Updateable Inner Product Argument with Logarithmic Verifier and Applications

Vanesa Daza[1,2], Carla Ràfols[1,2], and Alexandros Zacharakis[1]

[1] Pompeu Fabra University
{vanesa.daza,carla.rafols,alexandros.zacharakis}@upf.edu
[2] Cybercat

Abstract. We propose an improvement for the inner product argument of Bootle et al. (EUROCRYPT'16). The new argument replaces the unstructured common reference string (the commitment key) by a structured one. We give two instantiations of this argument, for two different distributions of the CRS. In the designated verifier setting, this structure can be used to reduce verification from linear to logarithmic in the circuit size. The argument can be compiled to the publicly verifiable setting in asymmetric bilinear groups. The new common reference string can easily be updateable. The argument can be directly used to improve verification of Bulletproofs range proofs (IEEE SP'18). On the other hand, to use the improved argument to prove circuit satisfiability with logarithmic verification, we adapt recent techniques from Sonic (ACM CCS'19) to work with the new common reference string. The resulting argument is secure under standard assumptions (in the Random Oracle Model), in contrast with Sonic and recent works that improve its efficiency (Plonk, Marlin, AuroraLight), which, apart from the Random Oracle Model, need either the Algebraic Group Model or Knowledge Type assumptions.

Keywords: Zero Knowledge · Inner Product · SNARKS · Range Proofs · Updateable

1 Introduction

Zero-Knowledge proofs have been an important primitive in the theory of cryptography since their introduction three decades ago. The classical applications of zero-knowledge proofs are numerous, including for example identification schemes, electronic voting, verifiable outsourced computation, or CCA secure public-key encryption. The common denominator of all of these is that zk-proofs are used to prove simple statements, like "this ciphertext is well-formed" or "I know a valid signature key". Although it was known that every NP statement could be proved in zero-knowledge [23], the cost of such general proofs was prohibitive and more sophisticated applications of zk-proofs were completely impractical.

This situation has changed radically in the last few years with the introduction of pairing-based zk-SNARKs [25]. The key element of these arguments

is that they are *succinct*, in fact, they are constant size, i.e. independent of the witness size and thus, very fast to verify. This is extremely powerful: in particular, a prover can show that it has executed correctly some large computation (expressed as a huge circuit) and a verifier will be convinced after doing only very few checks (e.g. computing 3 pairings in [26]). Besides their scientific interest, SNARKs have opened the door to new real-world privacy-preserving applications. Cryptocurrencies like Zcash [6] or Ethereum [36] are two of the most popular examples so far.

However, even the most efficient instantiations of pairing-based SNARKs [26,28] have a few drawbacks. On the efficiency side, the main ones are long common reference string and costly prover computation. On the security side, they are based on very strong hardness assumptions, and the setup is assumed to be trusted.

Recently, there are significant research efforts to propose alternatives which overcome some of these drawbacks following several dimensions. For instance, numerous works study how to reduce the trust in the common reference string, exploring weaker models such as subversion resistant SNARKs ([4,1,17]), updateable common reference strings ([27]) or transparent setup ([5]). Although SNARKs are unbeatable in some facets, different tradeoffs are compelling depending on the application scenario.

One of the most celebrated alternatives to SNARKs are the arguments of knowledge for Arithmetic Circuit Satisfiability of Bootle et al. [10] (and Bulletproofs, the improvement thereof by Bünz et al. [12]). Their dependence on weaker assumptions (the DLOG assumption and the Random Oracle if one wants to remove interaction via Fiat-Shamir), the absence of a trusted setup and the logarithmic size of the proofs are some of its most attractive features. Unfortunately, verification time scales linearly, even when batching techniques are used. The motivation of this paper is to improve the cost of the verifier in the aforementioned works, while keeping most of its advantages.

1.1 Related Work

In [10], Bootle et al. proposed an interactive zero-knowledge argument at the heart of which lies a recursive argument for an inner product relation of committed values. The argument has very interesting properties, most notably it is transparent. The communication complexity is $\mathcal{O}_\lambda(\log |\mathcal{C}|)$[3] and the verification cost is $(\mathcal{O}_\lambda(|\mathcal{C}|))$ which is the main drawback of the scheme, since verifying is asymptotically as costly as evaluating the circuit. Prover complexity is asymptotically optimal $(\mathcal{O}_\lambda(|\mathcal{C}|))$ but it heavily uses expensive public-key operations. Bünz et al. in [12] improved the concrete efficiency of the aforementioned protocol by a constant factor.

The Muggle-proofs based [39,40,37,38] proof systems build on the delegation scheme of Goldwasser, Kalai, and Rothblum [24]. These are very efficient schemes

[3] As explained in Section 2, $\mathcal{O}_\lambda(\cdot)$ hides linear factors that depend on the security parameter λ

for low depth computation, whose verification and communication complexity depend on $d \log W$, where d is the circuit depth, and W its width plus some communication overhead depending on the specific instantiation. Hyrax [37] is a DLOG-based transparent instantiation with an additional cost of $\mathcal{O}_\lambda(|w|^{\frac{1}{i}})$ for some i that can be fine-tuned. Recently, Libra [38] utilized and improved techniques from [14] to achieve an asymptotically optimal prover complexity and minimize public key operations. All these schemes need either a per-circuit setup or work for log-space uniform computations. Since they are inherently interactive they rely on the Fiat-Shamir transform to yield non interactive arguments.

Probabilistically Checkable Proofs (PCP) based constructions [5,7] originate from the works of Kilian [31] and Micali [33], and are based on Interactive Oracle Proofs [8] which generalize the classical PCP proofs in the interactive setting. They are based on symmetric primitives which results in transparent, plausibly post-quantum secure constructions. The main drawback is that they are still concretely inefficient, especially as far as prover complexity is concerned. In the same family, [22,30,2] build on the MPC-in-the-head paradigm [29] and share similar properties. The most efficient one is Ligero [2] which, while having good concrete efficiency, has communication complexity $\mathcal{O}_\lambda(\sqrt{|\mathcal{C}|})$ which can be bad for moderately large computations.

The line of work of Linear PCP constructions [21,35,16,26] that originates from the seminal work of Gennaro et al. [21] and abstracted in [9], are the most efficient when considering verification time and communication. Their proof size is constant and the verification cost is $\mathcal{O}_\lambda(|x|)$ where x is the public input. Note that this is optimal since the verifier has to, at least, read the statement to be proven. The main drawback is that they need a trusted setup.

To achieve a middle ground between efficiency and trust, Groth et al. [27] defined the Updateable model. In this model, everyone can non-interactively update the setup parameters. As long as one update is honest, soundness is guaranteed. The authors also presented a scheme which is updateable, but it has a universal common reference string of size quadratic in the maximal size of all supported circuits (although from the global setup a linear, circuit-specific string can be derived). Maller et al. presented Sonic [32], which improved this to a linear CRS by exploiting the reduction of [10]. Several works [20,15,19] have tried to improve the efficiency of Sonic concretely. However, all of these, including Sonic, are secure either in the Algebraic Group Model, or under knowledge type assumptions (apart from the Random Oracle Model). Recently, [13] uses the techniques of the aforementioned results to construct a SNARK sound in groups of unknown order. When instantiated in class groups it achieves a transparent setup and asymptotically improves over STARKS [5] by a logarithmic factor.

1.2 Our Contribution

We construct a public-coin Argument of Knowledge in the Universal Updateable Model based on the work of Bootle et al. [10]. The verification complexity is $\mathcal{O}_\lambda(|x| + \log|\mathcal{C}|)$ and communication complexity is $\mathcal{O}_\lambda(\log|\mathcal{C}|)$ where $|x|$ is the

public input size. The prover is linear in $|\mathcal{C}|$ but, as in [10], it needs to perform a lot of public-key operations. The two constructions are secure, respectively, under one assumption which reduces to asymmetric DLOG and another one to asymmetric q-DLOG. They can be made non-interactive with the Fiat-Shamir heuristic. Updating and verifying updates need time $\mathcal{O}_\lambda(|\mathcal{C}|)$, and communication complexity is $\mathcal{O}_\lambda(\log|\mathcal{C}|)$ (which can be reduced to $\mathcal{O}_\lambda(\log\log|\mathcal{C}|)$) and $\mathcal{O}_\lambda(1)$, respectively.

As far as we know, all recently proposed efficient and fully-succinct updateable schemes [32,20,15] rely on the Algebraic Group Model [18] or other Knowledge Type assumptions apart from the Random Oracle Model, while in our case the Random Oracle Model and a standard assumption is enough. However, the aforementioned schemes have a better communication complexity ($\mathcal{O}_\lambda(1)$) and, while asymptotically the verifier has the same complexity ($\mathcal{O}_\lambda(\log|\mathcal{C}|)$), in their case it works mainly on the field while ours works in the group, which is less efficient. Also, while the prover complexity in [32,20,15] is quasi-linear in $|\mathcal{C}|$ and ours is linear, theirs works mainly in the field. We report some concrete numbers in Table 1 for the overhead of each scheme (we do not include concrete numbers for other schemes in communication and verification since they are constant while ours are logarithmic in $|\mathcal{C}|$).

Finally, we observe that the major overhead in the general proof system is the delegation of (public) computation regarding the circuit structure and so, for fixed languages that may be of interest, we can use the same techniques to achieve better efficiency. We demonstrate that by applying this in range proofs improving on [12]. The main overhead compared to it is that we move to bilinear groups instead of standard ones, but we exponentially reduce the verification complexity.

1.3 Our Techniques

Distribution Parameterized Vector Commitments. We revisit the use of vector commitment schemes in zero-knowledge proof systems when working in groups: instead of using the classical Pedersen commitment key which is uniformly sampled, we add some limited structure which simultaneously allows more efficient representation of the key and efficient updateability. When combined with the properties of bilinear groups, only a compressed version of it is enough to allow a verifier to perform verification tasks exponentially faster.

In particular we propose two instantiations:

- The commitment key consisting of group encodings of all monomials of a secret x, i.e., $[1], [x], [x^2], \ldots, [x^{n-1}]$.
- The commitment key consisting of group encodings of all multilinear monomials monomials of a secret $x_1, \ldots, x_\nu$ i.e. $[1], [x_1], [x_2], [x_1x_2], \ldots, [x_1x_2\cdots x_\nu]$.

The structure of both commitment keys allows to non-interactively update the parameters and thus nullifying the trapdoors x or $x_1, \ldots, x_n$. We take advantage of this structure in bilinear groups to create compressed versions of these

	|CRS|	$\mathcal{P}$	$\mathcal{V}$	π	Assumptions
Sonic [32]	$36n$ $\mathbb{G}_1$	$273n$ E_1	$\mathcal{O}_\lambda(1)$	$\mathcal{O}_\lambda(1)$	AGM
Marlin [15]	$6m$ $\mathbb{G}_1$	$21m$ E_1	$\mathcal{O}_\lambda(1)$	$\mathcal{O}_\lambda(1)$	AGM or KT
Plonk [19]	$n+a$ $\mathbb{G}_1$	$9(n+a)$ E_1	$\mathcal{O}_\lambda(1)$	$\mathcal{O}_\lambda(1)$	AGM
This work	$n'\mathbb{G}_1(\mathcal{P})$ $\log n'\mathbb{G}_2(\mathcal{V})$	$(22+10M)n'$ E_1	$12\log n'$ E_1 $8\log n'$ P	$12\log n'$ $\mathbb{G}_1$ $4\log n'$ $\mathbb{F}$	A-DLOG or q-A-DLOG

Table 1. Comparison of the updateable SNARKSs in terms of the most expensive operations (exponentiations and pairings). n is the number of multiplication gates, a is the number of addition gates, m is the number of wires in the circuit and M is a parameter, which determines the processed circuit's fan-in and fan-out upper bound, and can be fine-tuned to balance the computations of the prover and verifier. n' is the size of the processed circuit which in the worst case is upper bounded by $n + \frac{2m}{M-1}$. Sonic empirically assumes $n' = 3n$ for $M = 3$ in its reported numbers rather than a worst case analysis. P refers to pairing operations and E_1 to $\mathbb{G}_1$ exponentiations. We omit constant factors. Our prover is essentially only performing multi-exponentiations and we consider we need k $\mathbb{G}_1$ exponentiations to do a k-multi-exponentiation, but we note that they can be implemented with $o(k)$ exponentiations, see e.g. [10]. In the assumptions columns KT refers to Knowledge Type assumptions, AGM to the Algebraic Group Model and A-DLOG, q-A-DLOG to variants of DLOG and q-DLOG in the asymmetric group setting. All schemes are interactive and can be turned to non-interactive in the Random Oracle model.

keys of size only $\log n$. For various languages, this allows the verifier to verify statements with the help of the prover without reading the whole commitment key. This leads to exponentially faster verification of proofs with minimal overhead for the prover, at the price of moving to bilinear instead of plain DLOG groups.

Inner Program Argument with Logarithmic Verifier. Using these techniques, we modify the inner product protocol of Bootle et al. [10] for proving that for given commitments $c_1 = \mathsf{Com}(\mathbf{a})$, $c_2 = \mathsf{Com}(\mathbf{b})$ and $z \in \mathbb{F}$, it holds that $\mathbf{a}^\top \mathbf{b} = z$. More specifically, we note that the overhead of the verifier in [10] is computing a new commitment key in each of the $\log n$ rounds of the protocol, where n is the vector dimension. This key depends on the previous key and the verifiers' challenges. Instead of doing that, we only give the verifier the compressed key (which is logarithmic in n) and have the prover convince the verifier that the reduced statement is w.r.t. a new key which is the correct one.

Universally Updateable NIZK AoK. Having this powerful tool allows us to aggregate linear and quadratic constraints and thus prove general statements. We follow the techniques of [10] to reduce a statement about a circuit w.r.t. a public input to an inner product one (which need not be zero knowledge) and we can then use the improved inner product argument. More concretely, the prover

convinces the verifier that $[\alpha], [\beta]$ are commitments to $\mathbf{a}, \mathbf{b}$ such that $\mathbf{a}^\top \mathbf{b} = z$. The former vector depends on the witness and the latter on the circuit structure, is public, and both depend on a random challenge issued by the verifier.

However, computing $[\beta]$ given universal parameters that work for any circuit (of bounded size) requires $\mathcal{O}_\lambda(|\mathcal{C}|)$ time making verification linear in the computation size. To overcome this, we delegate this computation to the prover who gives a succinct proof for the correct computation of $[\beta]$. To achieve that, we assume a specific structure for the circuit (basically that the gates have bounded fan-in and fan-out) and apply techniques similar to [32] adapted to our setting. These conditions can be imposed by pre-processing the circuit appropriately without asymptotically increasing the circuit size.

We note that when we have a fixed statement, we can make things much more efficient. The blueprint of the construction remains the same and we can appropriately fine-tune the parameter generation to avoid the delegation of computation of $[\beta]$ thus achieving a concretely more efficient verifier. We show how this can be applied in Range Proofs, and reduce exponentially the verification complexity of the similar construction of [12].

2 Preliminaries

2.1 Notations

We write $x \leftarrow S$ to denote uniformly sampling from S and assigning to x. When A is an algorithm we denote with $y \leftarrow A(x)$ the assignment of the output of A with input x to y, where we uniformly sample randomness from A if it is probabilistic. We write $A(x; r)$ to explicitly refer to the randomness of A when needed. We notate with $\mathcal{O}_\lambda(\cdot)$ asymptotic complexity that hides linear factors that depend on the security parameter λ.

We denote vectors with boldface letters. If $\mathbf{v}$ is a vector, we denote with normal font its components, that is v_i is its i-th component. We denote $\mathbf{e_n} \in \mathbb{F}^n$ the n-th element of the canonical basis. The symbol $\circ$ is used for denoting pairwise product, that is $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, \ldots, a_n \cdot b_n)$.

Groups are written in additive notation and its elements are written implicitly: if we fix a generator $g \in \mathbb{G}$, we denote with $[r]$ the group element rg. We extend this notation to vectors of group elements by denoting $[\mathbf{r}] = ([r_1], \ldots, [r_n])$. In the bilinear group setting, given some fixed generators $g_1, g_2, g_T = e(g_1, g_2)$, we use subscripts to specify the group. In this notation, $e([r]_1, [s]_2) = [rs]_T$.

Let $\mathbb{G}$ be a group of order q and $\mathbf{r} = (r_1, \ldots, r_n) \in \mathbb{Z}_q^n, \mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_q^n$. We denote $[\mathbf{a}^\top \mathbf{r}] = \sum_{i=1}^n a_i [r_i]$, that is, $[\mathbf{a}^\top \mathbf{r}]$ is a Vector Pedersen commitment of $\mathbf{a}$ w.r.t. to commitment key $[\mathbf{r}]$. Given a vector $\mathbf{r} = (r_1, \ldots, r_n)$, for even n, we denote $\mathbf{r}_{\frac{1}{2}} = (r_1, \ldots, r_{n/2})$ and $\mathbf{r}_{\frac{2}{2}} = (r_{n/2+1}, \ldots, r_n)$. We denote $\mathbf{x^n} = (1, x, \ldots, x^{n-1})$. Finally, let $x_1, \ldots, x_\nu \in \mathbb{Z}_q^n$. We denote as $\overline{\mathbf{x}}$ the vector that is constructed recursively by setting $\overline{\mathbf{x}} \leftarrow (1)$, $\{\overline{\mathbf{x}} \leftarrow (\overline{\mathbf{x}}, x_i \overline{\mathbf{x}})\}_{i \in [\nu]}$. Basically, $\mathbf{x^n}$ contains all the monomials of x up to degree $n-1$, and $\overline{\mathbf{x}}$ contains all the multilinear monomials where a "canonical" ordering has been imposed by its recursive definition.

2.2 (Zero Knowledge) Arguments

Interactive (Zero Knowledge) Arguments of Knowledge. We present the definitions and the relevant results we need for (Zero Knowledge) Arguments of Knowledge (ZKAoK). We follow the presentation of [10].

Let $\mathcal{L} \in \mathbf{NP}$ be a language and $\mathcal{R}_{\mathcal{L}}$ the corresponding relation for $\mathcal{L}$. A ZKAoK allows a prover to convince a verifier of knowledge of a witness w certifying membership of a public x in $\mathcal{L}$ that is $(x, w) \in \mathcal{R}_{\mathcal{L}}$. The zero knowledge property guarantees that the verifier learns nothing about the witness w apart from the fact that the prover knows such a witness.

Our final goal is a non-interactive argument, but we work in the interactive setting and then use standard techniques for transforming the interactive arguments to non-interactive.

Denote with $\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$ the transcript of an execution of $\mathcal{P}$ and $\mathcal{V}$ with respective inputs x, w and x. Let $\mathsf{view}_{\mathcal{V}}\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$ ($\mathsf{view}_{\mathcal{P}}\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$) be the views of $\mathcal{V}$ ($\mathcal{P}$) in a protocol execution (i.e. the input, randomness and all incoming messages), and finally let $\mathsf{out}_{\mathcal{V}}\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$ be the final verdict of the verifier (accept or reject).

Definition 1. *The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ is a Zero Knowledge Argument of Knowledge if it is public coin, it has perfect completeness, statistical witness extended emulation and perfect honest verifier zero Knowledge as defined next.*

Definition 2. *The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ has Perfect Completeness if for all $(x, w) \in \mathcal{R}_{\mathcal{L}}$ it holds that* $\Pr\left[\mathsf{out}_{\mathcal{V}}\langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle = 1\right] = 1$.

Definition 3. *The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ has Statistical Witness Extended Emulation if for all deterministic polynomial $\mathcal{P}^*$, there exists an expected polynomial time extractor $\mathcal{E}$, such that for all (unbounded) adversaries $\mathcal{A}$*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; \begin{array}{r} (x, s) \leftarrow \mathcal{A}(1^\lambda) \wedge \\ tr \leftarrow \langle \mathcal{P}^*(x, s), \mathcal{V}(x) \rangle \end{array} \right] - \Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; \begin{array}{r} (x, s) \leftarrow \mathcal{A}(1^\lambda) \wedge \\ (tr, w) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(x,s), \mathcal{V}(x) \rangle}(u) \wedge \\ \textit{if } tr \textit{ is accepting then } (x, w) \in \mathcal{R}_{\mathcal{L}} \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

Definition 4. *An $(n_1, \ldots, n_\mu)$-tree of accepting transcripts for the pair $\langle \mathcal{P}, \mathcal{V} \rangle$ with $2\mu + 1$ rounds is a tree where:*

- *Each node of the tree in level i is labeled with the transcript of the protocol used up to $\mathcal{V}$'s i-th message.*
- *Each node in the same level i is labeled with a transcript that uses fresh (uniformly distributed and independent) randomness for the verifier's i-th challenges.*
- *Level i has n_i descendants.*

– *The leafs are labeled with transcripts that are accepted by the verifier.*

Definition 5. *The pair $\langle \mathcal{P}, \mathcal{V} \rangle$ has $(n_1, \ldots, n_\mu)$-generalized special soundness if there exists a PPT extractor $\mathcal{E}$ such that given an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts for the pair $\langle \mathcal{P}, \mathcal{V} \rangle$, the extractor $\mathcal{E}$ outputs a valid witness for the statement.*

Definition 6. *An interactive proof system $\langle \mathcal{P}, \mathcal{V} \rangle$ is public coin if all messages from $\mathcal{V}$ to $\mathcal{P}$ are independent and uniformly distributed, and are uniquely defined by the randomness of the verifier alone.*

Definition 7. *A public coin interactive proof system $\langle \mathcal{P}, \mathcal{V} \rangle$ is perfect Honest Verifier Zero Knowledge (HVZK) if there exists a PPT simulator $\mathcal{S}$, such that for all PPT $\mathcal{A}$, it holds that*

$$\Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; (x, w, r) \leftarrow \mathcal{A}(1^\lambda) \wedge tr \leftarrow \langle \mathcal{P}^*(x, w), \mathcal{V}(x; r) \rangle \wedge (x, w) \in \mathcal{R}_\mathcal{L}\right] =$$
$$\Pr\left[1 \leftarrow \mathcal{A}(tr) \;\middle|\; (x, w, r) \leftarrow \mathcal{A}(1^\lambda) \wedge tr \leftarrow \mathcal{S}(x, r) \wedge (x, w) \in \mathcal{R}_\mathcal{L}\right].$$

Theorem 1. *Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be a $2\mu + 1$ round, public coin, interactive proof system with $(n_1, \ldots, n_\mu)$-generalized special soundness and $\prod_{i=1}^{\mu} n_i = \mathcal{O}(\lambda^c)$ for a constant c. Then $\langle \mathcal{P}, \mathcal{V} \rangle$ has witness extended emulation.*

The proof of the theorem is given in [10].

Updateable Non-Interactive (Zero Knowledge) Arguments of Knowledge. Informally, a non-interactive argument system in the common reference string model is a ZK argument with two rounds where the first is a setup round to create parameters that can be reused in many proofs. The most efficient constructions for general NP statements (e.g. Groth [26]) need a very expensive and inefficient trusted setup. To deal with this, Groth et al. [27] introduced the notion of an Updateable Setup where users can non-interactively update the parameters in a way that gives us the following guarantee: if an honest update takes place, then no PPT adversary can break soundness. We follow the model of Groth et al. [27], who show that for updateability it suffices to prove that an argument is secure in the following model.

– The adversary creates setup parameters.
– An honest update on these parameters takes place.
– The adversary updates the parameters.
– Circuit specific parameters are derived publicly for a circuit $\mathcal{C}$.
– Knowledge soundness is challenged w.r.t. these parameters.

We emphasize that the circuit-specific setup is done publicly: no secret is involved in it. Anyone can take the universal parameters, and deterministically compute the circuit-specific CRS. We present the definition of Updateable Non-Interactive (Zero Knowledge) Arguments of Knowledge.

Definition 8. *An Updateable Non-Interactive (Zero Knowledge) Argument of Knowledge is a tuple of algorithms* (USetup, Update, VrfySetup, VrfyUpdate, CircuitSetup, Prove, Vrfy) *where*

- $\sigma \leftarrow \mathsf{USetup}(1^\lambda, n)$*:* USetup *takes as input the security parameter* λ *and an upper bound on the derived circuit size* n*, and outputs a universal CRS* σ*.*
- $(\sigma', \pi_{\sigma'}) \leftarrow \mathsf{Update}(\sigma)$*:* Update *takes as input a universal CRS* σ*, and produces a new universal CRS* σ' *along with a proof of correct update* $\pi_{\sigma'}$*.*
- $0/1 \leftarrow \mathsf{VrfySetup}(\sigma, 1^\lambda, n)$*:* VrfySetup *takes as input a universal CRS* σ*, the security parameter* λ *and* n *and outputs a bit indicating the correctness of the structure of the universal CRS.*
- $0/1 \leftarrow \mathsf{VrfyUpdate}(\sigma', \sigma, \pi_{\sigma'})$*:* VrfyUpdate *takes as input the new and old CRS* σ' *and* σ*, and a proof* π'_σ*, and outputs a bit indicating the correctness of the update.*
- $\sigma_\mathcal{C} \leftarrow \mathsf{CircuitSetup}(\sigma, \mathcal{C})$*: is a deterministic algorithm that takes as input the description of a circuit with size bounded by* n*, and the universal CRS and outputs circuit specific parameters* $\sigma_\mathcal{C}$*.*
- $\pi \leftarrow \mathsf{Prove}(\sigma_\mathcal{C}, x, w)$*: takes as input the CRS* $\sigma_\mathcal{C}$*, the public and private input* x, w*, and outputs a proof* π*.*
- $0/1 \leftarrow \mathsf{Vrfy}(\sigma_\mathcal{C}, x, \pi)$*: takes as input the CRS* $\sigma_\mathcal{C}$*, the public input* x *and a proof* π*, and outputs a proof indicating its validity.*

which is Perfectly Complete, Knowledge Sound and Statistically Zero Knowledge as defined next.

Definition 9. *An Updateable Non-Interactive Argument of Knowledge is Perfectly Complete if for all* λ, n

$$\Pr\left[\mathsf{VrfySetup}(\sigma, 1^\lambda, n) = 1 \;\middle|\; \sigma \leftarrow \mathsf{USetup}(1^\lambda, n)\right] = 1,$$

for all λ, n, σ

$$\Pr\left[\begin{matrix}\mathsf{VrfySetup}(\sigma', 1^\lambda, n) = 1 \wedge \\ \mathsf{VrfyUpdate}(\sigma, \sigma', \pi_{\sigma'}) = 1\end{matrix} \;\middle|\; \begin{matrix}\mathsf{VrfySetup}(\sigma, 1^\lambda, n) = 1 \wedge \\ (\sigma', \pi_{\sigma'}) \leftarrow \mathsf{Update}(\sigma)\end{matrix}\right] = 1$$

and for all $\lambda, n, \sigma, \mathcal{C}, x, w$ *where* $\mathcal{C}$ *encodes a circuit of size bounded by* n *and* $\mathcal{R}_\mathcal{C}(x, w) = 1$

$$\Pr\left[\mathsf{Vrfy}(\sigma_\mathcal{C}, x, \pi) = 1 \;\middle|\; \begin{matrix}\mathsf{VrfySetup}(\sigma, 1^\lambda, n) = 1 \wedge \\ \sigma_\mathcal{C} \leftarrow \mathsf{CircuitSetup}(\sigma, \mathcal{C}) \wedge \\ \pi \leftarrow \mathsf{Prove}(\sigma_\mathcal{C}, x, w)\end{matrix}\right] = 1.$$

Definition 10. *An Updateable Non-Interactive Argument of Knowledge is Knowledge Sound if for all stateful PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists an extractor* $\mathcal{E}_\mathcal{A}$*, such that for all* λ*,* n*,* $\mathcal{C}$ *where* $\mathcal{C}$ *is a circuit of size bounded by* n

$$\Pr\left[\begin{matrix}\\ \mathsf{VrfySetup}(\sigma_1, 1^\lambda, n) = 1 \wedge \\ \mathsf{VrfyUpdate}(\sigma_3, \sigma_2, \pi_{\sigma_3}) = 1 \wedge \\ \mathsf{Vrfy}(\sigma_\mathcal{C}, x, \pi) = 1 \wedge \\ \mathcal{C}(x, w) \neq 1 \\ \\ \end{matrix} \;\middle|\; \begin{matrix}(\sigma_1, st_1) \leftarrow \mathcal{A}_1(1^\lambda, n) \wedge \\ (\sigma_2, \pi_{\sigma_2}) \leftarrow \mathsf{Update}(\sigma_1) \wedge \\ (\sigma_3, \pi_{\sigma_3}, st_2) \leftarrow \mathcal{A}_2(st_1, \sigma_2, \pi_{\sigma_2}) \wedge \\ \sigma_\mathcal{C} \leftarrow \mathsf{CircuitSetup}(\sigma_3, \mathcal{C}) \wedge \\ (x, \pi) \leftarrow \mathcal{A}_3(st_2, \sigma_\mathcal{C}; r) \wedge \\ w \leftarrow \mathcal{E}(\sigma_\mathcal{C}, x; r)\end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Definition 11. *An Updateable Non-Interactive Arguments of Knowledge is Statistically Zero knowledge in the Random Oracle model if there exists a pair of PPT algorithms $\mathcal{S}_1, \mathcal{S}_2$, where $\mathcal{S}_2$ is stateful, such that for all $\mathcal{A}$, and for all circuits $\mathcal{C}$ of size bounded by n, where $\mathcal{C}$ takes as input a public value x and a private value w then*

$$\Pr\left[\, b' = b \;\middle|\; \begin{array}{r} b \leftarrow \{0,1\} \;\wedge \\ \sigma \leftarrow \mathcal{A}^{H_b}(\mathsf{setup}, 1^\lambda, n) \wedge \\ \mathsf{VrfySetup}(\sigma, 1^\lambda, n) = 1 \wedge \\ \sigma_{\mathcal{C}} \leftarrow \mathsf{CircuitSetup}(\sigma, \mathcal{C}) \wedge \\ b' \leftarrow \mathcal{A}^{H_b, O_b}(\sigma_{\mathcal{C}}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where H is modeled as a Random Oracle and

$$O_0(x,w) \leftarrow \begin{cases} \perp, & \text{if } \mathcal{R}_{\mathcal{C}}(x,w) = 0 \\ \mathsf{Prove}(\sigma_{\mathcal{C}}, x, w), & \text{otherwise} \end{cases}, \qquad H_0(m) \leftarrow H(m),$$

$$O_1(x,w) \leftarrow \begin{cases} \perp, & \text{if } \mathcal{R}_{\mathcal{C}}(x,w) = 0 \\ \mathcal{S}_1(\sigma_{\mathcal{C}}, x), & \text{otherwise} \end{cases}, \qquad H_1(m) \leftarrow \mathcal{S}_2(m).$$

Note that this definition considers adversarially created parameters, i.e. Subversion Resistant ZK [4].

From HVZK Interactive AoK to Non Interactive ZK AoK. It is well-known that we can use the Fiat-Shamir heuristic to transform any public coin Perfect HVZK interactive argument to a non-interactive full-fledged Statistical Zero Knowledge argument in the Random Oracle Model.

2.3 Updateable Commitment Schemes

We define commitment schemes which have an updateability property as well. We do this to simplify proofs in the following sections. An updateable commitment will be enough to guarantee updateability of all the protocols in this work, since all the arguments presented hold regardless of parameters *unless* there is a breach in the binding property of the commitment scheme.

Definition 12. *An Updateable Commitment Scheme is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{VrfySetup}, \mathsf{Update}, \mathsf{VrfyUpdate}, \mathsf{Com}, \mathsf{Open})$ *such that*

- $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n)$ *takes as input the security parameter λ and the vector dimension n, and outputs a commitment key* ck.
- $(\mathsf{ck}', \pi_{\mathsf{ck}'}) \leftarrow \mathsf{Update}(\mathsf{ck})$*:* Update *takes as input a commitment key* ck *and produces a new commitment key* ck' *and a proof of correct update* $\pi_{\mathsf{ck}'}$.
- $0/1 \leftarrow \mathsf{VrfySetup}(\mathsf{ck}, 1^\lambda, n)$*:* $\mathsf{VrfySetup}$ *takes as input a commitment key* ck*, the security parameter λ and the dimension n, and outputs a bit indicating the correctness of the structure of the key.*

– $0/1 \leftarrow \mathsf{VrfyUpdate}(\mathsf{ck}', \mathsf{ck}, \pi_{\mathsf{ck}'})$: VrfyUpdate *takes as input a new key* ck'*, an old key* ck *and a proof* $\pi_{\mathsf{ck}'}$*, and outputs a bit indicating update correctness.*
– $(c, \tau) \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{m})$ *takes as input the commitment key and a message* $\mathbf{m} \in \mathcal{M}^n$*, and outputs a commitment* $c \in \mathcal{C}$ *and an opening trapdoor* $\tau \in \mathcal{T}$*.*
– $0/1 \leftarrow \mathsf{Open}(\mathsf{ck}, c, \mathbf{m}, \tau)$ *takes as input the commitment key, the message and the opening trapdoor and outputs a bit indicating the validity of the opening.*

which is Correct, Updateable Computationally Binding and Perfectly Hiding as defined next.

Definition 13. *An Updateable Commitment Scheme is correct if for all* λ, n

$$\Pr\left[\mathsf{VrfySetup}(\mathsf{ck}, 1^\lambda, n) = 1 \;\middle|\; \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, n)\right] = 1,$$

for all λ, n, ck

$$\Pr\left[\begin{array}{c}\mathsf{VrfySetup}(\mathsf{ck}', 1^\lambda, n) = 1 \wedge \\ \mathsf{VrfyUpdate}(\mathsf{ck}, \mathsf{ck}', \pi_{\mathsf{ck}'}) = 1\end{array} \;\middle|\; \begin{array}{c}\mathsf{VrfySetup}(\mathsf{ck}, 1^\lambda, n) = 1 \wedge \\ (\mathsf{ck}', \pi_{\mathsf{ck}'}) \leftarrow \mathsf{Update}(\mathsf{ck})\end{array}\right] = 1$$

and for all $\lambda, n, \mathsf{ck}, \mathbf{m}$

$$\Pr\left[\mathsf{Open}(\mathsf{ck}, c, \mathbf{m}, \tau) = 1 \;\middle|\; \begin{array}{c}\mathsf{VrfySetup}(\mathsf{ck}, 1^\lambda, n) = 1 \wedge \\ (c, \tau) \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{m})\end{array}\right] = 1.$$

Definition 14. *An Updateable Commitment Scheme has the Updateable Computational Binding property if for all stateful PPT* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, and for all* λ*,* n

$$\Pr\left[\begin{array}{c}\mathsf{VrfySetup}(\mathsf{ck}_1, 1^\lambda, n) = 1 \wedge \\ \mathsf{VrfyUpdate}(\mathsf{ck}_3, \mathsf{ck}_2, \pi_{\mathsf{ck}_3}) = 1 \wedge \\ \mathsf{Open}(\mathsf{ck}_3, c, \mathbf{m_1}, \tau_1) = 1 \wedge \\ \mathsf{Open}(\mathsf{ck}_3, c, \mathbf{m_2}, \tau_2) = 1 \wedge \\ \mathbf{m}_1 \neq \mathbf{m}_2\end{array} \;\middle|\; \begin{array}{r}(\mathsf{ck}_1, st_1) \leftarrow \mathcal{A}_1(1^\lambda, n) \wedge \\ (\mathsf{ck}_2, \pi_{\mathsf{ck}_2}) \leftarrow \mathsf{Update}(\mathsf{ck}_1) \wedge \\ (\mathsf{ck}_3, \pi_{\mathsf{ck}_3}, st_2) \leftarrow \mathcal{A}_2(st_1, \mathsf{ck}_2, \pi_{\mathsf{ck}_2}) \wedge \\ (c, \mathbf{m_1}, \tau_1, \mathbf{m_2}, \tau_2) \leftarrow \mathcal{A}_3(st_2)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 15. *An Updateable Commitment Scheme is perfectly hiding if, for all* $\lambda, n, \mathbf{m}$*, and all* ck *s.t.* $\mathsf{VrfySetup}(\mathsf{ck}, 1^\lambda, n) = 1$*, and all* c_1

$$\Pr\left[c = c_1 \;\middle|\; (c, \tau) \leftarrow \mathsf{Com}(\mathsf{ck}, \mathbf{m})\right] = \Pr\left[c = c_1 \;\middle|\; c \leftarrow \mathcal{C}\right].$$

3 Assumptions

We present the assumptions used in this work.

Definition 16. *(DLOG Assumption) The* DLOG *Assumption holds w.r.t. a group generator* GroupGen *if for all PPT adversaries* $\mathcal{A}$

$$\Pr\left[r = r' \;\middle|\; \mathsf{pp} \leftarrow \mathsf{GroupGen}(1^\lambda) \;\wedge\; r \leftarrow \mathbb{Z}_q \;\wedge\; r' \leftarrow \mathcal{A}(\mathsf{pp}, [r])\right] \leq \mathsf{negl}(\lambda).$$

We will also consider natural extensions of the DLOG Assumption. In the n-DLOG Assumption, the adversary receives n-powers of r, $[1], [r], \ldots, [r^n]$. In the Asymmetric DLOG Assumption in asymmetric bilinear groups, the adversary receives r in both groups $[r]_1, [r]_2$. Similarly, in the asymmetric n-DLOG Assumption, the adversary receives the powers of r in both groups. In either case, its goal is to compute $r \in \mathbb{Z}_q$.

The inner product argument of Bootle et al. [10] and the argument presented in this paper are based on the generalization of the DLOG Assumption presented next but with different vector distributions. The binding property of the vector commitments used in these arguments trivially reduces to this assumption.

Definition 17. *Let $n \in \mathbb{N}$. We call $\mathcal{D}_n$ a vector distribution if it outputs in PPT time, with overwhelming probability vectors in $\mathbb{Z}_q^n$.*

In this paper, $\mathcal{D}_n$ will typically be the distribution of the key of some perfectly hiding commitment scheme. More specifically, we will consider the distributions:

$$\mathcal{U}_n : \mathbf{r} = (1, x_1, \ldots, x_{n-1}), \qquad \mathcal{PW}_n : \mathbf{r} = \left(1, x, \ldots, x^{n-1}\right),$$

$$\mathcal{ML}_{2^\nu} : \mathbf{r} = (1, x_1, x_2, x_1 \cdot x_2, \ldots, x_1 \cdots x_\nu),$$

where $x, x_i \leftarrow \mathbb{Z}_q$. The first distribution is the uniform distribution, the second is the n-Power distribution and the last one is the multilinear monomial distribution with $n = 2^\nu$. Note that in the notation we introduced before, the power and multilinear monomial distribution can also be written as $\mathcal{PW}_n : \mathbf{r} = \mathbf{x^n}, x \leftarrow \mathbb{Z}_q$ and $\mathcal{ML}_{2^\nu} : \mathbf{r} = \overline{\mathbf{x}}, \mathbf{x} \leftarrow \mathbb{Z}_q^\nu$.

Definition 18. *The $\mathcal{D}_n$-Find-Rep Assumption holds with respect to* $\mathsf{GroupGen}$ *for all polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[[\mathbf{a}^\top \mathbf{r}] = [0] \ \wedge \ \mathbf{a} \neq \mathbf{0} \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{GroupGen}(1^\lambda) \wedge \\ \mathbf{r} \leftarrow \mathcal{D}_n \wedge \\ \mathbf{a} \leftarrow \mathcal{A}(\mathsf{pp}, [\mathbf{r}]) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

It is well known that the $\mathcal{U}_n$-Find-Rep (resp. $\mathcal{PW}_n$-Find-Rep) Assumption reduces to the DLOG (resp. q-DLOG) Assumption. For Multilinear Monomial distribution, we prove a similar result in Thm. 2. This assumption is inspired by the Naor-Reingold PRF [34].

In asymmetric bilinear groups, we define the Asymmetric $\mathcal{D}_n$-Find-Rep Assumption analogously except that the adversary receives $\mathbf{r}$ in both source groups $\mathbb{G}_1, \mathbb{G}_2$. We can prove similar reductions to asymmetric variants of the DLOG Assumption.

Theorem 2. *If there exists an adversary that runs in time $t(\lambda)$ and breaks the $\mathcal{ML}_{2^\nu}$-Find-Rep Assumption with probability $\epsilon(\lambda)$ with respect to a group generator* $\mathsf{BilGroupGen}(1^\lambda)$*, then there exists an adversary that breaks the Asymmetric Discrete Logarithm Assumption relative to* $\mathsf{BilGroupGen}(1^\lambda)$ *in time $\mathcal{O}_\lambda(2^\nu) + t(\lambda)$ with probability $\frac{\epsilon(\lambda)}{\nu}$.*

The proof of the theorem is presented in the full version.

4 Distribution Parameterized Vector Commitment

We can construct Updateable Commitment Schemes under the $\mathcal{D}_n$-Find-Rep assumptions we described. The Setup and Com are the same for all and they basically work as in the classical Pedersen Commitment.

We describe for the asymmetric $\mathcal{ML}_n, \mathcal{PW}_n$ distributions the algorithms related to the update (note that for $\mathcal{U}_n$, i.e. the Pedersen Vector Commitment, updateability trivially holds since the Setup is transparent). We present the $\mathcal{ML}_n$ case in detail and discuss which modifications are needed for the $\mathcal{PW}_n$ setting. For our application it is sufficient to give in $\mathbb{G}_2$ only the elements that define the commitment key, and not the whole key vector, i.e. $[\mathbf{x}]_2$ such that $\mathbf{r} = \overline{\mathbf{x}}$. Looking ahead, in the inner product argument $[\mathbf{x}]_2$ will be the compressed key the verifier has.

The update mechanism is fairly simple. To check a commitment key's structure, simply assert the various DDH relations that are implied by the $\mathcal{ML}_n$ distribution, and to update, pick a vector from $\mathcal{ML}_n$ and multiply it pairwise with the current key. NIZK PoK are used to assert that the previous randomness is taken into account in the new key and to ensure that any party updating knows its contribution to the final commitment key.

- $\mathsf{Setup}(1^\lambda, n)$
  - $\mathsf{pp} \leftarrow \mathsf{GroupGen}(1^\lambda)$.
  - $\mathbf{r} \leftarrow \mathcal{ML}_n$.
  - Output $\mathsf{pp}, [\mathbf{r}]_1, [\mathbf{x}]_2 \leftarrow ([r_1]_2, [r_2]_2, \ldots, [r_{2^i}]_2, \ldots, [r_{2^\nu}]_2)$.
- $\mathsf{VrfySetup}\,(\mathsf{pp}, [\mathbf{x}]_2, [\mathbf{r}]_1)$
  - Verify $[r_1]_1 = [1]_1$.
  - For $1 \leq i \leq \nu$, for $1 \leq j \leq 2^{i-1}$, check if $e([r_{2^{i-1}+j}]_1, [1]_2) = e([r_j]_1, [x_i]_2)$.
  - If all checks succeed output 1, otherwise output 0.
- $\mathsf{Update}\,(\mathsf{pp}, [\mathbf{x}]_2, [\mathbf{r}]_1)$
  - $\mathbf{y} \leftarrow \mathbb{Z}_q^\nu$.
  - Compute $[\mathbf{r}']_1 \leftarrow \overline{\mathbf{y}} \circ [\mathbf{r}]_1$, $[\mathbf{x}']_2 \leftarrow \mathbf{y} \circ [\mathbf{x}]_2$.
  - For $1 \leq i \leq \nu$, let $\pi_i \leftarrow \mathsf{NIZKAoK}\,\{([x_i]_2, [x_i']_2), (y_i) : [x_i']_2 = y_i[x_i]_2\}$.
  - Output $(\mathsf{pp}, [\mathbf{x}']_2, [\mathbf{r}']_1, \pi_1, \ldots, \pi_\nu)$.
- $\mathsf{VrfyUpdate}\,(\mathsf{pp}, [\mathbf{x}]_2, [\mathbf{x}']_2, [\mathbf{r}']_1, \pi_1, \ldots, \pi_\nu)$
  - If $\pi_1, \ldots, \pi_\nu$ are correct, output $\mathsf{VrfySetup}\,(\mathsf{pp}, [\mathbf{x}']_2, [\mathbf{r}']_1)$.
- $\mathsf{Com}\,(\mathsf{pp}, [\mathbf{r}]_1, \mathbf{m})$
  - Pick $\rho \leftarrow \mathbb{Z}_q$.
  - Compute $c \leftarrow [(\mathbf{m}, \rho)^\top \mathbf{r}]$.
  - Output (c, τ) where $\tau \leftarrow ([\mathbf{r}]_1, \rho)$.
- $\mathsf{Open}\,(\mathsf{pp}, [\mathbf{x}]_2, \mathbf{m}, c, \tau)$
  - Parse $\tau = ([\mathbf{r}]_1, \rho)$.
  - Output 1 iff $\mathsf{VrfySetup}\,(\mathsf{pp}, [\mathbf{x}]_2, [\mathbf{r}]_1)$ and $c = [(\mathbf{m}, \rho)^\top \mathbf{r}]$.

Theorem 3. *The $\mathcal{ML}_n$-Find-Rep Commitment scheme is Updateably Computationally Binding under the $\mathcal{ML}_n$-Find-Rep assumption, and the existence of a NIZK AoK for the relation $\mathcal{R} = \{(([x], [x']), y) \,|[x'] = y[x]\}$.*

The proof of the theorem is presented in the full version.

We can use a transparent scheme such as [12] to prove that an update is correctly performed, which will yield $\mathcal{O}_\lambda(\log\log n)$ proof size.

A similar construction works for the $\mathcal{PW}_n$ distribution. In this case, we simply need the element x encoded in $\mathbb{G}_2$ since this is enough to check that the key is drawn from the $\mathcal{PW}_n$ distribution. That is, for each i, it is enough to check that $e([r_i]_1, [1]_2) = e([r_{i-1}]_1, [x]_2)$. The Update and $\mathsf{VrfyUpdate}$ work in the same way but now a NIZK AoK is only needed for the element $[x]_2$.

As for concrete efficiency, the cost is dominated by the group exponentiations and the pairing operations for the verifier (the NIZK AoK statements are logarithmic in n). Setup and Update are dominated by n exponentiations in $\mathbb{G}_1$, $\mathsf{VrfySetup}$ and $\mathsf{VrfyUpdate}$ by n pairing operations, and Com and Open by one multi-exponentiation of size n in $\mathbb{G}_1$ which, if performed trivially needs n exponentiations. Proof size amounts to $\log n$ proofs of the NIZK AoK in the $\mathcal{ML}_n$ case and 1 in the $\mathcal{PW}_n$ case.

4.1 Commitments to Monomial Vectors

We will need to efficiently compute special commitments in the proof systems we present later. Specifically, given commitment schemes under $\mathcal{ML}_{2^\nu}$ and $\mathcal{PW}_{2^\nu}$ we will need to compute (non-hiding) commitments to $\mathbf{t^n}$ and $\overline{\mathbf{t}}$ where we know t and $t_1, \ldots, t_\nu$, respectively. Of course, these computations can be performed in time linear in the vector dimension, but we want to do so in sublinear (logarithmic in n) time. Since the univariate case reduces to the multilinear one by setting $t_i = t^{2^{i-1}}$, we only consider the most general case of computing $\overline{\mathbf{t}}$ when the keys are drawn from the $\mathcal{ML}_{2^\nu}$ distribution. We will need this in two different settings:

1. In the first case, let $\mathsf{ck} = (\mathsf{ck}_\mathcal{P}, \mathsf{ck}_\mathcal{V})$ be a commitment key. A prover, holding the whole commitment key $\mathsf{ck}_\mathcal{P}$, computes the commitment to $\overline{\mathbf{t}}$ w.r.t. ck, and gives it to a verifier, who holds only a compressed version of it, $\mathsf{ck}_\mathcal{V}$. It also gives a small proof that the issued commitment is a commitment to $\overline{\mathbf{t}}$ w.r.t. ck.
2. In the second case, given a commitment to $\mathbf{1^n}$ w.r.t. some commitment key $\mathsf{ck} = (\mathsf{ck}_\mathcal{P}, \mathsf{ck}_\mathcal{V})$ (which can be precomputed once), the verifier derives a commitment to $\overline{\mathbf{t}}$ w.r.t. a new commitment key $\mathsf{ck}' = (\mathsf{ck}'_\mathcal{P}, \mathsf{ck}'_\mathcal{V})$ in logarithmic time in n.

For the first case we use the following lemma:

Lemma 1. *Let* $\mathsf{ck} = (\mathsf{pp}, [\mathbf{x}]_2, [\mathbf{r}]_1)$ *be a commitment key where* $[\mathbf{r}]_1 = [\overline{\mathbf{x}}]$. *Then* $\mathsf{Com}_{\mathsf{ck}}(\overline{\mathbf{t}}) = \prod_{i=1}^{\nu}(1 + t_i x_i)[1]_1$.

Proof. We use induction on ν.

– When $\nu = 1$, we have $\overline{\mathbf{t}} = (1, t_1)$ and $\overline{\mathbf{x}} = (1, x_1)$. We get

$$\mathsf{Com}_{\mathsf{ck}}(\overline{\mathbf{t}}) = [r_1]_1 + t_1[r_2]_1 = [1]_1 + t_1 x_1 [1]_1 = (1 + t_1 x_1)[1]_1.$$

– For $\nu > 1$, we have $[\mathbf{r}_\nu]_1 = (\overline{\mathbf{x}}_{\nu-\mathbf{1}}[1]_1, x_\nu \overline{\mathbf{x}}_{\nu-\mathbf{1}}[1]_1)$ and $\overline{\mathbf{t}} = (\overline{\mathbf{t}}_{\nu-\mathbf{1}}, t_\nu \overline{\mathbf{t}}_{\nu-\mathbf{1}})$ and

$$\begin{aligned}\mathsf{Com}_{\mathsf{ck}}(\overline{\mathbf{t}}) = [\overline{\mathbf{t}}^\top \mathbf{r}_\nu]_1 &= [\overline{\mathbf{t}}_{\nu-\mathbf{1}}{}^\top \mathbf{r}_{\nu-\mathbf{1}}]_1 + [t_\nu \overline{\mathbf{t}}_{\nu-\mathbf{1}}{}^\top x_\nu \mathbf{r}_{\nu-\mathbf{1}}]_1 = \\ &= [\overline{\mathbf{t}}_{\nu-\mathbf{1}}{}^\top \mathbf{r}_{\nu-\mathbf{1}}]_1 + t_\nu x_\nu [\overline{\mathbf{t}}_{\nu-\mathbf{1}}{}^\top \mathbf{r}_{\nu-\mathbf{1}}]_1 = \\ &= (1 + t_\nu x_\nu)[\overline{\mathbf{t}}_{\nu-\mathbf{1}}{}^\top \mathbf{r}_{\nu-\mathbf{1}}]_1 = \\ &= (1 + t_\nu x_\nu) \prod_{i=1}^{\nu-1} (1 + t_i x_i)[1]_1,\end{aligned}$$

where the last equality follows from the induction hypothesis.

■

We take advantage of this structure by having the prover sending, for all $i \in \{1, \ldots, \nu\}$, the elements

$$[\tau_i]_1 \leftarrow \prod_{j=1}^{i} (1 + t_j x_j)[r]_1 = (1 + t_i x_i)[\tau_{i-1}]_1,$$

where $[\tau_0]_1 = [1]_1$. The verifier can then use the pairing to check

$$e(t_i[\tau_{i-1}]_1, [x_i]_2) = e([\tau_i - \tau_{i-1}]_1, [1]_2).$$

The prover needs to do $\log n$ $\mathbb{G}_1$ multi-exponentiations each of size 2^i for $i \in \left\{1, \ldots, \frac{n}{2}\right\}$, which can be implemented with n $\mathbb{G}_1$ exponentiations. The verifier needs to perform $\log n$ pairing operations and $2 \log n$ $\mathbb{G}_1$ exponentiations to verify.

For the second case, we do the following: suppose the verifier is given $\mathsf{Com}_{\mathsf{ck}_1}(\mathbf{1}) = [\mathbf{1}^\top \mathbf{r}]_1$. The verifier and the prover can compute a new verification key ck_2 as follows:

$$(\mathsf{ck}_2^{\mathcal{V}}, \mathsf{ck}_2^{\mathcal{P}}) = (([r]_1, t_1^{-1}[x_1]_2, \ldots, t_\nu^{-1}[x_\nu]_2), (\mathbf{r} \circ \overline{\mathbf{t}}^{-1})).$$

Then, we have:

$$[\mathbf{1}^\top \mathbf{r}]_1 = [(\mathbf{1} \circ \overline{\mathbf{t}})^\top (\mathbf{r} \circ \overline{\mathbf{t}}^{-1})]_1 = [\overline{\mathbf{t}}^\top (\mathbf{r} \circ \overline{\mathbf{t}}^{-1})]_1 = \mathsf{Com}_{\mathsf{ck}_2}(\overline{\mathbf{t}}).$$

The verifier needs $\log n$ $\mathbb{G}_2$ exponentiations and the prover can implicitly hold its key without computing it: when it needs to commit to $\mathbf{m}$ it can simply commit to $\mathbf{m} \circ \mathbf{t}^{-1}$ thus saving in expensive group operations.

5 Improved Inner Product Argument

In this section, we will first provide a high-level description of the inner product argument of [10], which has linear verification cost. Next, in subsection 5.2 we briefly discuss how to reduce the verification complexity to logarithmic in the designated verifier setting in the CRS model by changing the distribution of the commitment keys (still under the DLOG Assumption). In asymmetric bilinear groups, the construction can be "compiled" to achieve public verifiability, as discussed in subsection 5.3.

5.1 Inner Product Argument

We first briefly present the Inner Product Argument of [10]. The argument is a Proof of Knowledge of the openings of two (non-hiding) Vector Pedersen Commitments that satisfy an inner product relation. In [10], keys are sampled from $\mathcal{U}_n$. Formally, it is a proof of knowledge for the following language $\mathcal{L}_{\mathsf{IP}}$:

$$(\mathsf{pp}, [\mathbf{r}], [\mathbf{s}] \in \mathbb{G}^{2^\nu}, [\alpha], [\beta] \in \mathbb{G}, z \in \mathbb{Z}_q) \in \mathcal{L}_{\mathsf{IP}} \iff$$
$$\exists \mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^{2^\nu} \text{ s.t. } [\alpha] = [\mathbf{a}^\top \mathbf{r}] \wedge [\beta] = [\mathbf{b}^\top \mathbf{s}] \wedge \mathbf{a}^\top \mathbf{b} = z.$$

The idea of the protocol is to reduce this statement to an equivalent one of roughly half the size.

To do that, we create new commitment keys which have size half of the original one by splitting them in half and then combining them to a new key based on a challenge issued by the verifier. That is, the new commitment key will be $[\mathbf{r}'] = c^{-1}[\mathbf{r}_{\frac{1}{2}}] + c^{-2}[\mathbf{r}_{\frac{2}{2}}]$, where c is the verifier's challenge.

In order to prevent the prover from taking advantage of the split, we first ask her to give partial commitments $[\alpha_{-1}] = [\mathbf{a}_{\frac{1}{2}}^\top \mathbf{r}_{\frac{2}{2}}]$, $[\alpha_1] = [\mathbf{a}_{\frac{2}{2}}^\top \mathbf{r}_{\frac{1}{2}}]$.

The new witness will be $\mathbf{a}' = c\mathbf{a}_{\frac{1}{2}} + c^2 \mathbf{a}_{\frac{2}{2}}$. Note that both prover and verifier can compute the commitment to this new value, for every challenge c, from the partial commitments as follows:

$$\begin{aligned}
[\alpha'] = [\mathbf{a}'^\top \mathbf{r}'] &= [(\mathbf{a}_{\frac{1}{2}} c + \mathbf{a}_{\frac{2}{2}} c^2)^\top (c^{-1}\mathbf{r}_{\frac{1}{2}} + c^{-2}\mathbf{r}_{\frac{2}{2}})] \\
&= [\mathbf{a}_{\frac{1}{2}}^\top \mathbf{r}_{\frac{1}{2}}] + [\mathbf{a}_{\frac{2}{2}}^\top \mathbf{r}_{\frac{2}{2}}] + c^{-1}[\mathbf{a}_{\frac{1}{2}}^\top \mathbf{r}_{\frac{2}{2}}] + c[\mathbf{a}_{\frac{2}{2}}^\top \mathbf{r}_{\frac{1}{2}}] \\
&= [\alpha] + c^{-1}[\alpha_{-1}] + c[\alpha_1].
\end{aligned}$$

The same procedure is done for the second commitment $[\beta] = [\mathbf{b}^\top \mathbf{s}]$ with the inverse challenge c^{-1}.

Finally, the prover sends before seeing the challenge c the values $z_{-1} = \mathbf{a}_{\frac{2}{2}}^\top \mathbf{b}_{\frac{1}{2}}$ and $z_1 = \mathbf{a}_{\frac{1}{2}}^\top \mathbf{b}_{\frac{2}{2}}$, and based on these, the new inner product is computed as $z' = z_{-1}c + z + z_1 c^{-1}$. The new statement becomes $(\mathsf{pp}, [\mathbf{r}'], [\mathbf{s}'], [\alpha'], [\beta'], z') \in \mathcal{L}_{\mathsf{IP}}$.

Straightforward calculations assert that the new witness is indeed a witness for the new statement. The prover can now simply send the new witness $\mathbf{a}', \mathbf{b}'$ with cost half of what it would take to send $\mathbf{a}, \mathbf{b}$.

To achieve logarithmic complexity, the prover and the verifier recursively proceed in reducing the statement size until it is constant. The prover finally sends the witness. Under the generalized forking lemma the protocol remains sound.

We formally present the protocol next.

IPReduce

- Common input: $\sigma = (\mathsf{pp}, [\mathbf{r}], [\mathbf{s}])$, $[\alpha], [\beta]$, z.
- $\mathcal{P}$ input: $\mathbf{a}, \mathbf{b}$.
- Statement: $(\sigma, [\alpha], [\beta], z) \in \mathcal{L}_{\mathsf{IP}}$.

The prover and verifier proceed as follows:

- $\mathcal{P}$ computes

$$[\alpha_{-1}] \leftarrow [\mathbf{a}_{\frac{1}{2}}^\top \mathbf{r}_{\frac{2}{2}}], \qquad [\beta_{-1}] \leftarrow [\mathbf{b}_{\frac{1}{2}}^\top \mathbf{s}_{\frac{2}{2}}], \qquad z_{-1} \leftarrow \mathbf{a}_{\frac{2}{2}}^\top \mathbf{b}_{\frac{1}{2}},$$

$$[\alpha_1] \leftarrow [\mathbf{a}_{\frac{2}{2}}^\top \mathbf{r}_{\frac{1}{2}}], \qquad [\beta_1] \leftarrow [\mathbf{b}_{\frac{2}{2}}^\top \mathbf{s}_{\frac{1}{2}}], \qquad z_1 \leftarrow \mathbf{a}_{\frac{1}{2}}^\top \mathbf{b}_{\frac{2}{2}}.$$

- $\mathcal{P}$ sends $[\alpha_{-1}], [\alpha_1], [\beta_{-1}], [\beta_1], z_{-1}, z_1$ and $\mathcal{V}$ replies with $c \leftarrow \mathbb{Z}_q$.
- $\mathcal{P}$ computes

$$\mathbf{a}' \leftarrow \mathbf{a}_{\frac{1}{2}} c + \mathbf{a}_{\frac{2}{2}} c^2, \qquad \mathbf{b}' \leftarrow \mathbf{b}_{\frac{1}{2}} c^{-1} + \mathbf{b}_{\frac{2}{2}} c^{-2}.$$

- $\mathcal{P}$ and $\mathcal{V}$ compute

$$[\mathbf{r}'] \leftarrow c^{-1}[\mathbf{r}_{\frac{1}{2}}] + c^{-2}[\mathbf{r}_{\frac{2}{2}}], \qquad [\mathbf{s}'] \leftarrow c[\mathbf{s}_{\frac{1}{2}}] + c^2[\mathbf{s}_{\frac{2}{2}}],$$

$$[\alpha'] \leftarrow c^{-1}[\alpha_{-1}] + [\alpha] + c^1[\alpha_1], \qquad [\beta'] \leftarrow c^1[\beta_{-1}] + [\beta] + c^{-1}[\beta_1],$$
$$z' \leftarrow z_{-1}c^1 + z + z_1 c^{-1},$$
$$\sigma' \leftarrow (\mathsf{pp}, [\mathbf{r}'], [\mathbf{s}']).$$

- The reduced statement is $(\sigma', [\alpha'], [\beta'], z') \in \mathcal{L}_{\mathsf{IP}}$, with witness $\mathbf{a}', \mathbf{b}'$.

5.2 DV Inner Product Argument with Logarithmic Verifier

In this section we give the intuition on how to modify the above protocol with a $\mathcal{D}_n$-variant of the commitment scheme to achieve a logarithmic verifier. Full details are only given for the public verifiable scheme, which is very similar.

The linear overhead in the verifier's computation is computing the new key $\mathbf{r}'$. Having a structured commitment key allows to make this computation implicit for the verifier. If $\mathbf{r} \leftarrow \mathcal{ML}_n$, then $\mathbf{r} = (\mathbf{r}_{\frac{1}{2}}, \mathbf{r}_{\frac{2}{2}}) = (\mathbf{r}_{\frac{1}{2}}, x_\nu \mathbf{r}_{\frac{1}{2}})$. So, in the first round, the key for the next round is

$$[\mathbf{r}'] = c^{-1}[\mathbf{r}_{\frac{1}{\mathbf{2}}}] + c^{-2}[\mathbf{r}_{\frac{\mathbf{2}}{\mathbf{2}}}] = (c^{-1} + x_\nu c^{-2})[\mathbf{r}_{\frac{\mathbf{1}}{\mathbf{2}}}].$$

The new key is now determined by $[x_1], \ldots, [x_{\nu-1}]$ and the new generator $(c^{-1} + x_\nu c^{-2})[1]$. Further, this transformation respects the structure of the key, which can again be written as $\mathbf{r}' = (\mathbf{r}'_{\frac{1}{2}}, x_{\nu-1}\mathbf{r}'_{\frac{1}{2}})$, so the same argument can be applied again.

In the designated verifier case, we let the verifier know $x_1, \ldots, x_\nu$. It does not compute or read $[\mathbf{r}']$ in each round but just checks in the last round if:

$$[r'] = \prod_{i=1}^{\nu} (c_i^{-1} + x_{\nu-i+1} c_i^{-2})[1],$$

where c_i is the challenge at round i, and $[r']$ is the key in the last round (consisting of 1 element). The same holds for the second key $[s']$. Therefore, verification requires a logarithmic number of operations.

When $\mathbf{r} \leftarrow \mathcal{PW}_n$, the verification can also be reduced to logarithmic, as the structure of the key is very similar, namely, $\mathbf{r} = (\mathbf{r}_{\frac{1}{2}}, \mathbf{r}_{\frac{2}{2}}) = (\mathbf{r}_{\frac{1}{2}}, x^{2^{\nu-1}}\mathbf{r}_{\frac{1}{2}})$. The $\mathcal{PW}_{2^\nu}$ can be seen as a special case where $x_i = x^{2^{i-1}}$.

5.3 Inner Product Argument with Logarithmic Verifier

To allow public verifiability, we work in asymmetric bilinear groups. The verifier can no longer compute

$$\prod_{i=1}^{\nu}(c_i^{-1} + x_{\nu-i+1}c_i^{-2})[1],$$

but it lets the prover compute the intermediate values in each round (which it can compute without knowledge of x_i), and the verifier uses the pairing as a DDH oracle to verify this claim.

We now present the argument formally for the $\mathcal{ML}_{2^\nu}$ distribution (for $\mathcal{PW}_n$ the argument is defined similarly and we omit the details). First, we define the language of well structured commitments. We include the generator since it will be modified in each round.

$$(\mathsf{pp},[r]_1,[\mathbf{r}]_1,[\mathbf{x}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{ML}_{2^\nu}} \iff$$
$$[r_1]_1 = [r]_1 \wedge \forall i \in \{1,\ldots,\nu\}\, \forall j \in \left\{1,\ldots,2^{i-1}\right\} [r_{2^{i-1}+j}]_1 = x_i[r_j]_1.$$

The language to be proven and the reduction step are presented next.

$$\begin{aligned}
&(\mathsf{pp},[r]_1,[s]_1,[\mathbf{x}]_2,[\mathbf{y}]_2,[\alpha]_1,[\beta]_1,z) \in \mathcal{L}_{\mathsf{IP}} \iff \\
&\quad \exists\, [\mathbf{r}]_1,[\mathbf{s}]_1 \in \mathbb{G}^{2^\nu}, \mathbf{a},\mathbf{b} \in \mathbb{Z}_q^{2^\nu} \text{ s.t.} \\
&\quad (\mathsf{pp},[r]_1,[\mathbf{r}]_1,[\mathbf{x}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{ML}_{2^\nu}} \wedge (\mathsf{pp},[s]_1,[\mathbf{s}]_1,[\mathbf{y}]_2) \in \mathcal{L}_{\mathsf{Com}}^{\mathcal{ML}_{2^\nu}} \wedge \\
&\quad [\alpha]_1 = [\mathbf{a}^\top\mathbf{r}]_1 \wedge [\beta]_1 = [\mathbf{b}^\top\mathbf{s}]_1 \wedge \mathbf{a}^\top\mathbf{b} = z.
\end{aligned}$$

PVReduce

- Common input: $\sigma = (\mathsf{pp},[r]_1,[s]_1,[\mathbf{x}]_2,[\mathbf{y}]_2),[\alpha]_1,[\beta]_1,\, z$.
- $\mathcal{P}$ input: $\sigma_{\mathcal{P}} = (\mathsf{pp},[\mathbf{r}]_1,[\mathbf{s}]_1),\mathbf{a},\mathbf{b}$.
- Statement: $(\sigma,[\alpha]_1,[\beta]_1,z) \in \mathcal{L}_{\mathsf{IP}}$.

The prover and the verifier proceed as follows:

- $\mathcal{P}$ computes

$$\begin{aligned}
[\alpha_{-1}]_1 &\leftarrow [\mathbf{a}_{\frac{1}{2}}^\top\mathbf{r}_{\frac{2}{2}}]_1, & [\beta_{-1}]_1 &\leftarrow [\mathbf{b}_{\frac{1}{2}}^\top\mathbf{s}_{\frac{2}{2}}]_1, & z_{-1} &\leftarrow \mathbf{a}_{\frac{2}{2}}^\top\mathbf{b}_{\frac{1}{2}}, \\
[\alpha_1]_1 &\leftarrow [\mathbf{a}_{\frac{2}{2}}^\top\mathbf{r}_{\frac{1}{2}}]_1, & [\beta_1]_1 &\leftarrow [\mathbf{b}_{\frac{2}{2}}^\top\mathbf{s}_{\frac{1}{2}}]_1, & z_1 &\leftarrow \mathbf{a}_{\frac{1}{2}}^\top\mathbf{b}_{\frac{2}{2}}.
\end{aligned}$$

- $\mathcal{P}$ sends $[\alpha_{-1}]_1,[\alpha_1]_1,[\beta_{-1}]_1,[\beta_1]_1,z_{-1},z_1$ and $\mathcal{V}$ replies with $c \leftarrow \mathbb{Z}_q$

– $\mathcal{P}$ computes

$$\mathbf{a}' \leftarrow \mathbf{a}_{\frac{1}{2}}\, c + \mathbf{a}_{\frac{2}{2}}\, c^2, \qquad \mathbf{b}' \leftarrow \mathbf{b}_{\frac{1}{2}}\, c^{-1} + \mathbf{b}_{\frac{2}{2}}\, c^{-2},$$
$$[\mathbf{r}']_1 \leftarrow c^{-1}[\mathbf{r}_{\frac{1}{2}}]_1 + c^{-2}[\mathbf{r}_{\frac{2}{2}}]_1, \qquad [\mathbf{s}']_1 \leftarrow c[\mathbf{s}_{\frac{1}{2}}]_1 + c^2[\mathbf{s}_{\frac{2}{2}}]_1,$$
$$[r']_1 \leftarrow [r'_1]_1, \qquad [s']_1 \leftarrow [s'_1]_1,$$

$$\sigma'_{\mathcal{P}} = (\mathsf{pp}, [\mathbf{r}']_1, [\mathbf{s}']_1).$$

– $\mathcal{P}$ sends $[r']_1, [s']_1$.
– $\mathcal{V}$ checks the following pairing equations and aborts if any fail.

$$e([r']_1 - c^{-1}[r]_1, [1]_2) = e(c^{-2}[r]_1, [x_\nu]_2),$$
$$e([s']_1 - c[s]_1, [1]_2) = e(c^2[s]_1, [y_\nu]_2).$$

– Both compute

$$[\mathbf{x}']_2 \leftarrow ([x_i]_2)_{i\in\{1,\ldots,\nu-1\}}, \qquad [\mathbf{y}']_2 \leftarrow ([y_i]_2)_{i\in\{1,\ldots,\nu-1\}},$$
$$[\alpha']_1 \leftarrow c^{-1}[\alpha_{-1}]_1 + [\alpha]_1 + c[\alpha_1]_1, \qquad [\beta']_1 \leftarrow c[\beta_{-1}]_1 + [\beta]_1 + c^{-1}[\beta_1]_1,$$

$$z' = z_{-1}c + z + z_1 c^{-1},$$
$$\sigma' = \left(\mathsf{pp}, [r']_1, [s']_1, [\mathbf{x}']_2, [\mathbf{y}']_2\right).$$

– The reduced statement is $(\sigma', [\alpha']_1, [\beta']_1, z') \in \mathcal{L}_{\mathsf{IP}}$.

Theorem 4. *The protocol presented is a Public Coin, Argument of Knowledge for the relation $\mathcal{L}_{\mathsf{IP}}$ with $\log n$ round complexity, $\mathcal{O}_\lambda(n)$ prover complexity, and $\mathcal{O}_\lambda(\log n)$ communication and verification complexity under either the $\mathcal{ML}_n$-Find-Rep or the $\mathcal{PW}_n$-Find-Rep assumptions. The argument yields a Universally Updateable Non-Interactive AoK in the Random Oracle model. In the former case the proof size of an update is $\mathcal{O}_\lambda(\log n)$ and in the latter $\mathcal{O}_\lambda(1)$.*

Proof.
Completeness: We show that each reduction round leads to a valid reduced statement. It is enough to show that the prover and verifier compute the same key. Then, we can argue as in the case with uniform keys.

First, note that $[\mathbf{r}']_1 = c^{-1}[\mathbf{r}_{\frac{1}{2}}]_1 + c^{-2}[\mathbf{r}_{\frac{2}{2}}]_1$, which means that we "combine" all pair of elements that have distance $2^{\nu-1}$. That is, for all $j \leq 2^{\nu-1}$,

$$[r'_j]_1 = c^{-1}[r_j]_1 + c^{-2}[r_{2^{\nu-1}+j}]_1.$$

Also, note that, by construction of the commitment keys for all $i \in \{1,\ldots,\nu\}$ and $j \in \{1,\ldots,2^{i-1}\}$, it holds that $[r_{2^{i-1}+j}]_1 = x_i[r_j]_1$, which means that $[r']_1 = [r'_1]_1 = c^{-1}[r_1]_1 + c^{-2}[r_{2^{\nu-1}+1}]_1 = c^{-1}[r]_1 + c^{-2}x_\nu[r]_1$ and the verifier always accepts the pairing test.

It remains to show that $(\mathsf{pp}, [r']_1, [\mathbf{r}']_1, [\mathbf{x}']_2) \in \mathcal{L}_{\mathsf{Com}}$. It is evident that $[r'_1]_1 = [r']_1$. We show that the various Diffie-Hellman Relations hold for the reduced statement.

Let $i \in \{1, \ldots, \nu - 1\}$ and $j \in \left\{1, \ldots, 2^{i-1}\right\}$. It holds that $[r'_{2^{i-1}+j}]_1 = x_i[r'_j]_1$. Indeed,

$$\begin{aligned}[r'_{2^{i-1}+j}]_1 &= c^{-1}[r_{2^{i-1}+j}]_1 + c^{-2}[r_{2^{\nu-1}+2^{i-1}+j}]_1 = c^{-1}x_i[r_j]_1 + x_\nu x_i c^{-2}[r_j]_1 \\ &= x_i(c^{-1}[r_j]_1 + x_\nu c^{-2}[r_j]_1) = x_i[r'_j]_1.\end{aligned}$$

Similar calculations show the part related to $\mathbf{s}'$. We can now argue completeness exactly as in the $\mathcal{U}_{2^\nu}$ case.

Witness extended emulation: For witness extended emulation we need to prove that, for each round, we can extract the witness, i.e. the commitment key and the commitment openings w.r.t. it. We show next how to extract the commitment keys. After having these, we can argue as in [10] except that we use the corresponding $\mathcal{D}_n$-Find-Rep Assumption.

Assume we get two accepting transcripts for different challenges c from the prover. We show that given a witness for the reduced statement, we can extract the unique valid commitment keys $[\mathbf{r}]_1, [\mathbf{s}]_1$.

Let $[\mathbf{r}'_b]_1 = c_b^{-1}[\mathbf{r}_{\frac{1}{2}}]_1 + c_b^{-2}[\mathbf{r}_{\frac{2}{2}}]_1$ be the new commitment keys for two different challenges c_0, c_1. The matrix with rows (c_b^{-1}, c_b^{-2}) for $b \in \{0, 1\}$ is invertible, so we can take appropriate linear combination and extract $[\mathbf{r}_{\frac{1}{2}}]_1$, $[\mathbf{r}_{\frac{2}{2}}]_1$. We show that this is the commitment key. First note that since the transcript is accepting, we have that for both reduced keys $[r'_{2^{i-1}+j}]_1 = x_i[r'_j]_1$ which means that $[r_{2^{i-1}+j}]_1 = x_i[r_j]_1$ and $[r_{2^{\nu-1}+2^{i-1}+j}]_1 = x_i[r_{2^{\nu-1}+j}]_1$ for all $i \leq \nu - 1, j \leq 2^i$. In other words $[\mathbf{r}_{\frac{1}{2}}]_1$ and $[\mathbf{r}_{\frac{2}{2}}]_1$ are valid commitment keys w.r.t. the same $[x_1]_2, \ldots, [x_{\nu-1}]_2$. By the pairing test, we have that $[r'_b]_1 = c_b^{-1}[r]_1 + c_b^{-2}x_\nu[r]_1 = c_b^{-1}[r_{\frac{1}{2},1}]_1 + c_b^{-2}[r_{\frac{2}{2},1}]_1$. This equation holds for both challenges c_b, so it should be the case that $[r_{\frac{1}{2},1}]_1 = [r]$ and $[r_{\frac{2}{2},1}]_1 = x_\nu[r]$, thus the extracted key should be the unique key determined by $[x_1]_1, \ldots, [x_\nu]_1$. We argue for $[\mathbf{s}]_1$ in the same way. After extracting the keys the extractor works exactly as in [10] to extract $\mathbf{a}, \mathbf{b}$.

Complexity: It is evident that the protocol needs ν rounds. In each round the size of the witness is decreased in half, and we perform a constant number of communication, so we have $\mathcal{O}_\lambda(\nu)$ communication complexity. The prover in round i performs $\mathcal{O}_\lambda(2^{\nu+i-1})$ computations, so the prover complexity is $\mathcal{O}_\lambda\left(\sum_{i=1}^{\nu} 2^{\nu-i+1}\right) = \mathcal{O}_\lambda(2^\nu)$, while the verifier does $\mathcal{O}_\lambda(1)$ operations and therefore its complexity is $\mathcal{O}_\lambda(\nu)$. To be more concrete, the communication complexity is $8 \log n$ elements in $\mathbb{G}_1$ and $2 \log n$ elements in $\mathbb{Z}_q$. Prover complexity is dominated by 4 times $\log n$ multi-exponentiations of sizes $\frac{n}{2^i}$ in $\mathbb{G}_1$ to compute the first 4 messages in each round and less than $4n$ $\mathbb{G}_1$ exponentiations to compute all the keys. In total, $8n$ exponentiations in $\mathbb{G}_1$ with a non optimized implementation of multi-exponentiations.

■

6 Updateable Zero Knowledge SNARK for CSAT

We could use the improved inner product argument in a black box way to improve the verification of the zero knowledge protocol of Bootle et al. [10]. However, the source of inefficiency of verifier in [10] is twofold: the linear time needed in verifying the inner product argument, and some computation needed for the specific circuit. The latter is inherent to universal arguments since the verifier needs to, at least, read the circuit. The way to solve this is to add a circuit setup phase so the verifier will need to read the circuit only once. For a universal argument, this circuit setup should involve no secrets, that is, it should be a deterministic algorithm with input the Universal CRS and the circuit description. In this section, we give a sketch of the proof of Bootle et al. and explain where this source of inefficiency occurs in their construction. Then, we show how to overcome this using techniques similar to Sonic [32].

Roughly, the proof of [10] works as follows:

- $\mathcal{P}$ commits to its witness (a satisfying wire assignment) $\mathbf{w}$.
- $\mathcal{V}$ issues a random challenge y.
- $\mathcal{P}$ computes a polynomial $t(X) = \mathbf{q}_y(X)^\top(\mathbf{q}_y(X) \circ \mathbf{y^n} + 2\mathbf{s}_y(X)) + 2K$ where $\mathbf{q}_y(X)$ is a vector of polynomials that depends on $\mathbf{w}$ and y, and $\mathbf{s}_y$ on the circuit structure and the challenge y. K is a value that depends on the public input and y. The polynomial $t(X)$ has zero constant coefficient if and only if the circuit is satisfiable w.o.p. over the choice of y. It then sends a commitment to the polynomial $t(X)$ which has constant degree (it can commit to its coefficients using standard Pedersen Commitments).
- $\mathcal{V}$ picks and sends a random challenge x to the prover. $\mathcal{V}$ then computes commitments to $\mathbf{q}_y(x)$, $\mathbf{q}_y(x) \circ \mathbf{y^n}$, $\mathbf{s}_y(x)$ and K. The first two values are computed given a commitment to $\mathbf{w}$ and utilizing the homomorphic properties of the commitment scheme, and $\mathbf{s}_y$ is computed by the circuit description. K is computed efficiently by the public input.
- $\mathcal{P}$ decommits to $t_x = t(x)$. $\mathcal{V}$ checks this claim and the prover and verifier execute an inner product protocol to assert that $t(x) - 2K = \mathbf{q}_y(x)^\top(\mathbf{q}_y(x) \circ \mathbf{y^n} + 2\mathbf{s}_y(x))$. This convinces the verifier that the polynomial $t(x)$ has indeed a zero constant term and that it was computed honestly, thus the verifier is convinced about the claim.

The Verifier in [10] is linear in the circuit for three reasons:

- The inner product protocol in the last step needs linear time.
- Computing a commitment to $\mathbf{q}_y(x) \circ \mathbf{y^n}$ needs linear time.
- Computing a commitment to $\mathbf{s}_y(x)$ needs linear time.

The first two problems can be addressed easily: the first by using the improved inner product protocol, and the second by utilizing the structure of the $\mathcal{ML}_n$ or $\mathcal{PW}_n$ distributions to compute the commitment in logarithmic time. For the latter, the key homomorphic properties described in subsection 4.1 are utilized to efficiently obtain a commitment to $\mathbf{q}_y(x) \circ \mathbf{y^n}$ from a commitment to $\mathbf{q}_y(x)$.

The most subtle point is computing a commitment to $\mathbf{s}_y(x)$. This depends on the circuit structure and the challenge y. We solve it by applying similar techniques as Sonic [32]. We first preprocess the circuit to impose a specific structure that allows to "commit" to it efficiently. Then we use an aggregated Grand Product protocol which we introduce in the next section to delegate the computation of $\mathbf{s}_y(x)$ to the prover. We closely follow Sonic in the handling of this issue, but we differ from it in the setting: in this work we delegate computation of a vector commitment while in Sonic the prover decommits to bivariate polynomials of specific form by utilizing a univariate polynomial commitment scheme.

We present on the full version the preprocessing for the general case which only incurs in a constant overhead and so parameters remain optimal (i.e. linear in the size of computation).

6.1 Description of the ZK Argument

We assume that the circuit is preprocessed (see the full version) and has $n-1$ multiplication gates for $n = 2^\nu$ (the last element will be used as a blinding factor). The size of the public input and output is n'. The circuit is satisfiable iff the following constraints hold

$$\mathbf{a} \circ \mathbf{b} - \mathbf{c} = \mathbf{0},$$

$$\left\{a_i + \mathbf{w_{a,i}}^\top \mathbf{c} = 0\right\}_{i \in \{n'+1,\ldots,n-1\}}, \qquad \left\{a_i + \mathbf{w_{a,i}}^\top \mathbf{c} - \chi_i = 0\right\}_{i \in \{1,\ldots,n'\}},$$

$$\left\{b_i + \mathbf{w_{b,i}}^\top \mathbf{c} = 0\right\}_{i \in \{1,\ldots,n-1\}},$$

where $\mathbf{x} = (\chi_1, \ldots, \chi_{n'})$ is the public input and $\mathbf{w_{a,i}} = \mathbf{0}$ for $i \in \{1, \ldots, n'\}$. These equations are satisfied iff the circuit is satisfiable w.r.t. the input $\mathbf{x}$.

We can aggregate these equations as follows: First, add one extra zero element $a_n = b_n = 0$ to $\mathbf{a}, \mathbf{b}, \mathbf{c}$ to make them have 2^ν elements (these will be used as a blinding factor) and two extra zero constraints $a_n + \mathbf{0}^\top \mathbf{c} - 0 = 0$, $b_n + \mathbf{0}^\top \mathbf{c} - 0 = 0$ and set

$$p_m(Y) = (\mathbf{a} \circ \mathbf{b} - \mathbf{c})^\top \mathbf{Y^n} = \mathbf{a}^\top (\mathbf{b} \circ \mathbf{Y^n}) - \mathbf{c}^\top \mathbf{Y^n},$$

$$p_a(Y) = \sum_{i=1}^{n} \left(a_i + \mathbf{w_{a,i}}^\top \mathbf{c}\right) Y^{i-1} - \sum_{i=1}^{n'} \chi_i Y^{i-1} = \mathbf{a}^\top \mathbf{Y^n} + \mathbf{c}^\top \sum_{i=1}^{n} \mathbf{w_{a,i}} Y^{i-1} - \sum_{i=1}^{n'} \chi_i Y^{i-1},$$

$$p_b(Y) = \sum_{i=1}^{n} \left(b_i + \mathbf{w_{b,i}}^\top \mathbf{c}\right) Y^{i-1} = \mathbf{b}^\top \mathbf{Y^n} + \mathbf{c}^\top \sum_{i=1}^{n} \mathbf{w_{b,i}} Y^{i-1}.$$

Now, let

$$p(Y) = p_m(Y) + Y^n p_a(Y) + Y^{2n} p_b(Y).$$

The polynomial p should be identically zero iff the circuit is satisfiable. For a fixed y, we define $\mathbf{w_a}, \mathbf{w_b}, K$ as follows:

$$\mathbf{w_a} = \sum_{i=1}^{n} \mathbf{w_{a,i}} y^{i-1}, \qquad \mathbf{w_b} = \sum_{i=1}^{n} \mathbf{w_{b,i}} y^{i-1}, \qquad K = -y^n \sum_{i=1}^{n'} \chi_i y^{i-1}. \tag{1}$$

Note that these values only depend on the circuit, the input and the challenge. We now get

$$p(y) = \mathbf{a}^\top(\mathbf{b} \circ \mathbf{y^n}) + y^n \mathbf{a}^\top \mathbf{y^n} + y^{2n} \mathbf{b}^\top \mathbf{y^n} + \mathbf{c}^\top (y^n \mathbf{w_a} + y^{2n} \mathbf{w_b} - \mathbf{y^n}) + K.$$

We can now construct polynomials

$$\mathbf{q}(X) = \mathbf{a}X + \mathbf{b}X^{-1} + \mathbf{c}X^2 + \mathbf{d}X^3,$$
$$\mathbf{s}(X) = y^n \mathbf{y^n} X^{-1} + y^{2n} \mathbf{y^n} X + (y^n \mathbf{w_a} + y^{2n} \mathbf{w_b} - \mathbf{y^n}) X^{-2},$$
$$t(X) = \mathbf{q}(X)^\top (\mathbf{q}(X) \circ \mathbf{y^n} + 2\mathbf{s}(X)) + 2K.$$

Here $\mathbf{d}$ is some blinding factor chosen by the prover. The constant term of $t(X)$ equals $2p(y)$. The prover now can commit to the non-zero coefficients of t using standard Pedersen Commitment and then the verifier issues a new challenge x. The prover reveals t on this value, and the verifier needs to be convinced that the decommitted value is equivalent to computing the value on the right side. To do so, after agreeing on the (commitments of) vectors $\mathbf{q}(x), \mathbf{q}(x) \circ \mathbf{y^n} + 2\mathbf{s}(x)$, they execute an inner product protocol to assert that their inner product is $t(x) - 2K$. If that is the case, the verifier can be confident that the constant term of $t(x)$ is indeed zero, and thus the assignment satisfying. We sketch how the verifier computes the two commitments needed for the inner product protocol.

Let ck_1 be a commitment key defined in the CRS. The commitment to $\mathbf{q}(x)$ w.r.t. ck_1 can be computed by the homomorphic properties of the commitment scheme and commitments to $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$ w.r.t ck_1, which the provers issues in the first round.

Now, a commitment to $\mathbf{q}(x) \circ \mathbf{y^n} + 2\mathbf{s}(x)$ is needed to run the inner product argument. A commitment to $\mathbf{q}(x) \circ \mathbf{y^n}$, can be computed by the verifier, by deriving a new key ck_2, such that, the commitment to $\mathbf{q}(x)$ w.r.t. ck_1 is a commitment to $\mathbf{q}(x) \circ \mathbf{y^n}$ w.r.t. ck_2, as described in subsection 4.1. It remains to compute a commitment to $\mathbf{s}(x)$ w.r.t. ck_2.

Note that $\mathbf{s}(x)$ only depends on public values and the verifier can compute it, but would need linear time to do so. But if the verifier had commitments to $\mathbf{w_a}, \mathbf{w_b}$, it could compute the commitment to $\mathbf{s}(x)$ succinctly. To get such commitments, it delegates their computation to the prover. Assuming a preprocessed circuit, its description is given by matrices of the form $\mathbf{W_a} = \sum_{k=1}^{M} \mathbf{W_{a,k}}$ where $\mathbf{W_{a,k}}$ are matrices with, at most, one non-zero value in each column and row (respectively for $\mathbf{b}$). It follows by Eq. 1 and the structure of the preprocessed circuit matrix $\mathbf{W_a}$, that the verifier needs a commitment to

$$\mathbf{w_a} = \sum_{i=1}^{n} \mathbf{w_{a,i}} y^{i-1} = \mathbf{y^n}^\top \mathbf{W_a} = \mathbf{y^n}^\top \sum_{k=1}^{M} \mathbf{W_{a,k}} = \sum_{k=1}^{M} \sigma_k(\mathbf{y^n}) \circ \mathbf{w_{a,k}},$$

for known vectors $\mathbf{w_{a,k}}$ and permutations σ_k.

We sketch this delegation part in the next section, and provide a full description for it in the full version. A detailed description for the protocol is presented in the full version.

We state next the theorem which is the main result of our work.

Theorem 5. *There exists a Public Coin, Honest Verifier Zero Knowledge Argument of Knowledge for CSAT with $\mathcal{O}(\log|\mathcal{C}|)$ round complexity, $\mathcal{O}_\lambda(|\mathcal{C}|)$ prover complexity, and $\mathcal{O}_\lambda(\log|\mathcal{C}|)$ communication and verification complexity under either the $\mathcal{ML}_{|\mathcal{C}|}$-Find-Rep or the $\mathcal{PW}_{|\mathcal{C}|}$-Find-Rep assumptions. The argument yields a Universally Updateable NIZK AoK in the random oracle model. In the former case the proof size of an update is $\mathcal{O}_\lambda(\log|\mathcal{C}|)$ and in the latter $\mathcal{O}_\lambda(1)$.*

We note that, to achieve updateability, we rely on a NIZK AoK for proving correctness of the updates. One can be flexible in selecting such a NIZK AoK to fine tune efficiency measures. For example, one could combine the $\mathcal{ML}_{|\mathcal{C}|}$-Find-Rep scheme with [10] as the underline NIZK AoK for updateability to achieve $\mathcal{O}_\lambda(\log\log|\mathcal{C}|)$ proof size for proving correctness of an update.

7 Proof of Vector Permutation

We use techniques similar to Sonic to handle the computation regarding the structure of the circuit. We consider only the case of the left-wires for simplicity, i.e. the commitment to $\mathbf{w_a}$. The problem boils down to the following.

Let $\mathsf{ck}_1 = (\mathsf{ck}_1^{\mathcal{P}}, \mathsf{ck}_1^{\mathcal{V}}) = ([\mathbf{r}]_1, ([x_1]_2, \ldots, [x_\nu]_2))$, be a commitment key defined in the CRS, $[\omega_{a,1}]_1, \ldots, [\omega_{a,M}]_1$ be commitments to vectors $\mathbf{w_{a,1}}, \ldots, \mathbf{w_{a,M}}$ w.r.t. ck_1, $\sigma_{a,1}, \ldots, \sigma_{a,M}$ be commtiments to permutations w.r.t. ck_1 (i.e. $\mathsf{Com}_{\mathsf{ck}_1}(v_{a,i})$ where $v_{a,i} = (\sigma_{a,i}(1), \ldots, \sigma_{a,i}(n))$). These commitments succinctly encode the circuit structre. Given a value y and a commitment key $\mathsf{ck}_2 = (\mathsf{ck}_2^{\mathcal{P}}, \mathsf{ck}_2^{\mathcal{V}}) = (\mathbf{r} \circ \mathbf{y^{-n}}, ([x_1 y^{2^0}]_2, \ldots, [x_\nu y^{2^{\nu-1}}]_2))$, compute with the help of the prover a commitment $[\omega_a]_1$ to the vector $\mathbf{w_a} = \mathbf{w_{a,1}} \circ \sigma_{a,1}(\mathbf{y^n}) + \ldots + \mathbf{w_{a,M}} \circ \sigma_{a,M}(\mathbf{y^n})$ w.r.t. ck_2, where $\sigma_{a,i}(\mathbf{y^n}) = (y^{\sigma_{a,1}(1)-1}, \ldots, y^{\sigma_{a,M}(n)-1})$.

Note that, all the commitments that do not depend on the challenge y, can be computed once in a (deterministic) preprocessing phase, and can be reused in multiple proofs. The goal is to allow the verifier to compute the challenge dependent values in logarighmic time. These values are public and a linear time verifier could compute these on its own, though sacrifying succinctness.

The main difference with Sonic is in the setting. Sonic works with permutation polynomials, that is, polynomials of the form $p_i(X,Y) = \sum a_j X^j Y^{\sigma_i(j)}$ and the goal is to decommit to an evaluation in x, y for a polynomial $p(X,Y) = \sum_{i=1}^{M} p_i(X,Y)$, that is, the prover wants to reveal $p(x,y)$.

In both our work and Sonic, the heart of the protocol is a Permutation Argument which uses a Grand Product Argument [3,11]. We reduce the Grand Product Argument to an inner product and utilize the inner product argument of section 5, while Sonic, reduces it to verifying a value of a univariate polynomial and utilizes a univariate polynomial commitment scheme.

We next sketch the delegation protocol, and in the following subsection we describe how to reduce the Grand Product to an inner product.

To proceed the prover and the verifier do the following:

- The prover helps the verifier compute a commitment to $\mathbf{y^{-n}}$ w.r.t. ck_1, as explained in subsection 4.1.

– The prover provides values $[v_i]_1$, for $1 \leq i \leq M$, which it claims are commitments to $\sigma_{a,i}(\mathbf{y^n})$ w.r.t. ck_1.
– The prover gives $[\omega_{a,i}]_1$ and claims they are commitments to $\mathbf{w_{a,i}} \circ \sigma_{a,i}(\mathbf{y^n})$ w.r.t. ck_1.
– The prover gives $[\omega'_{a,i}]_1$ and claims they are commitments to $\mathbf{w_{a,i}} \circ \sigma_{a,i}(\mathbf{y^n}) \circ \mathbf{y^{-n}}$ w.r.t. ck_1. Equivalently, these are commitments to $\mathbf{w_{a,i}} \circ \sigma_{a,i}(\mathbf{y^n})$ w.r.t. ck_2.
– The prover and the verifier aggregate and reduce all the above claims to an inner product, which is verified by the improved inner product.
– The verifier sets $[\omega_a]_1 = [\omega'_{a,1}]_1 + \ldots + [\omega'_{a,M}]_1$ as a commitment to $\mathbf{w_a}$.

We present a sketch for reducing a Grand Product to an inner product in the next section. In the full version we present how we can aggregate all the above claims, and give a description of the protocol.

7.1 Proof of Grand Product

Let $\mathsf{ck}_1 = (\mathsf{ck}_1^{\mathcal{P}}, \mathsf{ck}_1^{\mathcal{V}}) = ([\mathbf{r}]_1, ([x_1]_2, \ldots, [x_\nu]_2))$, be a commitment key. Also, let $\mathbf{a_1} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b_1} = (b_1, b_2, \ldots, b_n)$, and $[\alpha_1]_1, [\beta_1]_1$ be commitments w.r.t. ck_1. The claim is that $\prod a_i = \prod b_i$.

Let $\mathbf{a_2} = (1, a_1, a_1a_2, \ldots, a_1 \cdots a_{n-1})$ be the vector of partial products and $\mathbf{a_3} = \mathbf{a_2} \circ \mathbf{a_1}$. We similarly define $\mathbf{b_2}, \mathbf{b_3}$. One can easily verify that $a_{3,n} = \prod_{i=1}^n a_i$ and $b_{3,n} = \prod_{i=1}^n b_i$. To convince the verifier, the prover gives commitments $[\alpha_2]_1$, $[\alpha_3]_1$, $[\beta_2]_1$, $[\beta_3]_1$ to vectors $\mathbf{a_2}$, $\mathbf{a_3}$, $\mathbf{b_2}$, $\mathbf{b_3}$ w.r.t. ck_1, convince it that they have the right form, and prove that $a_{3,n} = b_{3,n}$.

We express these requirements as a set of quadratic and linear constraints. We use different variables Y, W for the various groups of equations for presentational convenience, but we can use just one variable Y and set $W = Y^k$ for an appropriate k.

$$a_{3,n} = b_{3,n},$$

$$\begin{aligned} \mathbf{a_1} \circ \mathbf{a_2} &= \mathbf{a_3}, & \mathbf{b_1} \circ \mathbf{b_2} &= \mathbf{b_3}, \\ a_{2,1} &= 1, & b_{2,1} &= 1, \\ \{a_{2,i} &= a_{3,i-1}\}_{i=2}^n, & \{b_{2,i} &= b_{3,i-1}\}_{i=2}^n. \end{aligned}$$

We show how to reduce these equations to an inner product. We can aggregate the two Hadamard products by setting

$$p_1(Y) = \mathbf{a_1}^\top(\mathbf{a_2} \circ \mathbf{Y^n}) - \mathbf{a_3}^\top \mathbf{Y^n}, \quad p_2(Y) = \mathbf{b_1}^\top(\mathbf{b_2} \circ \mathbf{Y^n}) - \mathbf{b_3}^\top \mathbf{Y^n}.$$

We also set

$$p_3(Y) = (a_{2,1} - 1) + \sum_{i=2}^n (a_{2,i} - a_{3,i-1}) Y^{i-1} = \mathbf{a_2}^\top \mathbf{Y^n} - Y\mathbf{a_3}^\top(\mathbf{Y^n} - Y^{n-1}\mathbf{e_n}) - 1,$$

$$p_4(Y) = (b_{2,1} - 1) + \sum_{i=2}^n (b_{2,i} - b_{3,i-1}) Y^{i-1} = \mathbf{b_2}^\top \mathbf{Y^n} - Y\mathbf{b_3}^\top(\mathbf{Y^n} - Y^{n-1}\mathbf{e_n}) - 1,$$

$p_5(Y) = a_{3,n} - b_{3,n} = \mathbf{e_n}^\top \mathbf{a_3} - \mathbf{e_n}^\top \mathbf{b_3}$.

and $p(Y, W) = p_1(Y) + Wp_2(Y) + W^2 p_3(Y) + W^3 p_4(Y) + W^4 p_5(Y)$. The polynomial p is identically zero if and only if the constraints are satisfied. We use the technique of Bootle et al. to embed it in the constant term of a polynomial (similarly to the previous section). The resulting polynomials are

$$\begin{aligned}
\mathbf{q}(X) &= \mathbf{a_1}X + \mathbf{a_2}X^{-1} + w\mathbf{b_1}X^2 + \mathbf{b_2}X^{-2} + \mathbf{a_3}X^3 + \mathbf{b_3}X^4, \\
\mathbf{s}(X) &= w^2\mathbf{y^n}X + w^3\mathbf{y^n}X^2 \\
&\quad + \left(-w^2 y(\mathbf{y^n} - y^{n-1}\mathbf{e_n}) + w^4\mathbf{e_n} - \mathbf{y^n}\right) X^{-3} \\
&\quad + \left(-w^3 y(\mathbf{y^n} - y^{n-1}\mathbf{e_n}) - w^4\mathbf{e_n} - w\mathbf{y^n}\right) X^{-4}, \\
t(X) &= \mathbf{q}(X)^\top \left(\mathbf{q}(X) \circ \mathbf{y^n} + 2\mathbf{s}(X)\right) - 2w^2 - 2w^3.
\end{aligned}$$

The verifier computes a commitment to $\mathbf{q}(x) \circ \mathbf{y^n}$ w.r.t. the new commitment key -defined by the challenge y- as in the previous section. As for the commitment of $\mathbf{s}(x)$ w.r.t. this new key, however, the verifier can compute it itself: it only needs commitments to $\mathbf{y^n}$, and to $y^{n-1}\mathbf{e_n}$ w.r.t. the new key. For the first, it is given a commitment to $[o]_1 = [\mathbf{1^n}^\top \mathbf{r}]_1$ with the initial key, ck_1, and for the second, the last group element of the initial commitment key $[r_n]_1$. The desired commitments w.r.t the new key are $[o]_1$ and $[r_n]_1$. The prover and the verifier then proceed as in the CSAT case. Both $[o]_1$ and $[r_n]_1$ can be precomputed once. The detailed protocol is given in the full version.

Extending for multiple Grand Products. It is straightforward to extend these protocol to prove simultaneously M grand products. Also we can add kM quadratic equations of the form $\mathbf{c_1} \circ \mathbf{c_2} = \mathbf{c_3}$ to include the remaining constraints needed to compute the commitments needed for the CSAT case. We include the modified system of equation in the full version.

7.2 Proof of Known Permutation

Let $[\mathbf{r}]_1$ be a commitment key of size $n = 2^\nu$, $[\alpha]_1 = [\mathbf{a}^\top \mathbf{r}]_1$, $[\beta]_1 = [\mathbf{b}^\top \mathbf{r}]_1$ and $\sigma \in S_n$ be a permutation of $\{1, \ldots, n\}$. The prover wants to convince the verifier that, for all, i $b_i = a_{\sigma(i)}$.

In the same spirit as [32], we use the proof system of [3,11]. The verifier is given as input commitments to $(1, \ldots, n)$ and $(\sigma(1), \ldots, \sigma(n))$ denoted as $[\iota]_1, [\iota^\pi]_1$ respectively, and a commitment to $\mathbf{1^n}$ denoted as $[o]$. The idea is to reduce this problem to whether two vectors have equal grand products.

The verifier issues two challenges $t, u \in \mathbb{Z}_q$ and the prover needs to convince the verifier that

$$\prod_{i=1}^{n}(b_i + t\sigma_i - u) = \prod_{i=1}^{n}(a_i + ti - u)$$

Viewing these as polynomials in u, if their respective roots $\{b_i + t\sigma_i\}_{i\in\{1,\ldots,n\}}$ and $\{a_i + ti\}_{i\in\{1,\ldots,n\}}$ are different, they will be different in a fixed u with overwhelming probability (in a sufficiently large field). Also $b_i + t\sigma_i$ will be the σ

permutation of $a_i + ti$ only if for all i $b_i = a_{\sigma(i)}$, except with negligible probability. Thus, proving the grand product of the commitments $[\beta]_1 + t[\iota^\pi]_1 - u[o]_1$ and $[\alpha]_1 + t[\iota]_1 - u[o]_1$ (which are efficiently computable for the verifier) are equal is enough.

8 Range Proofs With Logarithmic Verifier

We present a new, more efficient aggregated range proof to allow a prover to convince a verifier that it knows openings for perfectly hiding commitments which all are in a range $[0, 2^m)$. This has applications in cryptocurrencies such as Monero to privatize transactions. Our approach resembles that of Bulletproofs [12]. The difference is that, in the inner product protocol of [12], the inner product claimed is encoded in the group (i.e. $\mathbf{a}^\top \mathbf{b}[r]$) while in our setting the inner product is given as an element of $\mathbb{Z}_q$. We thus slightly modify things to work in our setting. We exploit two things to achieve logarithmic verification time: the improved inner product argument, and the ability to compute structured commitments of the form $\mathbf{t^n}$ efficiently (either with the help of the prover or by modifying the commitment key). We present the blueprint of the scheme. Details for the protocol are presented in the full version.

Let $[0, 2^m)$ be the desired range and let ν be the smallest number such that $n = 2^\nu \geq m$. We first transform the statement to a set of linear and quadratic constraints, and we then construct a suitable inner product statement that holds if and only if the statement is correct w.o.p. Let $[\gamma]_1 = v[1] + \rho_c[r_2]_1$ ($[r_2]$ is used as a blinding factor for the commitment) be a hiding commitment to v. Equivalently, we can consider this as a binding commitment to the n-dimensional vector $\mathbf{c} = (v, \rho_c, 0, \ldots, 0)$, that is, $[\gamma]_1 = [\mathbf{c}^\top \mathbf{r}]_1$ for a given commitment key $[\mathbf{r}]_1$. The prover can compute the binary representation of v padding the end with zeros. Denote the padded representation $\mathbf{a}$. It is enough for the prover to show that:

- $\mathbf{a}^\top \mathbf{2^n} = \mathbf{c}^\top \mathbf{0^n}$ (note that we define $\mathbf{0^n}$ to have 1 as its first element).
- $\mathbf{a}$ has the first $m-1$ elements equal to either 0 or 1.
- $\mathbf{a}$ has all the other variables equal to zero.
- $\mathbf{c}$ has all but the first and second elements zero.

Now let $b_i = a_i - 1$ for $1 \leq i < m$, $b_i = 0$ for $i \geq m$. We express these constraints and aggregate them as follows:

$$\mathbf{a} \circ \mathbf{b} = \mathbf{0}, \qquad \{a_i - b_i - 1 = 0\}_{i=1}^{m-1}, \qquad \mathbf{a}^\top \mathbf{2^n} = \mathbf{c}^\top \mathbf{0^n},$$

$$\{a_i = 0\}_{i=m}^{n}, \qquad \{b_i = 0\}_{i=m}^{n}, \qquad \{c_i = 0\}_{i=3}^{n}.$$

Now let $\mathbf{Y_1} = (1, \ldots, Y^{m-2}, 0, \ldots, 0) \in \mathbb{Z}_q^n$, $\mathbf{Y_2} = (0, \ldots, 0, Y^{m-1}, \ldots, Y^{n-1}) \in \mathbb{Z}_q^n$ and $\mathbf{Y_3} = (0, 0, Y^2, \ldots, Y^{n-1}) \in \mathbb{Z}_q^n$. We define polynomials

$$p_1(Y) = \mathbf{a}^\top (\mathbf{b} \circ \mathbf{Y^n}),$$

$$p_2(Y) = \sum_{i=1}^{m-1}(a_i - b_i - 1)Y^{i-1} + \sum_{i=m}^{n} a_i Y^{i-1} = \mathbf{a}^\top \mathbf{Y^n} - \mathbf{b}^\top \mathbf{Y_1} - \mathbf{1^n}^\top \mathbf{Y_1},$$

$$p_3(Y) = \sum_{i=m}^{n} b_i Y^{i-1} = \mathbf{b}^\top \mathbf{Y_2}, \quad p_4(Y) = \sum_{i=3}^{n} c_i Y^{i-1} = \mathbf{c}^\top \mathbf{Y_3}.$$

The equations hold if and only if

$$p(Y) = p_1(Y) + Y^n p_2(Y) + Y^{2n} p_3(Y) + Y^{3n} p_4(Y) + Y^{4n}(\mathbf{a}^\top \mathbf{2^n} - \mathbf{c}^\top \mathbf{0^n}).$$

is identically zero. Similarly to the CSAT case, we define for fixed y

$$\begin{aligned}
\mathbf{q}(X) &= \mathbf{a}X + \mathbf{b}X^{-1} + \mathbf{c}X^2 + \mathbf{d}X^3, \\
\mathbf{s}(X) &= \left(y^n \mathbf{y^n} + y^{4n}\mathbf{2^n}\right) X^{-1} + \left(-y^n \mathbf{y_1} + y^{2n}\mathbf{y_2}\right) X \\
&\qquad + \left(y^{3n}\mathbf{y_3} - y^{4n}\mathbf{0^n}\right) X^{-2}, \\
\mathbf{t}(X) &= \mathbf{q}(x)^\top (\mathbf{q}(x) \circ \mathbf{y^n} + 2\mathbf{s}(X)) - y^n \mathbf{1^n}^\top \mathbf{y_1}.
\end{aligned}$$

Now the constant term of $t(X)$ should be zero for all Y, X if the constraints are satisfied so we proceed exactly as in the proof system of CSAT except that now it is easier to compute the vector $\mathbf{s}(x)$. In particular, the verifier can efficiently compute $\mathbf{s}(x)$, if it has commitments to $\mathbf{1^n}, (\mathbf{1^{m-1}}, \mathbf{0})$, $\mathbf{1^n} - (\mathbf{1^{m-1}}, \mathbf{0})$ and $(0, 0, 1, \ldots, 1, 1)$. By the key homomorphic properties of the commitment scheme these are commitments to $\mathbf{y^n}, \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}$ w.r.t. the new key. The prover and verifier can efficiently compute a commitment to the vector $\mathbf{2^n}$ w.r.t to the appropriate key as described in the polynomial commitment section. Finally, note that the inner product $\mathbf{1^n}^\top \mathbf{Y_1} = 1 + y + y^2 + \ldots + y^{m-2}$ can be efficiently computed by the verifier. Indeed, assuming w.l.o.g. (otherwise apply recursively) that $m - 2 + 1 = 2^\mu$ for some μ we have that $1 + y + y^2 + \ldots + y^{2^{\mu-1}} = (1 + y^{2^0})(1 + y^{2^1}) \cdots (1 + y^{2^{\mu-1}})$, and the verifier can compute this in logarithmic time. The full protocol is presented in the full version. We note that the aggregation techniques similar to [12] can be applied in the above.

We state the main theorem for the Range Proof protocol.

Theorem 6. *There exists a Public Coin, Honest Verifier Zero Knowledge Argument of Knowledge for the language* $\mathcal{L}_{RP} = \{m, [\alpha]_1, [1]_{1,2}, [r_2]_{1,2}, \mid \exists v, \rho_c \text{ s.t. } [\alpha]_1 = v[1]_1 + \rho_c[r_2]_1 \wedge v < 2^m\}$ *with* $\log m + \mathcal{O}(1)$ *round complexity,* $\mathcal{O}_\lambda(m)$ *prover complexity, and* $\mathcal{O}_\lambda(\log m)$ *communication and verification complexity under either the* $\mathcal{ML}_m$*-Find-Rep or the* $\mathcal{PW}_m$*-Find-Rep assumptions. The argument yields a Universally Updateable NIZK AoK in the Random Oracle model. In the former case the proof size of an update is* $\mathcal{O}_\lambda(\log m)$ *and in the latter* $\mathcal{O}_\lambda(1)$.

Acknowledgements. We would like to thank the anonymous reviewers of PKC 2020 and Helger Lipmaa for useful comments on this work.

The project that gave rise to these results received the support of a fellowship from "la Caixa" Foundation (ID 100010434). The fellowship code is

LCF/BQ/DI18/11660053. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713673. First author was supported by Project RTI2018-102112-B-I00 (AEI/FEDER,UE) and this paper is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 856879.

References

1. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A subversion-resistant SNARK. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 3–33. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_1
2. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134104
3. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_17
4. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53890-6_26
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 701–732. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_23
6. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from Bitcoin. Cryptology ePrint Archive, Report 2014/349 (2014), `http://eprint.iacr.org/2014/349`
7. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
8. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53644-5_2
9. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (Mar 2013). https://doi.org/10.1007/978-3-642-36594-2_18
10. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_12
11. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Takagi,

T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 336–365. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_12
12. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00020
13. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from dark compilers. Cryptology ePrint Archive, Report 2019/1229 (2019), https://eprint.iacr.org/2019/1229
14. Chiesa, A., Forbes, M.A., Spooner, N.: A zero knowledge sumcheck and its applications. Cryptology ePrint Archive, Report 2017/305 (2017), http://eprint.iacr.org/2017/305
15. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Report 2019/1047 (2019), https://eprint.iacr.org/2019/1047
16. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 532–550. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45611-8_28
17. Fuchsbauer, G.: Subversion-zero-knowledge SNARKs. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 315–347. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76578-5_11
18. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
19. Gabizon, A.: Auroralight: Improved prover efficiency and srs size in a sonic-like system. Cryptology ePrint Archive, Report 2019/601 (2019), https://eprint.iacr.org/2019/601
20. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumencial noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
21. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_37
22. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for boolean circuits. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 1069–1083. USENIX Association (Aug 2016)
23. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_11
24. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 113–122. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374396
25. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_19

26. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
27. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24
28. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 581–612. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63715-0_20
29. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007). https://doi.org/10.1145/1250790.1250794
30. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 525–537. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243805
31. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press (May 1992). https://doi.org/10.1145/129712.129782
32. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339817
33. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365746
34. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997). https://doi.org/10.1109/SFCS.1997.646134
35. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press (May 2013). https://doi.org/10.1109/SP.2013.47
36. Reitwiessner, C.: zksnarks in a nutshell (2016), `https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/`
37. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00060
38. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 733–764. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_24
39. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In: 2017 IEEE Symposium on Security and Privacy. pp. 863–880. IEEE Computer Society Press (May 2017). https://doi.org/10.1109/SP.2017.43
40. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: A zero-knowledge version of vSQL. Cryptology ePrint Archive, Report 2017/1146 (2017), `https://eprint.iacr.org/2017/1146`