Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments*

Helger Lipmaa[1,2]

[1] Cybernetica AS, Estonia
[2] Tallinn University, Estonia

Abstract. In 2010, Groth constructed a non-interactive zero-knowledge (NIZK) argument for circuit satisfiability with communication $\Theta(1)$, verifier's computation $\Theta(|C|)$, and common reference string length and prover's computation $\Theta(|C|^2)$, where $|C|$ is the circuit size. In the current paper, we show how to reduce the common reference string length to $O(|C|^{1+\varepsilon})$ for any $\varepsilon > 0$. For this, we show that for any $n > 0$, $[n]$ has a progression-free subset of odd integers of cardinality $\Omega(n^{1-\varepsilon})$. Moreover, the NIZK argument for circuit satisfiability is slightly shorter than that of Groth, due to a simpler element-wise product argument that may be of independent interest. One of the applications of the current paper is a perfect zap for circuit satisfiability with communication complexity $O(|C|^{1/2+\varepsilon})$.
Keywords. Bilinear pairings, circuit satisfiability, non-interactive zero-knowledge, progression-free sets.

1 Introduction

One of the central concepts in cryptography is zero-knowledge, introduced by Goldwasser, Micali and Rackoff [GMR85]. By using the power of randomness and (usually) interaction, zero-knowledge proofs make it possible for the prover to convince the verifier in the truth of a statement, without leaking any side information. Zero-knowledge proofs are usually required to be statistically complete, statistically sound and computationally zero-knowledge. Computationally-sound proofs are usually known as arguments. Due to wide applications of zero-knowledge in diverse areas of cryptography, it is of utmost importance to construct efficient zero-knowledge proofs. As usually, efficiency of protocols is measured by their round-complexity, communication complexity and computational complexity. In particular, *non-interactive zero-knowledge* proofs are extremely useful, since in many applications the proof is generated once but then have to be verified many times.

While it is known that non-interactive zero-knowledge (NIZK) proofs (or arguments) cannot be constructed in the plain model (i.e., without random oracles or any trusted setup), Blum, Feldman and Micali showed in [BFM88] how to construct NIZK proofs in the common reference string model. During the last years, substantial amount of research has been done towards constructing efficient NIZK proofs (and arguments). Since communication and verifier's computational complexity are arguably more important than prover's computational complexity (a NIZK proof/argument is generated once but can be verified many times), special effort has been put in minimizing these two parameters.

In some very recent NIZK proofs for NP-complete problems, see Tbl. 1, both the communication complexity and verifier's computational complexity are sublinear in a well-defined sense. Already in 1994, Micali [Mic94] proposed sublinear NIZK *arguments* for all NP-languages. However, his CS proofs are based on the PCP theorem (making them computationally unattractive) and on random oracles. Gentry [Gen09] noted that given his new fully-homomorphic cryptosystem, one can construct very efficient NIZK *proofs* for all NP — unfortunately, again in the random oracle model. Another NIZK argument for a NP-complete problem, circuit satisfiability, proposed by Groth in [Gro09], are also based on the random oracle model. While the random oracle model makes it possible to design efficient protocols, it is also well-known, see e.g., [CGH98,BP04], that there are many functionalities that are secure in the random oracle model and insecure in the plain model. Thus, it is important to design efficient NIZK proofs and arguments that do not use random oracles.

Recently, Groth [Gro10] proposed an efficient pairing-based NIZK argument for circuit satisfiability in the common reference string model. Groth's circuit satisfiability argument is based on two more primitive NIZK arguments for element-wise product and permutation, and in particular it inherits the

* Second eprint version, January 12, 2011, with minor revisions.

length-$\Theta(|C|^2)$ common reference string of those two basic arguments. Here, $|C|$ is the circuit size. The security of Groth's NIZK argument is based on two assumptions, a computational one (q-CPDH) and a knowledge one (q-PKE).

Our Contributions. By using a lowerbound on the size of the maximum cardinality of a progression-free set in $[n]$, we improve on [Gro10], by proposing a NIZK argument for circuit satisfiability with common reference string of size $O(|C|^{1+\varepsilon})$. This, in particular, also improves somewhat the computational complexity of all new arguments. Moreover, the new NIZK argument for circuit satisfiability is slightly simpler (requiring one less commitment and one less basic argument) than the on in [Gro10]. Thus, in the new argument, prover's computational complexity is $\Theta(|C|^2)$ multiplications in $\mathbb{Z}_p$ and $O(|C|^{1+\varepsilon})$ exponentiations in the bilinear group, the communication complexity is $\Theta(1)$, and verifier's computational complexity is $\Theta(|C|)$ offline multiplications and $\Theta(1)$ online pairings. (See Tbl. 1.) Finally, our contribution is also conceptual, by clarifying the ideas in [Gro10], and taking them one step further. The connection to progression-free sets is interesting in its own right.

As our underlying observation, we note that Groth's basic arguments are constructed by making use of $\Theta(n^2)$ generators of the underlying group, where n is the input size. Of those possible generators, only $\leq 2n$ generators are really needed in the description of the arguments, while the rest of the generators are inherit to the proof technique. When constructing the NIZK argument for circuit satisfiability, this results in $\Theta(|C|^2)$ used generators that are all part of the common reference string.

Product Argument. In a element-wise product argument, the prover argues that given (multi-)commitments, he knows how to open them to three length-n vectors a_{ij}, $i \in \{1,2,3\}$, such that $a_{3j} = a_{1j}a_{2j}$. The description of Groth's element-wise product argument [Gro10] uses $2n$ generators $g_i = g^{x^i}$ and $g_{i+n} = g^{x^{i(n+1)}}$, for $i \in [n]$. Here, a_1 and a_3 are committed to by using the first n generators, and a_2 is committed to by using the last n generators. Soundness of the product argument is based on inequalities $k(n+2) \neq i$, $k(n+2) \neq i(n+1)$ and $k(n+2) \neq i+j(n+1)$ for any $i,j,k \in [n]$, where $i \neq j$. However, while constructing the actual argument, one needs $\Theta(n^2)$ different products $g_i g_j = g^{x^{i+j(n+1)}}$ of all $n \times n$ generator pairs g_i and g_{j+n}.

Intuitively, the first question was whether either the statement of the argument and the soundness condition can be simplified. For example, can one just use n generators $g_i = g^{x^{\lambda_i}}$ for some well-chosen powers λ_i, so that all a_1, a_2, and a_3 are committed to by using the same generators $g^{x^{\lambda_i}}$? Indeed, it turns out that one can. As we show in Sect. 4, we can do it so that the soundness conditions are simplified: now it is just required that $2\lambda_k \neq \lambda_i$ and $2\lambda_k \neq \lambda_i + \lambda_j$ for any $i \neq j$. That is, in this "simplified case", it suffices that the set of $\Lambda = \{\lambda_i\}$ of exponents λ_i used in generators $g^{x^{\lambda_i}}$ 1) does not contain even numbers, and 2) is progression-free (that is, does not contain arithmetic progressions of length three).

Next, we use some deep results from additive combinatorics. Let $r_3(x)$ be the cardinality of maximal progression-free subset of $[x]$. According to [Beh46,Elk10], $r_3(x) = \Omega(x^{1-\varepsilon})$ for any $\varepsilon > 0$. Thus, there exists a $y = O(n^{1+\varepsilon})$ for any $\varepsilon > 0$, such that $[y]$ contains a size-n progression-free subset $\Lambda = \{\lambda_i\}_{i\in[n]}$. We show that this is even true when Λ is required to only contain odd numbers. (See Thm. 1.)

In Sect. 4, we describe the new element-wise product argument. It uses n generators $g_{\lambda_i} \leftarrow g^{x^{\lambda_i}}$, where $i \in [n]$, for committing to the vectors a_1, a_2, and a_3. While we still need all the "product" generators $g_{\lambda_i} \cdot g_{\lambda_j}$, due to the choice of Λ, there is only $O(n^{1+\varepsilon})$ of different product generators. Thus, the new entry-wise product argument uses a common reference string of length $O(n^{1+\varepsilon})$, and is otherwise asymptotically as efficient as Groth's (with constant communication and verifier's online computation, and with $\Theta(n^2)$ prover's computation). However, in concrete terms, while in [Gro10], prover's computation was dominated by $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$ and $\Theta(n^2)$ exponentiations in underlying bilinear group, in the new argument, the number of exponentiations can be also reduced to $O(n^{1+\varepsilon})$.

Finally, we observe that Groth used different generators to commit to a_1 and a_3 than to a_2, while we use the same n generators to commit to all three different vectors. Thus, the new element-wise product argument can be used more easily in several applications, like recursive element-wise product argument of 2^m elements for some $m > 1$. This is the second reason why the new argument for circuit satisfiability is slightly more efficient than the one from [Gro10].

Permutation Argument. The second basic argument in [Gro10] is for permutation. In the permutation argument, given two (multi-)commitments, the prover aims to convince the verifier that he knows to

	CRS size	Argument size	Prover comp.	Verifier comp.	Assumption
Gentry [Gen09]	$O(1)G$	$\|w\|\operatorname{poly}(\kappa)G$	$\|C\|\operatorname{poly}(\kappa)M$	$\|C\|\operatorname{poly}(\kappa)M$	lattice + random oracle
Groth [Gro09]	$O(\|C\|^{1/2})G$	$O(\|C\|^{1/2})G$	$O(\|C\|)M$	$O(\|C\|)M$	random oracle
Groth [Gro10]	$O(\|C\|^2)G$	$O(1)G$	$O(\|C\|^2)M$	$O(\|C\|)M$	PKE and CPDH
Groth [Gro10]	$O(\|C\|^{2/3})G$	$O(\|C\|^{2/3})G$	$O(\|C\|^{4/3})M$	$O(\|C\|)M$	PKE and CPDH
This paper	$O(\|C\|^{1+\varepsilon})G$	$O(1)G$	$O(\|C\|^2)M$	$O(\|C\|)M$	PKE and CPDH
This paper	$O(\|C\|^{1/3+\varepsilon})G$	$O(\|C\|^{2/3})M$	$O(\|C\|^{4/3})G$	$O(\|C\|)M$	PKE and CPDH
This paper	$O(\|C\|^{1/2+\varepsilon})G$	$O(\|C\|^{1/2})M$	$O(\|C\|^{3/2})G$	$O(\|C\|)M$	PKE and CPDH

Table 1. Comparison of NIZK arguments with sublinear argument size. $|C|$ is the size of circuit, $|w|$ is the length of witness, κ is security parameter, G is the length of group elements and M is the time of single multiplication. For verifier, we count here the total (offline + online) computation

open the commitments to $a_1 = (a_{1j})_{j\in[n]}$ and to $a_2 = (a_{2j})_{j\in[n]}$ respectively, such that for a fixed public permutation ϱ of $[n]$, $a_{2j} = a_{1,\rho(j)}$ for all $j \in [n]$. Similarly to the case of product argument, Groth's permutation argument makes use of $2n$ basic generators and of $\Theta(n^2)$ "product" generators, while we show that one can limit the number of "product" generators to $O(n^{1+\varepsilon})$. The basic idea of our modification is the same as in the case of element-wise product. That is, we commit to a_{1j} and a_{2j} by using generators g_{λ_j}. Additionally, generators $g_{\lambda_{\rho(i)}-\lambda_i}$ are used in the verification.We show that also in this case, it is sufficient that the set $\Lambda = \{\lambda_i\}$ is progression-free and does not contain even numbers. This argument is described in Sect. 5.

NIZK for Circuit Satisfiability. Groth [Gro10] proposed an efficient NIZK argument for circuit satisfiability that crucially used the product and permutation arguments. The circuit satisfiability argument inherits the length-$\Theta(|C|^2)$ common reference string of the subarguments, where $|C|$ is the circuit size. By using the new basic arguments, in Sect. 6 we construct a NIZK argument for circuit satisfiability with common reference string of length $O(|C|^{1+\varepsilon})$. Moreover, that argument has constant communication complexity, while prover's computational complexity is $\Theta(|C|^2)$ multiplications in $\mathbb{Z}_p$ and $O(|C|^{1+\varepsilon})$ exponentiations in the bilinear group, and verifier's offline computational complexity $\Theta(|C|)$ and online computational complexity $\Theta(1)$ bilinear pairings. The new NIZK argument for circuit satisfiability is also slightly more efficient than Groth's argument (omitting one commitment and one element-wise product argument) due to the simpler construction of the element-wise product argument of Sect. 4.

Balancing. As shown by Groth [Gro10], to achieve both sublinear common reference string length and communication, one can apply the same base protocols on length-$n^{1/3}$ inputs $n^{2/3}$ times in parallel. Groth's balanced circuit satisfiability argument has common reference string length $\Theta(n^{2/3})$ and communication $\Theta(n^{2/3})$. Note that prover's computational complexity is decreased to $\Theta(n^{4/3})$ while the verifier's online computational complexity is increased to $\Theta(n^{2/3})$. We refer to [Gro10] for details.

We can use the same balancing with the new argument (though the resulting common reference string will be shorter). In addition, we can balance the new argument by applying the same base protocols on length-$n^{1/2}$ inputs $n^{1/2}$ times in parallel. When balanced like that, the new circuit satsifiability argument has common reference string length $O(|C|^{1/2+\varepsilon})$ and communication $\Theta(|C|^{1/2})$. Prover's computational complexity is $\Theta(|C|^{3/2})$ and verifier's online computational complexity is $\Theta(|C|^{1/2})$. Full comparison is given in Tbl. 1.

Perfect Zaps. The presented NIZK argument is in the common reference string model, and thus requires some trusted third party to construct the common reference string. If trusted setup model is not available, then one can instead construct a sublinear-size zap [DN00] (a 2-move publicly-verifiable witness-indistinguishable argument, where verifier's first message can be reused many times), just by letting the common reference string to be verifier's first message σ, and then letting the prover to verify the well-formedness of the common reference string and then send his argument (polynomially many times). It is straightforward to see, that all common reference strings in this paper are publicly verifiable. This zap will be perfectly witness-indistinguishable. Due to the balancing, the new zap for circuit satisfiability has total communication $O(|C|^{1/2+\varepsilon})$, while Groth's zap from [Gro10] has communication complexity $\Theta(|C|^{2/3})$.

2 Preliminaries

Notation. Let $[n] = \{1, \ldots, n\}$. Let $\mathsf{Mat}_{m,n}(R)$ be the ring of $m \times n$ matrices over ring R. We denote matrices by capital letters (like A), and their elements by corresponding lowercase letters (like a). Thus $A = (a_{ij})$. We denote the ith row of matrix A as A_i. If $y = h^x$ then $\log_h y := x$. For any set A and $n > 1$, let $n.A = \{a_1 + \cdots + a_n : a_j \in A\}$ [TV06]. Let $2\mathbb{Z} + 1$ be the set of odd integers. Let κ be the security parameter. We write $(y; z) \leftarrow (\mathcal{A}||X_{\mathcal{A}})(x)$ if $\mathcal{A}$ on input x outputs y, and $X_{\mathcal{A}}$ on the same input (including the random tape of $\mathcal{A}$) outputs z.

Bilinear groups. Let $G_{\mathsf{bp}}(1^\kappa)$ be a bilinear group generator, that outputs a description of a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow G_{\mathsf{bp}}(1^\kappa)$ such that p is a κ-bit prime, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map (pairing) such that $\forall a, b$, $e(g^a, g^b) = e(g, g)^{ab}$. If g generates $\mathbb{G}$, then $e(g, g)$ generates $\mathbb{G}_T$. Moreover, it is efficient to decide the membership in $\mathbb{G}$ and $\mathbb{G}_T$, group operations and the pairing e are efficiently computable, generators are efficiently sampleable, and the descriptions of the groups and group elements each have length $O(\kappa)$ bits. For the sake of simplicity, we will use symmetric notation. However, with only some extra effort all new arguments can be modified to work together with an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

***q*-Computational Power Diffie-Hellman Assumption [Gro10].** We say that a bilinear group generator G_{bp} is $q(\kappa)$-*CPDH secure*, if for any non-uniform polynomial-time adversary $\mathcal{A}$,

$$\max_i \Pr\left[\begin{array}{l}(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow G_{\mathsf{bp}}(1^\kappa), g \leftarrow \mathbb{G} \setminus \{1\}, (x, \alpha) \leftarrow \mathbb{Z}_p^2 : \\ \mathcal{A}(p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, \ldots, g^{x^q}, g^\alpha, g^{\alpha x}, \ldots, g^{\alpha x^{i-1}}, g^{\alpha x^{i+1}}, \ldots, g^{\alpha x^q}) = g^{\alpha x^i}\end{array}\right]$$

is negligible in κ. As shown in [Gro10], q-CPDH holds in generic group model for any polynomial q. We recall the next lemma from [Gro10].

Lemma 1 (Groth [Gro10]). *If G_{bp} is q-CPDH secure, then given a random common reference string, no non-uniform probabilistic polynomial-time adversary can find a non-trivial linear combination $(a_0, \ldots, a_q)$, such that $\sum_{i=0}^q a_i x^i = 0$, except with a negligible probability.*

***q*-Power Knowledge of Exponent Assumption (*q*-PKE).** Abe and Fehr showed in [AF07] that no statistically zero-knowledge NIZK argument for an NP-complete language can have a "direct black-box" security reduction to a standard cryptographic assumption unless NP $\subseteq$ P/poly. Since then, several other authors have shown related impossibility results, see, e.g., [GW10]. (The best known related result is by Di Crescenzo and Lipmaa [DL08], who construct a 2-message zero-knowledge argument for NP under a knowledge assumption, but without using a common reference string.) Therefore, Groth [Gro10] based his NIZK argument for circuit satisfiability on the next knowledge assumption.

The bilinear group generator G_{bp} is q-*PKE secure* if for any non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ there exists a non-uniform probabilistic polynomial-time extractor $X_{\mathcal{A}}$, such that

$$\Pr\left[\begin{array}{c}(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow G_{\mathsf{bp}}(1^\kappa), g \leftarrow \mathbb{G} \setminus \{1\}, (\alpha, x) \leftarrow \mathbb{Z}_p^*, \\ \sigma \leftarrow (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, \ldots, g^{x^q}, g^\alpha, g^{\alpha x}, \ldots, g^{\alpha x^q}), \\ (c, \hat{c}; a_0, \ldots, a_q) \leftarrow (\mathcal{A}||X_{\mathcal{A}})(\sigma) : \hat{c} = c^\alpha \wedge c \neq \prod_{i=0}^n g^{a_i x^i}\end{array}\right]$$

is negligible. Groth [Gro10] proved that the q-PKE assumption holds in the generic group model. We emphasize that while knowledge assumptions are non-falsifiable, they seem to be more realistic than the random oracle assumption.

Trapdoor Commitment Schemes. A commitment scheme $(G_{\mathsf{com}}, \mathsf{Com}, \mathsf{Open})$ consists of three probabilistic polynomial-time algorithms: a randomized key generation algorithm G_{com}, a randomized commitment algorithm Com and an opening algorithm Open (that may output $\perp$ in the case of failure). It is required that for any $\mathsf{ck} \leftarrow G_{\mathsf{com}}(1^\kappa)$ and for any valid m and r, if $(c, d) = \mathsf{Com}_{\mathsf{ck}}(m; r)$ then

$\mathsf{Open}_{\mathsf{ck}}(c,d)=m$. Here, c is the commitment value and d is the opening value. For the sake of simplicity, we often denote $c \leftarrow \mathsf{Com}_{\mathsf{ck}}(m;r)$. In the case of a doubt, we write $c \leftarrow \mathsf{Com}^1_{\mathsf{ck}}(m;r)$. In the current paper, ck is a part of the common reference string.

Commitment scheme is *computationally binding*, if no non-uniform polynomial-time adversary $\mathcal{A}$ can produce a triple (c,d_1,d_2), such that $\bot \neq \mathsf{Open}_{\mathsf{ck}}(c,d_1) \neq \mathsf{Open}_{\mathsf{ck}}(c,d_2) \neq \bot$, with non-negligible probability. The probability is taken over the choice of ck and the private coins of $\mathcal{A}$. Commitment scheme is *statistically hiding*, if for any commitment key $\mathsf{ck} \in G_{\mathsf{com}}(1^\kappa)$, the statistical distance between distributions $\mathsf{Com}^1_{\mathsf{ck}}(m_1;\cdot)$ and $\mathsf{Com}^1_{\mathsf{ck}}(m_2;\dot{)}$ is negligible in κ.

We use the same commitment scheme as Groth [Gro10] but with a generalized choice of generators. That is, given a group $\mathbb{G}$ of prime order p, a generator $g \in \mathbb{G}$, and random $\alpha, x \leftarrow \mathbb{Z}_p$, the commitment-key is $\mathsf{ck} \leftarrow (g, g_1, \ldots, g_n, \hat{g}, \hat{g}_1, \ldots, \hat{g}_n)$, where $g_i = g^{x^{\lambda_i}}$ and $\hat{g}_i = g^{\alpha x^{\lambda_i}}$ for some fixed integers λ_i such that $0 < \lambda_i < \lambda_{i+1}$. To commit to $a = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$, the committing party chooses a random $r \leftarrow \mathbb{Z}_p$, and defines

$$\mathsf{Com}_{\mathsf{ck}}(a_1,\ldots,a_n;r) := (g^r \cdot \prod_{i=1}^n g_i^{a_i}, \hat{g}^r \cdot \prod_{i=1}^n \hat{g}_i^{a_i}) \ .$$

The opening value is just $(a_1,\ldots,a_n;r)$. As shown by Groth [Gro10], this commitment scheme is computationally binding and statistically hiding under the q-CPDH assumption. Note that $\mathsf{Com}_{\mathsf{ck}}(a;r) = (g^{r+\sum_{i=1}^n a_i x^i}, g^{\alpha(r+\sum_{i=1}^n a_i x^i)})$, thus by the q-PKE assumption, for every committer $\mathcal{A}$ there exists an extractor $X_{\mathcal{A}}$ that can open the commitment, given access to $\mathcal{A}$'s inputs and random tape. Since the commitment scheme is computationally binding, then the extracted opening has to be the same that $\mathcal{A}$ used. Clearly, this commitment scheme is also perfectly trapdoor, with the trapdoor being $\tau = x$: after trapdoor committing $\mathsf{A} \leftarrow \mathsf{Com}_{\mathsf{ck}}(0,\ldots,0;r) = g^r$ for $r \leftarrow \mathbb{Z}_p$, the committer can open it to any $(a_1,\ldots,a_n;r')$ where r' is chosen so that $r' + \sum a_i x^i = r$.

Non-Interactive Zero-Knowledge. Let $R = \{(C,w)\}$ be an efficiently computable binary relation. Here, C is a statement, and w is a witness. Let $L = \{C : \exists w, (C,w) \in R\}$ be an NP-language. Let n be some fixed concrete input length $n = |C|$. For fixed n, we have relation R_n and language L_n. A *non-interactive argument* for R consists of the next probabilistic polynomial-time algorithms: a common reference string generator G_{crs}, a prover P and a verifier V. For $\sigma \leftarrow G_{\mathsf{crs}}(1^\kappa, n)$, $P(\sigma, C, w)$ produces an argument π. The verifier $V(\sigma, C, \pi)$ outputs 1 (accept) or 0 (reject).

A non-interactive argument (G_{crs}, P, V) is *perfectly complete*, if for all $n = \mathrm{poly}(\kappa)$, all $\sigma \leftarrow G_{\mathsf{crs}}(1^\kappa, n)$ and all $(C,w) \in R_n$, it holds that $V(\sigma, C, P(\sigma, C, w)) = 1$.

A non-interactive argument (G_{crs}, P, V) is *computationally sound*, if for all non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$ and all $n = \mathrm{poly}(\kappa)$,

$$\Pr[\sigma \leftarrow G_{\mathsf{crs}}(1^\kappa, n), (C,\pi) \leftarrow \mathcal{A}(\sigma) : C \notin L \wedge V(\sigma, C, \pi) = 1]$$

is negligible.

A non-interactive argument (G_{crs}, P, V) is *perfectly witness-indistinguishable*, if (given that there are several possible witnesses) it is impossible to tell which witness the prover used. That is, for all all $n = \mathrm{poly}(\kappa)$, if $\sigma \in G_{\mathsf{crs}}(1^\kappa, n)$ and $(C,w_0), (C,w_1) \in R_n$, then the distributions $P(\sigma, C, w_0)$ and $P(\sigma, C, w_0)$ are equal.

A non-interactive argument (G_{crs}, P, V) is *perfectly zero-knowledge*, if there exists a polynomial-time simulator $S = (S_1, S_2)$, such that for all stateful interactive non-uniform probabilistic polynomial-time adversaries $\mathcal{A}$ and $n = \mathrm{poly}(\kappa)$,

$$\left| \Pr\left[\begin{array}{l} \sigma \leftarrow G_{\mathsf{crs}}(1^\kappa, n), (C,w) \leftarrow \mathcal{A}(\sigma), \\ \pi \leftarrow P(\sigma, C, w) : \\ (C,w) \in R_n \wedge \mathcal{A}(\pi) = 1 \end{array}\right] - \Pr\left[\begin{array}{l} (\sigma,\tau) \leftarrow S_1(1^\kappa, n), (C,w) \leftarrow \mathcal{A}(\sigma), \\ \pi \leftarrow S_2(\sigma, C, \tau) : \\ (C,w) \in R_n \wedge \mathcal{A}(\pi) = 1 \end{array}\right] \right|$$

is negligible. Here, τ is the *simulation trapdoor*.

3 Progression-Free Sets

A set of positive integers $u_1, u_2, \ldots$ is *progression-free* [TV06], if no three of the numbers are in arithmetic progression, so that $u_i + u_j = 2u_k$ only if $i = j = k$. Let $r_3(n)$ denote the cardinality of the largest progression-free set that belongs to $[n]$.

It is well known that that for any $n > 1$, the next subset is progression-free:

$$T(n) = \{1 \leq i \leq n : \text{no ternary digit of } i \text{ is equal to } 2\} \enspace .$$

Clearly, if $n = 3^k$, then $\sharp T(n) = 2^k - 1$, and thus $r_3(n) = \Omega(n^{\log_3 2})$. The set $T(n)$ can also be efficiently constructed. As shown by Behrend [Beh46], this idea can be generalized to non-ternary bases, with

$$r_3(n) = \Omega\left(\frac{n}{2^{2\sqrt{2}\sqrt{\log_2 n}} \cdot \log_2^{1/4} n}\right) \enspace .$$

Recently, Elkin [Elk10] improved upon Behrend's construction, by showing that

$$r_3(n) = \Omega\left(\frac{n}{2^{2\sqrt{2}\sqrt{\log_2 n}}} \cdot \log_2^{1/4} n\right) \enspace .$$

While both constructions employ the pigeonhole principle, Elkin's methodology can be used to compute the latter progression-free set in quasi-linear time $O(n2^{O(\sqrt{\log n})})$ [Elk10].

On the other hand, Bourgain [Bou98] showed that $r_3(n) = O(n \cdot (\log n/\log\log n)^{1/2})$, and very recently Sanders [San10] showed that $r_3(n) = O(n \cdot (\log\log n)^5/\log n)$. Thus, according to Behrend and Elkin, the minimal y such that $r_3(y) = n$ is $y = O(n^{1+\varepsilon})$ for any $\varepsilon > 0$, while according to Sanders, $y = \Theta(n/\log n^{1-o(1)})$.

We need all members of the progression-free subset also to be odd. For this, we prove the next theorem, which achieves Behrend's bound even in this case. (We can also achieve Elkin's bound, but the corresponding proof is much longer and thus omitted from the submitted version.)

Theorem 1. *For any fixed N, there exists $y = O(N^{1+\varepsilon})$, such that $[y] \cap (2\mathbb{Z}+1)$ contains a progression-free subset Λ of cardinality N.*

Proof. Let $r_3^{odd}(N)$ be the size of the largest progression-free set in $[N]$ that only consists of odd numbers. Assume first that d is an integer.

Let us consider n independent random variables $Y_1, \ldots, Y_n$, such that $Y_1 \in [0, d-1] \cap (2\mathbb{Z}+1)$ and $Y_i \in [0, d-1]$ for $i > 1$. Let $Z_i = Y_i^2$ and $Z = \sum_{i=1}^n Z_i$. Let $C = ([0, d-1] \cap (2\mathbb{Z}+1)) \times [0, d-1]^{n-1}$ be the domain of $Y_1 \times \cdots \times Y_n$. Clearly, $\mathbb{E}(Z_1) = \frac{2}{d-1} \cdot \sum_{j=0}^{(d-1)/2-1} (2j+1)^2 = \frac{d^2}{3} - \frac{2d}{3} = \frac{d^2}{3} + \Theta(d)$ if d is odd, and $\mathbb{E}(Z_1) = \frac{2}{d} \cdot \sum_{j=0}^{d/2-1} (2j+1)^2 = \frac{d^2}{3} - \frac{1}{3} = \frac{d^2}{3} + \Theta(1)$ if d is even. Analogously,

$$\mathbb{E}(Z_i) = \frac{1}{d} \cdot \sum_{j=0}^{d-1} j^2 = \frac{d^2}{3} + \Theta(d)$$

for $i > 0$. Let μ_Z denote the expectation of the random variable Z. Then $\mu_Z = \frac{nd^2}{3} + \Theta(nd)$. Moreover, the variance of Z_i is

$$\mathrm{var}(Z_i) = \mathbb{E}(Z_i^2) - \mathbb{E}(Z_i)^2 = \frac{d^4}{5} + O(d^3) - \left(\frac{d^4}{9} + O(d^3)\right) = \frac{4d^4}{45} + O(d^3)$$

for $i \in [n]$. Thus,

$$\mathrm{var}(Z) = \frac{4nd^4}{45} + O(nd^3) = \frac{4nd^4}{45} \cdot (1 + O(1/d)) \enspace ,$$

and the standard deviation of Z is

$$\sigma_Z = \frac{2\sqrt{n} \cdot d^2}{3 \cdot \sqrt{5}} \cdot (1 + O(1/d)) \enspace .$$

Now, according to Chebyshev's inequality, for any $t > 0$, $\Pr[|Z - \mu_Z] > t \cdot \sigma_Z) \leq \frac{1}{t^2}$. Thus, for a fixed value of $t > 0$, at least $(1 - \frac{1}{t^2})$-fraction of all vectors $a \in C$ have squared ℓ_2-norm $||a||_2^2$ such that $\mu_Z - t \cdot \sigma_Z \leq ||a||_2^2 \leq \mu_Z + t \cdot \sigma_Z$.

Since every $a \in C$ has an integer squared norm, then by the pigeonhole principle, there exists a value $k = K$ such that $\mu_Z - t \cdot \sigma_Z \leq K \leq \mu_Z + t \cdot \sigma_Z$, such that at least

$$\left(1 - \frac{1}{t^2}\right) \cdot \frac{1}{2t\sigma_Z + 1} \cdot \frac{d-1}{2} \cdot d^{n-1}$$

vectors from C have squared norm K. Let Λ' be this set. Thus,

$$|\Lambda'| \geq \left(1 - \frac{1}{t^2}\right) \cdot \frac{1}{2t\sqrt{n} \cdot d^2 + 1} \cdot \frac{3\sqrt{5}}{2} \cdot (1 - O(1/d)) \cdot \frac{d-1}{2} \cdot d^{n-1} \geq \frac{d^{n-2}}{\sqrt{n}} \cdot c$$

for a fixed positive small constant $c = c(t)$.

Setting $c = c(2)$ and $d = N^{1/n}/2$, we get $|\Lambda'| = \Omega(N^{1-2/n}/(2^n\sqrt{n}))$. Let us fix $n = \lceil \sqrt{2 \cdot \log_2 N} \rceil$. Then

$$|\Lambda'| = \Omega(N/(2^{2\sqrt{2}\cdot\sqrt{\log_2 N}} \cdot \log_2^{1/4} N)) \ .$$

Since all vectors in Λ' have the same norm $\sqrt{K}$, then for any three vectors $a, a', a''' \in \Lambda'$, $a + a' \neq a''$.

Next, we consider the coordinates of vectors from $\Lambda' \subset C$ as $(2d)$-nary digits of an integer. That is, for $a = (a_1, \ldots, a_n) \in \Lambda'$, let $\hat{a} = \sum_{i=0}^{n-1} a_{i+1}(2d)^i$. Define $\Lambda = \{\hat{a} : a \in \Lambda\}$. Due to the choice of C, Λ clearly only contains odd numbers. Since $\Lambda' \subset [0, d-1]^n$, the mapping $a \mapsto \hat{a}$ is one-to-one. Since for every $a \in \Lambda'$, $0 < a' \leq (2d)^n - 1 = N - 1$, we have

$$|\Lambda| = \Omega\left(\frac{N}{2^{2\sqrt{2}\cdot\sqrt{\log_2 N}} \cdot \log_2^{1/4} N}\right).$$

Finally, Λ is progression-free. If it were not, for some $\hat{a}, \hat{b}, \hat{c} \in \Lambda$, $2c = a + b$. But then $c = \sum_{i=0}^{n-1}(a_{i+1} + b_{i+1}/2) \cdot (2d)^i = \sum_{i=0}^{n-1} c_{i+1} \cdot (2d)^i$. Since all coordinates $a_i, b_i, c_i \in [0, d-1]$, then $2c_i = a_i + b_i$ for every i. Thus also $2c = a + b$, which contradicts the assumption that $||a|| = ||b|| = ||c||$. Thus, Λ is the required progression-free set.

Finally, if d is not integer, then one considers the same construction with d replaced with $\lfloor d \rfloor$. The claim holds. □

4 New Product Argument

In an *element-wise product argument*, the prover aims to convince the verifier that for given three commitments A_1, A_2 and A_3, he knows how to open them as $\mathsf{A}_i = \mathsf{Com}_{\mathsf{ck}}(a_{i1}, \ldots, a_{in}; r_i)$, such that $a_{3j} = a_{1j} \cdot a_{2j}$ for $j \in [n]$. In [Gro10], Groth constructed an element-wise product argument where the communication and the verifier's online computation are $\Theta(1)$, the verifier's offline computation is $\Theta(n)$, and the prover's computation and the length of the common reference string are $\Theta(n^2)$. In this section, we propose a new argument that is closely related to that of Groth in [Gro10], but with the common reference string of length $O(n^{1+\varepsilon})$ for any $\varepsilon > 0$. In addition, prover's concrete computation will be improved.

The full argument is given by Prot. 1. Intuitively, the discrete logarithm (on basis $h = e(g, g)$) of the left side of the verification equation Eq. (1) in Prot. 1 is a degree-$2n$ formal polynomial in x. That formal polynomial has a monomial cx^d with non-zero b exactly if either

1. $d = 0$, $d = \lambda_i$ for some $i \in [n]$, or $d = \lambda_i + \lambda_j$ for some $i, j \in [n]$ such that $i \neq j$, or
2. $d = 2\lambda_i$ for some $i \in [n]$.

In the first case, we say that d is a type 1 power, in the second case we say that d is a type 2 power. The coefficients corresponding to the type 2 powers ($d = 2\lambda_i$ for some $i \in [n]$) cancel out if and only if $a_{3j} = a_{1j}a_{2j}$ for $j \in [n]$.

The prover creates a short argument π that contains all type 1 monomials of the formal polynomial, and the verifier checks that those two formal polynomials are equal via Eq. (1). For soundness, it is required that the type 1 and type 2 powers do not intersect, that is, given the set $\Lambda = \{\lambda_i\}$ of all possible powers, $2\lambda_i \neq 0$, $2\lambda_i \neq \lambda_i + \lambda_j$ for $i \neq j$, and $\lambda_i \neq 0$. As we show, for this it is sufficient that Λ is a progression-free set of odd integers. To achieve soundness, the common reference string contains a different generator $\hat{g}$, and the values $\hat{g}^{x^d}$ for type 1 powers d. If also the the second verification equation holds, then the verifier is convinced that the π is correctly formed (i.e., the prover knows a representation of $\log_g \pi$ as a formal polynomial in x that has only type 1 powers), and then via the first verification equation, the verifier is convinced that $a_{3j} = a_{1j} \cdot a_{2j}$ for all $j \in [n]$.

Theorem 2. *Let G_{bp} be a $|2.\Lambda|$-CPDH secure bilinear group generator. Then Prot. 1 is a perfectly complete, computationally sound, and perfectly witness-indistinguishable element-wise product argument.*

1. **CRS generation** $G_{\mathsf{crs}}(1^\kappa)$**:** Let $(p,\mathbb{G},\mathbb{G}_T,e)\leftarrow G_{\mathsf{bp}}(1^\kappa)$. Choose $\lambda_i>0$, $i\in[n]$, such that (a) $\lambda_i+\lambda_j\neq 2\lambda_k$ if $i\neq j$, and (b) $2\lambda_i\neq\lambda_j$ for any i,j. Let $\Lambda=\{\lambda_i\}_{i\in[n]}$, $2.\Lambda:=\{i+j:i,j\in\Lambda\}$, and $\hat{S}:=\Lambda\cup\{i+j\in 2.\Lambda:i\neq j\}$. For $1\le i\le|2.\Lambda|$, let $g_i\leftarrow g^{x^i}$. Here and in what follows, let $h\leftarrow e(g,g)$. Choose a random generator $\hat{g}$, and $\alpha\leftarrow\mathbb{Z}_p$. The common reference string is
$$\sigma=\left(p,\mathbb{G},\mathbb{G}_T,e,g,\left\{g^i\right\}_{2\le i\le|2.\Lambda|},\hat{g},\left\{\hat{g}_i\leftarrow\hat{g}^{i}\right\}_{i\in\hat{S}}\right)\ ,$$
moreover, $\mathsf{ck}\leftarrow\sigma$.
2. **Argument generation:** Given commitments $\mathsf{A}_i\leftarrow\mathsf{Com}_{\mathsf{ck}}(a_{i1},\ldots,a_{in};r_i)=(g^{r_i}\cdot\prod_{j=1}^n g^{a_{ij}x^{\lambda_j}},\hat{g}^{r_i}\cdot\prod_{j=1}^n\hat{g}^{a_{ij}x^{\lambda_j}})$ for $i\in[3]$, the prover defines
$$\pi\leftarrow g^{r_1r_2}\cdot\prod_{i=1}^n g_{\lambda_i}^{r_1a_{2i}+r_2a_{1i}-r_3}\cdot\prod_{i=1}^n\prod_{j=1:j\neq i}^n g_{\lambda_i+\lambda_j}^{a_{1i}a_{2j}-a_{3i}}\ ,$$
$$\hat{\pi}\leftarrow\hat{g}^{r_1r_2}\cdot\prod_{i=1}^n\hat{g}_{\lambda_i}^{r_1a_{2i}+r_2a_{1i}-r_3}\cdot\prod_{i}\prod_{j=1:j\neq i}^n\hat{g}_{\lambda_i+\lambda_j}^{a_{1i}a_{2j}-a_{3i}}\ .$$
The prover sends $(\pi,\hat{\pi})$ to the verifier as the NIZK argument.
3. **Verification:** the verifier accepts iff
$$e(\mathsf{A}_1,\mathsf{A}_2)/e(\mathsf{A}_3,\prod_{i=1}^n g_i)=e(g,\pi)\ ,\quad\text{and}\qquad e(g,\hat{\pi})=e(\hat{g},\pi)\ . \tag{1}$$

Protocol 1: New argument for element-wise product.

Proof. Completeness. The second verification is straightforward. For the first one, note that
$$\begin{aligned}\log_h e(\mathsf{A}_1,\mathsf{A}_2)=&(r_1+\sum_{i=1}^n a_{1i}x^{\lambda_i})(r_2+\sum_{i=1}^n a_{2i}x^{\lambda_i})=r_1r_2+\sum_{i=1}^n(r_1a_{2i}+r_2a_{1i})x^{\lambda_i}+\sum_{i=1}^n\sum_{j=1}^n a_{1i}a_{2j}x^{\lambda_i+\lambda_j}\\=&r_1r_2+\sum_{i=1}^n(r_1a_{2i}+r_2a_{1i})x^{\lambda_i}+\sum_{i=1}^n a_{1i}a_{2i}x^{2\lambda_i}+\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{1i}a_{2j}x^{\lambda_i+\lambda_j}\end{aligned}$$
while
$$\log_h e(\mathsf{A}_3,\prod_{i=1}^n g_i)=(r_3+\sum_{i=1}^n a_{3i}x^{\lambda_i})(\sum_{i=1}^n x^{\lambda_i})=r_3\sum_{i=1}^n x^{\lambda_i}+\sum_{i=1}^n a_{3i}x^{2\lambda_i}+\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{3i}x^{\lambda_i+\lambda_j}\ .$$
Thus
$$\begin{aligned}\log_h(e(\mathsf{A}_1,\mathsf{A}_2)/e(\mathsf{A}_3,\prod_{i=1}^n g_i))=&r_1r_2+\sum_{i=1}^n(r_1a_{2i}+r_2a_{1i}-r_3)x^{\lambda_i}+\\&\sum_{i=1}^n(a_{1i}a_{2i}-a_{3i})x^{2\lambda_i}+\sum_{i=1}^n\sum_{j=1:j\neq i}^n(a_{1i}a_{2j}-a_{3i})x^{\lambda_i+\lambda_j}\ .\end{aligned}$$
If the prover is honest and $a_{3i}=a_{1i}a_{2i}$, then
$$\log_h e(\mathsf{A}_1,\mathsf{A}_2)/e(\mathsf{A}_3,\prod_{i=1}^n g_i)=r_1r_2+\sum_{i=1}^n(r_1a_{2i}+r_2a_{1i}-r_3)x^{\lambda_i}+\sum_{i=1}^n\sum_{j=1:j\neq i}^n(a_{1i}a_{2j}-a_{3i})x^{\lambda_i+\lambda_j}\ ,$$
and thus
$$e(\mathsf{A}_1,\mathsf{A}_2)/e(\mathsf{A}_3,\prod_{i=1}^n g_i)=h^{r_1r_2}\cdot\prod_{i=1}^n h_{\lambda_i}^{r_1a_{2i}+r_2a_{1i}-r_3}\cdot\prod_{i=1}^n\prod_{j=1:j\neq i}^n h_{\lambda_i+\lambda_j}^{a_{1i}a_{2j}-a_{3i}}=e(g,\pi)\ .$$

Thus, if $a_{3i} = a_{1i}a_{2i}$ for $i \in [n]$, then the first verification succeeds.

Soundness. For the soundness, we need that π can be represented as a product of only type 1 powers, that is, as some powers of g_{λ_i} and $g_{\lambda_i+\lambda_j}$, with $i \neq j$. The choice of set Λ guarantees that type 1 and type 2 powers do not intersect. The value $\hat{\pi}$ is used to verify that the latter is true. Now, the common reference string only contains values $\hat{g}_i$ only for such generators. If $a_{1j} \neq a_{2j} \cdot a_{2j}$ for some j, then there is a non-trivial linear combination of $1, x, \ldots, x^{|2.\Lambda|}$, and thus by Lem. 1 one has breached the q-CPDH assumption.

Witness-Indistinguishability: follows straightforwardly from the fact that there is exactly one possible argument $(\pi, \hat{\pi})$ that satisfies the verification equations. □

Theorem 3. *Let Λ be as described in Thm. 1. Let $\varepsilon > 0$. The communication (argument size) of the argument in Prot. 1 is two group elements. Prover's computational complexity is $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$ and $O(n^{1+\varepsilon})$ exponentiations in $\mathbb{G}$. Verifier's computational complexity is dominated by $n-1$ offline multiplications and 5 online bilinear pairings. The common reference string has length $O(n^{1+\varepsilon})$ for any $\varepsilon > 0$.*

Proof. It is clear that with this choice of Λ, the size of the common reference string is $\Theta(|\hat{S}|) = O(n^{1+\varepsilon})$. For prover's computation, note that π can be defined as

$$\pi \leftarrow g^{r_1 r_2} \cdot \prod_{i=1}^{n} g_{\lambda_i}^{r_1 a_{2i} + e_2 a_{1i} - r_3} \cdot \prod_{k=1}^{|2.\Lambda|} g_k^{\sum_{i,j \neq i : \lambda_i + \lambda_j = k} a_{1i} a_{2j} - a_{3i}} ,$$

and similarly for $\hat{\pi}$. Other statements are straightforward to prove. □

Comparison to [Gro10]. In Groth's product argument [Gro10], A_1 and A_2 are committed to by using two different tuples of generators $((g_i)_{i\in[n]}$ and $(g_{i(n+1)})_{i\in[n]}$, respectively). Since by multiplying those generators together, one gets $\Theta(n^2)$ intermediate generators that all have to be present in the common reference string, Groth's product argument requires the common reference string to be $\Theta(n^2)$ bits long. Secondly, the fact that we use the same n generators to commit to all A_1, A_2, and A_3 means that we can more readily reuse these commitments in different arguments. This will be seen in the new NIZK argument for circuit satisfiability, where Groth needed an additional argument to show that some commitments A and A' commit to the same values but under two different generator tuples. Finally, in Groth's argument, the prover performed $\Theta(n^2)$ exponentiations in $\mathbb{G}$ and $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$. Depending on the concrete implementation, prover's computation in the new product argument may be significantly more efficient. The same comment applies for the new permutation argument of Sect. 5 and also for the new argument for circuit satisfiability in Sect. 6.

5 New Permutation Argument

Assume $\mathsf{A}_1 = \mathsf{Com}_{\mathsf{ck}}(a_{11}, \ldots, a_{1n}; r_1) = g^{r_1} \cdot \prod_{j=1}^{n} g_{\lambda_j}^{a_{1j}}$ and $\mathsf{A}_2 = \mathsf{Com}_{\mathsf{ck}}(a_{21}, \ldots, a_{2n}; r_2) = g^{r_2} \cdot \prod_{j=1}^{n} g_{\lambda_j}^{a_{2j}}$. In a *permutation argument*, the prover aims to convince the verifier that for a publicly known permutation ϱ, $a_{2j} = a_{1,\varrho(j)}$. In [Gro10], Groth proposed a permutation argument that had similar complexity as his product argument, including a $\Theta(n^2)$-bit long common reference string. In this section, we propose a permutation argument that has a $O(n^{1+\varepsilon})$-bit common reference string, for any $\varepsilon > 0$.

The proposed permutation argument, see Prot. 2, follows the same outline as described in the beginning of Sect. 4. The logarithm of the left side of the first verification equation in Prot. 2 is a degree-$2n$ formal polynomial in x. The set of powers of x that are present in this formal polynomial are divided into the following two types: 1) $d = \lambda_i$, $d = \lambda_{\varrho(i)} - \lambda_i$, $d = \lambda_i + \lambda_j$, and $d = 2\lambda_{\varrho(j)} + \lambda_i - \lambda_j$ for $i, j \in [n]$, $i \neq j$ and ϱ being an arbitrary permutation, and 2) $d = 2\lambda_i$. For soundness, it is thus required that those to sets do not intersect for any permutation ϱ; we will achieve this by choosing the set Λ carefully (in fact, the same choice as in Sect. 4 suffices), and then including only elements $\tilde{g}_d$, where d is a type 1 power, to the common reference string. The full argument is given in Prot. 2.

Theorem 4. *Let G_{bp} be a $|4.\Lambda|$-CPDH secure bilinear group generator. Then Prot. 2 is a perfectly complete, computationally sound and perfectly witness-indistinguishable non-interactive permutation argument.*

1. **Common reference string generation:** Set $(p,\mathbb{G},\mathbb{G}_T,e)\leftarrow G_{\mathsf{bp}}(1^\kappa)$. Let $\Lambda=\{\lambda_i\}$ be defined exactly as in the NIZK argument of Sect. 4, and let $3.\Lambda=\{i+j+k : i,j,k\in\Lambda\}$. Define

$$\tilde{S}\leftarrow \Lambda\cup\{\lambda_i-\lambda_j : i,j\in[n]\}\cup\{\lambda_i+\lambda_j : i,j\in[n] : i\neq j\}\cup\{2\lambda_k+\lambda_i-\lambda_j : i,j,k\in[n], i\neq j\} \ .$$

   Denote $\beta\leftarrow\mathbb{Z}_p$, $g_i\leftarrow g^{x^i}$, $\tilde{g}_i\leftarrow g^{\beta x^i}$ and $\tilde{g}\leftarrow\tilde{g}_1$. The common reference string is equal to

$$\sigma\leftarrow\left(p,\mathbb{G},\mathbb{G}_T,e,g,\{g_i\}_{-|\Lambda|<i<|3.\Lambda|},\tilde{g},\{\tilde{g}_i\}_{i\in\tilde{S}}\right) \ .$$

   Moreover, $\mathsf{ck}\leftarrow\sigma$.
2. **Argument generation:** Given commitments $\mathsf{A}_i\leftarrow\mathsf{Com}_{\mathsf{ck}}(a_{i1},\ldots,a_{in};r_i)=(g^{r_i}\cdot\prod_{j=1}^n g^{a_{ij}x^{\lambda_j}},\tilde{g}^{r_i}\cdot\prod_{j=1}^n\tilde{g}^{a_{ij}x^{\lambda_j}})$ for $i\in[2]$, and a permutation ϱ on $[n]$, the prover defines

$$\pi\leftarrow\prod_{i=1}^n g_{\lambda_i}^{r_1}\cdot\prod_{i=1}^n g_{\lambda_{\varrho(i)}-\lambda_i}^{r_2}\cdot\prod_{i=1}^n\prod_{j=1:j\neq1}^n g_{\lambda_i+\lambda_j}^{a_{1i}}\cdot\prod_{i=1}^n\prod_{j=1:j\neq1}^n g_{2\lambda_{\varrho(j)}+\lambda_i-\lambda_j}^{a_{2i}} \ ,$$
$$\tilde{\pi}\leftarrow\prod_{i=1}^n \tilde{g}_{\lambda_i}^{r_1}\cdot\prod_{i=1}^n \tilde{g}_{\lambda_{\varrho(i)}-\lambda_i}^{r_2}\cdot\prod_{i=1}^n\prod_{j=1:j\neq1}^n \tilde{g}_{\lambda_i+\lambda_j}^{a_{1i}}\cdot\prod_{i=1}^n\prod_{j=1:j\neq1}^n \tilde{g}_{2\lambda_{\varrho(j)}+\lambda_i-\lambda_j}^{a_{2i}} \ .$$

   The prover sends $(\pi,\tilde{\pi})$ to the verifier as the argument.
3. **Verification:** the verifier accepts if

$$e(\mathsf{A}_1,\prod_{j=1}^n g_{\lambda_j})/e(\mathsf{A}_2,\prod_{j=1}^n g_{\lambda_{\varrho(j)}-\lambda_j})=e(g,\pi)$$

   and $e(g,\tilde{\pi})=e(\tilde{g},\pi)$.

Protocol 2: New argument for permutation

Proof. Completeness. The second verification is straightforward. For the first verification, note that

$$\log_h e(\mathsf{A}_1,\prod_{i=1}^n g_{\lambda_i})=(r_1+\sum_{i=1}^n a_{1i}x^{\lambda_i})(\sum_{i=1}^n x^{\lambda_i})=r_1\sum_{i=1}^n x^{\lambda_i}+\sum_{i=1}^n a_{1i}x^{2\lambda_i}+\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{1i}x^{\lambda_i+\lambda_j} \ ,$$

and

$$\begin{aligned}\log_h e(\mathsf{A}_2,\prod_{i=1}^n g_{2\lambda_{\varrho(i)}-\lambda_i})=&(r_2+\sum_{i=1}^n a_{2i}x^{\lambda_i})(\sum_{i=1}^n x^{2\lambda_{\varrho(i)}-\lambda_i})\\ =&r_2\sum_{i=1}^n x^{\lambda_{\varrho(i)}-\lambda_i}+\sum_{i=1}^n a_{2i}x^{2\lambda_{\varrho(i)}}+\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{2i}x^{2\lambda_{\varrho(j)}+\lambda_i-\lambda_j} \ .\end{aligned}$$

Thus,

$$\begin{aligned}\log_h\left(e(\mathsf{A}_1,\prod_{i=1}^n g_{\lambda_i})/e(\mathsf{A}_2,\prod_{i=1}^n g_{\lambda_{\varrho(i)}-\lambda_i})\right)=&r_1\sum_{i=1}^n x^{\lambda_i}-r_2\sum_{i=1}^n x^{\lambda_{\varrho(i)}-\lambda_i}+\\ &\sum_{i=1}^n(a_{1,\varrho(i)}-a_{2i})x^{2\lambda_{\varrho(i)}}+\\ &\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{1i}x^{\lambda_i+\lambda_j}-\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{2i}^{2\lambda_{\varrho(j)}+\lambda_i-\lambda_j} \ .\end{aligned}$$

If the prover is honest, that is, $a_{2i}=a_{1,\varrho(i)}$ for $i\in[n]$, then

$$\begin{aligned}\log_h(e(\mathsf{A}_1,\prod_{i=1}^n g_{\lambda_i})/e(\mathsf{A}_2,\prod_{i=1}^n g_{\lambda_{\varrho(i)}-\lambda_i}))=&r_1\sum_{i=1}^n x^{\lambda_i}-r_2\sum_{i=1}^n x^{\lambda_{\varrho(i)}-\lambda_i}+\\ &\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{1i}x^{\lambda_i+\lambda_j}-\sum_{i=1}^n\sum_{j=1:j\neq i}^n a_{2i}^{2\lambda_{\varrho(j)}+\lambda_i-\lambda_j} \ .\end{aligned}$$

Thus, with the choice of π as in Prot. 2, the first verification equation holds.

Soundness. As long as

$$\lambda_i \neq 2\lambda_k \wedge \lambda_i \neq \lambda_{\varrho(k)} - \lambda_k \wedge \lambda_i \neq 2\lambda_{\varrho(j)} - \lambda_j \wedge \lambda_i \neq \lambda_j \tag{2}$$

for $i, j, k \in [n]$ and $i \neq j$, we can use a similar argument to the previous section. The first inequality holds because Λ does not contain even numbers. For the second inequality, we need that $\lambda_{\rho(k)} \neq \lambda_k + \lambda_i$. For the third inequality, we need that $2\lambda_k \neq \lambda_i + \lambda_j$ for any $i \neq j$ and any k. Both inequalities hold since the set $\{\lambda_i\}$ is progression-free. The fourth inequality holds automatically.

Thus, similarly to the previous section, we need to show that

$$e(\mathsf{A}_1, \prod_{j=1}^{n} g_{\lambda_j}) / e(\mathsf{A}_2, \prod_{j=1}^{n} g_{\lambda_{\varrho(j)}-\lambda_j})$$

can be expressed as a product of some powers of g_{λ_i}, $g_{\lambda_{\varrho(i)}-\lambda_i}$, $g_{\lambda_i+\lambda_j}$ and $g_{2\lambda_{\varrho(j)}+\lambda_i-\lambda_j}$. For this we have defined a value π that cancels out all other powers of x in the exponents. Finally, $\tilde{\pi}$ is used to check that no other powers of x are used at all. This follows from the fact that the common reference string contains only values $\tilde{g}_\ell$ where either $\ell = \lambda_i + \lambda_j$ or $\ell = 2\lambda_k + \lambda_i - \lambda_j$ for some k and $i \neq j$. The statement follows from Lem. 1.

Witness-Indistinguishability: follows straightforwardly from the fact that there is exactly one possible argument $(\pi, \tilde{\pi})$ that satisfies the verification equations. □

Theorem 5. *Let Λ be the set from Thm. 1. Let $\varepsilon > 0$. The communication complexity (i.e., the argument size) of the argument in Prot. 2 is 2 group elements. Prover's computational complexity is $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$ and $O(n^{1+\varepsilon})$ exponentiations in $\mathbb{G}$. Verifier's computational complexity is dominated by $n-1$ offline multiplications and 5 online bilinear pairings. The common reference string has length $O(n^{1+\varepsilon})$.*

Proof. Similar to the proof of Thm. 3. □

Comparison to [Gro10]. he main difference is that instead of n generators g_{λ_i}, $i \in [n]$, Groth used $2n$ generators $g_i \in [n]$ and $g_{i(n+1)} \in [n]$ and this time also their their products in the verification equations. Since there are $\Theta(n^2)$ such products that have to be given in the common reference string, the common reference string length is $\Theta(n^2)$. In the new permutation argument, there are only $O(n^{1+\varepsilon})$ such products, and thus the common reference string is also $O(n^{1+\varepsilon})$. Finally, in Groth's argument, prover's computation was dominated by $\Theta(n^2)$ exponentiations in $\mathbb{G}$.

6 New NIZK Argument for Circuit Satisfiability

In a *NIZK argument for circuit satisfiability* (which is well-known to be NP-complete), the prover and the verifier share a circuit C. The prover aims to prove in non-interactive zero-knowledge that she knows an assignment of input values that makes the circuit output 1.

We have to take some special care about the used trapdoor commitment scheme. Namely, in the element-wise product argument and the permutation argument, the commitment keys were different, depending on different generators $\hat{g}$ and $\tilde{g}$, respectively. Since here we use both types of arguments with the same commitment keys, we cannot allow it here. Similarly to Groth [Gro10], we instead introduce a new generator g', and use that in the commitment scheme. We will also change the element-wise product argument and the permutation argument. In say the element-wise product argument, the common reference string now also contains $(g', \{g'_i \leftarrow (g')^i\}_{i \leq |2.\Lambda|})$, the argument contains an additional value $\pi' = \pi^\gamma$ (which is computed by the prover like π, but using generators g'_i), and the verifier additionally checks that $e(g, \pi') = e(g', \pi)$. It is clear that this version of the element-wise product also remains secure. The permutation argument is changed in the same way.

For the sake of simplicity, assume that the circuit is only composed of the NAND gates. Let the circuit have $n-1$ gates. Assuming that the output gate of the circuit is $n-1$, we add on top of the circuit another output gate n, which just NAND's the output of the $n-1$th gate with itself, $U_n = \neg(U_{n-1} \wedge U_{n-1})$. Note that then $U_n = \neg U_{n-1} = 1 - U_{n-1}$. For every gate $j \in [n]$ of the resulting circuit C, let the input

1. **Common reference string generation:** Set $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow G_{\mathsf{bp}}(1^\kappa)$. Define Λ, $3.\Lambda$ and $\hat{S}$ as in Prot. 1, and $\tilde{S}$ as in Prot. 2. Generate random $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_p$. Set $\hat{g} \leftarrow g^\alpha$, $\tilde{g} \leftarrow g^\beta$, and $g' \leftarrow g^\gamma$. Denote $\hat{g}_i \leftarrow g^{\alpha x^i}$, $\tilde{g}_i \leftarrow g^{\beta x^i}$, and $g'_i \leftarrow g^{\gamma x^i}$. The common reference string is
$$\sigma \leftarrow (p, \mathbb{G}, \mathbb{G}_T, e, g, \{g_i\}_{-|\Lambda|<i<|3.\Lambda|}, \hat{g}, \{\hat{g}_i\}_{i\in\hat{S}}, \tilde{g}, \{\tilde{g}_i\}_{i\in\tilde{S}}, g', \{g'_i\}_{-|\Lambda|<i<|3.\Lambda|}) \ .$$
Set $\mathsf{ck} \leftarrow (p, \mathbb{G}, \mathbb{G}_T, e, g, \{g_i\}_{-|\Lambda|<i<|3.\Lambda|}, g', \{g'_i\}_{-|\Lambda|<i<|3.\Lambda|})$. In this argument, thus $\mathsf{Com}_{\mathsf{ck}}(a_1, \ldots, a_n; r) := (g^r \cdot \prod_{j=1}^n g_j^{a_j}, (g')^r \cdot \prod_{j=1}^n (g'_j)^{a_j})$.
2. Argument:
   (a) The prover generates corresponding random strings $r_1, \ldots, r_5 \leftarrow \mathbb{Z}_q$, and then computes
   $$\mathsf{LR} \leftarrow \mathsf{Com}_{\mathsf{ck}}(L_1, \ldots, L_n, R_1, \ldots, R_n; r_1) \ , \qquad \mathsf{RL} \leftarrow \mathsf{Com}_{\mathsf{ck}}(R_1, \ldots, R_n, L_1, \ldots, L_n; r_2) \ ,$$
   $$\mathsf{RZ} \leftarrow \mathsf{Com}_{\mathsf{ck}}(R_1, \ldots, R_n, 0, \ldots, 0; r_3) \ , \qquad \mathsf{OZ} \leftarrow \mathsf{Com}_{\mathsf{ck}}(U_1, \ldots, U_n, 0, \ldots, 0; r_4) \ ,$$
   $$\mathsf{OX} \leftarrow \mathsf{Com}_{\mathsf{ck}}(U_1, \ldots, U_n, X_1, \ldots, X_n; r_5) \ .$$
   He sends LR, RL, RZ, OZ, and OX to the verifier.
   (b) The prover proves that he knows an opening of LR that consists of Boolean values, by using the element-wise product argument, by showing that $\mathsf{Open}_{\mathsf{ck}}(\mathsf{LR})^2 = \mathsf{Open}_{\mathsf{ck}}(\mathsf{LR})$.
   (c) The prover proves that LR, RL and RZ are mutually consistent:
      i. Show that LR and RL are mutually consistent by a permutation argument.
      ii. Show that RL and RZ are mutually consistent by showing that $\mathsf{Open}_{\mathsf{ck}}(\mathsf{RZ})$ is an entry-wise product of $\mathsf{Open}_{\mathsf{ck}}(\mathsf{RL})$ and $\mathsf{Open}_{\mathsf{ck}}(\prod_{i=1}^n g_{\lambda_i}) = (1, \ldots, 1, 0, \ldots, 0)$.
   (d) The prover proves that OX and OZ are mutually consistent by showing that $\mathsf{Open}_{\mathsf{ck}}(\mathsf{OZ})$ is an entry-wise product of $\mathsf{Open}_{\mathsf{ck}}(\mathsf{OX})$ and $\mathsf{Open}_{\mathsf{ck}}(\prod_{i=1}^{n-1} g_{\lambda_i})$.
   (e) The prover proves that the opening of OZ is a NAND of the openings of LR and RZ, or more precisely that $\mathsf{Open}_{\mathsf{ck}}(\prod_{i=1}^{n-1} g_{\lambda_i} \cdot \mathsf{OZ}^{-1}) = \mathsf{Open}_{\mathsf{ck}}(\mathsf{LR}) \cdot \mathsf{Open}_{\mathsf{ck}}(\mathsf{RZ})$: by using the element-wise product argument.
   (f) The prover shows that the values are internally consistent with the wires:
      i. Create a permutation ϱ' on $[2n]$, such that for any values $L_{i_1}, \ldots, L_{i_s}, R_{j_1}, \ldots, R_{j_t}$ that correspond to the same wire, ϱ' contains a cycle $i_1 \to i_2 \to \cdots \to i_s \to j_1 + n \to \cdots \to j_t + n \to i_1$.
      ii. Give a permutation argumentation that $\mathsf{Open}_{\mathsf{ck}}(\mathsf{LR})$ is related to itself by permutation ϱ'. This shows that $L_{i_1} = \ldots L_{i_s} = R_{j_1} = R_{j_t}$.
   (g) The prover gives a permutation argument that shows that the gate input values $\mathsf{Open}_{\mathsf{ck}}(\mathsf{LR})$ and the "output" values $\mathsf{Open}_{\mathsf{ck}}(\mathsf{OX})$ are mutually consistent.

Protocol 3: New NIZK argument for circuit satisfiability

wire of its jth gate be L_j and R_j, and let U_j be one of its output wires. We let X_j be other values that correspond to some L_k or R_k (e.g., inputs to the circuit, or duplicates of output wires). Note that $(U_1, \ldots, U_n, X_1, \ldots, X_n)$ is chosen so that for some permutation ϱ, (U, X) is a ϱ-permutation of (L, R). The argument is given by Prot. 3.

Theorem 6. *Let G_{bp} be a $|4.\Lambda|$-CPDH and $|4.\Lambda|$-PKE secure bilinear group generator. Then Prot. 3 is a perfectly complete, computationally sound and perfectly zero-knowledge non-interactive argument for circuit satisfiability.*

Proof. Perfect completeness: follows from the perfect completeness of the product and permutation arguments.

Computational soundness: Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time adversary that creates a circuit C and an accepting NIZK argument π. By the q-PKE assumption, there exists a non-uniform probabilistic polynomial-time extractor $X_{\mathcal{A}}$ that, running on the same input and seeing $\mathcal{A}$'s random tape, extracts all openings. From the soundness of he product and permutation arguments it follows that by the q-CPDH assumption, the corresponding relations are satisfied between the opened values. Moreover, by the q-CPDH assumption, the opened values belong to corresponding sets $\hat{S}$ and $\tilde{S}$.

Let $(L_1, \ldots, L_n, R_1, \ldots, R_n)$ be the opening of LR and let $(U_1, \ldots, U_n, X_1, \ldots, X_n)$ be the opening of OX. The first multiplication argument shows that $L_i, R_i \in \{0, 1\}$. The second and the third argument show that RZ commits to $(R_1, \ldots, R_n, 0, \ldots, 0)$ and is thus consistent with the opening of LR. The fourth argument gives us that OZ commits to $(U_1, \ldots, U_{n-1}, U_n = 0, 0, \ldots, 0)$ which is consistent with the opening of OX. The fifth argument proves that $\neg(L_i \wedge R_i) = U_i$ for $i \in [n]$ while $\neg(L_n \wedge R_n) = 0$ which shows that circuit outputs 0. The last two arguments show that the wiring of the circuit is consistent.

Perfect zero-knowledge: the simulator creates a correctly formed common reference string together with a simulation trapdoor, so as the trapdoor commitments can be opened to any values. When simulating an argument, the simulator creates LR, RL, RZ, OZ and OX as commitments to $(0,\ldots,0)$. Due to the trapdoor, the simulator can simulate all product and permutation arguments. More precisely, he uses $L_i = R_i = U_i = 1$ to simulate all product and permutation arguments, except the one on step 2e where he uses $U_i = 0$ instead. (Obviously, also RZ and OZ commit to $(1,\ldots,1,0,\ldots,0)$.) To show that this argument π_2 simulates the real argument π_0, note that π_0 is perfectly indistinguishable from the simulated NIZK argument π_1 where one makes trapdoor commitments but opens them to *real* witnesses L_i, R_i when making product and permutation arguments. On the other hand, also π_1 and π_2 are perfectly indistinguishable, and thus so are π_0 and π_2. □

Theorem 7. *Let Λ be chosen as in Thm. 1. Let $\varepsilon > 0$. The communication (argument size) of the argument in Prot. 3 is* 31 *group elements. Prover's computational complexity is $\Theta(|C|^2)$ multiplications in $\mathbb{Z}_p$ and $\Theta(|C|^{1+\varepsilon})$ exponentiations in $\mathbb{G}$. Verifier's computational complexity is dominated by $\Theta(|C|)$ offline multiplications and* 35 *online bilinear pairings. The common reference string has length $O(|C|^{1+\varepsilon})$.*

Proof. Prot. 3 employs 4 element-wise product arguments, and 3 permutation arguments for *knowledge commitment*. Due to Thm. 3 and Thm. 5, the argument size is $5 \cdot 2$ group elements (for commitments) and $(4+3)(2+1) = 21$ group elements (for $4+3$ basic arguments). Statements about computational complexity and the length of the common reference string follow directly from Thm. 3 and Thm. 5. □

Compared to Groth's argument from [Gro10] we saved a commitment and a element-wise product argument. This is since Groth needed to commit to $(L_1,\ldots,L_n,R_1,\ldots,R_n)$ twice, by using two different set of generators g^{x^i} and $g^{x^{i(n+1)}}$, needed for two multiplicands in his element-wise product argument. Since we use the same set of generators for both multiplicands, we save also some computation.

Acknowledgments. We would like to thank Jens Groth for comments. The author was supported by Estonian Science Foundation, grant #8058, and European Union through the European Regional Development Fund.

References

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with Adaptive Soundness. In Salil Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 118–136, Amsterdam, The Netherlands, February 21–24, 2007. Springer Verlag.

[Beh46] Felix A. Behrend. On the Sets of Integers Which Contain No Three in Arithmetic Progression. *Proceedings of the National Academy of Sciences*, 32(12):331–332, December 1946.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications. In *STOC 1988*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

[Bou98] Jean Bourgain. On Triples in Arithmetic Progression. *Geometric and Functional Analysis*, 9(5):968–984, 1998.

[BP04] Mihir Bellare and Adriana Palacio. Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62, Jeju Island, Korea, December 5-9 2004. Springer-Verlag.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. In *STOC 1998*, pages 209–218, New York, May 23–26, 1998.

[DL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP Proofs from An Extractability Assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Computability in Europe, CIE 2008*, volume 5028 of *LNCS*, pages 175–185, Athens, Greece, June 15–20, 2008. Springer-Verlag.

[DN00] Cynthia Dwork and Moni Naor. Zaps and Their Applications. In *FOCS 2000*, pages 283–293, Redondo Beach, California, USA, November 12–14, 2000. IEEE Computer Society Press.

[Elk10] Michael Elkin. An Improved Construction of Progression-Free Sets. In Moses Charikar, editor, *SODA 2010*, pages 886–905, Austin, Texas, USA, January 17–19, 2010. SIAM.

[Gen09] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In Michael Mitzenmacher, editor, *STOC 2009*, pages 169–178, Bethesda, Maryland, USA, May 31 — June 2, 2009. ACM Press.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems. In *STOC 1985*, pages 291–304, Providence, Rhode Island, USA, May 6–8, 1985. ACM Press.

[Gro09] Jens Groth. Linear Algebra with Sub-linear Zero-Knowledge Arguments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 192–208, Santa Barbara, California, USA, August 16–20, 2009. Springer-Verlag.

[Gro10] Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, December 5–9 2010. Springer-Verlag.

[GW10] Craig Gentry and Daniel Wichs. Separating Succinct Non-Interactive Arguments From All Falsifiable Assumptions. Technical Report 2010/610, International Association for Cryptologic Research, November 29, 2010. Available at http://eprint.iacr.org/2010/610.

[Mic94] Silvio Micali. CS Proofs. In Shafi Goldwasser, editor, *FOCS 1994*, pages 436–453, Los Alamitos, California, USA, November 1994. IEEE, IEEE Computer Society Press.

[San10] Tom Sanders. On Roth's Theorem on Progressions. Technical Report arXiv:1011.0104v1, arXiv.org, October 30, 2010. Available from http://arxiv.org/abs/1011.0104.

[TV06] Terrence Tao and Van Vu. *Additive Combinatorics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006.