Sub-linear Size Pairing-based Non-interactive Zero-Knowledge Arguments

Jens Groth*

University College London
E-mail: j.groth@ucl.ac.uk

Abstract. We construct non-interactive zero-knowledge arguments for circuit satisfiability and arithmetic circuits with perfect completeness, perfect zero-knowledge and computational (co-)soundness. The non-interactive zero-knowledge arguments have sub-linear size and very efficient public verification. Our construction uses bilinear groups and is only proven secure in the generic group model, but does not rely on random oracles.

Keywords: Sub-linear size non-interactive zero-knowledge arguments, circuit satisfiability, pairing-based cryptography, generic group model.

1 Introduction

Zero-knowledge proofs introduced by Goldwasser, Micali and Rackoff [GMR89] are fundamental building blocks in cryptography that are used in numerous protocols. Zero-knowledge proofs are protocols that enable a prover to convince a verifier of the truth of a statement without leaking any other information. The central properties are captured in the notions of completeness, soundness and zero-knowledge.

Completeness: The prover can convince the verifier if the statement is true. We will in this paper focus on statements of the form $x \in L$, where L is an NP-language, and the efficient-prover case, where the polynomial time prover knows an NP-witness w for $x \in L$ when she creates the zero-knowledge proof.

Soundness: A malicious prover cannot convince the verifier if the statement is false. We distinguish between computational soundness that protects against polynomial time cheating provers and statistical or perfect soundness where even an unbounded prover cannot convince the verifier of a false statement. We will call computationally sound proofs for *arguments*.

Zero-knowledge: A malicious verifier learns nothing except that the statement is true. We distinguish between computational zero-knowledge, where a polynomial time verifier learns nothing from the protocol interaction and statistical or perfect zero-knowledge, where even a verifier with unlimited resources learns nothing from the proof.

1.1 Non-interactive Zero-Knowledge Proofs

While the first zero-knowledge proofs were interactive, there are many cryptographic tasks that are done off-line, for instance signing messages or encrypting messages. For these situations, it is desirable to have *non-interactive* zero-knowledge (NIZK) proofs, where there is no interaction and a proof just consists of a single message from the prover to the verifier. Only trivial languages in BPP have NIZK proofs in the plain model without any setup [Ore87,GO94,GK96]. However, Blum, Feldman and Micali [BFM88] introduced NIZK proofs in the *common reference string model*, where both the prover and verifier have access to a common reference string generated in a trusted way. Such NIZK proofs have many applications, ranging from the first chosen ciphertext attack secure construction of public-key encryption [DDN00] to recent advanced

* Part of this research was done while visiting IPAM, UCLA.

signature schemes [CGS07,BW06]. For this reason there has been a lot of research into the underlying assumptions [FLS99,BCNP04,GO07], the efficiency [Dam92,DDP02,KP98], and the security guarantees offered by NIZK proofs [DP92,Sah01,DDO$^+$02].

NIZK proofs based on standard cryptographic assumptions used to be inefficient and not useful in practice. To get around this inefficiency, applied cryptographers have relied on the so-called Fiat-Shamir heuristic for transforming public-coin interactive zero-knowledge proofs into NIZK arguments by using a cryptographic hash-function to compute the verifier's challenges. The Fiat-Shamir heuristic can give very efficient NIZK arguments that are secure in the random oracle model [BR93], where the cryptographic hash-function is modeled as a random function. Unfortunately, there are several examples of protocols that are secure in the random oracle model, but do not have any secure standard model instantiation no matter which hash-function is used [CGH98,CGH04,MRH04,BBP04,Nie02]. Particularly relevant here is Goldwasser and Kalai's [GK03] demonstration of a signature scheme built from a public-coin identification scheme that is secure in the random oracle model, but insecure in real life. While it is possible that the Fiat-Shamir heuristic is secure for "natural" protocols, it is worthwhile to investigate alternative approaches.

Another way to get around the inefficiency of traditional NIZK proofs is to use non-interactive designated verifier proofs. In a designated verifier proof, the proof is not publicly verifiable, it can only be verified by a designated verifier. Damgård, Fazio and Nicolosi [DFN06] gave an efficient linear size non-interactive designated verifier proof for circuit satisfiability based on an assumption related to Paillier encryption. There are cases where designated verifier proofs suffice, for instance Cramer and Shoup's chosen ciphertext attack secure public-key cryptosystem [CS98]. In many other cases, the lack of public verifiability is problematic though. When there is only one designated verifier, it is for instance not possible to use them to construct advanced digital signatures, such as ring signatures and group signatures, since here public verifiability is needed for non-repudiation.

Recent work in NIZK proofs has used bilinear groups to improve efficiency. Groth, Ostrovsky and Sahai [GOS06b,GOS06a] gave NIZK proofs for circuit satisfiability where the proof consists of $O(|C|)$ group elements, with $|C|$ being the number of gates in the circuit. Their NIZK proofs have the property that they can be set up to give either perfect soundness and computational zero-knowledge, or alternatively computational (co-)soundness and perfect zero-knowledge. Works by Boyen, Waters, Groth and Sahai [BW06,BW07,Gro06,GS08] have explored how to build efficient NIZK proofs that are directly applicable in bilinear groups instead of going through circuit satisfiability.

In some special cases, for instance in the ring signature of Chandran, Groth and Sahai [CGS07] these techniques lead to sub-linear size NIZK proofs, but in general the number of group elements in an NIZK proof grows linearly in the size of the statement. Looking at the general NP-complete problem of circuit satisfiability, the reason these NIZK proofs grow linearly in the circuit size is that they encrypt the value of each wire in the circuit. Abe and Fehr [AF07] gave a construction based on commitments instead of encryptions, but since each wire is committed by itself they also get a linear growth in the size of the circuit. The goal in this paper is to break the linear size barrier and propose NIZK arguments with sub-linear size without using the Fiat-Shamir heuristic.

1.2 Our Contribution

We construct NIZK arguments for circuit satisfiability and arithmetic circuits with perfect completeness, computational (co-)soundness and perfect zero-knowledge. The NIZK arguments are both of sub-linear size and very efficient to verify, but the prover uses a super-linear number of group operations. Table 1 gives a quick comparison to other NIZK proofs and arguments for circuit satisfiability, where G stands for the size of a group elements, and M, E, and P, are the costs of respectively multiplication, exponentiation and pairings in a bilinear group (see Section 2.1). We refer to Section 11 for further details on the efficiency of our NIZK argument including the exact constants hidden by the Big-Oh notation.

	CRS size	Proof size	Prover comp.	Verifier comp.	Soundness	Zero-knowledge	Assumption
Kilian-Petrank [KP98]	$O(\|C\|\lambda^2)$ G	$O(\|C\|\lambda^2)$ G	$O(\|C\|\lambda^2)$ E	$O(\|C\|\lambda^2)$ M	statistical	computational	trapdoor permutation
Groth-Ostrovsky-Sahai	$O(1)$ G	$O(\|C\|)$ G	$O(\|C\|)$ E	$O(\|C\|)$ P	perfect	computational	subgroup decision or
[GOS06b,GOS06a]	$O(1)$ G	$O(\|C\|)$ G	$O(\|C\|)$ E	$O(\|C\|)$ E	computational	perfect	linear decision
Abe-Fehr [AF07]	$O(1)$ G	$O(\|C\|)$ G	$O(\|C\|)$ E	$O(\|C\|)$ E	computational	perfect	DLog and knowledge of expo.
Groth [Gro09]	$O(\sqrt{\|C\|})$ G	$O(\sqrt{\|C\|})$ G	$O(\|C\|)$ M	$O(\|C\|)$ M	computational	perfect	DLog and Fiat-Shamir
This paper	$O(\|C\|^{\frac{3}{4}})$ G	$O(\|C\|^{\frac{3}{4}})$ G	$O(\|C\|^{\frac{5}{4}})$ M	$O(\|C\|)$ M	computational	perfect	generic bilinear group

Table 1. Comparison of NIZK proof and arguments.

TRANSFERABILITY AND EFFICIENCY. Our NIZK arguments are publicly verifiable. This means that unlike interactive zero-knowledge proofs they are transferable; they can be copied and distributed to many different entities that can do their own independent verification.

Due to their transferability, there are many situations where it is reasonable to accept a super-linear computational cost for the prover in return for less communication and very efficient verification. In situations where many entities want to verify the same NIZK argument, the one-off cost of creating the NIZK argument may be acceptable due to bandwidth savings in distributing the NIZK argument to many parties. Further, the NIZK argument is as efficient to verify as state of the art interactive zero-knowledge arguments, so the prover's computational overhead may be offset by many verifiers' highly efficient verification process.

SETUP AND ZERO-KNOWLEDGE/WITNESS-INDISTINGUISHABILITY. We assume a trusted setup, where a common reference string is made available to the prover and verifier. If the trust is violated and the common reference string is adversarially chosen, we are in better shape than with traditional zero-knowledge proofs though. Most NIZK proofs are proofs of knowledge, where the adversary might learn all of the prover's witness if the prover uses an adversarially chosen common reference string. In contrast, sub-linear size NIZK arguments have the advantage that there is an information theoretic upper bound on how many bits of the witness may be leaked.

The common reference string in our NIZK argument has a particular structure; it is not a uniform random string. The common reference string is verifiable though; the prover can verify that the common reference string does have the right structure. This implies that even if the trust assumption is violated and the common reference string is chosen by the adversary, the argument will still be perfectly witness-indistinguishable, i.e., it will be impossible for the adversary to tell which out of potentially many witnesses the prover used when creating the argument. Dwork and Naor [DN00] introduced the notion of Zaps, 2-move witness-indistinguishable proofs, where the verifier's initial message can be used over and over again. The results in this paper imply the existence of communication-efficient Zaps with perfect witness-indistinguishability and computational soundness.

ASSUMPTIONS AND SOUNDNESS/CO-SOUNDNESS. As we will demonstrate, it is infeasible for an adversary restricted to the generic bilinear group operations to give a valid argument for a false statement. Even stronger, in the generic group model [Nec94,Sho97] there is negligible probability of constructing a valid argument without being able to extract a valid witness. Based on this we may conjecture that our NIZK argument is a computationally sound argument of knowledge for circuit satisfiability.

Basing security on the generic group model is risky. Just as there exist protocols with no secure instantiation that are provably secure in the random oracle model, so there exists a signature scheme with no secure instantiation that is secure in the generic group model as demonstrated by Dent [Den02]. We believe our use of the generic group model is more benign than the typical use of the random oracle model though. To make a sub-linear size NIZK argument in the random oracle model, soundness is usually proved through rewinding the protocol and then running the protocol again with a different output from the random oracle. In real life it is of course not possible for a deterministic hash-function to have two different outputs. In contrast, we do not use rewinding techniques to prove security of our NIZK argument in the generic group model. The issue of rewinding can in theory be overcome though, Pass [Pas03] and Fischlin [Fis05] have given NIZK arguments with straight-line extractors in the random oracle model. While it is quite speculative to argue the merits of one heuristic model over another, we believe in either case that it is desirable to develop a set of techniques that is completely different from the previous random oracle model approach.

Another concern regarding soundness is that any NIZK argument with statistical or perfect zero-knowledge is non-falsifiable in Naor's terminology [Nao03]. Given an adversary producing a statement and a valid argument, it may be impossible to efficiently check whether the statement is false and therefore impossible to verify that soundness has been broken. Abe and Fehr [AF07] showed that only languages in P/poly can have an NIZK argument with statistical zero-knowledge, where soundness is proved using a "direct black-box" proof to a falsifiable cryptographic assumption. All known NIZK arguments with statistical or perfect zero-knowledge, including Abe and Fehr's NIZK argument, therefore rely on non-falsifiable assumptions. Bellare and Palacio [BP04], have shown that it may be possible to combine a standard cryptographic assumption and a falsifiable assumption to get a falsifiable assumption though. So if a NIZK argument is composed with another cryptographic protocol, it may be that the security of this composition is falsifiable.

To capture a falsifiable notion of security that can undergo cryptanalytic scrutiny, Groth, Ostrovsky and Sahai [GOS06b] introduced the weaker notion co-soundness. When defining co-soundness, we restrict ourselves to languages in $\mathrm{NP} \cap \mathrm{coNP}$. An NIZK argument is co-sound if it is infeasible for the adversary to create a valid argument for a false statement *together with a* coNP *witness that the statement is false*. The extra requirement of the adversary having to produce a witness for the statement being false, leads to the co-soundness of an NIZK argument being falsifiable. As a consequence, it is possible to prove from natural cryptographic assumptions that the perfect zero-knowledge variants of Groth, Ostrovsky and Sahai's [GOS06b,GOS06a] NIZK arguments are co-sound. One might worry about the restriction to languages in $\mathrm{NP} \cap \mathrm{coNP}$, however, in the authors' experience most statements arising in the design of cryptographic protocols do actually belong to $\mathrm{NP} \cap \mathrm{coNP}$. Consider for example the setting of verifiable encryption, where we may want to argue that the plaintext contains the string "$ 1,000,000". Here the decryption key may be a coNP witness for the plaintext not containing this string. We remark that in protocols using NIZK arguments with co-soundness, the adversary is not expected to *explicitly* produce the coNP witness, it suffices that the coNP witness is available in the security reduction when proving security of the protocol.

This leads us to the weaker conjecture that our NIZK argument is computationally co-sound. While falsifiable, we still consider this a strong assumption to make. In favor of the assumption, we note that it is a standard-model computational assumption, where the adversary is given an input and has to produce an output that can be checked whether it constitutes a breach. The amount of the adversary's input and output depends on the size of statements we want to give NIZK arguments for. The assumption therefore falls into the class of "q-style" assumptions in pairing-based cryptography of which the q-SDH assumption [BB04] is arguably the most prominent member.

Finally, it is worth noting the trade-off between soundness and zero-knowledge. It is well-known that it is not possible to have both statistical/perfect soundness and statistical/perfect zero-knowledge at the same time for non-trivial languages in NP. In many applications, soundness is checked shortly after the generation of the NIZK argument whereas confidentiality should be preserved many years into the future. NIZK proofs with statistical/perfect soundness may be the wrong choice since technical advances may lead to a breach of the computational zero-knowledge property. For such situations, it is better to be guaranteed long-term zero-knowledge and have soundness that is based on a cryptographic assumption that nobody knows how to break with current cryptanalytic techniques.

2 Preliminaries

Given two functions $f, g : \mathbb{N} \to [0,1]$ we write $f(\lambda) \approx g(\lambda)$ when $|f(\lambda) - g(\lambda)| = O(\lambda^{-c})$ for every constant $c > 0$. We say that f is *negligible* when $f(\lambda) \approx 0$ and that it is *overwhelming* when $f(\lambda) \approx 1$.

We write $y = A(x;r)$ when the algorithm A, on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x;r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S. We will assume it is possible to sample uniformly at random from sets such as $\mathbb{Z}_p$. We define $[n]$ to be the set $\{1, 2, \ldots, n\}$.

2.1 Bilinear Groups

Our NIZK argument will use prime order bilinear groups. Let $\mathcal{G}$ take a security parameter λ written in unary as input and output a description of a bilinear group $(p, G, G_T, e, g) \leftarrow \mathcal{G}(1^\lambda)$ such that

1. p is a λ-bit prime.
2. G, G_T are cyclic groups of order p.
3. $e : G \times G$ is a bilinear map (pairing) such that $\forall a, b : e(g^a, g^b) = e(g,g)^{ab}$.
4. g generates G and $e(g,g)$ generates G_T.
5. Membership in G, G_T can be efficiently decided, group operations and the pairing e are efficiently computable and the descriptions of the groups and group elements each have size $O(\lambda)$ bits.

See for instance Boneh and Franklin's paper [BF03] for an elliptic curve example of a candidate bilinear group with these properties. Their example has the additional advantage that it is possible to verify whether the output (p, G, G_T, e, g) of $\mathcal{G}$ actually describes a bilinear group. Furthermore, in their example it is possible to pick the prime p first and then construct a bilinear group on top of that prime. This latter property is useful in case we have a specific field $\mathbb{Z}_p$ in mind for which we want to argue satisfiability of an arithmetic circuit.

EFFICIENCY. When giving efficiency estimates in the paper, we will use the following notation. P is the computational cost of computing a pairing, E is the computational cost of computing an exponentiation and M is the computational cost of computing a multiplication in G. We write G for the size of a group element in G. We will assume membership of G is always verified, but ignore the cost of verifying membership of G, since on elliptic curves it can be done with a few multiplications and will therefore be insignificant compared to other computational costs in our NIZK argument.

2.2 The Schwartz-Zippel Lemma

For completeness we state a variation of the well-known Schwartz-Zippel lemma that we will use several times in the paper.

Lemma 1 (Schwartz-Zippel). *Let* poly *be a non-zero multivariate polynomial of degree* d *over* $\mathbb{Z}_p$*, then the probability of* $\mathrm{poly}(x_1, \ldots, x_m) = 0$ *for randomly chosen* $x_1, \ldots, x_m \leftarrow S \subset \mathbb{Z}_p$ *is at most* $d/|S|$.

The Schwartz-Zippel lemma is frequently used in polynomial identity testing. Given two multi-variate polynomials poly_1 and poly_2 we can test whether $\mathrm{poly}_1(x_1, \ldots, x_m) - \mathrm{poly}_2(x_1, \ldots, x_m) = 0$ for random $x_1, \ldots, x_m \leftarrow S$. If the two polynomials are identical this will always be true, whereas if the two polynomials are different then there is only probability $\max(d_1, d_2)/|S|$ for the equality to hold.

3 Non-interactive Arguments

Let R be an efficiently computable trinary relation. For triples $(\sigma, x, w) \in R$ we call x the statement and w the witness. Let L_σ be the NP-language consisting of statements with witnesses in R given σ. If σ is the empty string this is the standard definition of an NP-language. We will use the more general definition where the relation may or may not depend on an input σ.

A non-interactive argument for a relation R consists of a common reference string generator algorithm K, a prover algorithm P and a verifier algorithm V that run in probabilistic polynomial time. The common reference string generator takes as input a security parameter λ and may also take additional inputs and produces a common reference string σ of length $\Omega(\lambda)$. In our case, the additional input to the key generation algorithm may be a value $n \in \mathbb{N}$ specifying the size of statements we are interested in. The prover takes as input (σ, x, w) and produces an argument π. The verifier takes as input (σ, x, π) and outputs 1 if the argument is acceptable and 0 if rejecting the argument. We call (K, P, V) an argument for R if it has the completeness and soundness property described below.

PERFECT COMPLETENESS. For all adversaries $\mathcal{A}$ and $n = \lambda^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, x, w) : V(\sigma, x, \pi) = 1 \text{ if } (\sigma, x, w) \in R\right] = 1.$$

COMPUTATIONAL SOUNDNESS. For all non-uniform polynomial time adversaries $\mathcal{A}$ and $n = \lambda^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, \pi) \leftarrow \mathcal{A}(\sigma) : x \notin L_\sigma \text{ and } V(\sigma, x, \pi) = 1\right] \approx 0.$$

COMPUTATIONAL ARGUMENT OF KNOWLEDGE. We call (K, P, V) an argument of knowledge if there is an extractor that can compute a witness whenever the adversary produces a valid argument. The extractor gets full access to the adversary's state, including any random coins. Since extraction of a witness implies the existence of such a witness, an argument of knowledge is always sound.

Formally, we require that for all non-uniform polynomial time adversaries $\mathcal{A}$ there exists a non-uniform polynomial time extractor $E_\mathcal{A}$ using the same random coins such that for all $n = \lambda^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, \pi) \leftarrow \mathcal{A}(\sigma); w \leftarrow E_\mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ and } V(\sigma, x, \pi) = 1\right] \approx 0.$$

COMPUTATIONAL CO-SOUNDNESS. Let R_{co} be a polynomial time decidable relation consisting of statements and witnesses w_{co} so $(\sigma, x, w_{\mathrm{co}}) \in R_{\mathrm{co}}$ implies $x \notin L_\sigma$. We say the argument is co-sound if for all non-uniform polynomial time adversaries $\mathcal{A}$ and $n = \lambda^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, \pi, w_{\mathrm{co}}) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w_{\mathrm{co}}) \in R_{\mathrm{co}} \text{ and} V(\sigma, x, \pi) = 1\right] \approx 0.$$

PERFECT WITNESS-INDISTINGUISHABILITY. We say a non-interactive argument (K, P, V) is perfectly witness-indistinguishable if it is impossible to tell which witness the prover used in case there are many possible witnesses. For all stateful interactive adversaries $\mathcal{A}$ and $n = \lambda^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, w_0, w_1) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, x, w_0) : (\sigma, x, w_0), (\sigma, x, w_1) \in R \text{ and } \mathcal{A}(\pi) = 1\right]$$
$$= \Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, w_0, w_1) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, x, w_1) : (\sigma, x, w_0), (\sigma, x, w_1) \in R \text{ and } \mathcal{A}(\pi) = 1\right].$$

PERFECT ZERO-KNOWLEDGE. We say a non-interactive argument (K, P, V) is perfect zero-knowledge if there exists a polynomial time simulator $S = (S_1, S_2)$ with the following zero-knowledge property. S_1 outputs a simulated common reference string and a simulation trapdoor. S_2 takes the common reference string, the simulation trapdoor and a statement as input and produces a simulated argument. For all stateful interactive adversaries $\mathcal{A}$ and $n = \lambda^{O(1)}$ we require

$$\Pr\left[\sigma \leftarrow K(1^\lambda, n); (x, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, x, w) : (\sigma, x, w) \in R \text{ and } \mathcal{A}(\pi) = 1\right]$$
$$= \Pr\left[(\sigma, \tau) \leftarrow S_1(1^\lambda, n); (x, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow S_2(\sigma, \tau, x) : (\sigma, x, w) \in R \text{ and } \mathcal{A}(\pi) = 1\right].$$

4 Common Reference String

We will now describe how to generate the common reference string for our NIZK argument. The common reference string consists of multiple parts, which each serve a particular purpose that will be explained later in the paper. When simulating the NIZK argument, we will be using a common reference string with an identical distribution, so we describe how to generate the simulation trapdoor as well.

Define the multivariate monomial P of degree n and the n multivariate monomials P_j of degree $n-1$ as follows

$$P(x_1, \ldots, x_n) = \prod_{i \in [n]} x_i \qquad \forall j \in [n] : P_j(x_1, \ldots, x_n) = \prod_{i \in [n] \setminus \{j\}} x_i.$$

Common Reference String Generation:

1. Generate a bilinear group $(p, G, G_T, e, g) \leftarrow \mathcal{G}(1^\lambda)$ and set $gk = (p, G, G_T, e)$.
2. Choose at random $x_1, \ldots, x_n \leftarrow \mathbb{Z}_p^*$ and set $ck = (g, g^{x_1}, \ldots, g^{x_n})$.
3. Choose at random $\alpha \leftarrow \mathbb{Z}_p^*$ and set $\sigma_{\text{know}} = (g^\alpha, g^{\alpha x_1}, \ldots, g^{\alpha x_n})$.
4. Set $\sigma_{\text{prod}} = (\{g^{x_i x_j}\}_{i,j \in [n], i \neq j})$.
5. Choose at random $y_1, \ldots, y_n \leftarrow \mathbb{Z}_p^*$ and set $\sigma_{\text{perm}} = (\{g^{P_i(\boldsymbol{x}) y_j}\}_{i,j \in [n]}, \{g^{x_i P_j(\boldsymbol{x}) y_k}\}_{i,j,k \in [n], i \neq j})$.
6. For $k = 1$ to n:
   Choose at random $z_{1k}, \ldots, z_{nk} \leftarrow \mathbb{Z}_p^*$ and set $\sigma_{\text{rot},k} = (\{g^{z_{ik}}\}_{i \in [n]}, \{g^{x_j z_{ik}}\}_{i,j \in [n], j \neq k})$.

The common reference string is $\sigma = (gk, ck, \sigma_{\text{know}}, \sigma_{\text{prod}}, \sigma_{\text{perm}}, \{\sigma_{\text{rot},k}\}_{k \in [n]})$. The simulation trapdoor is $tk = (x_1, \ldots, x_n)$.

VERIFIABILITY OF THE COMMON REFERENCE STRING AND ZAPS. The common reference string described above has a particular mathematical structure and we do not know of an extraction procedure that can generate it from a public string of *random* bits. However, provided one can verify that (p, G, G_T, e, g) does describe a bilinear group, it is also possible to verify that σ is a well-formed common reference string. First, one checks that all group elements in σ are non-trivial. This demonstrates that all the secret exponents

x_i, α, y_i, z_{ik} are non-zero. Next, one uses the pairing operation to verify the structure of the common reference string. For instance, $e(g, g^{\alpha x_i}) = e(g^{\alpha}, g^{x_i})$ and $e(g, g^{x_i x_j}) = e(g^{x_i}, g^{x_j})$. This verifiability of the common reference string gives us 2-move arguments with perfect witness-indistinguishability, also known as Zaps [DN00]. The verifier in the first move sends a common reference string and the prover then can give many publicly verifiable arguments (second moves) for different statements using the same common reference string.

4.1 Generic Group Model

We will provide heuristic evidence for the soundness of our NIZK argument by proving that it cannot be broken by an adversary that only uses generic bilinear group operations. For this purpose, let us define a generic bilinear group model, where the adversary learns what the bilinear group is but instead of seeing any group elements it only sees a random representation of the group elements. More precisely, we choose random permutations $[\cdot] : G \to G$ and $[\cdot]_T : G_T \to G_T$ and instead of seeing for instance g^{x_i} and $e(g, g)$, the adversary will only see $[g^{x_i}]$ and $[e(g, g)]_T$. The adversary gets access to an oracle that enables it to compute the group operations and the pairings even though it only sees random representations of those elements. The oracle $\mathcal{O}$ works as follows:

Initialization:

- Pick two random permutations $[\cdot] : G \to G$ and $[\cdot]_T : G \to G_T$.
- Pick at random $x_1, \ldots, x_n, \alpha, y_1, \ldots, y_n, z_{11}, \ldots, z_{nn} \leftarrow \mathbb{Z}_p^*$.
- Define

$$\sigma_{\text{know}} = ([g^{\alpha}], \{[g^{\alpha x_i}]\}) \qquad \sigma_{\text{prod}} = (\{[g^{x_i x_j}]\}_{i \neq j}) \qquad \sigma_{\text{perm}} = (\{[g^{P_i(\boldsymbol{x}) y_j}]\}, \{[g^{x_i P_j(\boldsymbol{x}) y_k}]\}_{i \neq j})$$
$$ck = ([g], \{[g^{x_i}]\}) \qquad \sigma_{\text{rot},1} = (\{[g^{z_{i1}}]\}, \{[g^{x_j z_{i1}}]\}_{j \neq 1}) \;\; \ldots \;\; \sigma_{\text{rot},n} = (\{[g^{z_{in}}]\}\{[g^{x_j z_{in}}]\}_{j \neq n}).$$

- Give $\sigma = (ck, \sigma_{\text{know}}, \sigma_{\text{prod}}, \sigma_{\text{perm}}, \sigma_{\text{rot},1}, \ldots, \sigma_{\text{rot},n})$ as initial input to the adversary.

Operations:

- On $(\text{mult}, [a], [b])$ return $[a + b]$.
- On $(\text{mult}_T, [a]_T, [b]_T)$ return $[a + b]_T$.
- On $(\text{pair}, [a], [b])$ return $[ab]_T$.

The adversary can use the generic group operations to compute a random group element in $O(\log p)$ steps by picking a random exponent and using the oracle to go through the square-and-multiply algorithm to get $[g^r]$ starting from $[g]$. Since the adversary gets no advantage of not knowing the discrete logarithm of a group element, we can without loss of generality assume the adversary only queries the oracle with encodings $[a], [b]$ or $[a]_T, [b]_T$ that has been given to it from the oracle at some point.

Theorem 1. *In the generic group model it is infeasible for the adversary to find a non-trivial linear relation between the group elements given in the common reference string using only a polynomial number of queries to the generic group oracle.*

Proof. We want to prove

$$\Pr\Big[(p, G, G_T, e, g) \leftarrow \mathcal{G}(1^{\lambda}); (r_1, r_{x_1}, \ldots, r_{x_{n-1} z_{nn}}) \leftarrow \mathcal{A}^{\mathcal{O}}(p, G, G_T, e, g) :$$
$$(r_1, \ldots, r_{x_{n-1} z_{nn}}) \neq \{\mathbf{0}\} \quad \text{and} \quad g^{r_1} \prod_{i=1}^{n} (g^{x_i})^{r_{x_i}} \cdots \prod_{i \in [n]} \prod_{j \in [n] \setminus \{n\}} (g^{x_j z_{in}})^{r_{x_j z_{in}}} = 1\Big] \approx 0$$

Consider an algorithm $\mathcal{B}$ getting (p, G, G_T, e, g) and the generic σ from $\mathcal{O}$ as input that runs a copy of $\mathcal{A}(p, G, G_T, e, p)$ simulating all oracle queries made by $\mathcal{A}$. In the end, $\mathcal{A}$ makes an output $(r_1, \ldots, r_{x_{n-1}z_{nn}})$ and $\mathcal{B}$ uses this as its output. $\mathcal{B}$ keeps two lists $L = \{(p_i, \xi_i)\}$ and $L_T = \{(p_{i,T}, \xi_{i,T})\}$ containing pairs of the form $(p_i, \xi_i) \in \mathbb{Z}_p[X_1, \ldots, X_n, A, Y_1, \ldots, Y_n, Z_{11}, \ldots, Z_{nn}] \times \mathbb{Z}_p$, where $\xi_i, \xi_{i,T}$ are values the oracle has given to $\mathcal{A}$.

At first the lists are initialized as follows $L = \{(1, \xi_1), (X_1, \xi_2), (X_2, \xi_3), \ldots, (X_{n-1}Z_{nn}, \xi_{2n^3+n^2+n+2})\}$ and $L_T = \emptyset$, where $\xi_1, \ldots, \xi_{2n^3+n^2+n+2}$ are the values $\mathcal{B}$ got from the oracle. The list L thus starts out by containing the polynomials used in the exponents when generating a common reference string and the matching encodings of those group elements. $\mathcal{B}$ aborts the simulation if there are any collisions, i.e., $\xi_i = \xi_j$ for some $i \neq j$.

Upon input $(\mathrm{mult}, \xi_i, \xi_j)$ it looks up (p_i, ξ_i) and (p_j, ξ_j) in L. If there is an entry $(p_i + p_j, \xi) \in L$ it returns ξ. Otherwise, it picks $\xi \leftarrow G \setminus \{\xi_1, \ldots\}$, appends $(p_i + p_j, \xi)$ to L and returns ξ. Inputs of the form $(\mathrm{mult}_T, \xi_{i,T}, \xi_{j,T})$ are treated similarly using the list L_T. On input $(\mathrm{pair}, \xi_i, \xi_j)$ it looks up (p_i, ξ_i) and (p_j, ξ_j) in L. If there is an entry $(p_i p_j, \xi_T)$ in L_T it returns ξ_T. Otherwise, it picks $\xi_T \leftarrow \mathcal{G}_T \setminus \{\xi_{1,T}, \ldots\}$, appends $(p_i p_j, \xi_T)$ to L_T and returns ξ_T.

Let us argue that B's simulation of the oracle is statistically indistinguishable from $\mathcal{O}$. We can imagine $\mathcal{O}$ keeping track of polynomials p_i and answers ξ_i corresponding to the oracle queries just as $\mathcal{B}$ does. It is possible with the concrete choices of $x_1, \ldots, z_{nn} \leftarrow \mathbb{Z}_p^*$ made by $\mathcal{O}$ that we have some i, j so $p_i(x_1, \ldots, z_{nn}) = p_j(x_1, \ldots, z_{nn})$ and hence $\mathcal{O}$ would return $\xi_i = [g^{p_i(x_1, \ldots, z_{nn})}] = [g^{p_j(x_1, \ldots, z_{nn})}] = \xi_j$, whereas $\mathcal{B}$ will always use $\xi_i \neq \xi_j$. A similar problem can occur for G_T. The maximal degree of these polynomials is $n+1$ in L and $2n+2$ in L_T, so by the Schwartz-Zippel lemma each pair of polynomials (p_i, p_j) has no more than negligible probability $\frac{2n}{p-1}$ of giving a collision. With a total of at most a polynomial number of queries there is negligible chance of encountering a collision for any pair of oracle answers. Since $\mathcal{O}$ uses a random permutation to disguise the actual group elements, the answers it returns look random just like $\mathcal{B}$'s answers, as long as no collision happens.

In the generic group model the common reference string σ contains evaluations of monomials in $\mathbb{Z}_p[X_1, \ldots, X_n, A, Y_1, \ldots, Y_n, Z_{11}, \ldots, Z_{nn}]$. It is easy to see that all the monomials are different, so there is no non-trivial linear combination $r + r_1 x_1 + \cdots r_{n-1,n,n} x_{n-1} z_{nn}$ that gives us the zero-polynomial in $\mathbb{Z}_p[X_1, \ldots, Z_{nn}]$. The adversary must therefore find a linear combination for the concrete evaluation of the monomials that the oracle has chosen, when it picked $x_1, \ldots, z_{nn}$ at random. The probability of outputting a non-trivial linear combination $(r_1, r_{x_1}, \ldots, r_{x_{n-1}z_{nn}})$ of the group elements underlying σ is a negligible $\frac{n+1}{p-1}$ if we have no information at all about the group elements. By the Schwartz-Zippel lemma there is negligible probability for the oracle outputting a σ with a collision and since the group elements are encoded using a random permutation they give no information away except the fact that there is no collision. This means $\mathcal{B}$ has negligible probability of outputting a non-trivial linear combination. Further, since $\mathcal{A}$ without loss of generality can check in a polynomial number of oracle queries whether the linear combination is valid and it cannot distinguish the simulation made by $\mathcal{B}$ from the oracle, we conclude that $\mathcal{A}$ also has negligible probability of outputting a non-trivial linear combination of the group elements underlying the common reference string. $\square$

5 Commitment

It is well-known that the Pedersen commitment [Ped91] can be extended to commit to many values. Let G be a cyclic group of prime order p and $g, g_1, \ldots, g_n$ be random generators of G. To commit to messages $a_1, \ldots, a_n$ using randomness $r \leftarrow \mathbb{Z}_p$ compute the commitment $c = g^r \prod_{i \in [n]} g_i^{a_i}$. We call $(r, a_1, \ldots, a_n)$

an opening of c. The commitment is computationally binding assuming the discrete logarithm problem is hard, i.e., it is infeasible to find two different openings of a commitment c. The commitment is perfectly hiding, i.e., no matter the choice of $a_1, \ldots, a_n$ the randomizer r ensures that the commitment is a uniformly distributed group element. Further, the commitment scheme is trapdoor; if the discrete logarithms of $g_1, \ldots, g_n$ with respect to g are known, it is possible to create a commitment and later open it to any set of values.

Common reference string: σ including commitment key $ck = (g, g_1, \ldots, g_n)$.
Commitment: To commit to $a_1, \ldots, a_n$ sample randomness $r \leftarrow \mathbb{Z}_p$ and compute the commitment

$$c = g^r \prod_{i \in [n]} g_i^{a_i}.$$

Trapdoor commitment: To make a trapdoor commitment sample randomness $t \leftarrow \mathbb{Z}_p$ and compute the commitment $c = g^t$ and the trapdoor randomness t.
Trapdoor opening: The trapdoor opening algorithm takes as input a commitment $c \in G$, messages $a_1, \ldots, a_n \in \mathbb{Z}_p$, trapdoor randomness $t \in \mathbb{Z}_P$ and trapdoor key $tk = (x_1, \ldots, x_n)$ so $c = g^t$ and for all $i \in [n] : g_i = g^{x_i}$. It returns randomizer $r = t - \sum_{i \in [n]} a_i x_i$, so we have $c = g^r \prod_{i \in [n]} g_i^{a_i}$.

Theorem 2. *The commitment scheme is perfectly hiding and perfectly trapdoor.*

Proof. It is easy to see the commitment scheme is perfectly hiding, since no matter what the values $a_1, \ldots, a_n$ are, the commitment is uniformly distributed in G. Consider the probability distributions of (c, r) for respectively commitments and trapdoor commitments that are trapdoor opened conditioned on messages $a_1, \ldots, a_n \in \mathbb{Z}_p$. In the first case r is uniformly random and given r the commitment $c = g^r \prod_{i \in [n]} g_i^{a_i}$ is uniquely determined conditioned on $a_1, \ldots, a_n$. In the second case, t is uniformly random and conditioned on $a_1, \ldots, a_n$ this makes $r = t - \sum_{i \in [n]} a_i x_i$ uniformly random. The trapdoor opening satisfies $c = g^{t - \sum_{i \in [n]} a_i x_i} \prod_{i \in [n]} g_i^{a_i}$. In both cases, we therefore have the same probability distribution for (c, r) and conclude that the commitment scheme is perfectly trapdoor. □

It is well-known that the Pedersen commitment scheme for multiple messages is computationally binding assuming the discrete logarithm problem is hard. Here the common reference string contains more information about the discrete logarithms $x_1, \ldots, x_n$ but as the following corollary to Theorem 1 shows the commitment scheme is still binding in the generic group model.

Corollary 1. *In the generic group model it is infeasible to find two different openings* $(r, a_1, \ldots, a_n) \neq (s, b_1, \ldots, b_n)$ *so*

$$g^r \prod_{i \in [n]} g_i^{a_i} = g^s \prod_{i \in [n]} g_i^{b_i}.$$

EFFICIENCY. We summarize the cost of commitments in Table 2. The receiver of a commitment just checks whether it belongs to G, which we assume carries negligible cost compared to other parts of the NIZK argument that will be presented in this paper. We note as a special case, the computation involved in making a commitment is cheaper when the values are bits.

6 Argument of Knowledge

We will give an argument of knowledge of opening of a commitment. This argument can be seen as a natural extension of arguments of knowledge for standard Pedersen commitments with a single message based on a knowledge of exponent assumption [BP04,AF07].

Size of ck	Commitment size	Computation	Computation for $a_i \in \{0,1\}$	Receiver computation
$n+1$ G	1 G	$n+1$ E	1 E + n M	-

Table 2. Cost of commitments.

Common reference string: σ including $ck = (g, g_1, \ldots, g_n)$ and $\sigma_{\text{know}} = (g^{\alpha}, g^{\alpha x_1}, \ldots, g^{\alpha x_n})$.
Statement: Commitment c.
Prover's witness: Opening $a_1, \ldots, a_n, r \in \mathbb{Z}_p$ so $c = g^r \prod_{i\in[n]} g_i^{a_i}$.
Argument: Compute the argument as

$$\pi = (g^{\alpha})^r \prod_{i\in[n]} (g^{\alpha x_i})^{a_i}.$$

Verification: Output 1 if and only if

$$e(c, g^{\alpha}) = e(g, \pi).$$

Theorem 3. *The non-interactive argument described above has perfect completeness and perfect witness-indistinguishability.*

Proof. To argue perfect completeness, observe $\pi = c^{\alpha}$ giving us $e(c, g^{\alpha}) = e(g, \pi)$. Perfect witness-indistinguishability follows from g^{α} being a generator for G, so there is only one acceptable argument π. By perfect completeness, any opening of the commitment will result in the same argument π and we therefore have perfect witness-indistinguishability. □

Theorem 4. *In the generic group model it is infeasible to construct a commitment c and a valid argument of knowledge π without knowing an opening $r, a_1, \ldots, a_n$ of the commitment.*

Proof. We will show that there is an polynomial time extractor E that takes the adversary's input (p, G, G_T, e, g) and the generic common reference string σ, the adversary's outputs $[c], [\pi]$, and a list of inputs and outputs to the generic bilinear group oracle, which with overwhelming probability outputs an opening $r, a_1, \ldots, a_n$ of c in case π is a valid argument of knowledge for c.

Let us first consider how $\mathcal{A}$ can be successful in producing encodings $[c], [\pi]$ of a commitment and a valid argument of knowledge. Suppose one or both of $[c]$ and $[\pi]$ are not part of the generic common reference string or output by the generic group oracle. In that case, since $[\cdot]$ is a random permutation over G and g, g^{α} are generators for G there is negligible probability of $e(c, g^{\alpha}) = e(g, \pi)$. We can therefore from now on assume $[c], [\pi]$ have been generated by the generic group oracle.

Now, let us as in the proof of Theorem 1 keep track of queries made by $\mathcal{A}$ and the corresponding polynomials. The extractor maintains two lists $L = \{(p_i, \xi_i)\}$ and $L_T = \{(p_{i,T}, \xi_{i,T})\}$ containing pairs of the form $(p_i, \xi_i) \in \mathbb{Z}_p[X_1, \ldots, X_n, A, Y_1, \ldots, Y_n, Z_{11}, \ldots, Z_{nn}] \times \mathbb{Z}_p$, where $\xi_i, \xi_{i,T}$ are values the oracle has given to $\mathcal{A}$. At first the lists are initialized as follows $L = \{(1, \xi_1), (X_1, \xi_2), (X_2, \xi_3), \ldots, (X_{n-1}Z_{nn}, \xi_{2n^3+n^2+n+2})\}$ and $L_T = \emptyset$, where $\xi_1, \ldots, \xi_{2n^3+n^2+n+2}$ are the values from the generic common reference string. The list L thus starts out by containing the polynomials used in the exponents when generating a common reference string and matching random encodings of those group elements. Upon input $(\text{mult}, \xi_i, \xi_j)$ giving answer ξ the extractor appends $(p_i + p_j, \xi)$ to L unless it has already been stored in L. Inputs of the form $(\text{mult}_T, \xi_{i,T}, \xi_{j,T})$ are treated similarly using the list L_T. On input $(\text{pair}, \xi_i, \xi_j)$ giving response ξ_T it appends $(p_i p_j, \xi_T)$ to L_T unless such a tuple already exists.

Consider now the polynomials $c(X_1, \ldots, Z_{nn})$ and $\pi(X_1, \ldots, Z_{nn})$ for the adversary's output $[c], [\pi]$. If we have $\pi(X_1, \ldots, Z_{nn}) \neq Ac(X_1, \ldots, Z_{nn})$ we will argue as in Theorem 1 that there is negligible chance of $\mathcal{A}$ being successful. Since the generic oracle uses a random permutation to encode the group elements, we can simulate it by picking random distinct values for each new polynomial generated during the queries to the oracle. By the Schwartz-Zippel lemma, there is negligible probability of having a collision, i.e., two distinct polynomials that evaluate to the same group elements for the oracle's concrete choice of $x_1, \ldots, z_{nn}$. The simulation is perfect conditioned on no such collision happening, and as the adversary in the simulation gets no information about $x_1, \ldots, z_{nn}$ in the simulation there is negligible chance over the choice of $x_1, \ldots, z_{nn} \leftarrow \mathbb{Z}_p^*$ for the event $\pi(x_1, \ldots, z_{nn}) = \alpha c(x_1, \ldots, z_{nn})$.

Remaining is the possibility that $\pi(X_1, \ldots, Z_{nn}) = Ac(X_1, \ldots, Z_{nn})$. Since the two polynomials belong to L, which can only grow by adding polynomials in L when the adversary makes a $(\text{mult}, \xi_i, \xi_j)$ query to the oracle, we have

$$c(X_1, \ldots, Z_{nn}) = c_1 + c_{x_1}X_1 + \cdots + c_\alpha A + c_{\alpha x_1}AX_1 + \cdots + c_{x_1x_2}X_1X_2 + \cdots c_{x_{n-1}z_{nn}}X_{n-1}Z_{nn}$$
$$\pi(X_1, \ldots, Z_{nn}) = \pi_1 + \pi_{x_1}X_1 + \cdots + \pi_\alpha A + \pi_{\alpha x_1}AX_1 + \cdots + \pi_{x_1x_2}X_1X_2 + \cdots \pi_{x_{n-1}z_{nn}}X_{n-1}Z_{nn},$$

for known coefficients $c_1, \ldots, c_{x_{n-1}z_{nn}}, \pi_1, \ldots, \pi_{x_{n-1}z_{nn}} \in \mathbb{Z}_p$.

The equality $\pi(X_1, \ldots, Z_{nn}) = Ac(X_1, \ldots, Z_{nn})$ implies that we can simplify

$$\pi(X_1, \ldots, Z_{nn}) = \pi_\alpha A + \pi_{\alpha x_1}AX_1 + \cdots + \pi_{\alpha x_n}AX_n,$$

since all the other parts are distinct monomials with degree 0 in A. This means we have

$$c(X_1, \ldots, Z_{nn}) = \pi_\alpha A + \pi_{\alpha x_1}X_1 + \cdots \pi_{\alpha x_n}X_n.$$

By Theorem 1 this is the only linear combination giving $c(X_1, \ldots, Z_{nn})$, so $r = c_1 = \pi_\alpha, a_1 = c_{x_1} = \pi_{\alpha x_1}, \ldots, a_n = c_{x_n} = \pi_{\alpha x_n}$ gives us the desired opening of c. □

EFFICIENCY. We summarize the cost of arguments of knowledge in Table 3. If there are many arguments of knowledge it is possible to use batch-verification techniques to lower the cost of verification. Consider for instance the case, where we have N commitments and arguments of knowledge $c_1, \pi_1, \ldots, c_N, \pi_N$. The verifier picks $t_1, \ldots, t_n \leftarrow [2^\ell]$ at random for some security parameter $\ell < \log p$ and verifies

$$e(\prod_{i \in [n]} c_i^{t_i}, g^\alpha) = \prod_{i \in [n]} e(c_i, g^\alpha)^{t_i} = \prod_{i \in [n]} e(g, \pi_i)^{t_i} = e(g, \prod_{i \in [n]} \pi_i^{t_i}),$$

which has at most chance $2^{-\ell}$ of being true unless $\forall i \in [n] : e(c_i, g^\alpha) = e(g, \pi_i)$. The last column in the table lists the costs of batch-verifying N arguments of knowledge.

Size of σ_{know}	Argument size	Prover comp.	Prover comp. $\{0,1\}$	Verifier comp.	Verifier comp. N
$n+1$ G	1 G	$n+1$ E	1 E + n M	2 P	2 P + $2N$ E

Table 3. Cost of arguments of knowledge.

7 Products of Committed Values

Consider three commitments c, d and v to values $a_1, \ldots, a_n, b_1, \ldots, b_n$ and $u_1, \ldots, u_n$. We will give a non-interactive perfectly witness-indistinguishable argument for the committed $u_1, \ldots, u_n$ being the products $a_1b_1, \ldots, a_nb_n$.

Common reference string: σ including $ck = (g, g_1, \ldots, g_n)$ and $\sigma_{\text{prod}} = (\{g^{x_ix_j}\}_{i,j\in[n],i\neq j})$.
Statement: Commitments $c, d, v \in G$.
Prover's witness: Openings $r, a_1, \ldots, a_n$ and $s, b_1, \ldots, b_n$ and $t, u_1, \ldots, u_n$ so

$$c = g^r \prod_{i\in[n]} g_i^{a_i} \quad \text{and} \quad d = g^s \prod_{i\in[n]} g_i^{b_i} \quad \text{and} \quad v = g^t \prod_{i\in[n]} g_i^{u_i} \quad \text{and} \quad \forall i \in [n] : u_i = a_ib_i.$$

Argument: Compute the argument as

$$\pi = g^{rs} \prod_{i\in[n]} \left((g_i)^{a_is+b_ir-t} \prod_{j\in[n]\setminus\{i\}} (g^{x_ix_j})^{a_ib_j-u_i} \right).$$

Verification: Output 1 if and only if

$$e(c, d) = e(v, \prod_{j\in[n]} g_j)e(g, \pi).$$

Theorem 5. *The non-interactive argument described above has perfect completeness and perfect witness-indistinguishability.*

Proof. To argue perfect completeness, consider the three pairings the verifier has to compute.

$$\begin{aligned} e(c, d) &= e(g^{r+\sum_i a_ix_i}, g^{s+\sum_j b_jx_j}) = e(g, g)^{\sum_i a_ib_ix_i^2} e(g, g)^{rs+s\sum_i a_ix_i+r\sum_j b_jx_j+\sum_{i\neq j} a_ib_jx_ix_j}. \\ e(v, \prod_{j\in[n]} g_j) &= e(g^{t+\sum_i u_ix_i}, g^{\sum_j x_j}) = e(g, g)^{\sum_i u_ix_i^2} e(g, g)^{t\sum_j x_j+\sum_{i\neq j} u_ix_ix_j}. \\ e(g, \pi) &= e(g, g^{rs} \prod_{i\in[n]} [(g^{x_i})^{a_is+b_ir-t} \prod_{j\in[n]\setminus\{i\}} (g^{x_ix_j})^{a_ib_j-u_i}]) \\ &= e(g, g)^{rs+\sum_i(a_ir+b_is-t)x_i+\sum_{i\neq j}(a_ib_j-u_i)x_ix_j} \end{aligned}$$

Since $u_i = a_ib_i$ we see that $\sum_{i\in[n]} a_ib_ix_i^2 = \sum_{i\in[n]} u_ix_i^2$. Looking at the three pairings above we then get

$$e(c, d) = e(v, \prod_{j\in[n]} g_j)e(g, \pi)$$

so the verification accepts the argument π.

Perfect witness-indistinguishability follows from g being a generator of G, since there is exactly one unique argument π that will make the verification accept. By the perfect completeness, all valid witnesses give an accepting argument and therefore for fixed c, d, u all openings with $u_i = a_ib_i$ result in the same argument π. □

Theorem 6. *In the generic group model it is infeasible to find three commitments with their respective openings and a valid argument unless indeed the third set of values contains the entry-wise product of the first two sets.*

Proof. We want to prove

$$\Pr\Big[(p,G,G_T,e,g)\leftarrow\mathcal{G}(1^\lambda);(r,a_1,\ldots,s,b_1,\ldots,t,u_1,\ldots,[\pi])\leftarrow\mathcal{A}^{\mathcal{O}}(p,G,G_T,e,g):$$
$$e(c,d)=e(v,\prod_{j\in[n]}g_j)e(g,\pi)\quad\text{and}\quad\exists i\in[n]:u_i\neq a_ib_i\Big]\approx 0,$$

where we define $c=g^r\prod_{i\in[n]}g_i^{a_i}$, $d=g^s\prod_{i\in[n]}g_i^{b_i}$ and $v=g^t\prod_{i\in[n]}g_i^{u_i}$.

Let us first consider how $\mathcal{A}$ can be successful in producing the encoding $[\pi]$ of a valid argument. Suppose $[\pi]$ is not part of the generic common reference string or output by the generic group oracle. In that case, since $[\cdot]$ is a random permutation over G and g is a generator there is negligible probability of $e(c,d)=e(v,\prod_{j\in[n]}g_j)e(g,\pi)$. We can therefore from now on assume $[\pi]$ has been generated by the generic group oracle at some point.

Now, let us as in the proof of Theorem 1 keep track of queries made by $\mathcal{A}$ and the corresponding polynomials. We maintain two lists $L=\{(p_i,\xi_i)\}$ and $L_T=\{(p_{i,T},\xi_{i,T})\}$ containing pairs of the form $(p_i,\xi_i)\in\mathbb{Z}_p[X_1,\ldots,X_n,A,Y_1,\ldots,Y_n,Z_{11},\ldots,Z_{nn}]\times\mathbb{Z}_p$, where $\xi_i,\xi_{i,T}$ are values the oracle has given to $\mathcal{A}$. At first the lists are initialized as follows $L=\{(1,\xi_1),(X_1,\xi_2),(X_2,\xi_3),\ldots,(X_{n-1}Z_{nn},\xi_{2n^3+n^2+n+2})\}$ and $L_T=\emptyset$, where $\xi_1,\ldots,\xi_{2n^3+n^2+n+2}$ are the values from the generic common reference string. The list L thus starts out by containing the polynomials used in the exponents when generating a common reference string and matching random encodings of those group elements. Upon input $(\mathrm{mult},\xi_i,\xi_j)$ giving answer ξ the extractor appends (p_i+p_j,ξ) to L unless it has already been stored in L. Inputs of the form $(\mathrm{mult}_T,\xi_{i,T},\xi_{j,T})$ are treated similarly using the list L_T. On input $(\mathrm{pair},\xi_i,\xi_j)$ giving response ξ_T it appends (p_ip_j,ξ_T) to L_T unless such a tuple already exists.

Consider now the polynomial $\pi(X_1,\ldots,Z_{nn})$ for the adversary's output $[\pi]$. If we have

$$(r+\sum_{i\in[n]}a_iX_i)(s+\sum_{j\in[n]}b_jX_j)\neq(t+\sum_{i\in[n]}u_iX_i)\sum_{j\in[n]}X_j+\pi(X_1,\ldots,Z_{nn})$$

we can argue as in Theorem 1 that there is negligible chance of $\mathcal{A}$ being successful, i.e., there is negligible chance that $e(c,d)=e(v,\prod_{j\in[n]}g_j)e(g,\pi(x_1,\ldots,z_{nn}))$ for the concrete values of $x_1,\ldots,z_{nn}$ chosen by the oracle. Since the generic oracle uses a random permutation to encode the group elements, we can simulate it by picking random distinct values for each new polynomial generated during the queries to the oracle. By the Schwartz-Zippel lemma, there is negligible probability of having a collision, i.e., two distinct polynomials that evaluate to the same group elements for the oracle's concrete choice of $x_1,\ldots,z_{nn}$. The simulation is perfect conditioned on no such collision happening, and as the adversary in the simulation gets no information about $x_1,\ldots,z_{nn}$ in the simulation there is negligible chance over the choice of $x_1,\ldots,z_{nn}\leftarrow\mathbb{Z}_p^*$ for the event $e(c,d)=e(v,\prod_{j\in[n]}g_j)e(g,\pi(x_1,\ldots,z_{nn}))$.

Remaining is the possibility that $(r+\sum_{i\in[n]}a_iX_i)(s+\sum_{j\in[n]}b_jX_j)=(t+\sum_{i\in[n]}u_iX_i)\sum_{j\in[n]}X_j+\pi(X_1,\ldots,Z_{nn})$. Since $\pi(X_1,\ldots,Z_{nn})$ belongs to L, which can only grow by adding polynomials in L when the adversary makes a $(\mathrm{mult},\xi_i,\xi_j)$ query to the oracle, we have

$$\pi(X_1,\ldots,Z_{nn})=\pi_1+\pi_{x_1}X_1+\cdots+\pi_\alpha A+\pi_{\alpha x_1}AX_1+\cdots+\pi_{x_1x_2}X_1X_2+\cdots\pi_{x_{n-1}z_{nn}}X_{n-1}Z_{nn},$$

for known coefficients $\pi_1, \dots, \pi_{x_{n-1}z_{nn}} \in \mathbb{Z}_p$.

Expanding $(r + \sum_{i\in[n]} a_i X_i)(s + \sum_{j\in[n]} b_j X_j) = (t + \sum_{i\in[n]} u_i X_i) \sum_{j\in[n]} X_j + \pi(X_1, \dots, Z_{nn})$ on both sides gives us

$$\sum_{i\in[n]} a_i b_i X_i^2 + \left[rs + s \sum_{i\in[n]} a_i X_i + r \sum_{j\in[n]} b_j X_j + \sum_{i\in[n]} \sum_{j\in[n]\setminus\{i\}} a_i b_j X_i X_j \right]$$
$$= \sum_{i\in[n]} u_i X_i^2 + \left[t \sum_{j\in[n]} X_j + \sum_{i\in[n]} \sum_{j\in[n]\setminus\{i\}} u_i X_i X_j + \pi(X_1, \dots, Z_{nn}) \right].$$

Since $\pi(X_1, \dots, Z_{nn})$ has no X_i^2 elements, neither do any of the parts within the brackets. We conclude that for the two polynomials to be equal, we must have $u_i = a_i b_i$ for all $i \in [n]$. $\square$

EFFICIENCY. Table 4 summarizes the cost of product arguments. As is the case with arguments of knowledge, batch-verification techniques may reduce the cost of verifying many statements and product arguments.

Size of σ_{prod}	Argument size	Prover comp.	Prover comp. $\{0,1\}$	Verifier comp.	Verifier comp. N
$n^2 - n$ G	1 G	$n^2 + 1$ E	$n + 1$ E + $n^2 - n$ M	3 P	N P + $2N$ E

Table 4. Cost of product arguments.

8 Permutation of Committed Values within a Commitment

Consider two commitments c and d to values $a_1, \dots, a_n$ and $b_1, \dots, b_n$. We will give a non-interactive perfectly witness-indistinguishable argument for the committed $b_1, \dots, b_n$ being a publicly known permutation of $a_1, \dots, a_n$.

Common reference string: σ including $\sigma_{\text{perm}} = (\{g^{P_i(\boldsymbol{x})y_j}\}_{i,j\in[n]}, \{g^{x_i P_j(\boldsymbol{x})y_k}\}_{i,j,k\in[n],i\neq j})$.
Statement: Commitments $c, d \in G$ and permutation $\rho \in S_n$.
Prover's witness: Openings $r, a_1, \dots, a_n \in \mathbb{Z}_p$ and $s, b_1, \dots, b_n \in \mathbb{Z}_p$ so

$$c = g^r \prod_{i\in[n]} g_i^{a_i} \quad \text{and} \quad d = g^s \prod_{j\in[n]} g_i^{b_j} \quad \text{and} \quad \forall i \in [n] : b_i = a_{\rho(i)}.$$

Argument: Compute the argument as

$$\pi = \left(\prod_{i\in[n]} g^{P_i(\boldsymbol{x})y_i} \right)^r \left(\prod_{i\in[n]} g^{P_i(\boldsymbol{x})y_{\rho(i)}} \right)^{-s} \prod_{i\in[n]} \left[\left(\prod_{j\in[n]\setminus\{i\}} g^{x_i P_j(\boldsymbol{x})y_j} \right)^{a_i} \left(\prod_{j\in[n]\setminus\{i\}} g^{x_i P_j(\boldsymbol{x})y_{\rho(j)}} \right)^{-b_i} \right].$$

Verification: Output 1 if and only if

$$e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}}) e(g, \pi).$$

Theorem 7. *The non-interactive argument described above has perfect completeness and perfect witness-indistinguishability.*

Proof. To argue perfect completeness, consider the three pairings the verifier has to compute.

$$e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(g^{r+\sum_i a_i x_i}, g^{\sum_j P_j(\boldsymbol{x})y_j}) = e(g,g)^{\sum_i a_i y_i P(\boldsymbol{x})} e(g,g)^{r\sum_j P_j(\boldsymbol{x})y_j + \sum_{i\neq j} a_i x_i P_j(\boldsymbol{x}) y_j}.$$

$$\begin{aligned} e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}}) &= e(g^{s+\sum_i b_i x_i}, g^{\sum_j P_j(\boldsymbol{x})y_{\rho(j)}}) \\ &= e(g,g)^{\sum_i b_i y_{\rho(i)} P(\boldsymbol{x})} e(g,g)^{s\sum_j P_j(\boldsymbol{x})y_{\rho(j)} + \sum_{i\neq j} b_i x_i P_j(\boldsymbol{x}) y_{\rho(j)}}. \\ e(g,\pi) &= e(g, (\prod_{i\in[n]} g^{P_i(\boldsymbol{x})y_i})^r (\prod_{i\in[n]} g^{P_i(\boldsymbol{x})y_{\rho(i)}})^{-s} \prod_{i\in[n]} [(\prod_{j\in[n]\setminus\{i\}} g^{x_i P_j(\boldsymbol{x})y_j})^{a_i} (\prod_{j\in[n]\setminus\{i\}} g^{x_i P_j(\boldsymbol{x})y_{\rho(j)}})^{-b_i}] \\ &= e(g,g)^{r\sum_i P_i(\boldsymbol{x})y_i - s\sum_i P_i(\boldsymbol{x})y_{\rho(i)} + \sum_{i\neq j} a_i x_i P_j(\boldsymbol{x})y_j - \sum_{i\neq j} b_i x_i P_j(\boldsymbol{x}) y_{\rho(j)}}. \end{aligned}$$

Since $b_i = a_{\rho(i)}$ we have $\sum_{i\in[n]} a_i y_i = \sum_{i\in[n]} b_i y_{\rho(i)}$. Looking at the three pairings above we then get

$$e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}}) e(g,\pi)$$

so the verification accepts the argument π.

Since g is a generator for G, there is exactly one argument π that will make the verification accept. By the perfect completeness, all valid openings with $b_i = a_{\rho(i)}$ give an accepting argument and therefore all valid witnesses result in the same argument. $\square$

Theorem 8. *In the generic group model it is infeasible to find a permutation, two commitments with their respective openings, and a valid argument unless indeed the openings obey the permutation.*

Proof. We want to prove

$$\Pr\Big[(p,G,G_T,e,g) \leftarrow \mathcal{G}(1^\lambda); (\rho, r, a_1, \ldots, a_n, s, b_1, \ldots, b_n, [\pi]) \leftarrow \mathcal{A}^{\mathcal{O}}(p,G,G_T,e,g) : \rho \in S_n \text{ and}$$
$$e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}}) e(g,\pi) \quad \text{and} \quad \exists i \in [n] : b_i \neq a_{\rho(i)}\Big] \approx 0,$$

where we define $c = g^r \prod_{i\in[n]} g_i^{a_i}$ and $d = g^s \prod_{i\in[n]} g_i^{b_i}$.

Let us first consider how $\mathcal{A}$ can be successful in producing the encoding $[\pi]$ of a valid argument. Suppose $[\pi]$ is not part of the generic common reference string or output by the generic group oracle. In that case, since $[\cdot]$ is a random permutation over G and g is a generator there is negligible probability of $e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}}) e(g,\pi)$. We can therefore from now on assume $[\pi]$ has been generated by the generic group oracle at some point.

Now, let us as in the proof of Theorem 1 keep track of queries made by $\mathcal{A}$ and the corresponding polynomials. We maintain two lists $L = \{(p_i, \xi_i)\}$ and $L_T = \{(p_{i,T}, \xi_{i,T})\}$ containing pairs of the form $(p_i, \xi_i) \in \mathbb{Z}_p[X_1, \ldots, X_n, A, Y_1, \ldots, Y_n, Z_{11}, \ldots, Z_{nn}] \times \mathbb{Z}_p$, where $\xi_i, \xi_{i,T}$ are values the oracle has given to $\mathcal{A}$. At first the lists are initialized as follows $L = \{(1,\xi_1), (X_1, \xi_2), (X_2, \xi_3), \ldots, (X_{n-1}Z_{nn}, \xi_{2n^3+n^2+n+2})\}$ and $L_T = \emptyset$, where $\xi_1, \ldots, \xi_{2n^3+n^2+n+2}$ are the values from the generic common reference string. The list

L thus starts out by containing the polynomials used in the exponents when generating a common reference string and matching random encodings of those group elements. Upon input $(\mathrm{mult}, \xi_i, \xi_j)$ giving answer ξ the extractor appends $(p_i + p_j, \xi)$ to L unless it has already been stored in L. Inputs of the form $(\mathrm{mult}_T, \xi_{i,T}, \xi_{j,T})$ are treated similarly using the list L_T. On input $(\mathrm{pair}, \xi_i, \xi_j)$ giving response ξ_T it appends $(p_i p_j, \xi_T)$ to L_T unless such a tuple already exists.

Consider now the polynomial $\pi(X_1, \ldots, Z_{nn})$ for the adversary's output $[\pi]$. If we have

$$(r + \sum_{i\in[n]} a_i X_i)(\sum_{j\in[n]} P_j(\boldsymbol{X})Y_j) \neq (s + \sum_{i\in[n]} b_i X_i)(\sum_{j\in[n]} P_j(\boldsymbol{X})Y_{\rho(j)}) + \pi(X_1, \ldots, Z_{nn})$$

we will argue as in Theorem 1 that there is negligible chance of $\mathcal{A}$ being successful, i.e., there is negligible chance that $e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}})e(g, \pi)$ for the concrete values of $x_1, \ldots, z_{nn}$ chosen by the oracle. Since the generic oracle uses a random permutation to encode the group elements, we can simulate it by picking random distinct values for each new polynomial generated during the queries to the oracle. By the Schwartz-Zippel lemma, there is negligible probability of having a collision, i.e., two distinct polynomials that evaluate to the same group elements for the oracle's concrete choice of $x_1, \ldots, z_{nn}$. The simulation is perfect conditioned on no such collision happening, and as the adversary gets no information about $x_1, \ldots, z_{nn}$ in the simulation there is negligible chance over the choice of $x_1, \ldots, z_{nn} \leftarrow \mathbb{Z}_p^*$ for the event $e(c, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_j}) = e(d, \prod_{j\in[n]} g^{P_j(\boldsymbol{x})y_{\rho(j)}})e(g, \pi)$.

Remaining is the possibility that

$$(r + \sum_{i\in[n]} a_i X_i)(\sum_{j\in[n]} P_j(\boldsymbol{X})Y_j) = (s + \sum_{i\in[n]} b_i X_i) \sum_{j\in[n]} P_j(\boldsymbol{X})Y_{\rho(j)} + \pi(X_1, \ldots, Z_{nn}).$$

Since $\pi(X_1, \ldots, Z_{nn})$ belongs to L, which can only grow by adding polynomials in L when the adversary makes a $(\mathrm{mult}, \xi_i, \xi_j)$ query to the oracle, we have

$$\pi(X_1, \ldots, Z_{nn}) = \pi_1 + \pi_{x_1}X_1 + \cdots + \pi_\alpha A + \pi_{\alpha x_1} A X_1 + \cdots + \pi_{x_1 x_2} X_1 X_2 + \cdots \pi_{x_{n-1} z_{nn}} X_{n-1} Z_{nn},$$

for known coefficients $\pi_1, \ldots, \pi_{x_{n-1}z_{nn}} \in \mathbb{Z}_p$.

Expanding $(r + \sum_{i\in[n]} a_i X_i)(\sum_{j\in[n]} P_j(\boldsymbol{X})Y_j) = (s + \sum_{i\in[n]} b_i X_i)(\sum_{j\in[n]} P_j(\boldsymbol{X})Y_{\rho(j)}) + \pi(X_1, \ldots, Z_{nn})$ on both sides gives us

$$\begin{aligned} &\sum_{i\in[n]} a_i P(\boldsymbol{X})Y_i + \left[r \sum_{j\in[n]} P_j(\boldsymbol{X})Y_j + \sum_{i\in[n]} \sum_{j\in[n]\setminus\{i\}} a_i X_i P_j(\boldsymbol{X})Y_j \right] \\ &= \sum_{i\in[n]} b_i P(\boldsymbol{X})Y_{\rho(i)} + \left[s \sum_{j\in[n]} P_j(\boldsymbol{X})Y_{\rho(j)} + \sum_{i\in[n]} \sum_{j\in[n]\setminus\{i\}} b_i X_i P_j(\boldsymbol{X})Y_{\rho(j)} + \pi(X_1, \ldots, Z_{nn}) \right]. \end{aligned}$$

Since $\pi(X_1, \ldots, Z_{nn})$ has no elements of the form $P(\boldsymbol{X})Y_i$, neither do any of the parts within the brackets. We conclude that for the two polynomials to be equal, we must have $b_i = a_{\rho(i)}$ for all $i \in [n]$. □

EFFICIENCY. Table 5 summarizes the cost of permutation arguments. We note that if we carry out many permutation arguments, it is possible to pre-compute all the products $\prod_{j\in[n]\setminus\{i\}} g^{x_i P_j(\boldsymbol{x})y_j}$ to reduce the number of multiplications. Similarly, it is possible to pre-compute $\prod_{i\in[n]} g^{P_i(\boldsymbol{x})y_i}$. We have therefore ignored these computational costs for the prover and verifier in Table 5.

Size of σ_{perm}	Argument size	Prover comp.	Prover comp. $\{0,1\}$	Verifier comp.	Verifier comp. N
n^3 G	1 G	$2n+2$ E + n^2 M	2 E + n^2 M	3 P + $n-1$ M	$N+2$ P + $2N$ E + $Nn-N$ M

Table 5. Cost of permutation arguments.

9 Rotating Values between Commitments

Consider two sets of n commitments $c_1,\dots,c_n$ and $d_1,\dots,d_n$, with openings $a_{i1},\dots,a_{in},r_i$ and $b_{i1},\dots,b_{in},s_i$ for $i=1,\dots,n$. We can view these committed values as rows of two matrices A and B. We will give an argument for the first column of B being the first column of A rotated one step up, the second column of B being the second column of A rotates two steps up, the third column of B being the third column of A rotated three step up, etc. Since a rotation of n steps corresponds to staying in place the last column remains unchanged. In other words, we want to show that for all i,j we have $b_{ij}=a_{i+j \bmod n,j}$. This means each commitment d_i contains exactly one value from each of the commitments $c_1,\dots,c_n$ and vice versa.

We will split the argument into n separate arguments; one for each column. We now give an argument for column k of B being equal to column k of A rotated k steps upwards.

Common reference string: σ including $\sigma_{\mathrm{rot},k}=(\{g^{z_{ik}}\}_{i\in[n]},\{g^{x_j z_{ik}}\}_{i,j\in[n],j\neq k})$.

Statement: Commitments $c_1,\dots,c_n,d_1,\dots,d_n\in G$.

Prover's witness: Openings $\{r_i,a_{i1},\dots,a_{in}\}_{i\in[n]}$ and $\{s_i,b_{i1},\dots,b_{in}\}_{i\in[n]}$ so

$$\forall i: c_i=g^{r_i}\prod_{j\in[n]} g_j^{a_{ij}} \quad\text{and}\quad \forall i: d_i=g^{s_i}\prod_{j\in[n]} g_j^{b_{ij}} \quad\text{and}\quad \forall i: b_{ik}=a_{i+k \bmod n,k}.$$

Argument: Compute the argument as

$$\pi=\prod_{i\in[n]}\left[(g^{z_{ik}})^{r_i}(g^{z_{i+k \bmod n,k}})^{-s_i}\prod_{j\in[n]\setminus\{k\}}(g^{x_j z_{ik}})^{a_{ij}}(g^{x_j z_{i+k \bmod n,k}})^{-b_{ij}}\right].$$

Verification: Output 1 if and only if

$$\prod_{i\in[n]} e(c_i,g^{z_{ik}})=e(g,\pi)\prod_{i\in[n]} e(d_i,g^{z_{i+k \bmod n,k}}).$$

Theorem 9. *The non-interactive argument described above has perfect completeness and perfect witness-indistinguishability.*

Proof. To argue perfect completeness, consider the three products of pairings the verifier has to compute.

$$\begin{aligned}
\prod_{i\in[n]} e(c_i, g^{z_{ik}}) &= e(g,g)^{\sum_i (r_i+\sum_j a_{ij}x_j)z_{ik})} \\
&= e(g,g)^{\sum_i a_{ik}x_k z_{ik}} e(g,g)^{\sum_i r_i z_{ik} + \sum_i \sum_{j\neq k} a_{ij}x_j z_{ik}}. \\
\prod_{i\in[n]} e(d_i, g^{z_{i+k \bmod n,k}}) &= e(g,g)^{\sum_i (s_i+\sum_j b_{ij}x_j)z_{i+k \bmod n,k}} \\
&= e(g,g)^{\sum_i b_{ik}x_k z_{i+k \bmod n,k}} e(g,g)^{\sum_i s_i z_{i+k \bmod n,k} + \sum_i \sum_{j\neq k} b_{ij}x_j z_{i+k \bmod n,k}}. \\
e(g,\pi) &= e(g, \prod_{i\in[n]} [(g^{z_{ik}})^{r_i}(g^{z_{i+k \bmod n,k}})^{-s_i} \prod_{j\in[n]\setminus\{k\}} (g^{x_j z_{ik}})^{a_{ij}}(g^{x_j z_{i+k \bmod n,k}})^{-b_{ij}}]) \\
&= e(g,g)^{\sum_i (r_i z_{ik} - s_i z_{i+k \bmod n,k} + \sum_{j\neq k}(a_{ij}x_j z_{ik} - b_{ij}x_j z_{i+k \bmod n,k}))}
\end{aligned}$$

Since $b_{ik} = a_{i+k \bmod n,k}$ we have $\sum_{i\in[n]} b_{ik}x_k z_{i+k \bmod n,k} = \sum_{i\in[n]} a_{i+k \bmod n,k}x_k z_{i+k \bmod n,k} = \sum_{i\in[n]} a_{ik}x_k z_{ik}$. Looking at the three pairings above we then get $\prod_{i\in[n]} e(c_i, g^{z_{ik}}) = e(g,\pi)\prod_{i\in[n]} e(d_i, g^{z_{i+k \bmod n,k}})$ so the verification accepts the argument π.

Since g is a generator for G, there is exactly one argument π that will make the verification accept. By the perfect completeness, all openings with $b_{ik} = a_{i+k \bmod n,k}$ give an accepting argument and therefore all valid witnesses result in the same argument. $\square$

Theorem 10. *In the generic group model it is infeasible to find two sets of commitments with their respective openings and a valid argument unless indeed the matrices A and B given by the commitments have the property that column k of B is equal to column k of A rotated k step upwards.*

Proof. We want to prove

$$\Pr\Big[(p,G,G_T,e,g) \leftarrow \mathcal{G}(1^\lambda); (\{r_i, a_{i1},\ldots,a_{in}, s_i, b_{i1},\ldots,b_{in}\}_{i\in[n]}, [\pi]) \leftarrow \mathcal{A}^{\mathcal{O}}(p,G,G_T,e,g) : \\ \prod_{i\in[n]} e(c_i, g^{z_{ik}}) = e(g,\pi)\prod_{i\in[n]} e(d_i, g^{z_{i+k \bmod n,k}}) \quad \text{and} \quad \exists i\in[n] : b_{ik} \neq a_{i+k \bmod n,k}\Big] \approx 0,$$

where we define $c_i = g^{r_i}\prod_{j\in[n]} g_j^{a_{ij}}$ and $d_i = g^{s_i}\prod_{j\in[n]} g_j^{b_{ij}}$.

Let us first consider how $\mathcal{A}$ can be successful in producing the encoding $[\pi]$ of a valid argument. Suppose $[\pi]$ is not part of the generic common reference string or output by the generic group oracle. In that case, since $[\cdot]$ is a random permutation over G and g is a generator there is negligible probability of $\prod_{i\in[n]} e(c_i, g^{z_{ik}}) = e(g,\pi)\prod_{i\in[n]} e(d_i, g^{z_{i+k \bmod n,k}})$. We can therefore from now on assume $[\pi]$ has been generated by the generic group oracle at some point.

Now, let us as in the proof of Theorem 1 keep track of queries made by $\mathcal{A}$ and the corresponding polynomials. We maintain two lists $L = \{(p_i, \xi_i)\}$ and $L_T = \{(p_{i,T}, \xi_{i,T})\}$ containing pairs of the form $(p_i, \xi_i) \in \mathbb{Z}_p[X_1,\ldots,X_n,A,Y_1,\ldots,Y_n,Z_{11},\ldots,Z_{nn}] \times \mathbb{Z}_p$, where $\xi_i, \xi_{i,T}$ are values the oracle has given to $\mathcal{A}$. At first the lists are initialized as follows $L = \{(1,\xi_1),(X_1,\xi_2),(X_2,\xi_3),\ldots,(X_{n-1}Z_{nn},\xi_{2n^3+n^2+n+2})\}$ and $L_T = \emptyset$, where $\xi_1,\ldots,\xi_{2n^3+n^2+n+2}$ are the values from the generic common reference string. The list L thus starts out by containing the polynomials used in the exponents when generating a common reference string and matching random encodings of those group elements. Upon input $(\mathrm{mult}, \xi_i, \xi_j)$ giving answer ξ the extractor appends $(p_i + p_j, \xi)$ to L unless it has already been stored in L. Inputs of the form

$(\mathrm{mult}_T, \xi_{i,T}, \xi_{j,T})$ are treated similarly using the list L_T. On input $(\mathrm{pair}, \xi_i, \xi_j)$ giving response ξ_T it appends $(p_i p_j, \xi_T)$ to L_T unless such a tuple already exists.

Consider now the polynomial $\pi(X_1, \ldots, Z_{nn})$ for the adversary's output $[\pi]$. If

$$\sum_{i\in[n]}(r_i + \sum_{j\in[n]} a_{ij}X_j)Z_{ik} \neq \pi(X_1, \ldots, Z_{nn}) + \sum_{i\in[n]}(s_i + \sum_{j\in[n]} b_{ij}X_j)Z_{i+k \bmod n,k},$$

we can argue as in Theorem 1 that there is negligible chance of $\mathcal{A}$ being successful, i.e., there is negligible chance that $\prod_{i\in[n]} e(c_i, g^{z_{ik}}) = e(g,\pi)\prod_{i\in[n]} e(d_i, g^{z_{i+k \bmod n,k}})$ for the concrete values of $x_1, \ldots, z_{nn}$ chosen by the oracle. Since the generic oracle uses a random permutation to encode the group elements, we can simulate it by picking random distinct values for each new polynomial generated during the queries to the oracle. By the Schwartz-Zippel lemma, there is negligible probability of having a collision, i.e., two distinct polynomials that evaluate to the same group elements for the oracle's concrete choice of $x_1, \ldots, z_{nn}$. The simulation is perfect conditioned on no such collision happening, and as the adversary gets no information about $x_1, \ldots, z_{nn}$ in the simulation there is negligible chance over the choice of $x_1, \ldots, z_{nn} \leftarrow \mathbb{Z}_p^*$ for the event $\prod_{i\in[n]} e(c_i, g^{z_{ik}}) = e(g,\pi)\prod_{i\in[n]} e(d_i, g^{z_{i+k \bmod n,k}})$.

Remaining is the possibility that

$$\sum_{i\in[n]}(r_i + \sum_{j\in[n]} a_{ij}X_j)Z_{ik} = \pi(X_1, \ldots, Z_{nn}) + \sum_{i\in[n]}(s_i + \sum_{j\in[n]} b_j X_j)Z_{i+k \bmod n,k}.$$

Since $\pi(X_1, \ldots, Z_{nn})$ belongs to L, which can only grow by adding polynomials in L when the adversary makes a $(\mathrm{mult}, \xi_i, \xi_j)$ query to the oracle, we have

$$\pi(X_1, \ldots, Z_{nn}) = \pi_1 + \pi_{x_1}X_1 + \cdots + \pi_\alpha A + \pi_{\alpha x_1}AX_1 + \cdots + \pi_{x_1x_2}X_1X_2 + \cdots \pi_{x_{n-1}z_{nn}}X_{n-1}Z_{nn},$$

for known coefficients $\pi_1, \ldots, \pi_{x_{n-1}z_{nn}} \in \mathbb{Z}_p$.

Expanding $\sum_{i\in[n]}(r_i + \sum_{j\in[n]} a_{ij}X_j)Z_{ik} = \pi(X_1, \ldots, Z_{nn}) + \sum_{i\in[n]}(s_i + \sum_{j\in[n]} b_{ij}X_j)Z_{i+k \bmod n,k}$ on both sides gives us

$$\begin{aligned}
&\sum_{i\in[n]} a_{ik}X_kZ_{ik} + \sum_{i\in[n]}\left[r_iZ_{ik} + \sum_{j\in[n]\setminus\{k\}} a_{ij}X_jZ_{ik}\right] \\
&= \sum_{i\in[n]} b_{ik}X_kZ_{i+k \bmod n,k} + \pi(X_1, \ldots, Z_{nn}) + \sum_{i\in[n]}\left[s_iZ_{i+k \bmod n,k} + \sum_{j\in[n]\setminus\{k\}} b_{ij}X_jZ_{i+k \bmod n,k}\right].
\end{aligned}$$

Neither $\pi(X_1, \ldots, Z_{nn})$ nor any of the parts within the brackets have elements of the form X_kZ_{ik}. We conclude that for the two polynomials to be equal, we must have

$$\sum_{i\in[n]} a_{ik}X_kZ_{ik} = \sum_{i\in[n]} b_{ik}X_kZ_{i+k \bmod n,k}.$$

This in turn implies $b_{ik} = a_{i+k \bmod n,k}$ for all $i \in [n]$. □

EFFICIENCY. We summarize the cost of the rotation argument in Table 6. In the NIZK argument for circuit satisfiability, we will typically be considering 2 sets of n commitments, where we want to make a rotation argument for each of the n columns. The rotation argument has a structure that permits significant batch-verification savings in this case, which we will now present. Suppose we have commitments $c_{11}, d_{11}, \ldots, c_{nN}, d_{nN}$ and corresponding rotation arguments $\pi_{11}, \ldots, \pi_{Nn}$, where π_{jk} is the rotation argument for column k between the sets $c_{1j}, \ldots, c_{nj}$ and $d_{1j}, \ldots, d_{nj}$. The verifier picks at random $s_1, \ldots, s_N, t_1, \ldots, t_n \leftarrow [2^\ell]$ and verifies

$$\prod_{i\in[n]} e\Big(\prod_{j\in[N]} c_{ij}^{s_j}, \prod_{k\in[n]} (g^{z_{ik}})^{t_k}\Big) = e\Big(g, \prod_{j\in[N]}\prod_{k\in[n]} \pi_{jk}^{s_j t_k}\Big) \prod_{i\in[n]} e\Big(\prod_{j\in[N]} d_{ij}^{s_j}, \prod_{k\in[n]} (g^{z_{i+k \bmod n,k}})^{t_k}\Big).$$

By the bilinear properties of the pairing, this corresponds to a "two-layer" batch-verification

$$\prod_{j\in[N]}\left(\prod_{k\in[n]}\left[\prod_{i\in[n]} e(c_{ij}, g^{z_{ik}})\right]^{t_k}\right)^{s_j} = \prod_{j\in[N]}\left(\prod_{k\in[n]}\left[e(g,\pi_{jk})\prod_{i\in[n]} e(d_{ij}, g^{z_{i+k \bmod n,k}})\right]^{t_k}\right)^{s_j},$$

with at most probability $2^{1-\ell}$ of succeeding unless all of the Nn rotation arguments are valid.

	Size of $\sigma_{\mathrm{rot},k}$	Argument size	Prover comp.	Prover comp. $\{0,1\}$	Verifier comp.	Verifier comp. N
One column	n^2 G	1 G	n^2 E	n E + $n^2 - n$ M	$n+1$ P	2 P + Nn E
n columns	n^3 G	n G	n^3 E	n^2 E + $n^3 - n^2$ M	n^2+n P	$2n+1$ P + $3Nn+2n^2$ E

Table 6. Cost of rotation arguments.

10 Circuit Satisfiability

We will now give a NIZK argument for Circuit Satisfiability. Without loss of generality, we will for simplicity just consider circuits consisting of NAND-gates; the construction can be generalized to circuits containing a mix of different types of binary gates. Further, we will without loss of generality assume $3|n$ and that the circuit has exactly $\frac{1}{3}n^4$ gates.

Common reference string: $\sigma = (gk, ck, \sigma_{\text{know}}, \sigma_{\text{prod}}, \sigma_{\text{perm}}, \sigma_{\text{rot},1}, \ldots, \sigma_{\text{rot},n})$.
Statement: Circuit C consisting of $N = \frac{1}{3}n^4$ NAND-gates.
Prover's witness: An assignment of values in $\{0,1\}$ to the input wires so the circuit outputs 1.
Argument:

1. Make commitments $c_1, \ldots, c_{\frac{1}{3}n^3}$ and $d_1, \ldots, d_{\frac{1}{3}n^3}$ to the inputs and commitments $v_1, \ldots, v_{\frac{1}{3}n^3}$ to the outputs of the NAND-gates. We think of the gates as being numbered $(i,j) \in [\frac{1}{3}n^3] \times [n]$ and the committed values a_{ij}, b_{ij} and u_{ij} correspond to the two inputs and the output of gate (i,j).
2. For each commitment give an argument of knowledge of the committed values as in Section 6.
3. For each commitment prove that all values belong to $\{0,1\}$. This can be done by using the product argument from Section 7, since $a = a^2$ implies $a \in \{0,1\}$.
4. Let $(*, \ldots, *, a, *, \ldots, *)$ we the values of the commitment containing the circuit's output a. We prove that the circuits output is $a = 1$ by using a product argument from Section 7 for the product of $(*, \ldots, *, a, *, \ldots, *)$ and $(0\ldots, 0, 1, 0, \ldots, 0)$ being $(0, \ldots, 0, 1, 0, \ldots, 0)$.

5. Prove that the committed values respect the NAND-gates. This can be done by using the product argument from Section 7 and the homomorphic property of the commitment scheme, since $u_{ij} = \neg(a_{ij} \wedge b_{ij})$ if and only if $1 - u_{ij} = a_{ij}b_{ij}$, when $a_{ij}, b_{ij}, u_{ij} \in \{0, 1\}$.
6. Prove that the wires have been given consistent value assignments, i.e., for each wire all the committed values corresponding to this wire are the same. This argument is described in Section 10.1.

Verification: Return 1 if and only if all arguments are valid.

Theorem 11. *The argument given above has perfect completeness and perfect zero-knowledge.*

Proof. Perfect completeness follows from perfect completeness of the underlying arguments of knowledge, products, permutations and rotations.

We now turn to proving perfect zero-knowledge. First we describe the simulator (S_1, S_2). S_1 generates the common reference string correctly, but also outputs the trapdoor for the commitment scheme $tk = (x_1, \ldots, x_n)$. The simulation algorithm S_2 gets the circuit C, the common reference string σ and the commitment trapdoor tk as input. It creates a simulated argument as follows:

1. Make commitments $c_1, \ldots, c_{\frac{1}{3}n^3}$ and $d_1, \ldots, d_n$ and $v_1, \ldots, v_{\frac{1}{3}n^3}$ all containing 0 values. Use the trapdoor for the commitment scheme to compute openings of $v_1, \ldots, v_{\frac{1}{3}n^3}$ to 1 values.
2. For each commitment give an argument of knowledge of the committed values as in Section 6 as in the real argument.
3. For each commitment prove that all values belong to $\{0, 1\}$ using the product argument from Section 7 as in the real argument.
4. Using the trapdoor openings of the v_i's to 1 values, give an argument for the committed circuit output a being 1 as in the real argument.
5. Using the trapdoor openings of the v_i's to 1 values use the product argument from Section 7 to prove that the committed values respect the NAND-gates as in the real argument.
6. Prove that the wires have been given consistent value assignments, i.e., for each wire all the committed values corresponding to this wire are the same. As in the real argument, we do that using the wire consistency argument described in Section 10.1 and the argument goes through since all committed values are 0.

The common reference string is generated in the same way by the simulator as in a real argument, so to argue perfect zero-knowledge we just need to show that a simulated argument has the same probability distribution as a real argument. Consider for that purpose the following hybrid argument where we get both the trapdoor for the commitment scheme and the witness. First we commit to 0 in all commitments just as the simulator, but then afterwards we use the trapdoor for the commitment scheme to find openings of the commitments that correspond to the real inputs and outputs of the circuit and run the real prover algorithm to generate the rest of the argument. By the perfect trapdoor property of the commitment scheme, it is indistinguishable whether we first commit to the real values or whether we make trapdoor commitments and then open them to the real values. The hybrid argument is therefore perfectly indistinguishable from a real argument. On the other hand, the perfect witness-indistinguishability of all the underlying arguments of knowledge, products and wire-consistency implies that the hybrid argument is perfectly indistinguishable from a simulated argument. □

Theorem 12. *In the generic group model, the argument above demonstrates knowledge of a satisfying witness for the circuit C.*

Proof. By Theorem 4 there is an extractor that given the commitments, proofs of knowledge and a list of the oracle queries and answers made by the adversary can extract openings of all commitments. If those committed values are internally consistent with an honest prover's argument using a real witness for the circuit being satisfiable, then we will be able to deduce a witness for the circuit being satisfiable. What remains is to prove that the committed values are indeed consistent with an honest argument.

According to Theorem 1 the commitments are binding, so we can talk about the opening of a commitment without ambiguity. The argument contains a set of arguments showing that all the commitments contain values $a_1, \dots, a_n$ so $a_i^2 = a_i$. By Theorem 6 this implies that all the committed values belong to $\{0, 1\}$. Also, for the output wire a, there is an argument for $a \cdot 1 = a$. The next set of product arguments show that for each set of n gates with input $a_1, \dots, a_n, b_1, \dots, b_n$ and outputs $u_1, \dots, u_n$ we have $1 - u_i = a_i b_i$. This means the inputs and the outputs conform to the NAND-gates. Finally, there is an argument of wire consistency that shows that for each wire, all the committed values corresponding to that wire are identical. We conclude that the committed values do indeed represent a valid assignment of values to wires that make the circuit output 1, so we have extracted a witness for the circuit's satisfiability. $\square$

ARITHMETIC CIRCUIT. In an arithmetic circuit with multiple outputs, the wires may contain any value in $\mathbb{Z}_p$. The circuit has a mix of addition and multiplication gates which each have two inputs and one output. Some of the inputs and outputs may be fixed and specified in the statement. Without loss of generality, we can assume each gate has at most one of the wires specified as part of the public statement since gates with two specified wires are trivial and can be eliminated from the statement in a preprocessing step. We will adapt the argument for circuit satisfiability to the case of arithmetic circuits with multiple outputs.

Common reference string: $\sigma = (gk, ck, \sigma_{\text{know}}, \sigma_{\text{prod}}, \sigma_{\text{perm}}, \sigma_{\text{rot},1}, \dots, \sigma_{\text{rot},n})$.

Statement: Circuit C consisting of $\frac{1}{3}n^4$ addition and multiplication gates and a specification of fixed values for some of the inputs and outputs.

Prover's witness: An assignment of values in $\mathbb{Z}_p$ to the unspecified wires so the circuit is consistent with the specified inputs and outputs.

Argument:

1. Group the gates in six categories: addition with one specified input, addition with one specified output, addition with no specified values, multiplication with one specified input, multiplication with one specified output, multiplication with no specified values.
   Without loss of generality, we assume the number of gates in each category is divisible by n. For each block of n gates with specified inputs and each block of n gates with specified outputs we will in the following use trivial randomness 0 in the commitment so it is easy to verify that the inputs and outputs are as specified in the statement.
2. Make commitments $c_1, \dots, c_{\frac{1}{3}n^3}$ and $d_1, \dots, d_{\frac{1}{3}n^3}$ to the inputs and commitments $v_1, \dots, v_{\frac{1}{3}n^3}$ to the outputs of the gates. We think of the gates as being numbered $(i, j) \in [\frac{1}{3}n^3] \times [n]$ and the committed values a_{ij}, b_{ij} and u_{ij} correspond to the two inputs and the output of gate (i, j).
   For the addition gates, the randomizers are chosen so $c_i d_i = v_i$, which by the homomorphic property of the commitment scheme implies $a_{ij} + b_{ij} = u_{ij}$.
3. For each non-trivial commitment give an argument of knowledge of the committed values as in Section 6.
4. For each block of n multiplication gates, make a product argument for the corresponding commitments as in Section 7.
5. Prove that the wires have been given consistent value assignments, i.e., for each wire all the committed values corresponding to this wire are the same. This argument is described in Section 10.1.

Verification: Return 1 if and only if all commitments with specified values are correct, for all addition gates we have $c_i d_i = v_i$, and all arguments are valid.

Theorem 13. *The argument given above has perfect completeness and perfect zero-knowledge.*

Proof. Perfect completeness follows from perfect completeness of the underlying arguments of knowledge, products, permutations and rotations.

We now turn to proving perfect zero-knowledge. First we describe the simulator (S_1, S_2). S_1 generates the common reference string correctly, but also outputs the trapdoor for the commitment scheme $tk = (x_1, \ldots, x_n)$. The simulation algorithm S_2 gets the circuit C, the common reference string σ and the commitment trapdoor tk as input. It creates a simulated argument as follows:

1. As in the real argument, group the gates in six categories: addition with one specified input, addition with one specified output, addition with no specified values, multiplication with one specified input, multiplication with one specified output, multiplication with no specified values.
   For each block of n gates with specified inputs and each block of n gates use trivial randomness 0 in the commitment. Use the trapdoor opening property to compute an opening of these commitments to 0 values.
2. Make commitments $c_1, \ldots, c_{\frac{1}{3}n^3}$ and $d_1, \ldots, d_n$ and $v_1, \ldots, v_{\frac{1}{3}n^3}$ for blocks of unspecified values all contain 0 values instead of inputs and outputs of the gates.
   For addition gates, select the randomizers so $c_i d_i = v_i$. For triples involving a commitment to specified values, this can be done by using the trapdoor opening of that commitment to 0.
3. For each commitment give an argument of knowledge of the committed values as in Section 6 as in a real argument.
4. As in the real argument, use the product argument from Section 7 to prove that the committed values respect the product gates. For triples involving a commitment to specified values, this may require using the trapdoor opening of that commitment.
5. As in a real argument, use the technique described in Section 10.1 to prove that the wires have been given consistent value assignments, i.e., for each wire all the committed values corresponding to this wire are the same. This can be done by using the trapdoor openings to 0 values for the commitments to specified values.

The common reference string is generated in the same way by the simulator as in a real argument, so to argue perfect zero-knowledge we just need to show that a simulated argument has the same probability distribution as a real argument. Consider for that purpose the following hybrid argument where we get both the trapdoor for the commitment scheme and the witness. First we make all commitments just as the simulator, but then afterwards we use the trapdoor for the commitment scheme to find openings of the commitments that correspond to the real inputs and outputs of the circuit and run the real prover algorithm to generate the rest of the argument. By the perfect trapdoor property of the commitment scheme, it is indistinguishable whether we first commit to the real values or whether we make trapdoor commitments and then open them to the real values. The hybrid argument is therefore perfectly indistinguishable from a real argument. On the other hand, the perfect witness-indistinguishability of all the underlying arguments of knowledge, products and wire-consistency implies that the hybrid argument is perfectly indistinguishable from a simulated argument. □

Theorem 14. *In the generic group model, the argument above demonstrates knowledge of an assignment of values to the wires of the circuit C that is consistent with the specified values and the addition and multiplication gates.*

Proof. By Theorem 4 there is an extractor that given the commitments, proofs of knowledge and a list of the oracle queries and answers made by the adversary can extract openings of all commitments. If those committed values are consistent with an honest prover's argument using a real witness for the circuit being satisfiable, then we will be able to deduce a witness for the circuit being satisfiable. What remains is to prove that the committed values are indeed consistent with an honest argument.

According to Theorem 1 the commitments are binding, so we can talk about the opening of a commitment without ambiguity. The prover uses trivial randomness in commitments to specified values, so it can be checked that they are correct. The argument contains commitments c_i, d_i, v_i for the addition gates such that $c_i d_i = v_i$, which by the homomorphic property shows that the committed values respect the addition gates. The product arguments show that the multiplication gates are respected. Finally, there is an argument of wire consistency that shows that for each wire, all the committed values corresponding to that wire are identical. We conclude that the committed values do indeed represent a valid assignment of values to wires that is consistent with the gate sin the circuit and the specified values. □

10.1 Wire Consistency

A wire may appear several places in a circuit, both as an output of a gate and as input to other gates. If we have committed values $a_1, \ldots, a_{n^4}$, a particular wire may be committed at indices $i_1, i_2, \ldots, i_k$. We will describe and argument for all committed values to a wire being identical.

The idea in the argument is as follows, if ρ is a permutation that contains the cycle $i_1 \to i_2 \to \cdots \to i_n \to i_1$, then if $a_{i_1} = a_{\rho(i_1)}, \ldots, a_{i_k} = a_{\rho(i_k)}$ we can conclude that $a_{i_1} = a_{i_2} = \cdots = a_{i_k}$. Generalizing, if ρ is a permutation that contain Hamiltonian cycles over the indices for each wire, then by proving that the committed values satisfy $a_1 = a_{\rho(1)}, \ldots, a_{n^4} = a_{\rho(n^4)}$ we show that each wire has been assigned the same value in each place it appears in the commitments.

We will now describe how to argue committed values $a_1, \ldots, a_{n^4}$ and $b_1, \ldots, b_{n^4}$ satisfy $b_1 = a_{\rho(1)}, \ldots, b_{n^4} = a_{\rho(n^4)}$. In our construction, we will use the permutation argument from Section 8 which can be used to show that two commitments c and d contain a publicly known permutation of each other's committed values. We will also use the rotation argument in Section 9. Observe, if we have two sets of n commitments $c_1, \ldots, c_n$ and $d_1, \ldots, d_n$, committing to two sets of n^2 values $a_{11}, \ldots, a_{nn}$ and $b_{11}, \ldots, b_{nn}$, we may combine n rotation arguments to demonstrate for all $i, j \in [n]$ that $b_{ij} = a_{i+j \bmod n, j}$. This implies that the values have been distributed evenly using a permutation such that each commitment b_i contains exactly one value from each of $c_1, \ldots, c_n$.

Consider first two sets of n commitments $c_1, \ldots, c_n, d_1, \ldots, d_n$ to values $a_{11}, \ldots, a_{nn}$ and $b_{11}, \ldots, b_{nn}$. We will use a Clos-network [Clo53] to give an argument for the two sets of committed values being permutations of each other, for a publicly known permutation $\rho \in S_{n^2}$, i.e., for all $i, j \in [n]$ we have $b_{ij} = a_{\rho(ij)}$. Clos observed that all permutations of n^2 elements can be obtained as follows. First divide the elements into n blocks of n elements and permute the elements within each block. Next, distribute the elements in each block evenly on n blocks, i.e., we get a new set of n blocks, each containing one element from each of the previous blocks. Again permute the elements in each block. Once again, distribute the elements in each block evenly on n blocks. Finally, permute the elements within this block to get the elements permuted in the desired order.

To build a Clos-network for a permutation $\rho \in S_{n^2}$, we make 4 sets of n intermediate commitments $\{c'_i\}_{i\in[n]}, \{v_i\}_{i\in[n]}, \{v'_i\}_{i\in[n]}, \{d'_i\}_{i\in[n]}$ and argue knowledge of their contents. Each commitment corresponds to a commitment to a block of n values. We use the argument of a permutation within a commitment to show that for all $i \in [n]$ the pairs of commitments $(c_i, c'_i), (d_i, d'_i)$ and (v_i, v'_i) contain the same elements just in permuted order. We use n rotation arguments to show that $c'_1, \ldots, c'_n$ and $v_1, \ldots, v_n$ contain values $a'_{11}, \ldots, a'_{nn}$ and $u_{11}, \ldots, u'_{nn}$ so for all $i, j \in [n]$ we have $u_{ij} = a_{i+j \bmod n, j}$. Similarly, we use n rotation arguments to show that $v'_1, \ldots, v'_n$ and $b_1, \ldots, b_n$ contain values $u'_{11}, \ldots, u'_{nn}$ and $b'_{11}, \ldots, b'_{nn}$ so for all $i, j \in [n]$ we have $b'_{ij} = u'_{i+j \bmod n, j}$. By using appropriately chosen intermediate permutations within the commitments, this gives us $b_{ij} = a_{\rho(ij)}$ for $i, j \in [n]$ as desired.

We will now use recursion to build a Clos-network for $\rho \in S_{n^4}$. Given two sets of n^3 commitments $c_1, \ldots, c_{n^3}$ and $d_1, \ldots, d_{n^3}$ to values $a_1, \ldots, a_{n^4}$ and $b_1, \ldots, b_{n^4}$ and a publicly know permutation $\rho \in S_{n^4}$ we can use a similar Clos-network argument, to demonstrate $b_i = a_{\rho(i)}$ for $i \in [n^4]$. Divide the n^3 commitment $c_1, \ldots, c_{n^3}$ into n^2 groups of n commitments to n^2 values and similarly for the n^3 commitments $d_1, \ldots, d_n$. Using the Clos-network argument above for the case n^2 we can give an argument of n^2 values being permuted within each set of n commitment. Using n^3 rotation arguments, we can argue correctness of an even distribution of n^2 committed values within a group onto n^2 different groups of commitments. Repeating the Clos-network construction given above for n^2 we now have an argument for two sets of n^4 committed values satisfying $b_i = a_{\rho(i)}$ for $i \in [n^4]$.

The argument for a known permutation of n^4 committed values has perfect completeness and perfect witness-indistinguishability because the underlying commitments are perfectly hiding and the underlying arguments are perfectly witness-indistinguishable. The argument is sound in the generic group model, since we prove knowledge of the contents of the commitments, the commitments are binding and the underlying arguments for permutations within commitments and rotations of values between commitments are sound in the generic group model according to Theorems 8 and 10.

11 Efficiency

THE COMMON REFERENCE STRING. The common reference string contains $2n^3 + n^2 + n + 2$ group elements as well as the description of the group. With L being the language of circuits that has $|C| = \frac{1}{3}n^4$ gates this gives a common reference string with less than $5|C|^{3/4}$ group elements. Except for g all these group elements require an exponentiation to be constructed, so the key generation does less than $5|C|^{3/4}$ exponentiations.

THE ARGUMENTS. We summarize the cost of the various arguments in Table 7. In the table, we only include the dominant parts of the cost ignoring smaller additive factors. This is done under the assumptions that $N \geq n^2$ and $n \geq \lambda$. Our arguments are well-suited for the use of multi-exponentiation techniques [Pip80,Lim00], and $n \geq \lambda$ implies that n multiplications is cheaper than 1 exponentiation or pairing. We have ignored the cost of exponentiations and pairings when the cost is dominated by the cost of the multiplications.

	Argument size	Prover comp.	Prov. comp. $\{0,1\}$	Verifier comp.	Ver. comp. N
Commit	1 G	n E	n M	-	-
Knowledge	1 G	n E	n M	2 P	$2N$ E
Product	1 G	n^2 E	n^2 M	3 P	N P + $2N$ E
Permutation	1 G	n^2 M	n^2 M	n M	Nn M
Rotation (one column)	1 G	n^2 E	n^2 M	n P	Nn E
Rotation (n columns)	n G	n^3 E	n^3 M	n^2 P	$3Nn$ E

Table 7. Cost of arguments.

CIRCUIT SATISFIABILITY. The non-interactive argument for circuit satisfiability requires the construction of n^3 commitments together with their n^3 corresponding arguments of knowledge. It contains n^3 product arguments to show that each gate's inputs and output are 0 or 1. Moreover, it has $\frac{1}{3}n^3$ product arguments that show that the inputs and outputs satisfy the NAND-gates. The main bulk of effort is in the argument for wire consistency. This argument for a permutation over n^4 committed values requires $16n^3$ new commitments, $16n^3$ corresponding arguments of knowledge, $9n^3$ permutation arguments, and $8n^3$ rotation arguments. Adding up these parts gives a total of $17n^3$ commitments, $17n^3$ arguments of knowledge, $1\frac{1}{3}n^3$ product arguments, $9n^3$ permutation arguments and $8n^3$ rotation arguments. The $8n^3$ rotation arguments can be divided into $8n^2$ rotation arguments for n columns. Using Table 7 and $|C| = \frac{1}{3}n^4$ we get the cost of arguing circuit satisfiability listed in Table 8.

ARITHMETIC CIRCUIT. The non-interactive argument for arithmetic circuits requires the construction of n^3 commitments with their corresponding arguments of knowledge. There may be up to $\frac{1}{3}n^4$ multiplication gates for a cost of $\frac{1}{3}n^3$ product arguments. Add this to the cost of the wire-consistency argument, we get a total cost of up to $17n^3$ commitments, $17n^3$ arguments of knowledge, $\frac{1}{3}$ product arguments, $9n^3$ permutation arguments and $8n^2$ arguments for n columns. Using Table 7 and $|C| = \frac{1}{3}n^4$ we get the cost in Table 8 of arguing the arithmetic circuit is consistent with the specified values.

	CRS size	CRS comp.	Argument size	Prover comp.	Verifier comp.
Circuit satisfiability	$5\|C\|^{\frac{3}{4}}$ G	$5\|C\|^{\frac{3}{4}}$ E	$120\|C\|^{\frac{3}{4}}$ G	$73\|C\|^{\frac{5}{4}}$ M	$27\|C\|$ M
Arithmetic circuit	$5\|C\|^{\frac{3}{4}}$ G	$5\|C\|^{\frac{3}{4}}$ E	$117\|C\|^{\frac{3}{4}}$ G	$33\|C\|^{\frac{5}{4}}$ E	$27\|C\|$ M

Table 8. Cost of arguments.

12 Conclusion

We have proposed sub-linear size NIZK arguments with perfect completeness, perfect zero-knowledge and computational (co-)soundness in the common reference string model. Our construction directly yields a Zap with perfect completeness, perfect witness-indistinguishability and computational (co-)soundness in the plain model, where the verifier's first move consists of picking a verifiable common reference string for the NIZK argument. The NIZK argument is highly efficient to verify, matching state-of-the art interactive zero-knowledge arguments. The prover uses super-linear computation, but since the NIZK arguments are transferable this trade-off may be acceptable in cases where communication is expensive or when each NIZK argument has to be verified by many verifiers.

Abe and Fehr [AF07] showed that NIZK arguments with perfect zero-knowledge do not have a "direct black-box" reduction to a standard intractability problem. We do therefore not expect that the security of our NIZK argument can be reduced to a standard intractability assumption. The co-soundness of our NIZK argument, however, is a standard "q-style" assumption about the bilinear group used in the NIZK argument. We leave it as an interesting open problem to construct a sub-linear size co-sound NIZK argument from a simpler intractability assumption.

References

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 118–136, 2007.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, 2004.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188, 2004. Full paper available at `http://eprint.iacr.org/2003/077`.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112, 1988.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62, 2004.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73, 1993.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444, 2006.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15, 2007.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *STOC*, pages 209–218, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57, 2004.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434, 2007.

[Clo53] Charles Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.

[CS98] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, 1998. Full paper available at `http://eprint.iacr.org/2001/108`.

[Dam92] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 341–355, 1992.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.

[DDO$^+$02] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598, 2002.

[DDP02] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-optimal characterization of two NP proof systems. In *RANDOM*, volume 2483 of *Lecture Notes in Computer Science*, pages 179–193, 2002.

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 100–109, 2002.

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 41–59, 2006.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *FOCS*, pages 283–293, 2000.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS*, pages 427–436, 1992.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168, 2005.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal of Computing*, 25(1):169–192, 1996.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, pages 102–113, 2003. Full paper available at `http://eprint.iacr.org/2003/034`.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 323–341, 2007. Full paper available at `http://www.cs.ucla.edu/~rafail/PUBLIC/index.html`.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero-knowledge for NP. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, 2006.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT*, volume 4248 of *Lecture Notes in Computer Science*, pages 444–459, 2006. Full paper available at `http://www.brics.dk/~jg/NIZKGroupSignFull.pdf`.

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 192–208, 2009.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, 2008. Full paper available at `http://eprint.iacr.org/2007/155`.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[Lim00] Chae Hoon Lim. Efficient multi-exponentiation and application to batch verification of digital signatures, 2000. `http://dasan.sejong.ac.kr/~chlim/pub/multi_exp.ps`.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, 2004.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, 2003.

[Nec94] Vasilii I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mat. Zametki*, 55(2):91–101, 1994.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, 2002.

[Ore87] Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs. In *FOCS*, pages 462–471, 1987.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337, 2003.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, 1991.

[Pip80] Nicholas Pippenger. On the evaluation of powers and monomials. *SIAM Journal of Computing*, 9(2):230–250, 1980.

[Sah01] Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 2001.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, 1997.