed to provide more details about each of our contributions.

Fast protocols for SZK-AoK of a small integer vector secret. Recent techniques that targeted ZK-AoK of a small integer vector secret are described in [4, 23, 52]. The protocols in these papers, as does the basic protocol in this work, apply to an additively homomorphic OWF over integer vectors (ivOWF) where the secret is a pre-image of this function. However, these protocols are computational ZK-AoK (CZK-AoK) whereas our basic protocols is SZK-AoK. Further, the knowledge extractors for these protocols have a relative *soundness slack*, i.e. they extract a pre-image with a norm bound that is a factor more than that required of an honest prover, whereas our protocol provides either a minimal *absolute slack* or no slack but with a small completeness error. Amortized versions of such protocols [22, 21, 21] have a low *overhead*, i.e. they generate few values using the OWF. The number of repetitions required in [4] is fairly large (for a statistical security of $k = 128$, it is $64k(1 + \log_2 k) = 2^{16}$), and this leads to larger amortization batches and higher latency. Repetitions are eliminated in [23], however relative soundness slack and high latency (due to cut-and-choose) remain. We are not aware of experimental results for these techniques. In comparison, our amortized protocol is overhead-efficient (i.e. one hash per secret) and no repetitions, matching [23]. In addition, it ran at a rate > 100 MBps in our experiments. On the other hand, we show our amortized protocol is SWH/SWI, but not ZK. Finally, whereas these protocols aim at complexity measures, ours aim at execution speed on commodity hardware.

In our basic protocol we devise a new technique for a SZK-AoK of a small integer secret $\boldsymbol{x}$ using a new verifiable secret sharing scheme. In this scheme, $\boldsymbol{x}$ is split into small integer vector shares $\boldsymbol{a}, \boldsymbol{b}$. We use a fast lattice-based hash function h secure in the standard model to get commitments $h(\boldsymbol{a}), h(\boldsymbol{b})$ for these shares. We utilize the linearity and homomorphic properties of the hash function to construct a SZK-AoK of $\boldsymbol{x}$ using the commitments for $\boldsymbol{a}, \boldsymbol{b}$. This construction is based on a new technique for hiding secrets in a small subset of a larger cyclic group that we hope is of independent interest.

An amortized protocol for SWI/SWH-AoK of a given set of small integer secrets $\{\boldsymbol{x}^{[e]}\}_{e=1}^{v}, v \in \mathbb{N}$ is introduced. In it, the basic protocol is first applied to a random small integer vector secret. Then, the resulting SZK-AoK is extended to the given (non-random) secrets $\{\boldsymbol{x}^{[e]}\}_{e=1}^{v}$ at a very small amortized cost per such secret, yielding weaker SWI/SWH properties as compared to SZK.

A number of protocol variations are obtained using known transformations that apply to the introduced protocols. Each variation is secure in some hybrid model. Variations for NI, non-HV, and rZK/rWI protocols are given.

How to apply the protocols in practice. The use of authenticated data structures to extend the protocol to large secrets is described. A large secret is split into small pieces from which a Merkle tree is constructed as an example structure. The root hash of the Merkle tree yields the public value corresponding to the large secret. In an extended protocol for a large secret, the (unextended) protocol is applied to each secret corresponding to a leaf node in the Merkle tree.

A choice of parameters for which the concrete security against known attacks is at least 2^{100} based on a cryptanalysis by Lyubashevsky et al [44] are given. In particular, a different choice of parameters for our protocol and its variations is used, due to the requirements of the commitment scheme used.

Experimental results showing our protocols are practical for large secrets. Experimental results on commodity hardware confirming that our implementation of our protocol is practical for large secrets are discussed. The results demonstrate that the basic variation of our protocol is practical for secret sizes of at least 1 MB. On a single thread, the hasher, prover, and verifier ran at a rate of about 40, $1/6$ and $1/2$ MBps respectively. On all 8 available threads, the corresponding rates are about 175, $4/5$, and $2 1/2$ MBps. The amortized protocol ran about two orders of magnitude faster than the basic protocol.

1.4 Preliminaries

Notation. Plain (resp. bold or bold capital) letters such as x (resp. $\boldsymbol{x}$ or $\boldsymbol{M}$) denote scalars (resp. vectors or matrixes). The jth element of $\boldsymbol{x}$ is $\boldsymbol{x}_j$. Indexes in superscript square-brackets, as in $x^{[i]}$ or $x^{[i,j]}$, distinguish between distinct objects. $a \oplus b$ means $a + b \mod 2$ for $a, b \in \{0, 1\}$. The notation $[a|b]$ stands for a vertical (resp. horizontal) concatenation of vectors (resp. matrices) a, b. The uniform distribution over a set or multiset b is $\mathcal{U}(b)$. $a \leftarrow b$ means that a (a scalar, vector or matrix) is drawn (and assigned) according to $\mathcal{U}(b)$. $a \sim b$ means that a is claimed to uniformly distribute over b. The notation $\mathcal{D}\{f(a)|a \leftarrow b\}$ denotes the distribution of $f(a)$ given $a \leftarrow b$. For a distribution D, $a \leftarrow D$ means a is drawn according to D. $D_1 \approx D_2$ means that D_1, D_2 are (statistically or computationally) indistinguishable. For $n \in \mathbb{N}$, we write $[n]$ for $\{1, \ldots, n\}$ and $\mathbb{Z}_n$ for $\{0, \ldots, n-1\}$. Element-wise product is denoted by $\odot$. The size in bits of s is denoted $\#s$. In $|\boldsymbol{x}|$, the absolute value is applied element-wise. Except where said otherwise, $+$ (resp. $-$) is used interchangably with 1 (resp. -1).

Organization. The rest of this paper is organized as follows. Section 2 describes constructions used in devising our protocols. Our basic and amortized protocols, along with some variations, are described in Section 3. Section 4 describes experimental results for our implementation of our protocols. Section 5 concludes.

2 Constructions

2.1 Statistically Hiding Commitment

Scheme. We show a statistically hiding commitment scheme to be used in the construction of our protocols. Our commitment scheme depends on lattice-based hash functions that are conjectured to be secure against quantum attacks [50]. It is an adaptation of one from [38]. Both schemes apply a lattice-based hash function to the concatenation of the secret and a random string (this will be shown explicitly below). However, it differs in several ways. First, it uses SWIFFT hash functions [44] that are defined for ideal lattices and admit a highly optimized implementation, rather than Ajtai hash functions [1] which are more general and less efficient. Second, the security of this commitment scheme is conditioned on the hardness of SIS (Small Integers Solution problem) over ideal lattices with parameters for which a concrete cryptanalysis is available [44], rather than that of GapSVP (Gap Small Vector Problem). Third, while both are secure in the standard model, our commitment scheme is secure for a smaller $p = 257$, rather than a larger $p \geq 2^{16}$ as implied by a lower bound in [38].

The setting is as follows. Let $p, m, n \in \mathbb{N}$ be parameters. Let $S := \{0, 1\}$, $D := \mathbb{Z}_p$. Denote by $\mathrm{SWS}(p, m, n)$ the problem of finding a small pre-image for a given SWIFFT function with parameters p, m, n. An instance of SWS is also one of SIS. Fix p, m, n and let $m' := 2m$. Suppose $\mathrm{SWS}(p, m, n)$ and $\mathrm{SWS}(p, m', n)$ are computationally hard (abusing terminology, here this refer to classical and/or quantum hardness in the average case). Let $\boldsymbol{F}, \boldsymbol{G} \in \mathbb{Z}_p^{n\times m}$ be randomly chosen SWIFFT matrices, which may be chosen as in [44]. Let $f, g : S^m \to D^n$ be the corresponding SWIFFT functions $f(\boldsymbol{x}) := \boldsymbol{F}\boldsymbol{x} \mod p, g(\boldsymbol{x}) := \boldsymbol{G}\boldsymbol{x} \mod p$.

A commitment scheme to be used is described as follows. Let $\boldsymbol{x} \in S^m$ be a secret of Alice. To make a commitment for $\boldsymbol{x}$, Alice chooses a random $\boldsymbol{r} \in S^m$, computes $\boldsymbol{y} := \mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r}) := f(\boldsymbol{x}) + g(\boldsymbol{r}) \mod p$, sets $\boldsymbol{y}' := \boldsymbol{y}$, and sends $\boldsymbol{y}'$ to Bob. To decommit, Alice sends $\boldsymbol{x}, \boldsymbol{r}$ to Bob. Bob verifies the decommitment by checking that $\boldsymbol{y}' = \mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r})$.

Lemma 1 *Let $p, m, n, S, D, \mathrm{hcom}$ be as defined above. Suppose $\mathrm{S}\,WS(p, m, n)$ and $\mathrm{S}\,WS(p, m', n)$ are computationally hard and that $w := \lfloor 2^m/p^n \rfloor \geq 2$. Then for any secret $\boldsymbol{x} \in S^m$, $\boldsymbol{r} \leftarrow S^m$ the commitment $\boldsymbol{y} := \mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r})$ is statistically hiding and computationally binding, and except with negligible probability there are $\geq w$ distinct vectors $\boldsymbol{r}'$ such that $\boldsymbol{y} = \mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r}')$.*

Proof. Clearly, an honest decommitment is accepted by the verifier.
Statistically hiding. The distributions $\left(\mathcal{D}\{\mathrm{hcom}(\boldsymbol{x}^{[i]}, \boldsymbol{r}^{[i]}) | \boldsymbol{r}^{[i]} \leftarrow S^m\}\right)_{i\in\{0,1\}}$ are statistically indistinguishable for any $\boldsymbol{x}^{[0]}, \boldsymbol{x}^{[1]} \in S^m$. This follows from [46] showing that $\mathcal{D}\{g(\boldsymbol{r})|\boldsymbol{r} \leftarrow S^m\} \approx \mathcal{U}(\mathbb{Z}_p^n)$ and, by adding $f(\boldsymbol{x})$ to the distribution, that $\mathcal{D}\{\mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r})|\boldsymbol{r} \leftarrow S^m\} \approx \mathcal{U}(\mathbb{Z}_p^n)$ for any $\boldsymbol{x} \in S^m$.
Computationally binding. We show that opening a commitment to more than one value or finding a collision for the commitment are both computationally hard for our choice of parameters. Finding $\boldsymbol{x}', \boldsymbol{r}' \in S^m$ such that $\mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r}) = \mathrm{hcom}(\boldsymbol{x}', \boldsymbol{r}')$ for given $\boldsymbol{x}, \boldsymbol{r} \in S^m$ is an instance of $\mathrm{SWS}(p, m, n)$ which by assumption is computationally hard. We now show that finding $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{x}', \boldsymbol{r}' \in S^m$ such that $\mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r}) = \mathrm{hcom}(\boldsymbol{x}', \boldsymbol{r}')$ is computationally hard as well. Since $\mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r})$ is defined as $\boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{r}) \mod p$, this is equivalent to solving $\boldsymbol{F}\boldsymbol{a} + \boldsymbol{G}\boldsymbol{b} = 0 \mod p$ for $\boldsymbol{a}, \boldsymbol{b} \in T^m \equiv \{-1, 0, 1\}^m$ where $\boldsymbol{a} := \boldsymbol{x} - \boldsymbol{x}'$, $\boldsymbol{b} := \boldsymbol{r} - \boldsymbol{r}'$. In turn, this is equivalent to solving $\boldsymbol{H}\boldsymbol{c} = 0 \mod p$ for $\boldsymbol{c} \in T^2$ where $\boldsymbol{H} := [\boldsymbol{F}|\boldsymbol{G}]$, $\boldsymbol{c} := [\boldsymbol{a}|\boldsymbol{b}]$. This is an instance of $\mathrm{SWS}(p, m', n)$ assumed computationally hard.
Multiple witnesses. Given that $w := \lfloor 2^m/p^n \rfloor \geq 2$, and since the domain size of $\boldsymbol{r}$ is 2^m and the range size of hcom is p^n, then on average there are $2^m/p^n$ distinct $\boldsymbol{r}'$ such that $\mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r}') = \boldsymbol{y}$ for any choice of $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y} := \mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r})$. Since $\mathcal{D}\{\mathrm{hcom}(\boldsymbol{x}, \boldsymbol{r})|\boldsymbol{x}, \boldsymbol{r} \in S^m\}$ is indistinguishable from a uniform distribution

then except with negligble probability there are at least $\lfloor 2^m/p^n \rfloor$ distinct $\boldsymbol{r'}$ such that $\mathrm{hcom}(\boldsymbol{x},\boldsymbol{r'}) = \boldsymbol{y}$ for any choice $\boldsymbol{x} \in S^m, \boldsymbol{r} \leftarrow S^m$, $\boldsymbol{y} := \mathrm{hcom}(\boldsymbol{x},\boldsymbol{r})$. □

We will later use the linearity of the commitment in this form:

$$\mathrm{hcom}(\boldsymbol{x}^{[0]},\boldsymbol{r}^{[0]}) + \mathrm{hcom}(\boldsymbol{x}^{[1]},\boldsymbol{r}^{[1]}) = \mathrm{hcom}(\boldsymbol{x}^{[0]}+\boldsymbol{x}^{[1]},\boldsymbol{r}^{[0]}+\boldsymbol{r}^{[1]}) \pmod p \ . \quad (1)$$

SWIFFT functions, and thus our commitment scheme, extend to any finite Abelian group (or subgroup), which is a direct sum of cyclic groups: a matrix $\boldsymbol{F}$ acts on a vector $\boldsymbol{g} := (g^{\boldsymbol{x}_i})_{i=1}^m$ of elements of a (multiplicative) cyclic group G of order p with generator g as $\boldsymbol{F}\boldsymbol{g} \mod p := (g^{(\boldsymbol{F}\boldsymbol{x} \mod p)_i})_{i=1}^n$.

Pre-processing. Stronger security is obtained by pre-processing that applies a secure invertible transform. An all-or-nothing-transform (AONT), introduced by Rivest [57], or one of its variations, qualifies and is used in the rest of this paper. Fast [16] and low-cost [59] AONTs are known, including in the standard model [18]. Formally, given an AONT $\mathcal{T} : \{0,1\}^{m'} \to \{0,1\}^{m'}$, let the (non-pre-processed) secrets be $[\boldsymbol{x'}|\boldsymbol{r'}]$, let $[\boldsymbol{x}|\boldsymbol{r}] := \mathcal{T}([\boldsymbol{x'}|\boldsymbol{r'}])$ (randomness in $\boldsymbol{r'}$ is passed to $\mathcal{T}$), and apply hcom to $\boldsymbol{x},\boldsymbol{r}$. Without loss of generality, it is sufficient to use an l-AONT defined as follows: $\mathcal{T}$ is easy to invert given all bits of $[\boldsymbol{x}|\boldsymbol{r}]$ yet is computationally (or statistically) hard to extract any information on $[\boldsymbol{x'}|\boldsymbol{r'}]$ unless all except l bits ($l < m'$ and l large enough for hardness) of $[\boldsymbol{x}|\boldsymbol{r}]$ are known. Since $\mathcal{T}$ is invertible, knowledge of (resp. an AoK for) $[\boldsymbol{x}|\boldsymbol{r}]$) implies knowledge of (resp. AoK for) $[\boldsymbol{x'}|\boldsymbol{r'}]$. Further, for a set $X := \{[\boldsymbol{x}|\boldsymbol{r}]\}_{i=1}^v$ it may be computationally (or statistically) hard to extract any bit of $\{\mathcal{T}^{-1}([\boldsymbol{x}|\boldsymbol{r}])\}_{i=1}^v$ even given knowledge of certain relations on vectors in X.

2.2 Splits and Embeddings

Informally, a split transforms a cyclic group element into a pair of elements of the same group that compose back to the element. A split-embedding transforms the pair from the cyclic group to another (at least as large) cyclic group while maintaining the split structure. This is formalized next.

Definition 1 *Let $p \in \mathbb{N}, p > 1$, and let G be a (additive) cyclic group with elements $g_0,\ldots,g_{p-1}$ such that $\forall i,j \in \mathbb{Z}_p : g_i + g_j = g_{i+j \mod p}$. For convenience of notation, denote $g_p := g_0$. For a given $k \in \mathbb{N}, k \le p$, a k-*split* of G is the pair of sets $G^{[k,+]}, G^{[k,-]}$, each of cardinality k, where $G^{[k,+]} := \{g_0,\ldots,g_{k-1}\}$, $G^{[k,-]} := \{g_{p-k+1},\ldots,g_p\}$. We denote $G^{[k,\pm]} := G^{[k,+]} \cup G^{[k,-]}$. In the vector case, given $\boldsymbol{\alpha} \in \{+,-\}^m, m \in \mathbb{N}$ we denote $\boldsymbol{G}^{[k,i]} := \times_{j=1}^m G^{[k,i\boldsymbol{\alpha}_j]}$ for $i \in \{+,-\}$ and $\boldsymbol{G}^{[k,\pm]} := \boldsymbol{G}^{[k,+\boldsymbol{\alpha}]} \cup \boldsymbol{G}^{[k,-\boldsymbol{\alpha}]}$.*

Definition 2 *Let $m,k,p,G,G^{[k,+]},G^{[k,-]},G^{[k,\pm]}$ be as in Definition 1. A k-*split* of $g \in G^{[k,\pm]}$ is a pair of group elements $(g^{[+]},g^{[-]}) \in G^{[k,+]} \times G^{[k,-]}$ such that $g^{[+]} + g^{[-]} = g$. In the vector case, for $\boldsymbol{\alpha} \in \{+,-\}^m$, we denote $\boldsymbol{g}^{[+\boldsymbol{\alpha}]} + \boldsymbol{g}^{[-\boldsymbol{\alpha}]} = \boldsymbol{g}$.*

Definition 3 *Let $m, k, p, G, G^{[k,+]}, \{g_i\}_{i=0}^{p}$ be as in Definition 1. Let $q \in \mathbb{N}, q \geq p$, and let Z be a cyclic group of order q with $z, \{z_i\}_{i=0}^{q}, Z, Z^{[k,+]}, Z^{[k,-]}, Z^{[k,\pm]}$ defined analogously to Definition 1. The k-*split-embedding *of G in Z is the mapping tuple $(E^{[k,i]} : G^{[k,i]} \to Z^{[k,i]})_{i\in\{\pm,+,-\}}$ such that $E^{[k,\pm]}(g_i) := E^{[k,+]}(g_i) := z_i, E^{[k,\pm]}(g_{p-i}) := E^{[k,-]}(g_{p-i}) := z_{q-i}$ for $i \in \mathbb{Z}_k$. In the vector case, we denote $\boldsymbol{E}^{[k,i]}(\boldsymbol{g}) := \left(E^{[k,i]}(\boldsymbol{g}_j)\right)_{j=1}^{m}$ for $i \in \{+,-,\pm\}$.*

Definition 4 *A k-*RE-split *is a k-split such that $g^{[+]} \sim G^{[k,+]}, g^{[-]} \sim G^{[k,-]}$.*

RE-split stands for Random Embeddable split.

The following result shows that a k-split-embedding maintains the k-split structure; it extends to vectors element-wise.

Proposition 2 *Let notation be as in Definitions 2,3. Let $z := E^{[k,\pm]}(g), z^{[+]} := E^{[k,+]}(g^{[+]}), z^{[-]} := E^{[k,-]}(g^{[-]})$. Then $z = z^{[+]} + z^{[-]}$.*

Proof. Without loss of generality, $g^{[+]} = g_i, g^{[-]} = g_{p-j}, g = g_{i-j \mod p}$ for some $i, j \in \mathbb{Z}_p$. Applying the k-split-embedding mappings, we get $z^{[+]} = z_i, z^{[-]} = z_{q-j}, z = z_{i-j \mod q}$ and hence $z^{[+]} + z^{[-]} = z$. $\square$

For simplicity of exposition, and without loss of generality, attention will be restricted to random 2-split-embeddings in $\mathbb{Z}_p^n$. In particular, RE-split will mean 2-RE-split in the remainder of the paper. This will be sufficient for use with SWIFFT hash functions in our protocols. It is stressed that our theoretical results hold for general finite Abelian groups.

An RE-split to be used in our protocols is constructed as follows. An RE-split of a secret $x \in \{0, 1\}$ yields secret shares $x^{[0]}, x^{[1]} \in T$ where $T := \{-1, 0, 1\}$. It may be obtained by choosing $\alpha \leftarrow \{-1, 1\}, i \leftarrow \{0, 1\}, x^{[i]} \leftarrow \{0, \alpha\}$ and setting $x^{[1-i]} = \alpha x - x^{[i]}$. The RE-split has the property that revealing either $x^{[0]}$ or $x^{[1]}$ leaks no information about x. In fact, it has a stronger property: the RE-split of x yields a perfectly hiding secret sharing scheme embeddable in $\mathbb{Z}_p$ for any $p \in \mathbb{N}, p > 1$. The following lemma formalizes this.

Lemma 3 *Let $x, \alpha, x^{[0]}, x^{[1]}$ be as above. Then $\forall j, v \in \{0, 1\}, w \in \{-1, 0, 1\}$: $\Pr(x = v \mid x^{[j]} = w) = 1/2$. Moreover, $x^{[0]} + x^{[1]} = \alpha x$ holds even when embedded in $\mathbb{Z}_p$ for any $p \in \mathbb{N}, p > 1$.*

Proof. A proof sketch is given by case analysis over the choices $\alpha \leftarrow \{-1, 1\}, i \leftarrow \{0, 1\}, x^{[i]} \leftarrow \{0, \alpha\}$ followed by setting $x^{[1-i]} = \alpha x - x^{[i]}$. The 8 equally-probable cases are shown in the following table:

x	0	0	0	0	1	1	1	1
α	-1	-1	1	1	-1	-1	1	1
αx	0	0	0	0	-1	-1	1	1
$x^{[i]}$	0	-1	0	1	0	-1	0	1
$x^{[1-i]}$	0	1	0	-1	-1	0	1	0

Thus, for each of the two possible values of x, for each $j \in \{0,1\}$ there is an equal number of cases, namely two (resp. one), of $x^{[j]} = 0$ (resp. $x^{[j]} = 1$ or $x^{[j]} = -1$). This implies $\forall j, v \in \{0,1\}, w \in \{-1,0,1\} : \Pr(x = v \mid x^{[j]} = w) = 1/2$. Hence, the scheme is perfectly hiding. Moreover, since $(x^{[i]}, x^{[1-i]})$ is a 2-split of αx, it follows from Proposition 2 that $x^{[i]} + x^{[1-i]} = \alpha x$ holds when embedded in $\mathbb{Z}_p$ for any $p \in \mathbb{N}, p > 1$. □

The RE-split construction extends naturally to vectors. Let $m \in \mathbb{N}$, let $\boldsymbol{x} \in \{0,1\}^m$ be a secret, and let $\boldsymbol{x}^{[0]}, \boldsymbol{x}^{[1]} \in T^m$. A (vector) RE-split may be obtained by choosing $\boldsymbol{\alpha} \leftarrow \{-1,1\}^m$, $\boldsymbol{i} \leftarrow \{0,1\}^m$, $\boldsymbol{x}_j^{[\boldsymbol{i}_j]} \leftarrow \{0, \boldsymbol{\alpha}_j\}$ for $j \in [m]$, and setting $\boldsymbol{x}_j^{[1-\boldsymbol{i}_j]} = \boldsymbol{\alpha}_j \boldsymbol{x}_j - \boldsymbol{x}_j^{[\boldsymbol{i}_j]}$ for $j \in [m]$. This leads to the *RE-split equation*

$$\boldsymbol{x}^{[0]} + \boldsymbol{x}^{[1]} = \boldsymbol{\alpha} \odot \boldsymbol{x} \tag{2}$$

by applying Lemma 3 element-wise. The linearity and embedding properties of Equation 2 are useful in the construction of our protocol. In particular, linearity is preserved under composition with linear functions over $\mathbb{Z}_p$. Let $h : \mathbb{Z}_p^m \to \mathbb{Z}_p^n$ be a linear hash function where $p, m, n \in \mathbb{N}$. Embedding in $\mathbb{Z}_p$ and applying h to the equation, we have

$$h(\boldsymbol{x}^{[0]}) + h(\boldsymbol{x}^{[1]}) = h(\boldsymbol{\alpha} \odot \boldsymbol{x}) \pmod{p} \ . \tag{3}$$

This yields a verifiable secret sharing scheme where the public values are the hash values in Equation 3.

2.3 Slack

Some of the analyses below use notions of slack, referring to the gap between the domain of the secret from which an honest prover chooses and the domain in which an accepting verifier is convinced, as defined next.

Definition 5 *A* slack element *of $\boldsymbol{x}$ is one that is not in $G^{[k,\pm]}$. Its index in $\boldsymbol{x}$ is called a* slack index. *A* slack set *Ξ is a maximal set of slack indexes over any $\boldsymbol{x}$ given that the verifier accepted. The* absolute slack *ξ is the minimal non-negative integer (in $\mathbb{Z}_+$) such that $\boldsymbol{x}_j \in \{-k+1-\xi, \ldots, k-1+\xi\} \pmod{p}$ for $j \in [m]$ over any $\boldsymbol{x}$ given that the verifier accepted. The* slack error *ϕ is $\Pr(\xi > 0)$. All probabilities are over the prover and verifier randomness.*

We will often use these definitions for a worst-case $\boldsymbol{x}$.

We proceed to describe protocols for AoKoS in $S^m \equiv \{0,1\}$. We will implicitly use $k = 2$ for RE-splits and suppress k in the rest of this paper.

3 Protocols

We are now ready to describe our protocols. We begin with a (naïve but) leaky protocol, correct it to a basic protocol, and finish with an amortized protocol.

The setting involves $p, m, n, S, D, \text{hcom}$ such that Lemma 1 applies and hash function $h([\boldsymbol{a}|\boldsymbol{b}]) := \text{hcom}(\boldsymbol{a}, \boldsymbol{b})$ where $\boldsymbol{a}, \boldsymbol{b} \in \{-1,0,1\}^m$. Alice, the prover, wishes to make an AoKoS to Bob, the verifier.

3.1 Leaky Protocol

The leaky protocol is defined for the lattice-based hash function parameters, a security parameter u, pre-determined secrets $\boldsymbol{x} \in \{0,1\}^{m'}, \boldsymbol{\alpha} \leftarrow \{-1,1\}^{m'}$ where $\boldsymbol{x} := [\boldsymbol{x}'|\boldsymbol{r}], \boldsymbol{x}' \in \{0,1\}^m, \boldsymbol{r} \leftarrow \{0,1\}^m$, and for public $\boldsymbol{y} := h(\boldsymbol{\alpha} \odot \boldsymbol{x}) \in \mathbb{Z}_p^n$. It is specified in Figure 1.

Leaky protocol. The steps of the leaky protocol with parameters $p, m, n, u \in \mathbb{N}$ public and $\boldsymbol{x} \in \{0,1\}^{m'}, \boldsymbol{\alpha} \in \{-1,1\}^{m'}$ secret to Alice are as follows:

1. A sub-protocol, described in steps 2 through 8, is executed u times in parallel[a].
2. Alice makes a fresh RE-split $\boldsymbol{x}^{[0]}, \boldsymbol{x}^{[1]}$ of $\boldsymbol{\alpha} \odot \boldsymbol{x}$ except that $\boldsymbol{\alpha}$ is reused.
3. Alice sets $\boldsymbol{y}^{[0]} := h(\boldsymbol{x}^{[0]}), \boldsymbol{y}^{[1]} := h(\boldsymbol{x}^{[1]})$.
4. Alice sends $\boldsymbol{y}^{[0]}, \boldsymbol{y}^{[1]}$ to Bob.
5. Bob verifies that $\boldsymbol{y}^{[0]}, \boldsymbol{y}^{[1]} \in \mathbb{Z}_p^n$ and $\boldsymbol{y}^{[0]} + \boldsymbol{y}^{[1]} = \boldsymbol{y}$.
6. Bob chooses a bit $b \leftarrow \{0,1\}$ and sends b to Alice.
7. Alice reveals $\boldsymbol{x}^{[b]}$ to Bob.
8. Bob verifies that $h(\boldsymbol{x}^{[b]}) = \boldsymbol{y}^{[b]}$.

[a] We drop an implicit execution index e to simplify notation.

Fig. 1: Steps of the leaky protocol.

The sub-protocol of the leaky protocol has the following property.

Lemma 4 *Suppose that* $\mathrm{S}\,\mathit{WS}(p, m, n)$ *and* $\mathrm{S}\,\mathit{WS}(p, m', n)$ *are computationally hard. Then the sub-protocol of the leaky protocol is a perfect ZK (PZK) AoKoS.*

A proof of Lemma 4 is deferred to Appendix A.

However, the (full) protocol is leaky as multiple rounds do leak information – the entire secret after enough rounds. In particular, for each j, Bob infers that $\boldsymbol{\alpha}_j \boldsymbol{x}_j$ is 0 (resp. 1 or -1) when observations of $\boldsymbol{x}_j^{[b]}$ follow draws from the multiset $\{-1,0,0,1\}$ (resp. $\{0,1\}$ or $\{0,-1\}$) and recovers $\boldsymbol{x}_j$ as $|\boldsymbol{\alpha}_j \boldsymbol{x}_j|$. The leak arises from reusing $\boldsymbol{\alpha}$; however, this reuse is needed to prove with the same $\boldsymbol{y}$ in all rounds. The following protocol overcomes this difficultly.

3.2 Basic Protocol

Intuitively, the basic protocol works as follows. Each round of the leaky protocol is replaced with a double-round. An argument for a double-round uses partial cheating that results in a balanced distribution of the revealed shares, i.e. the same distribution for both cases $\boldsymbol{x}_j = 0$ and $\boldsymbol{x}_j = 1$ even if $\boldsymbol{\alpha}_j$ is public, for each $j \in [m]$. Prior to revealing, Alice chooses a random permutation of the 4 hidden shares for a double-round and commits to them. Using the linear statistically hiding commitment scheme, Alice is able to hide any elements. Alice and Bob each choose a bit that controls which share of each round of the double-round must be revealed. This allows Alice to avoid revealing one share of choice of one round of the double-round, which in turn allows Alice to balance the distribution of the revealed shares. However, Alice does not control which share of the other round must be revealed, which leads to soundness error 1/2 for each double-round.

The basic protocol is defined for the lattice-based hash function parameters, a security parameter u, pre-determined secrets $\boldsymbol{x} \in \{0,1\}^{m'}$ where $\boldsymbol{x} := [\boldsymbol{x}'|\boldsymbol{r}], \boldsymbol{x}' \in \{0,1\}^m, \boldsymbol{r} \leftarrow \{0,1\}^m$, and for public $\boldsymbol{\alpha} \in \{-1,1\}^{m'}$. It is specified in Figure 2. The basic protocol is a sigma protocol since Bob communicates only in step 8.

Basic protocol. The steps of the basic protocol with parameters $p, m, n, u \in \mathbb{N}, \boldsymbol{\alpha} \in \{-1,1\}^{m'}$ public and $\boldsymbol{x} \in \{0,1\}^{m'}$ secret to Alice are as follows:

1. Alice sets $\boldsymbol{y} := h(\boldsymbol{\alpha} \odot \boldsymbol{x})$ and sends $\boldsymbol{y}$ to Bob.
2. Bob verifies that $\boldsymbol{y} \in \mathbb{Z}_p^n$.
3. A sub-protocol, described in steps 4 through 13, is executed u times in parallel[a].
4. Alice chooses $k, l \leftarrow \{0,1\}$.
5. Alice draws RE-splits $(\boldsymbol{x}^{[i,0]}, \boldsymbol{x}^{[i,1]})$ of $\boldsymbol{\alpha} \odot \boldsymbol{x}$ for $i \in \{0,1\}$, with a reused $\boldsymbol{\alpha}$ and a modified splitting distribution: Alice sets D_j to $\{(0,0,0,0), (\boldsymbol{\alpha}_j, -\boldsymbol{\alpha}_j, 0, 0)\}$ if $\boldsymbol{x}_j = 0$ and otherwise to $\{(0, \boldsymbol{\alpha}_j, 0, \boldsymbol{\alpha}_j), (0, \boldsymbol{\alpha}_j, \boldsymbol{\alpha}_j, 0)\}$, and then draws $(\boldsymbol{x}_j^{[k,l]}, \boldsymbol{x}_j^{[k,1-l]}, \boldsymbol{x}_j^{[1-k,l]}, \boldsymbol{x}_j^{[1-k,1-l]}) \leftarrow D_j$, for $j \in [m']$.
6. For $i \in \{0,1\}$, Alice sets $\boldsymbol{y}^{[i,0]} := h(\boldsymbol{x}^{[i,0]})$ and sends $\boldsymbol{y}^{[i,0]}$ to Bob.
7. Bob verifies that $\boldsymbol{y}^{[i,0]} \in \mathbb{Z}_p^n$ and recovers $\boldsymbol{y}^{[i,1]} = \boldsymbol{y} - \boldsymbol{y}^{[i,0]} \mod p$ for $i \in \{0,1\}$.
8. Bob chooses $b \leftarrow \{0,1\}$ and sends b to Alice.
9. Alice sets $b_k = l$, $b_{1-k} = b \oplus b_k$ and reveals b_0.
10. Bob verifies $b_0 \in \{0,1\}$ and recovers $b_1 = b \oplus b_0$.
11. For $i \in \{0,1\}$, Alice sets $\boldsymbol{x}^{[i]} := \boldsymbol{x}^{[i,b_i]}$ and reveals $\boldsymbol{x}^{[i]}$ to Bob.
12. Bob verifies that $(\boldsymbol{x}_j^{[0]}, \boldsymbol{x}_j^{[1]}) \in \{(0,0), (0, \boldsymbol{\alpha}_j), (\boldsymbol{\alpha}_j, 0)\}$ for $j \in [m']$.
13. Bob verifies that $h(\boldsymbol{x}^{[i]}) = \boldsymbol{y}^{[i,b_i]}$ for $i \in \{0,1\}$.

[a] We drop an implicit execution index e to simplify notation.

Fig. 2: Steps of the basic protocol.

The security properties of the basic protocol are formalized as follows.

Theorem 5 *If* $\mathrm{S}\,\mathit{WS}(p,m,n)$ *and* $\mathrm{S}\,\mathit{WS}(p,m',n)$ *are computationally hard then the basic protocol is a SZK-AoKoS with soundness error* 2^{-u} *and worst-case slack error* ϕ *of* $1-(1-2^{-u})^{m'}$ *for an absolute slack* ξ *of* 1*. Further, the verifier can efficiently tell when* $\xi = 0$*.*

A proof of Theorem 5 is deferred to Appendix B.

Corollary 6 *A zero-slack protocol, defined as accepting if the basic protocol accepts and* $\xi = 0$*, is a SZK-AoKoS with soundness error* 2^{-u}*, worst-case slack error* 0*, and completeness error* $1-(1-2^{-u})^{m'}$*.*

This follows directly from Definition 5 for slack error; the verifier efficiently rejecting when $\xi = 0$ results in slack-error shifting to completeness-error.

Remark 7 Since $\boldsymbol{\alpha}$ may be public per Theorem 5, Alice may use a fixed $\boldsymbol{\alpha}$ made known to Bob once and reused with any number of secrets.

Remark 8 Given sufficiently large u', m, the slack error ϕ may be adjusted to $2^{-u'}$ by setting $u \approx 1 + u' + \log_2 m$, as derived using Stirling's approximation: $\phi = 1 - (1 - 2^{-u})^{m'} \approx 1 - (e^{-2^{-u}})^{m'} = 1 - e^{-2^{1-u+\log_2 m}} \approx 2^{1-u+\log_2 m} = 2^{-u'}$.

We proceed to analyze the resources required by the protocol. We focus on the number of random bits, communicated bits, and hash invocations required. For a given hash function used in the protocol, let s be its input size (i.e. excluding the random string) in bits and t be its output size in bits. For efficiency, since $\boldsymbol{x}^{[i]} \in \times_{j=1}^{m'} \{0, \boldsymbol{\alpha}_j\}$, it is sufficient in step 11 for Alice to send m' bits and in step 12 for Bob to recover $\boldsymbol{x}^{[i]}$ from these bits, for $i \in \{0, 1\}$. This is reflected in Table 1 that summarizes the resources required by the basic protocol – the top (resp. bottom) part is on a per step (resp. subtotal factor, e.g. u is per double-round) basis. The required random bits, communicated bits, and hash invocations are respectively $O(su), O((s+t)u), O(u)$ with very low hidden constants.

Table 1: Resources required by the basic protocol.

step number	random bits	communicated bits	hash invocations	by whom
1		t	1	Alice
4	$2u$			Alice
5	$2su$			Alice
6		$2tu$	$2u$	Alice
8	u	u		Bob
9		u		Alice
11		$4su$		Alice
13			$2u$	Bob
subtotal factor	random bits	communicated bit	hash invocations	by whom
1		t	1	Alice
u	$2 + 2s$	$1 + 4s + 2t$	2	Alice
u	1	1	2	Bob

3.3 Amortized Protocol

We begin with a brief overview of the amortized protocol. Alice has u secrets, each in $\{0,1\}^m$, to prove knowledge of. First, Alice and Bob execute the basic protocol once on $\boldsymbol{z}$, where $\boldsymbol{z} \in \{0,1\}^{m'}$ is a secret randomly chosen by Alice. Then, for each execution e, Alice makes an RE-split of $\boldsymbol{\alpha} \odot \boldsymbol{z}$ with a $\mathcal{T}$-preprocessing (cf. Section 2.1) of the eth secret as one of its inputs. This is carefully

Amortized protocol. The steps of the amortized protocol with parameters $p, m, n, u, v \in \mathbb{N}, \boldsymbol{\alpha} \in \{-1,1\}^{m'}$ public and $\{\boldsymbol{x}^{[e]} \in \{0,1\}^{m'}\}_{e=1}^{v}$ pre-processed with $\mathcal{T}$ that are secret to Alice are as follows:

1. Alice draws $\boldsymbol{z} \leftarrow \{0,1\}^{m'}$.
2. Alice and Bob execute an instance of the basic protocol with parameters $p, m, n, u, \boldsymbol{\alpha}, \boldsymbol{z}$. Alice (resp. Bob) rejects if the instance is rejected. Otherwise, Bob learns $h(\boldsymbol{\alpha} \odot \boldsymbol{z})$.
3. For $j \in [m']$, Alice sets $\boldsymbol{\beta}_j := \boldsymbol{\alpha}_j(2\boldsymbol{z}_j - 1)$.
4. A sub-protocol described in steps 5 through 8 is executed v times in parallel[a].
5. Now $\boldsymbol{y} := h(\boldsymbol{\beta} \odot \boldsymbol{x})$ yet Alice does not compute $\boldsymbol{y}$.
6. Alice sets $\boldsymbol{w} := \boldsymbol{\beta} \odot \boldsymbol{x} - \boldsymbol{\alpha} \odot \boldsymbol{z}$ and sends $\boldsymbol{w}$ to Bob.
7. For $j \in [m']$, Bob verifies that $\boldsymbol{w}_j \in \{0, -\boldsymbol{\alpha}_j\}$.
8. Bob recovers $\boldsymbol{y}$ as $h(\boldsymbol{w}) + h(\boldsymbol{\alpha} \odot \boldsymbol{z}) \mod p$.

[a] We drop an implicit execution index e to simplify notation, e.g. we write $\boldsymbol{x}, \boldsymbol{y}$ in place of $\boldsymbol{x}^{[e]}, \boldsymbol{y}^{[e]}$ respectively.

Fig. 3: Steps of the amortized protocol.

designed to keep the secrets hidden. Bob verifies by checking that the RE-splits are valid. The extra (amortized) work per secret of the amortized protocol over the basic protocol is very low.

We proceed to a formal description. The amortized protocol is defined for the lattice-based hash function parameters, a security parameter u, a number (of secrets) v, pre-determined secrets $\boldsymbol{x}^{[e]} \in \{0,1\}^{m'}$ pre-processed with $\mathcal{T}$ where $\boldsymbol{x}^{[e]} := [\boldsymbol{x}'^{[e]} | \boldsymbol{r}^{[e]}], \boldsymbol{x}'^{[e]} \in \{0,1\}^m$, $\boldsymbol{r}^{[e]} \leftarrow \{0,1\}^m$ for $e \in [v]$, and for public $\boldsymbol{\alpha} \in \{-1,1\}^{m'}$. It is specified in Figure 3. The amortized protocol is a sigma protocol since Bob communicates bits only by invoking the basic protocol.

The security properties of the amortized protocol are formalized as follows.

Theorem 9 *If $\mathrm{S}WS(p,m,n)$ and $\mathrm{S}WS(p,m',n)$ are computationally hard, if $\mathcal{T}$ is an l-AONT, and if $2^m/p^n \geq 2$ then the amortized protocol is a SWH/SWI-AoKoS with soundness error 2^{-u}, knowledge error at most 2^{-u}, and worst-case slack error ϕ of $1-(1-2^{-u})^{m'}$ for an absolute slack ξ of 1. Further, the verifier can efficiently tell when $\xi = 0$.*

A proof of Theorem 9 is deferred to Appendix C.

Corollary 10 *A zero-slack variation, defined as accepting if the amortized protocol accepts and $\xi = 0$, is a SWH/SWI-AoKoS with soundness error 2^{-u}, knowledge error 2^{-u}, worst-case slack error 0, and completeness error $1-(1-2^{-u})^{m'}$.*

This follows similarly to Corollary 6. Remarks 7 and 8 apply similarly.

We proceed to analyze the resources required by the amortized protocol. For efficiency, since $\boldsymbol{w} \in \times_{j=1}^{m'}\{0, -\boldsymbol{\alpha}_j\}$, it is sufficient in step 6 for Alice to send

m' bits and in step 7 for Bob to recover $\boldsymbol{w}$ from these bits. This is reflected in Table 2 that summarizes the resources required by the amortized protocol. It is structured the same as Table 1. The required amortized (per each of the v secrets) resources for random bits, communicated bits, and hash invocations are respectively $O(s), O(s+t), O(1)$ with very low hidden constants. The required overhead (per batch of v secrets) resources are low as well.

step number	random bits	communicated bits	hash invocations	by whom
1	$2s$			Alice
2	†	†	†	†
6		$2sv$		Alice
8			v	Bob
subtotal factor	random bits	communicated bit	hash invocations	by whom
1	$2s$	t	1	Alice
u	$2+2s$	$1+4s+2t$	2	Alice
u	1	1	2	Bob
v		$2s$		Alice
v			1	Bob

Table 2: Resources required by the amortized protocol.

† See Table 1

3.4 Protocol Variations in Hybrid Models

This section describes several transformations yielding variations of the basic and amortized protocol that are secure in some hybrid model.

NI variation. The Fiat-Shamir [29] transform is used to obtain a NI variation secure in the random oracle model. This transform replaces the verifier challenge in a sigma protocol by the output of a pseudo-random function (PRF) on the prover commitment data. Thus, in the resulting protocol variation the prover computes the challenge on its own and the variation is NI.

Non-HV variation. A transform described in [37], based on Blum's coin-flipping [15], is used to obtain a non-HV variation of the protocol secure in the commitment-hybrid model. This transform replaces each bit of the verifier's challenge with the result of a (sequential) coin-flipping between the prover and the verifier. The coin-flipping ensures that the verifier's challenge is not controlled by the verifier, and hence the verifier is not required to be honest.

rZK and rWI variations. A transform due to Canetti et al [19] is used to obtain a rZK (resp. rWI) variation of the basic (resp. amortized) protocol. The admissibility requirements of the transform are satisfied by both protocols. The resulting transformed protocol variations are secure in a hybrid model that requires a perfectly binding (and computationally hiding) commitment scheme for the prover and a perfectly hiding (and computationally binding) commitment scheme for the verifier. In rZK and rWI protocols, the verifier may reset the prover to use the same randomness. Roughly, the transform adds a commitment by the verifier to the challenge before the prover makes its commitment. The former commitment is opened before the latter one. Moreover, the verifier commitment is used as the determining message for the prover; specifically, the prover uses a secure pseudo random function (PRF) to determine its (pseudo random) actions based on the commitment of the verifier. Thus, even with the power to reset the prover, it is infeasible to make the prover produce two different responses for the same verifier challenge that underlies its commitment.

3.5 Extension to Large Secrets

In order to extend our protocols to large secrets, we take the approach of using an authenticated data structure. Many such structures fit our approach; here we describe a Merkle tree structure, which we chose to use. The leaves of the tree correspond to blocks of secret data. Each parent node of the tree corresponds to a hash value for its c children nodes (c is determined in Section 3.6). This hash value is effectively for the secret data corresponding to the leaves of the subtree of the parent node. In particular, the hash value of the root node is for secret data corresponding to the leaves of the entire tree.

The Merkle tree is used in a *hashing* process for a large secret. In this process, the hash value for the data corresponding to each leaf is computed and associated with it. Then, the process recurses up to the root of the tree, by computing the hash value for each parent node using the hash values of its children node as data. This is well-defined when the hash function contracts by a factor of c, i.e. the ratio of its input size to its output size is c.

Each of our protocols is extended to large secrets as follows. First, the depth d of a Merkle tree for a large secret s, where $\#s > 0$, is determined. For a block size b, the tree depth is $d := \lceil 1 + \log_c(\#s/b) \rceil$. If s does not exactly fit d, i.e. $1 + \log_c(\#s/b) \neq d$, then the secret may be padded to fit it. Standard padding techniques requiring a small constant work and memory (without storing the pad) exist. To avoid a length extension attack, some standard padding techniques append the length of s in a suffix block prior to zero-padding. We denote the padded secret s'. The hash value for each node in a Merkle tree of s' is computed. Zero-padding enables an optimization in this computation: the hash value of a zero block is a zero hash value that need not be explicitly computed. The hash value y computed for the root node is used as the public hash value for s. A ZK-AoKoS for s given its public hash value y proceeds as follows. First, the prover uses an instance of our basic protocol with the (padded) secret data associated with each leaf node. If any instance is rejected, the verifier rejects. Otherwise, the verifier sees a hash value for each leaf node. The verifier applies the hashing process to compute the hash value y' for the root node. Finally, the verifier accepts if $y = y'$ and rejects otherwise. A SWH/SWI-AoKoS using the amortized protocol proceeds analogously. The use of a single hash function in the extended protocols leads to the same concrete security for large secrets as for small ones.

3.6 Choice of Parameters

For small secrets, we use a SWIFFT hash function. We choose the parameters $p = 257, m = 1024, n = 64$ and define the SWIFFT hash function $H^{[1]}$ with parameters (p, m, n). Thus, we have $H^{[1]} : \{0,1\}^{1024} \to \mathbb{Z}_{257}^{64}$. The output of $H^{[1]}$ may be represented in 65 bytes. The concrete security of $H^{[1]}$ is at least 106 bits of security against known attacks (the fastest of which also requires at least 2^{102} space as well), as shown in a cryptanalysis by Lyubashevsky et al [44]. The cryptanalysis is similar when the statistically hiding commitment of Section 2.1 is considered. One alternative is to allocate half the domain size (64 bytes) to the secret and half to the random string. In this case, the cryptanalysis does not change. The number of witnesses (i.e. decommitment alternatives an unbounded attacker views) is $2^m/p^n > 2^{511}$ and Lemma 1 applies. However, security may be weaker than suggested by this; specifically, some types of attackers may be able to reject many alternative witnesses, since randomization is not full, and thus obtain non-negligible information about the secret. A second alternative is to double the domain size, allocating the added space to the random string. In this case, we define the SWIFFT hash function $H^{[2]}$ with parameters $(p, m', n), m' := 2m$. Thus, we have $H^{[2]} : \{0,1\}^{2048} \to \mathbb{Z}_{257}^{64}$. The cryptanalysis changes only slightly[1]. However, the average number of witnesses $2^{m'}/p^n > 2^{1535}$ is much larger (so Lemma 1 applies) and the randomization involved prevents leaks. This is the alternative used in our implementation described in Section 4.

For large secrets, the Merkle tree of Section 3.5 is used. When applying the basic protocol or its variations to a small secret for a leaf node, we use H for which Equation 1 holds. Here H stands for either $H^{[1]}$ or $H^{[2]}$ per the alternative used. When recursing up the tree in the hashing process, we use a modified hash function H' that outputs a truncation to 64 bytes of the representation of the output of H. This truncation is quite tiny and has virtually no impact on concrete security. Thus, H' is a function contracting by a factor of 4, from 256 to 64 bytes. We set the tree arity c to 4. Since the input to the hash for a parent node is defined as the concatenation of the hashes for each of its children nodes and since the contracting factor matches the tree arity, the concatenation size matches the input size of H', and so the recursion is well-defined. The amount of memory required by the recursion is proportional to the depth d of the Merkle tree. In many practical applications, d is relatively small: e.g. for a secret of size 1 MB (resp. 1 gigabyte, or GB) we have $d = 7$ (resp. $d = 12$). The relative overhead of extra hash values associated with internal nodes (i.e. beyond needed for hiding the input) converges to $1/3$.

[1] A generalized birthday attack would require twice as many groups, doubling the required work. An inversion attack would require sampling a larger space $\{0,1\}^{1984}$ when trying to invert while the probability that inversion would yield a binary vector remains $2^{64}/|\mathbb{Z}_{257}^{64}| \approx 2^{-448}$. A lattice attack would require an analysis over a much larger lattice dimension of 2048.

4 Experimental Results

We implemented the basic and amortized protocols described in Sections 3.2 and 3.3 and report here on experimental results obtained for them.
Structure. We report on these implementation stages:

- **hash** – compute a (root) hash for a large secret
- **prove** – Alice's sub-protocol executions (for leaves)
- **verify** – Bob's sub-protocol executions (for leaves)

For large secrets, Alice's (resp. Bob's) effort involves proving (resp. verifying) and hashing for committing (resp. for verifying a commitment) to a large secret.
Setup. The statistical security parameter u for the basic protocol is set to 100. The parameters (p, m, n) are set to $(257, 1024, 64)$. The SWIFFT hash function is set with parameters (p, m', n) where $m' := 2m$. With this setup, the hash input size s is equal to $m = 1024$ bits and its output size t is equal to $n\lceil\log p\rceil$, which is 520 bits when rounded up to the nearest byte. As shown in Table 1, the argument size A (counting bits communicated by the prover) is $t + u(1 + 4s + 2t) = 514220$ bits, yielding a blowup factor $B := A/s \approx 502$. In comparison, as shown in Table 2, the argument size of the amortized protocol A' is $t + u(1 + 4s + 2t) + 2sv$, yielding a blowup factor $B' := A'/(sv) = 2 + B/v$. Thus, $B' < B$ for $v > 1$ and B' converges to 2 (and very close to 2 for $v > 10^4$).

The experiment setup is as follows. We experimented with a single-threaded and a multi-threaded version of our implementation. In both cases, the statistical security parameter u was set to 100. The speed is approximately linear in the size of the secret and is reported as a rate. All experiments reported here were performed by running a 64-bit executable on a commodity machine with 8 GB DDR4-2133 RAM and with a processor having 4 hyper-threaded (for a total of 8 threads) cores at up to 3.5 GHz. The execution was performed in a single process and hence does not involve inter-process-communication or networking. Initialization, memory allocation, randomness generation, and pre-processing are not included. We compiled our implementation code using GCC with AVX2 instructions enabled and optimization for speed. Our implementation includes a highly-optimized SWIFFT hash function utilizing AVX2 instructions. Multi-threading was implemented using OpenMP.
Measurements. The reported measurements focus on the sustained rate of stages of the protocol. The input hashing rate in MBps and megahashes per second (MHps) was measured by number of threads as shown in Figure 4. The secret size used was 256 MB, and therefore the depth of the tree was $d = 11$. The rate was about 42 MBps using 1 thread and about 175 MBps using all 8 threads. Up to 4 threads, the average rate was about 36 MBps per thread. Beyond 4 threads hyper-threads must be used and the rate per thread was lower.

The sub-protocol execution times and input rates for the prover and verifier stages of the basic and amortized protocols are shown in Figures 5 and 6. The measurements are the same yet in different units: each execution in Figure 5 corresponds to 128 input bytes in Figure 6. The basic protocol handles one secret per u executions while the amortized protocol handles one secret per

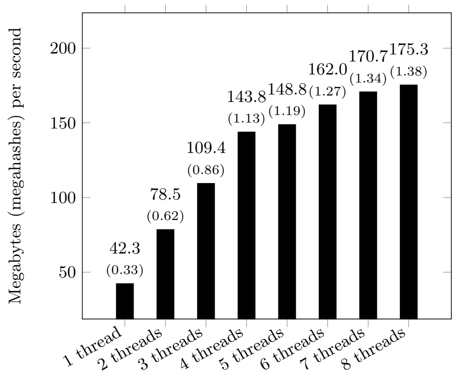

Fig. 4: Input hashing rate by number of threads ($d = 11$).

execution (out of v). The running times for the basic protocol were measured at a few microseconds per execution for 1 thread and about 5 times less for 8 threads. The amortized prover was about twice as fast while the amortized verifier was about two orders of magnitude faster than the correpsonding stages of the basic protocol stages. Latency is captured by the running time per secret on one thread. For the basic protocol, the latency is u times the running time per execution; for $u = 100$, proving (resp. verifying) latency was about 800 (resp. 300) microseconds. For the amortized protocol, the respective latencies were about $1/30$ and 1 microsecond. On 8 threads, the basic prover (resp. verifier) rate was about $4/5$ (resp. $2 1/2$) MBps; thus the basic protocol meets our goal. The respective amortized protocol rates were about 8000 and 500 MBps. With hashing (on 8 threads), the basic protocol rates were slightly slower than without hashing, while for the amortized protocol they were about 170 and 250 MBps respectively. Thus, the amortization speedup was about two orders of magnitude.

5 Conclusions

We presented the first provably secure, in the standard model under a hardness assumption on ideal lattices, ZK-AoKoS and WI/WH-AoKoS protocols that are practical for large secrets. Our protocols use a new verifiable secret sharing scheme introduced. This scheme enables hiding secrets in a small subset of a finite Abelian group using embeddable splits, a new construction we hope is of independent interest. It further enables applying SWIFFT, a fast lattice-based hash function having a concrete security guarantee, to the secret shares in constructing our protocols. The SWIFFT hash function is conjectured to be secure against quantum attacks, and our protocols inherit this property. We argued that our basic protocol is about an order of magnitude faster than other current techniques while our amortized protocols is overhead-efficient and runs about two orders of magnitude faster yet. The argument size grows linearly, with amortization slope 2. The prover and verifier resources involve small constants.

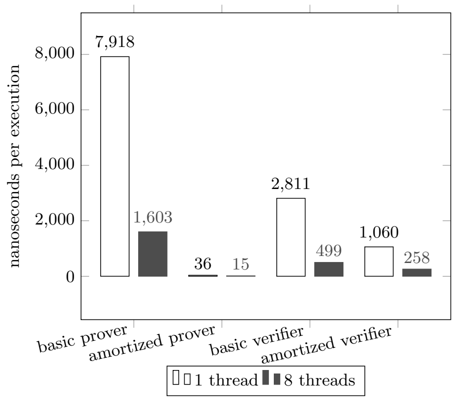

Fig. 5: Sub-protocol execution time by protocol stage.

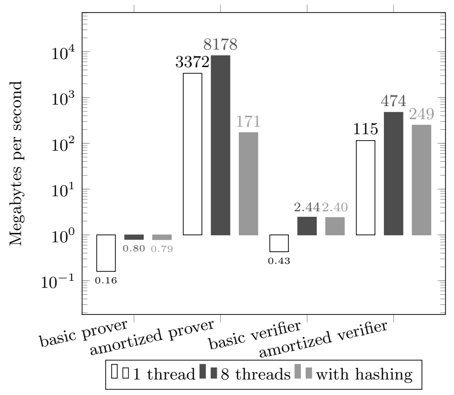

Fig. 6: Sub-protocol input rate by protocol stage ($u = 100$).

Our basic protocol is SZK while our amortized protocol is only shown to be SWI/SWH, a somewhat weaker property. Nevertheless, SWI/SWH is acceptable in many practical applications, and the amortized protocol offers an attractive speedup over the basic protocol. Each of the protocols has a negligible soundness error, a minimal worst-case absolute slack, and a variation with a zero slack at the expense of a negligible completeness error. Each is a sigma protocols that can be transformed using standard technique into NI, non-HV, and rZK/rWI protocol variations. The protocols can be used in the construction of larger (UC-)secure protocols. Thus, the protocols are relevant to a wide range of applications.

We showed how to apply our protocols in practice. Extensions to large secrets and a choice of parameters having concrete security at least 2^{100}, were described. Finally, experimental results on commodity hardware were presented. The results show a hashing rate of about 175 MBps, and proving and verifying rates of about $^4/_5$ MBps and $2^1/_2$ MBps (resp. about 8000 MBps and 500 MBps) for the basic (resp. amortized) protocol. With hashing, the respective amortized rates were about 170 and 250 MBps. The amortization speedup was about two orders of magnitude with a latency (for one secret) of about 1 microsecond.

References

1. Ajtai, M.: Generating Hard Instances of Lattice Problems (Extended Abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing. pp. 99–108. ACM (1996)
2. Arora, S., Safra, S.: Probabilistic Checking of Proofs; a New Characterization of NP. In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science. pp. 2–13. SFCS '92, IEEE Computer Society, Washington, DC, USA (1992)
3. Arora, S.: How NP Got a New Definition: A Survey of Probabilistically Checkable Proofs. CoRR cs.CC/0304038 (2002)

4. Baum, C., Damgård, I., Larsen, K.G., Nielsen, M.: How to Prove Knowledge of Small Secrets, pp. 478–498. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
5. Beaver, D., Micali, S., Rogaway, P.: The Round Complexity of Secure Protocols. In: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing. pp. 503–513. STOC '90, ACM, New York, NY, USA (1990)
6. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with Linear Communication Complexity. In: Proceedings of the 5th Conference on Theory of Cryptography. pp. 213–230. TCC'08, Springer-Verlag, Berlin, Heidelberg (2008)
7. Bellare, M., Hoang, V.T., Keelveedhi, S., Rogaway, P.: Efficient Garbling from a Fixed-Key Blockcipher. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy. pp. 478–492. SP '13, IEEE Computer Society, Washington, DC, USA (2013)
8. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In: CRYPTO. pp. 90–108. Springer (2013)
9. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable Zero Knowledge via Cycles of Elliptic Curves, pp. 276–294. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
10. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct Non-interactive Zero Knowledge for a Von Neumann Architecture. In: Proceedings of the 23rd USENIX Conference on Security Symposium. pp. 781–796. SEC'14, USENIX Association, Berkeley, CA, USA (2014)
11. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture. In: 23rd USENIX Security Symposium (USENIX Security 14). pp. 781–796. USENIX Association, San Diego, CA (2014)
12. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.: Short PCPs Verifiable in Polylogarithmic Time. In: Proceedings of the 20th Annual IEEE Conference on Computational Complexity. pp. 120–134. CCC '05, IEEE Computer Society, Washington, DC, USA (2005)
13. Bendlin, R., Damgård, I.: Threshold Decryption and Zero-knowledge Proofs for Lattice-based Cryptosystems. In: Proceedings of the 7th International Conference on Theory of Cryptography. pp. 201–218. TCC'10, Springer-Verlag, Berlin, Heidelberg (2010)
14. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct Non-interactive Arguments via Linear Interactive Proofs. In: Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography. pp. 315–333. TCC'13, Springer-Verlag, Berlin, Heidelberg (2013)
15. Blum, M.: Coin Flipping by Telephone a Protocol for Solving Impossible Problems. SIGACT News 15(1), 23–27 (Jan 1983)
16. Boyko, V.: On All-or-nothing Transforms and Password-authenticated Key Exchange Protocols. Ph.D. thesis, Cambridge, MA, USA (2000), aAI0801832
17. Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. Comput. Syst. Sci. 37(2), 156–189 (Oct 1988)
18. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient Functions and All-or-nothing Transforms. In: Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques. pp. 453–469. EUROCRYPT'00, Springer-Verlag, Berlin, Heidelberg (2000)
19. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-knowledge (Extended Abstract). In: Proceedings of the Thirty-second Annual ACM Sympo-

sium on Theory of Computing. pp. 235–244. STOC '00, ACM, New York, NY, USA (2000)
20. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile Verifiable Computation. In: Proceedings of the 2015 IEEE Symposium on Security and Privacy. pp. 253–270. SP '15, IEEE Computer Society, Washington, DC, USA (2015)
21. Cramer, R., Damgård, I., Keller, M.: On the amortized complexity of zero-knowledge protocols. J. Cryptology 27, 284–316 (2014)
22. Cramer, R., Damgård, I., Pastro, V.: On the Amortized Complexity of Zero Knowledge Protocols for Multiplicative Relations, pp. 62–79. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
23. Cramer, R., Damgård, I., Xing, C., Yuan, C.: Amortized Complexity of Zero-Knowledge Proofs Revisited: Achieving Linear Soundness Slack. In: EUROCRYPT (1). pp. 479–500. Springer (2017)
24. Dai, W.: Crypto++ – Speed Comparison of Popular Crypto Algorithms (retrieved July 16th, 2017). http://cryptopp.com/benchmarks
25. Damgård, I., López-Alt, A.: Zero-Knowledge Proofs with Low Amortized Communication from Lattice Assumptions. In: Proceedings of the 8th International Conference on Security and Cryptography for Networks. pp. 38–56. SCN'12, Springer-Verlag, Berlin, Heidelberg (2012)
26. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square Span Programs with Applications to Succinct NIZK Arguments, pp. 532–550. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
27. Even, S., Selman, A.L., Yacobi, Y.: The complexity of promise problems with applications to public-key cryptography. Information and Control 61(2), 159 – 173 (1984)
28. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: in 22nd STOC. pp. 416–426. ACM Press (1990)
29. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Proceedings on Advances in cryptology—CRYPTO '86. pp. 186–194. Springer-Verlag, London, UK, UK (1987)
30. Franklin, M., Yung, M.: Communication Complexity of Secure Computation (Extended Abstract). In: Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing. pp. 699–710. STOC '92, ACM, New York, NY, USA (1992)
31. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and Succinct NIZKs without PCPs, pp. 626–645. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
32. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008)
33. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster Zero-Knowledge for Boolean Circuits. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 1069–1083. USENIX Association, Austin, TX (2016)
34. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments, pp. 321–340. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
35. Gueron, S., Lindell, Y., Nof, A., Pinkas, B.: Fast Garbling of Circuits Under Standard Assumptions. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 567–578. CCS '15, ACM, New York, NY, USA (2015)

36. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient Arguments Without Short PCPs. In: Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity. pp. 278–291. CCC '07, IEEE Computer Society, Washington, DC, USA (2007)
37. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from Secure Multiparty Computation. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing. pp. 21–30. STOC '07, ACM, New York, NY, USA (2007)
38. Kawachi, A., Tanaka, K., Xagawa, K., Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently Secure Identification Schemes and Ad Hoc Anonymous Identification Schemes Based on the Worst-Case Hardness of Lattice Problems. In: Proceedings of Asiacrypt. LNCS, vol. 5350, pp. 372–389. Springer-Verlag, Berlin, Heidelberg (2008)
39. Kolesnikov, V., Schneider, T.: Improved Garbled Circuit: Free XOR Gates and Applications. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II. pp. 486–498. ICALP '08, Springer-Verlag, Berlin, Heidelberg (2008)
40. Laboratories, R.: TWIRL and RSA Key Size (retrieved July 16th, 2017. http://emc.com/emc-plus/rsa-labs/historical/twirl-and-rsa-key-size.htm (2016)
41. Lipmaa, H.: Progression-free Sets and Sublinear Pairing-based Non-interactive Zero-knowledge Arguments. In: Proceedings of the 9th International Conference on Theory of Cryptography. pp. 169–189. TCC'12, Springer-Verlag, Berlin, Heidelberg (2012)
42. Lipmaa, H.: Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes. In: Part I of the Proceedings of the 19th International Conference on Advances in Cryptology - ASIACRYPT 2013 - Volume 8269. pp. 41–60. Springer-Verlag New York, Inc., New York, NY, USA (2013)
43. Lipmaa, H.: Prover-Efficient Commit-and-Prove Zero-Knowledge SNARKs, pp. 185–206. Springer International Publishing, Cham (2016)
44. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A Modest Proposal for FFT Hashing, pp. 54–72. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
45. Micali, S.: Computationally Sound Proofs. SIAM J. Comput. 30(4), 1253–1298 (Oct 2000)
46. Micciancio, D.: The Geometry of Lattice Cryptography, pp. 185–210. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
47. Naor, M., Pinkas, B., Sumner, R.: Privacy Preserving Auctions and Mechanism Design. In: Proceedings of the 1st ACM Conference on Electronic Commerce. pp. 129–139. EC '99, ACM, New York, NY, USA (1999)
48. Narayanan, A., Shmatikov, V.: Myths and Fallacies of "Personally Identifiable Information". Commun. ACM 53(6), 24–26 (Jun 2010)
49. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly Practical Verifiable Computation. Commun. ACM 59(2), 103–112 (Jan 2016)
50. Peikert, C.: A Decade of Lattice Cryptography. Found. Trends Theor. Comput. Sci. 10(4), 283–424 (Mar 2016)
51. Peikert, C., Vaikuntanathan, V.: Noninteractive Statistical Zero-Knowledge Proofs for Lattice Problems. In: Proceedings of the 28th Annual Conference on Cryptology: Advances in Cryptology. pp. 536–553. CRYPTO 2008, Springer-Verlag, Berlin, Heidelberg (2008)

52. del Pino, R., Lyubashevsky, V.: Amortization with Fewer Equations for Proving Knowledge of Small Secrets. In: Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. pp. 365–394 (2017)
53. Polk, W.T., Dodson, D.F., Burr, W.E., Ferraiolo, H., Cooper, D.: Cryptographic Algorithms and Key Sizes for Personal Identity Verification (2014), `http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-4.pdf`
54. Rabin, M.O., Mansour, Y., Muthukrishnan, S., Yung, M.: Strictly-black-box Zero-knowledge and Efficient Validation of Financial Transactions. In: Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I. pp. 738–749. ICALP'12, Springer-Verlag, Berlin, Heidelberg (2012)
55. Ranellucci, S., Tapp, A., Zakarias, R.W.: Efficient Generic Zero-Knowledge Proofs from Commitments (Extended Abstract). In: Information Theoretic Security - 9th International Conference, ICITS 2016, Tacoma, WA, USA, August 9-12, 2016, Revised Selected Papers. pp. 190–212 (2016)
56. Regev, O.: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. J. ACM 56(6), 34:1–34:40 (Sep 2009)
57. Rivest, R.L.: All-or-Nothing Encryption and the Package Transform. In: Proceedings of the 4th International Workshop on Fast Software Encryption. pp. 210–218. FSE '97, Springer-Verlag, London, UK, UK (1997)
58. S. Goldwasser, S. Micali, C.R.: The knowledge complexity of interactive proof systems. SIAM Journal of Computing 18, 186–208 (1989)
59. Schartner, P.: A Low-cost Alternative for OAEP. In: Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systemss. pp. 1:1–1:6. S&D4RCES '11, ACM, New York, NY, USA (2011)
60. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology. pp. 239–252. CRYPTO '89, Springer-Verlag, London, UK, UK (1990)
61. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (Nov 1979)
62. Yao, A.C.C.: How to Generate and Exchange Secrets. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science. pp. 162–167. SFCS '86, IEEE Computer Society, Washington, DC, USA (1986)

A Proof of Lemma 4

This section sketches a proof that if SWS(p, m, n) and SWS(p, m', n) are computationally hard then the sub-protocol of the leaky protocol is a PZK-AoKoS by analyzing completeness, soundness, and perfect zero-knowledge.
Completeness. We have $\boldsymbol{y} \equiv h(\boldsymbol{\alpha} \odot \boldsymbol{x}) \equiv h(\boldsymbol{x}^{[0]} + \boldsymbol{x}^{[1]}) \equiv h(\boldsymbol{x}^{[0]}) + h(\boldsymbol{x}^{[1]}) \equiv \boldsymbol{y}^{[0]} + \boldsymbol{y}^{[1]} \pmod{p}$ where the third equivalence follows from Equation 1.
Soundness. This follows from the computational hardness of $\mathrm{S}WS(p, m, n)$ and $\mathrm{S}WS(p, m', n)$ that prevents Alice, except with negligible probability, from finding a value other than $\boldsymbol{x}^{[b]}$ that hashes to $\boldsymbol{y}^{[b]}$. Moreover, a knowledge extractor can extract Alice's secret using two challenges $0, 1$ in any round to reveal two shares for the same RE-split of the secret. The soundness error is $1/2$ since a cheating Alice will have (at least) one fake share, which would not pass verification if revealed, and with probability $1/2$ Bob will challenge to reveal it.

Perfect zero-knowledge. This follows from the RE-split being perfectly hiding. Formally, the simulator produces a transcript by choosing $b \leftarrow \{0,1\}$, choosing $\boldsymbol{x}^{[b]}$ as in an RE-split, and setting $\boldsymbol{y}^{[b]} := h(\boldsymbol{x}^{[b]}), \boldsymbol{y}^{[1-b]} := \boldsymbol{y} - \boldsymbol{y}^{[b]}$. The revealed share in a simulated round is drawn from a distribution identical to that in a real round, and the same is true for the bit. Since the revealed share and the bit determine the transcript, it follows that the distribution of simulated transcripts is identical to that of real transcripts.

B Proof of Theorem 5

This section sketches a proof that if SWS(p,m,n) and SWS(p,m',n) are computationally hard then the basic protocol of Section 3.2 is a SZK-AoKoS, that its soundeness error is 2^{-u}, that its worst-case slack error is $1-(1-2^{-u})^{m'}$ for an absolute slack $\xi = 1$, and that the verifier can tell when $\xi = 0$. The properties of completeness, soundness, statistical zero-knowledge, and slack are analyzed.

$\boldsymbol{\alpha}_j \boldsymbol{x}_j$	0	0	+	+
$\boldsymbol{x}_j^{[k,b_k]}$	0	+	0	0
$\boldsymbol{x}_j^{[k,1-b_k]}$	0	-	+	+
$\boldsymbol{x}_j^{[1-k,b_{1-k}]}$	0	0	0	+
$\boldsymbol{x}_j^{[1-k,1-b_{1-k}]}$	0	0	+	0

Table 3: Possibilities for variables in the basic protocol.

The analysis will refer to Table 3 which shows the possibilities for the shares in the basic protocol. Each of the possibilities over the choices of $\boldsymbol{x}_j, k, l$ and the draw from D_j in double-round e (for $e \in [u], j \in [m']$) of the basic protocol are listed on the right side of Table 3. Values in the table are shown for variables listed on its left side. The symbols $+, -$ stand for $\boldsymbol{\alpha}_j, -\boldsymbol{\alpha}_j$ respectively.

Completeness. We show the RE-split equation $\boldsymbol{x}^{[k,0]} + \boldsymbol{x}^{[k,1]} = \boldsymbol{\alpha} \odot \boldsymbol{x}$ holds for each $k \in \{0,1\}$ and each element $j \in [m']$. If $\boldsymbol{x}_j = 0$ then $(\boldsymbol{x}_j^{[k,l]}, \boldsymbol{x}_j^{[k,1-l]})$ is either $(\boldsymbol{\alpha}_j, -\boldsymbol{\alpha}_j)$ or $(0,0)$; either way $\boldsymbol{x}^{[k,0]} + \boldsymbol{x}^{[k,1]} = 0 = \boldsymbol{\alpha}_j \boldsymbol{x}_j$. Otherwise $\boldsymbol{x}_j = 1$ and then $(\boldsymbol{x}_j^{[k,l]}, \boldsymbol{x}_j^{[k,1-l]})$ is either $(0, \boldsymbol{\alpha}_j)$ or $(\boldsymbol{\alpha}_j, 0)$; either way $\boldsymbol{x}_j^{[k,0]} + \boldsymbol{x}_j^{[k,1]} = \boldsymbol{\alpha}_j = \boldsymbol{\alpha}_j \boldsymbol{x}_j$. Thus, in all cases the RE-split equation holds. Moreover, in both $\boldsymbol{x}_j$ cases, the share $\boldsymbol{x}^{[k,1-l]} \equiv \boldsymbol{x}^{[k,1-b_k]}$ is not revealed while the revealed shares $(\boldsymbol{x}_j^{[0,b_0]}, \boldsymbol{x}_j^{[1,b_1]}) \sim \{(0,0),(0,0)(0,\boldsymbol{\alpha}_j),(\boldsymbol{\alpha}_j,0)\}$ as seen in Table 3. Finally, for $i \in \{0,1\} : \boldsymbol{y} \equiv h(\boldsymbol{x}) \equiv h(\boldsymbol{x}^{[i,0]} + \boldsymbol{x}^{[i,1]}) \equiv h(\boldsymbol{x}^{[i,0]}) + h(\boldsymbol{x}^{[i,1]}) \equiv \boldsymbol{y}^{[i,0]} + \boldsymbol{y}^{[i,1]} \pmod{p}$ where the third equivalence follows from Equation 1. This proves completeness.

Soundness. A cheating Alice is forced to have one fake share (i.e. one causing a rejection in step 12 of the basic protocol) among $\boldsymbol{y}^{[k,0]}, \boldsymbol{y}^{[k,1]}$ for $k \in \{0,1\}$ due to the computational hardness assumption. After Bob chooses the challenge b, Alice may choose only one of b_0, b_1 and the other one is determined by b due to the verification that $b_0 \oplus b_1 = b$. Hence, Alice can avoid revealing a fake share for one round (of the double-round) while for the other round the cheating failur probability is $1/2$. Thus, the soundness error is $1/2$ for a double-round and 2^{-u} for u double-rounds. Moreover, a knowledge extractor can use two challenges $0, 1$ in any double-round to reveal two shares for the same RE-split of the secret.

Statistical zero-knowledge. This follows from the RE-split being perfectly hiding and the commitment scheme being statistically hiding. Formally, the simulator produces a transcript as follows. The simulator is given $\boldsymbol{\alpha} \in \{-1,1\}^{m'}, \boldsymbol{y} \in \mathbb{Z}_p^n$. For each double-round, the simulator performs these steps:

- choose $b, k, l \leftarrow \{0,1\}$ and set $b_k = l, b_{1-k} = b \oplus b_k$;
- for $k \in \{0,1\}, j \in [m']$ draw $(\boldsymbol{x}_j^{[k,b_k]}, \boldsymbol{x}_j^{[1-k,b_{1-k}]}) \leftarrow \{(0,0),(0,\boldsymbol{\alpha}_j)\}$;
- for $i \in \{0,1\}$, set $\boldsymbol{x}^{[i]} := \boldsymbol{x}^{[i,b_i]}$;
- for $i \in \{0,1\}$, set $\boldsymbol{y}^{[i,b_i]} := h(\boldsymbol{x}^{[i,b_i]})$;
- for $i \in \{0,1\}$, set $\boldsymbol{y}^{[i,1-b_i]} := \boldsymbol{y} - \boldsymbol{y}^{[i,b_i]} \mod p$.

The revealed vector shares in a simulated double-round are drawn from a distribution identical to that in a real double-round, while the real and simulated choices b, k, l for $i, j \in \{0,1\}$ are distributed identically as well. Since these determine the transcript, the distribution of simulated transcripts is identical to that of real ones. SZK follows by Lemma 1 applied to h.

Slack. Given that Bob accepts, a cheating Alice can only obtain an absolute slack of 1 by setting $D_{j'} := (0, 2\boldsymbol{\alpha}_{j'}, \boldsymbol{\alpha}_{j'}, \boldsymbol{\alpha}_{j'})$ for at least one element j' (a higher absolute slack has negligible success probability as it forces Alice to choose one element of a share not in $\{0, \boldsymbol{\alpha}_{j'}\}$ in each round of each double-round). Hence, at each such slack index j', the revealed element of the shares in each double-rounds would be $0, \boldsymbol{\alpha}_{j'}$ (but never $0, 0$). However, from an honest prover one expects $(\boldsymbol{x}_j^{[0,b_0]}, \boldsymbol{x}_j^{[1,b_1]}) \sim \{(0,0),(0,\boldsymbol{\alpha}_j)\}$ for $j \in [m']$ in each double-round. We now define and calculate probabilities of the following events:

1. Given $i \in [u], j \in [m']$, event $A_{i,j}$ occurs if $(0, \boldsymbol{\alpha}_j)$ elements are revealed for index j in double-round i. We have $\Pr(A_{i,j}) = 1/2$.
2. Given $j \in [m']$, event B_j occurs if $\forall i \in [u] : A_{i,j}$. We have $\Pr(B_j) = 2^{-u}$.
3. Event C occurs if $\exists j \in [m'] : B_j$. We have $\Pr(C) = 1 - (1 - 2^{-u})^{m'}$.

Event C corresponds to a slack set $\Xi \neq \emptyset$ for absolute slack $\xi = 1$. Hence, the slack error is $1 - (1 - 2^{-u})^{m'}$ for an absolute slack $\xi = 1$. Moreover, Bob can efficiently tell when $\xi = 0$ by checking that event C does not occur.

C Proof of Theorem 9

This section sketches a proof that if $\mathrm{SWS}(p, m, n)$ and $\mathrm{SWS}(p, m', n)$ are computationally hard, if $\mathcal{T}$ is an l-AONT and if $2^m/p^n \geq 2$ then the amortized protocol is SWH/SWI-AoKoS with soundeness error 2^{-u}, knowledge error at most 2^{-u}, and worst-case slack error $1 - (1 - 2^{-u})^{m'}$ for an absolute slack of 1,

$\boldsymbol{\alpha}_j \boldsymbol{z}_j$	0	0	+	+
$\boldsymbol{x}_j$	0	1	0	1
$\boldsymbol{\beta}_j$	-	-	+	+
$\boldsymbol{\beta}_j \boldsymbol{x}_j$	0	-	0	+
$\boldsymbol{w}_j$	0	-	-	0

Table 4: Possibilities for variables in the amortized protocol.

and that the verifier can efficiently tell when the slack error is 0. The properties of completeness, soundness, knowledge, WI, WH, and slack are analyzed.

The analysis will refer to Table 4 which shows the possibilities for the variables in the amortized protocol over the choices of $\boldsymbol{z}_j$ and of $\boldsymbol{x}_j$ in execution e. Values in the table are shown for variables listed on its left side. The symbols $+, -$ stand for $\boldsymbol{\alpha}_j, -\boldsymbol{\alpha}_j$ respectively. Each of $\boldsymbol{\beta}_j, \boldsymbol{\beta}_j \boldsymbol{x}_j, \boldsymbol{w}_j$ is determined by other variables shown in the table. We begin with lemmata we will need.

Lemma 11 *In each of the v executions of the amortized sub-protocol, $(\boldsymbol{w}, \boldsymbol{\alpha} \odot \boldsymbol{z})$ is an RE-split of $\boldsymbol{\beta} \odot \boldsymbol{x}$.*

Proof. $\boldsymbol{\alpha}_j \boldsymbol{z}_j \sim \{0, \boldsymbol{\alpha}_j\}, \boldsymbol{w}_j \sim \{0, -\boldsymbol{\alpha}_j\}$ per Table 4 while step 6 of the amortized protocol implies $\boldsymbol{w} + \boldsymbol{\alpha} \odot \boldsymbol{z} = \boldsymbol{\beta} \odot \boldsymbol{x}$. The lemma follows using Definition 4. $\square$

Lemma 12 *Fix $k, m, n, l, r \in \mathbb{N}, l < m \leq n, r < m$. Let $\mathcal{T} : \{0,1\}^m \to \{0,1\}^n$ be a statistical (resp. computational) l-AONT with the last r bits for randomness. For $i \in [k]$, let $\boldsymbol{x}^{[i]} \leftarrow D^{[i]}$, $\boldsymbol{y}^{[i]} := \mathcal{T}(\boldsymbol{x}^{[i]})$ where $D^{[i]}$ is any distribution on $\{0,1\}^m$ with the last r bits random. Let R be any relation on $\{\boldsymbol{y}^{[i]}\}_{i=1}^k$ with one degree of freedom. Then for any $j \in [k]$ and any $L \subset [m]$, $|L| < m - l$, the distribution $\mathcal{D}\{\boldsymbol{x}^{[j]} | R, \boldsymbol{y}_L^{[j]}\}$ where $\boldsymbol{y}_L^{[j]} := \{(i, \boldsymbol{y}_i^{[j]})\}_{i \in L}$ is statistically (resp. computationally) indistinguishable from $D^{[j]}$.*

Proof. The conditions for l-AONT and $D^{[j]}$ imply $\mathcal{D}\{\boldsymbol{x}^{[j]} | \boldsymbol{y}_L^{[j]}\} \approx D^{[j]}$, while R having one degree of freedom on $\{\boldsymbol{y}^{[i]}\}_{i=1}^k$ implies $\mathcal{D}\{\boldsymbol{y}^{[j]} | R\} \approx \mathcal{D}\{\boldsymbol{y}^{[j]}\}$. Therefore, $\mathcal{D}\{\boldsymbol{x}^{[j]} | R, \boldsymbol{y}_L^{[j]}\} \approx \mathcal{D}\{\boldsymbol{x}^{[j]} | \boldsymbol{y}^{[j]} \leftarrow \mathcal{D}\{\boldsymbol{y}^{[j]} | \boldsymbol{y}_L^{[j]}\}\} \approx \mathcal{D}\{\boldsymbol{x}^{[j]} | \boldsymbol{y}_L^{[j]}\} \approx D^{[j]}$. $\square$

Completeness. Step 8 of the amortized protocol recovers $\boldsymbol{y}$ in each execution:

$$\begin{aligned} \boldsymbol{y} = h(\boldsymbol{\beta} \odot \boldsymbol{x}) &= h(\boldsymbol{\beta} \odot \boldsymbol{x} - \boldsymbol{\alpha} \odot \boldsymbol{z} + \boldsymbol{\alpha} \odot \boldsymbol{z}) \\ &= h(\boldsymbol{\beta} \odot \boldsymbol{x} - \boldsymbol{\alpha} \odot \boldsymbol{z}) + h(\boldsymbol{\alpha} \odot \boldsymbol{z}) = h(\boldsymbol{w}) + h(\boldsymbol{\alpha} \odot \boldsymbol{z}) \end{aligned} \tag{4}$$

where the third equality follows from Equation 1.

Soundness. Since, for $e \in [v]$, any one of $\boldsymbol{\alpha} \odot \boldsymbol{z}, \boldsymbol{\beta} \odot \boldsymbol{x}^{[e]}$ determines the other (given $\boldsymbol{w}^{[e]}$) by step 6 of the amortized protocol, cheating about knowing one or both of $\boldsymbol{z}, \{\boldsymbol{x}^{[e]}\}_{e=1}^v$ is equivalent. The soundness error is 2^{-u}, as for one secret.

Knowledge. The challenge in the amortized protocol is equal to the challenge in the basic protocol it invokes. Therefore, as proven for the basic protocol in Appendix B, a knowledge extractor is able to extract $\boldsymbol{z}$ given responses to two different challenges, each from a domain of size 2^u. Hence, the knowledge error

for $\boldsymbol{z}$ is at most 2^{-u}. The extractor will also be able to extract each of $\{\boldsymbol{x}^{[e]}\}_{i=1}^{v}$ since these are determined by their revealed shares (appearing in the protocol transcript) together with $\boldsymbol{z}$. Therefore, the extractor is able to extract the (non pre-processed) secrets $\{\mathcal{T}^{-1}(\boldsymbol{x}^{[e]})\}_{i=1}^{v}$ with knowledge error at most 2^{-u} as well.
Witness indistinguishability. Lemma 1 implies that except with negligible probability $\boldsymbol{y}$ has multiple witnesses that are statistically indistinguishable and that the same is true for each $\{\boldsymbol{y}^{[e]}\}_{e=1}^{v}$. It remains to show that the pre-processing with $\mathcal{T}$ preserves SWI. By Theorem 5, $\boldsymbol{z}$ in the amortized protocol is statistically hidden with soundness error 2^{-u}. For $e \in [v]$, Lemmata 11 and 3 imply that $\boldsymbol{x}^{[e]}$ and its unrevealed share are statistically hidden. Step 6 of the amortized protocol is the only one exposing information: for $e \in [v]$, a difference $\boldsymbol{w}^{[e]} \in \times_{j=1}^{m'}\{0, -\boldsymbol{\alpha}_j\}$ between $\boldsymbol{\beta} \odot \boldsymbol{x}^{[e]}$ and $\boldsymbol{\alpha} \odot \boldsymbol{z}$ is exposed. Consequently, $\boldsymbol{x}_j^{[e_1]} \oplus \boldsymbol{x}_j^{[e_2]}$ for each $e_1, e_2 \in [v], j \in [m']$ are exposed. Nevertheless, Lemma 12 with R determined by $(\boldsymbol{y}^{[i_1]} \oplus \boldsymbol{y}^{[i_2]})_{i_1,i_2\in[k]}$ (having one degree of freedom) guarantees an attacker learns no useful information on $\{\mathcal{T}^{-1}(\boldsymbol{x}^{[e]})\}_{e=1}^{v}$. Since, for $e \in [v]$, any one of $\boldsymbol{\alpha} \odot \boldsymbol{z}, \boldsymbol{\beta} \odot \boldsymbol{x}^{[e]}$ determines the other (with $\boldsymbol{w}^{[e]}$ known), an attacker can learn any information on each of $\{\boldsymbol{x}^{[e]}\}_{e=1}^{v}$ only by learning all except $\leq l$ bits of any one of $\{\mathcal{T}^{-1}(\boldsymbol{x}^{[e]})\}_{e=1}^{v}$. Thus, SWI is preserved.
Witness hiding. The process for random drawing of $\boldsymbol{r}$ satisfies the conditions of a generator[2] with uniform distribution of inputs for the function $\mathcal{T}(\text{hcom}(\boldsymbol{x}', \cdot))$, for any choice of $\boldsymbol{x}'$, that in turn satisfies the conditions of a proper claw-free function[3]. Hence, a WI-to-WH theorem[4] applies, proving the protocol is SWH.
Slack. Due to Lemma 11, if Alice is honest about $\boldsymbol{z}$ then Alice is also honest about $\boldsymbol{x}^{[e]}$. Hence, to get a non-zero slack for $\boldsymbol{x}^{[e]}$ for any $e \in [v]$, Alice must cheat about $\boldsymbol{z}$. For this case Appendix B shows that $\boldsymbol{z}_{j'} = 2$ at each slack index j'. As steps 7 and 8 of the amortized protocol imply that $\boldsymbol{w}_{j'}^{[e]} \in \{0, -\boldsymbol{\alpha}_{j'}\}$ and $\boldsymbol{w}_{j'}^{[e]} + \boldsymbol{\alpha}_{j'}\boldsymbol{z}_{j'} = \boldsymbol{\beta}_{j'}\boldsymbol{x}_{j'}^{[e]} \pmod p$ at each slack index j', Alice may only cheat with $\boldsymbol{w}_{j'}^{[e]} = 0$ so that $\boldsymbol{z}_{j'} := \boldsymbol{x}_{j'}^{[e]} = 2$ (and $\boldsymbol{\beta}_{j'} := \boldsymbol{\alpha}_{j'}$). This yields an absolute slack $\xi = 1$ at any slack index j' of Alice's choice. Hence, using a similar argument as in Appendix B, the worst-case slack error is $1 - (1 - 2^{-u})^{m'}$ for $\xi = 1$ and Bob can efficiently tell when $\xi = 0$. Note that since from an honest prover one expects $\boldsymbol{w}_j^{[e]} \sim \{0, -\boldsymbol{\alpha}_j\}$ for $j \in [m']$, one may account (similarly to Appendix B) for observations on $\{\boldsymbol{w}^{[e]}\}_{e=1}^{v}$ in analyzing the slack error; the details are omitted.

[2] A definition is found in [28]. G is a generator for relation R if on input 1^n it produces instances $(x, w) \in R$ of length n.

[3] Informal description is found in [28]; For a claw-free function it is intractable in non-uniform polynomial time to find a claw: two arguments which map to the same image. A claw-free function is proper if any image has at least two pre-images.

[4] Theorem 4.1 in [28]: Let G be a generator for a proper claw free function f, which generates pairs (x, w) where $x = f(w)$, with uniform distribution over the arguments w. Let (P, V) be a proof of knowledge system for proving knowledge of a pre-image of x. Then if (P, V) is WI over f, then it is WH over (f, G).