New Non-Interactive Zero-Knowledge Subset Sum, Decision Knapsack And Range Arguments

First eprint version, September 19, 2012

Helger Lipmaa[1] and Bingsheng Zhang[2]

[1] University of Tartu, Estonia
[2] State University of New York at Buffalo, USA

Abstract. We propose two basic NIZK arguments, one for Hadamard product of two vectors, and another one for a shift of a vector. The first argument is based on the corresponding argument of Lipmaa (TCC 2012), but makes use of Fast Fourier Transform and Pippenger's multi-exponentiation algorithm to achieve quasilinear (as opposed quadratic) computational complexity. The shift argument seems to be novel.

Based on the new basic arguments, we propose a NIZK argument for subset sum. This seems to be the only known (direct) sublinear NIZK argument for some other NP-complete language than Circuit-SAT. Moreover, it is significantly more efficient than the known sublinear Circuit-SAT arguments by Groth (Asiacrypt 2010) and Lipmaa. In addition, we show that the new arguments can be used to speed up the recent range argument by Chaabouni, Lipmaa and Zhang (FC 2012). Finally, we combine the subset sum argument and the range argument to propose a direct sublinear NIZK argument for another NP-complete language, decision knapsack.

Keywords. Decision knapsack argument, FFT, Hadamard product argument, non-interactive zero knowledge, progression-free sets, range argument, shift argument, subset sum argument.

1 Introduction

By using a zero-knowledge proof [GMR85], a prover can convince a verifier in the truth of a claim, without revealing any additional details about the proof. Standard zero-knowledge proofs are interactive, requiring the prover to participate in every verification. In most applications, it is desirable to have non-interactive zero-knowledge (NIZK) proofs [BFM88], where the prover has to be present only once while creating the proof. For example, in a typical e-voting protocol [CGS97,DJ01], every voter has to prove that her ballot contains the name of a legit candidate (this can be implemented as a range proof [Bou00,Lip03]); it is unreasonable to expect every voter to be available during the subsequent tallying process. NIZK proofs are divided into computational zero-knowledge proofs (where the zero-knowledge property holds against polynomial-time adversaries and the soundness property is unconditional) and computationally-sound proofs (also known as arguments, where the soundness property holds against polynomial-time adversaries and the zero-knowledge property is unconditional).

Efficient NIZK proofs play an important role in the design of cryptographic protocols. Hence, a large number of recent papers aim to design more efficient NIZK proofs (and arguments). In practice, the most interesting case is when the NIZK arguments (sublinear statistically sound NIZK proofs clearly do not exist) are succinct, that is, sublinear — or even logarithmic — in the input size. As shown in [GW11], sublinear NIZK proofs are only possible under non-standard (for example, knowledge) assumptions. Following [Gro10,Lip12], we base our arguments on knowledge assumptions.[1]

Many general techniques for constructing efficient NIZK proofs have been discovered while designing NIZK proofs for concrete languages like Circuit-SAT (an NP-complete language), shuffle, and range. In [Gro10], Groth constructed a NIZK Circuit-SAT argument based on two basic arguments, for Hadamard product and for permutation. Let $n = |C|$ be the circuit size. Both of Groth's basic arguments have quadratic CRS size (in group elements) and prover's computation (in exponentiations; more precisely, a small constant number of $\Theta(n^2)$-wide bilinear-group multi-exponentiations), while the communication (in group elements) and verifier's computation (in bilinear pairings) are constant; see

[1] We only mention NIZK proofs that work in the common reference string (CRS) model (where all parties have access to a honestly generated CRS) and not in the random oracle model, since random oracles cannot always be instantiated [CGH98,GK03].

Table 1. Comparison of knowledge-assumption based adaptive NIZK arguments for NP-complete languages with (worst-case) sublinear argument size. Note that the summary length of the CRS and the argument corresponds to the zap length. n is the size of circuit, $N = r_3^{-1}(n) = o(n2^{2\sqrt{2\log_2 n}})$ and $N^* = r_3^{-1}(\sqrt{n}) = o(\sqrt{n} \cdot 2^{2\sqrt{\log_2 n}})$ (if using Elkin's progression-free set), m is the balancing parameter, G corresponds to 1 group element and $\mathfrak{a}/\mathfrak{m}/\mathfrak{m}_b/\mathfrak{e}/\mathfrak{p}$ corresponds to 1 addition/multiplication in $\mathbb{Z}_p$/multiplication in bilinear group/exponentiation/pairing

m	CRS length	Argument length	Prover comp.	Verifier comp.
Adaptive Circuit-SAT arguments from [Gro10]				
1	$\Theta(n^2)G$	$42G$	$\Theta(n^2)\mathfrak{e}$	$\Theta(n)\mathfrak{m} + \Theta(1)\mathfrak{p}$
$n^{1/3}$	$\Theta(n^{\frac{2}{3}})G$	$\Theta(n^{\frac{2}{3}})G$	$\Theta(n^{4/3})\mathfrak{e}$	$\Theta(n)\mathfrak{m} + \Theta(n^{\frac{2}{3}})\mathfrak{p}$
Adaptive Circuit-SAT arguments from [Lip12]				
1	$\Theta(N)G$	$39G$	$\Theta(n^2)\mathfrak{a} + \Theta(N)\mathfrak{e}$	$\Theta(n)\mathfrak{e} + 62\mathfrak{p}$
$\sqrt{n}$	$\Theta(N^*)G$	$\Theta(\sqrt{n})G$	$\Theta(n^{3/2})\mathfrak{a} + \Theta(\sqrt{n} \cdot N^*)\mathfrak{e}$	$\Theta(n)\mathfrak{e} + \Theta(\sqrt{n})\mathfrak{p}$
Adaptive subset sum and decision knapsack arguments from the current paper				
1	$\Theta(N)G$	$\Theta(1)G$	$\Theta(N \log n)\mathfrak{m} + \Theta(N)\mathfrak{m}_b$	$\Theta(n)\mathfrak{m} + \Theta(1)\mathfrak{p}$
$\sqrt{n}$	$\Theta(N^*)G$	$\Theta(\sqrt{n})G$	$\Theta(\sqrt{n} \cdot N^* \log n)\mathfrak{m} + \Theta(\sqrt{n} \cdot N^*)\mathfrak{m}_b$	$\Theta(n)\mathfrak{m} + \Theta(\sqrt{n})\mathfrak{p}$

Tbl. 1. Thus, Groth's arguments offer essentially optimal communication and verifier's computational complexity, but they are quite inefficient in other parameters. In particular, they will probably not be able to handle circuits of size 2^{10} or more.

Subsequently, Lipmaa [Lip12] improved Groth's basic arguments — and therefore also Groth's Circuit-SAT argument — by using the theory of progression-free sets. Namely, let $r_3(N)$ be the size of the largest known progression-free subset of $[N] = \{1, \ldots, N\}$. Currently [Elk11] (see also Sect. 2),

$$r_3(N) = O((N \cdot \log^{1/4} N)/2^{2\sqrt{2\log 2N}}) \ .$$

Thus, $r_3^{-1}(n) = o(n2^{2\sqrt{2\log_2 n}})$. Lipmaa showed how to decrease the CRS size to $\Theta(r_3^{-1}(n))$ group elements and the prover's computational complexity so that it is dominated by $\Theta(n^2)$ scalar additions and two $\Theta(r_3^{-1}(n))$-wide bilinear-group multi-exponentiations. Here, $n = |C|$ is again the circuit size. An improved construction of progression-free sets will therefore automatically result in more efficient NIZK arguments. As shown in [Lip12], Lipmaa's product and permutation arguments can be used to construct a Circuit-SAT argument with similar asymptotic complexity, see Tbl. 1. (The verifier's computation in Lipmaa's argument in Tbl. 1 differs from what was claimed in [Lip12], that forgot to include part of the computational cost in their complexity estimate. That slightly incorrect claim from [Lip12] was also replicated in [CLZ12]. See Remark 1 on page 19 for a clarification.)

In a range argument, the prover aims to convince the prover that the committed value belongs to an integer range $[L, H]$. While the problem setting is simple, construction of range arguments has proven to be an excellent test case of zero-knowledge techniques, see, for example, [Bou00,LAN02,Lip03,CCs08,CLs10]. Construction of NIZK range arguments has only taken off during the last few years [RKP09,CLZ12]. In [CLZ12], Chaabouni, Lipmaa and Zhang used the product and permutation arguments of [Lip12] to construct the first known constant-communication (interactive or non-interactive) range argument that works in prime-order groups. While they used the same basic arguments as [Gro10,Lip12], they combined these basic arguments with several different (and unrelated) techniques that have been developed specifically for range proofs [LAN02,CLs10].

Finally, Lipmaa and Zhang [LZ12] constructed a so called 1-sparsity argument and used this to construct an efficient shuffle argument. Their 1-sparsity argument, while constructed by following a similar framework, has linear CRS size and communication and computational complexities. The Lipmaa-Zhang shuffle is only the second known efficient NIZK shuffle argument after [GL07].

The product and permutation arguments of Groth and Lipmaa can obviously be used to construct other complex arguments, though the full power of the "NIZK programming language" that consists of these two arguments is yet unknown. Moreover, as demonstrated in [LZ12], following the same framework, one can construct other basic arguments — for 1-sparsity — and use them to construct efficient (complex)

arguments. It is an important open problem to increase the library of efficient basic arguments even further, and to investigate which (more complex) arguments can be solved by using the new basic arguments. In addition, the basic arguments of Groth and Lipmaa are still computationally intensive for the prover, and construction of more efficient basic arguments (that at the same time have meaningful applications) is therefore an important open problem.

Our Contributions. We make the product argument of Lipmaa [Lip12] more efficient, and we also propose a new efficient shift argument. We then show how to use the more efficient product argument and the new shift argument to construct an efficient argument for subset sum (another NP-complete language), and how to make the range argument of [CLZ12] more efficient. Note that Groth and Lipmaa constructed effectively a "NIZK programming language" consisting of permutation and product arguments, and then used these two arguments to design a Circuit-SAT argument. What we do here is somewhat similar, but instead of the fully fledged permutation argument we only have a significantly more efficient argument for shift. That is, we show how to construct subset sum, range and decision knapsack arguments in the somewhat simpler NIZK programming language that consists of only the product and shift arguments.

We first modify the knowledge commitment scheme of [Lip12]. The commitment scheme from [Lip12] has as a parameter a progression-free set $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ of positive odd integers with $\lambda_1 \leq \lambda_i < \lambda_{i+1} \leq \lambda_n$. The new commitment scheme introduces an additional integer parameter $\upsilon = \Theta(\lambda_n - \lambda_1)$. If the commitment scheme of the current paper is used, in the product argument of [Lip12], Λ does not have to consist of positive odd integers. This is important conceptually, making it clear that one really requires progression-freeness of Λ (and nothing else) in similar arguments. For the commitment scheme to be binding and the (product or shift) argument to be secure, υ has to be chosen sufficiently large. On the other hand, for efficiency reasons, one wants to have as small $|\upsilon|$ as possible.

Second, we show how to use Fast Fourier Transform (FFT, [CT65]) based polynomial multiplication [GS66] techniques to reduce the prover's computational complexity in the product argument from $\Theta(n^2)$ to $\Theta(r_3^{-1}(n) \cdot \log r_3^{-1}(n))$ multiplications in $\mathbb{Z}_p$. In addition, one has to evaluate two $\Theta(n)$-wide and two $\Theta(r_3^{-1}(n))$-wide bilinear-group multi-exponentiations. Due to this, the new product argument has complexity parameters that are at most quasilinear. We note that FFT-based techniques are not applicable to optimize the arguments of Groth [Gro10], since there the largest element of Λ is $\Theta(n^2)$. We were also unable to apply FFT-based techniques to the permutation argument from [Lip12]; this is since Lipmaa's product argument has a special FFT-friendly construction while the permutation argument has a more complex structure.

Third, we use Pippenger's [Pip80] multi-exponentiation algorithm to eliminate the need for both the prover and the verifier to compute any exponentiations in bilinear groups. To evaluate two $\Theta(r_3^{-1}(n))$-wide bilinear-group multi-exponentiations that the prover has to execute in Lipmaa's product argument, by using Pippenger's algorithm, the prover has to perform $\Theta(r_3^{-1}(n))$ bilinear-group multiplications. This number is smaller than the number of multiplications in $\mathbb{Z}_p$, but since bilinear-group multiplications are more expensive, we will count them separately. (While [Lip12] mentioned that one can use efficient multi-exponentiation algorithms, it provided no analysis.)

Fourth, we propose a new shift argument. The shift argument has constant communication and verifier's computational complexity, and linear prover's computational complexity and CRS length, and can work with a large choice of sets Λ. As a drawback, we prove its security only by reduction to Φ-PSDL assumption [CLZ12] (see also Sect. 3), which is a non-trivial generalization of the Λ-PSDL assumption from [Lip12]. To show that the Φ-PSDL assumption is reasonable, we prove that the Φ-PSDL assumption is secure in the generic group model [Sho97].

Efficient Subset Sum Argument. We show how to construct an efficient NIZK subset sum argument (the prover knows a non-zero subset of the given integer set that sums to 0), where the communication and computational complexity are dominated by two product arguments and one shift argument. Therefore, the new subset sum argument has quasilinear CRS length and prover's computational complexity and constant communication and verifier's computational complexity. We note that in this case n denotes the size of the input domain, that is, the public set S is known to belong to $[n]$.

When using the balancing techniques of [Gro10,Lip12] (where, instead of applying the arguments to length n-vectors, one applies them in parallel to m length-(n/m) vectors), if $m = \sqrt{n}$, we obtain a

balanced subset sum NIZK argument with the parameters, given in the last row of Tbl. 1. (This also means that by using the techniques of [Gro10], one can construct a perfect zap [DN00] for subset sum with the same complexity parameters.) See Tbl. 1 for more comparison with previous work, and [Gro10,Lip12] for more background about the balancing techniques.

Efficient Range Argument. The new basic arguments can be used to optimize the NIZK range argument from [CLZ12], reducing the prover's computation from $\Theta(n^2)$ to $\Theta(r_3^{-1}(n) \cdot \log r_3^{-1}(n))$ multiplications in $\mathbb{Z}_p$ and from $\Theta(r_3^{-1}(n))$ bilinear-group exponentiations to $\Theta(r_3^{-1}(n))$ bilinear-group multiplications. See Sect. 6 for more information and comparison to the previous work. In addition, we note in Sect. 6 that [CLZ12] replicated the small mistake of [Lip12] (see Remark 1) and therefore the computational complexity of the unmodified argument of [CLZ12] is larger than claimed in [CLZ12]. We propose a simple additional modification of their range argument to make it even more efficient. We also discuss balanced versions of the new range argument that obtain better prover's computational complexity but have larger communication.

Efficient Decision Knapsack Argument. As the final contribution, we show that one can combine subset sum and range arguments to construct a decision knapsack argument. We recall that decision knapsack is another NP-complete language, and that the knapsack problem has direct cryptographic applications.

Concurrent Work. In an unpublished eprint [GGPR12], Gennaro, Gentry, Parno, and Raykova showed how to construct a more efficient (linear CRS, quasilinear prover's computational complexity, and constant communication and verifier's computational complexity) but non-adaptive (that is, the CRS depends on the circuit — in their construction, the CRS contains elements of form $g^{f(\sigma)}$, where σ is the secret key and f are polynomials depending on the concrete circuit) NIZK argument for Circuit-SAT. Thus, their construction is not directly comparable to adaptive constructions of [Gro10,Lip12] and the current paper.

2 Preliminaries

Let $[L, H] = \{L, L+1, \ldots, H-1, H\}$ and $[H] = [1, H]$. By $\boldsymbol{a}$, we denote the vector $\boldsymbol{a} = (a_1, \ldots, a_n)$. If A is a value, then $x \leftarrow A$ means that x is set to A. If A is a set, then $x \leftarrow A$ means that x is picked uniformly and randomly from A. If $y = h^x$, then let $\log_h y := x$. Let κ be the security parameter. We abbreviate probabilistic polynomial-time as PPT, and let $\mathrm{negl}(\kappa)$ be a negligible function.

Additive Combinatorics. If Λ_1 and Λ_2 are subsets of some additive group ($\mathbb{Z}$ or $\mathbb{Z}_p$ in this paper), then $\Lambda_1 + \Lambda_2 = \{\lambda_1 + \lambda_2 : \lambda_1 \in \Lambda_1 \wedge \lambda_2 \in \Lambda_2\}$ is their *sum set* and $\Lambda_1 - \Lambda_2 = \{\lambda_1 - \lambda_2 : \lambda_1 \in \Lambda_1 \wedge \lambda_2 \in \Lambda_2\}$ is their *difference set*. If Λ is a set, then $k\Lambda = \{\lambda_1 + \cdots + \lambda_k : \lambda_i \in \Lambda\}$ is an *iterated sumset*, $k \cdot \Lambda = \{k\lambda : \lambda \in \Lambda\}$ is a *dilation* of Λ, and $2\hat{}\Lambda = \{\lambda_1 + \lambda_2 : \lambda_1 \in \lambda \wedge \lambda_2 \in \Lambda \wedge \lambda_1 \neq \lambda_2\} \subseteq \Lambda + \Lambda$ is a *restricted sumset*. (See [TV06] for more notation and background.)

Progression-Free Sets. A set $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ is *progression-free* [ET36,TV06], if no three elements of Λ are in arithmetic progression, that is, $\lambda_i + \lambda_j = 2\lambda_k$ only if $i = j = k$. That is, $2\hat{}\Lambda \cap 2 \cdot \Lambda = \emptyset$. Let $r_3(N)$ be the cardinality of the largest progression-free set $\Lambda \subseteq [N]$. Recently, Elkin [Elk11] improved an old result of Behrend [Beh46], by proving the following result.

Fact 1 ([Elk11]) $r_3(N) = \Omega((N \cdot \log^{1/4} N)/2^{2\sqrt{2\log_2 N}})$.

(See [GW10] or [Lip11] for a *relatively* short proof of this result.) Thus, for any fixed $n > 0$, there exists $N = o(n2^{2\sqrt{2\log_2 n}})$, such that $[N]$ contains an n-element progression-free subset. In [ET36], Erdős and Turán proposed a progression-free subset of $[N]$ that has cardinality $N^{\log_3 2}$, and is denser than Elkin's set for say $N \leq 2^{16}$. The Erdős-Turán progression-free set consists of all integers in $[N]$ that have no 2-s in their ternary representation. On the other hand, it is known that $r_3(N) = O(N(\log\log N)^5/\log N)$ [San11].

Polynomial factorization. It is well-known that polynomial factorization algorithm in $\mathbb{Z}_p[X]$ can be done in polynomial time [vHN10]. Let $\mathsf{PolyFact}$ be an efficient polynomial factorization algorithm that on input a degree-d polynomial f outputs all $d+1$ roots of f.

Multi-Exponentiation Algorithms. Let $y_1, \ldots, y_M$ be monomials over the indeterminates $x_1, \ldots, x_N$. For every $y = (y_1, \ldots, y_M)$, let $L(y)$ be the minimum number of multiplications sufficient to compute $y_1, \ldots, y_M$ from $x_1, \ldots, x_N$ and the identity 1. Let $L(M, N, B)$ denote the maximum of $L(y)$ over all y for which the exponent of any indeterminate in any monomial is at most B. In [Pip80], Pippenger proved that

Fact 2 ([Pip80]) *$L(M, N, B) = \min\{M, N\} \log B + \frac{h}{\log h} \cdot U((\log\log h / \log h)^{1/2}) + O(\max\{M, N\})$, where $h = MN \cdot \log(B+1)$, and $U(\ldots)$ denotes a factor of the form $\exp(O(\ldots))$, and if the quantity represented by the ellipsis tends to 0, then $U(\ldots)$ is equivalent to $1 + O(\ldots)$.*

Bilinear Groups. Let $\mathcal{G}_{\mathsf{bp}}$ be a bilinear group generator [BF01] that outputs a description of a bilinear group $\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$ such that p is a κ-bit prime, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of order p (and both have an identity element denoted by 1), $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing such that $\forall a, b \in \mathbb{Z}$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$. If g_t generates $\mathbb{G}_t$ for $t \in \{1, 2\}$, then $\hat{e}(g_1, g_2)$ generates $\mathbb{G}_T$. We also make the common assumption that it is efficient to decide membership in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, group operations and the pairing $\hat{e}$ are efficiently computable, generators are efficiently sampleable, and the descriptions of the groups and group elements each are $O(\kappa)$ bit long. One can implement an optimal Ate pairing [HSV06] over a subclass of Barreto-Naehrig curves [BN05,PSNB11] very efficiently. In that case, at security level of 128-bits, an element of $\mathbb{G}_1/\mathbb{G}_2/\mathbb{G}_T$ can be represented in respectively 256/512/3072 bits.

Commitment Schemes. A trapdoor commitment scheme $\Gamma = (\mathcal{G}\mathrm{com}, \mathcal{C}\mathrm{om}, \mathcal{G}\mathrm{com}_{td}, \mathcal{C}\mathrm{om}_{td}, \mathcal{O}\mathrm{pen}_{td})$ consists of five PPT algorithms: a randomized common reference string (CRS) generation algorithm $\mathcal{G}\mathrm{com}$, a randomized commitment algorithm $\mathcal{C}\mathrm{om}$, a randomized trapdoor CRS generation algorithm $\mathcal{G}\mathrm{com}_{td}$, a randomized trapdoor commitment algorithm $\mathcal{C}\mathrm{om}_{td}$, and a trapdoor opening algorithm $\mathcal{O}\mathrm{pen}_{td}$. More precisely,

1. the CRS generation algorithm $\mathcal{G}\mathrm{com}(1^\kappa)$ produces a CRS ck,
2. the commitment algorithm $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r)$, with a new randomizer r, outputs a commitment value A. A commitment $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r)$ is opened by revealing $(\boldsymbol{a}, r)$,
3. the trapdoor CRS generation algorithm $\mathcal{G}\mathrm{com}_{td}(1^\kappa)$ outputs a CRS ck_{td}, that has the same distribution as $\mathcal{G}\mathrm{com}(1^\kappa)$, and a trapdoor td,
4. the randomized trapdoor commitment algorithm $\mathcal{C}\mathrm{om}_{td}(\mathsf{ck}_{td}; r)$ takes ck_{td} and a randomizer r as inputs, and outputs $\mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \mathbf{0}; r)$, and
5. the trapdoor opening algorithm $\mathcal{O}\mathrm{pen}_{td}(\mathsf{ck}_{td}; \mathsf{td}, \boldsymbol{a}; r)$ outputs an r_{td} such that $\mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \mathbf{0}; r) = \mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \boldsymbol{a}; r_{td})$.

A commitment scheme $\Gamma = (\mathcal{G}\mathrm{com}, \mathcal{C}\mathrm{om}, \mathcal{G}\mathrm{com}_{td}, \mathcal{C}\mathrm{om}_{td}, \mathcal{O}\mathrm{pen}_{td})$ is *computationally binding*, if for every non-uniform PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathcal{G}\mathrm{com}(1^\kappa), (\boldsymbol{a_1}, r_1, \boldsymbol{a_2}, r_2) \leftarrow \mathcal{A}(\mathsf{ck}) : \\ (\boldsymbol{a_1}, r_1) \neq (\boldsymbol{a_2}, r_2) \wedge \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_1}; r_1) = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_2}; r_2)\end{array}\right] = \mathrm{negl}(\kappa) \ .$$

A commitment scheme $\Gamma = (\mathcal{G}\mathrm{com}, \mathcal{C}\mathrm{om}, \mathcal{G}\mathrm{com}_{td}, \mathcal{C}\mathrm{om}_{td}, \mathcal{O}\mathrm{pen}_{td})$ is *perfectly hiding*, if for any $\mathsf{ck} \in \mathcal{G}\mathrm{com}(1^\kappa)$ and any two messages $\boldsymbol{a_1}, \boldsymbol{a_2}$, the distributions $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_1}; \cdot)$ and $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_2}; \cdot)$ are equal.

Non-Interactive Zero-Knowledge. Let $\mathcal{R} = \{(C, w)\}$ be an efficiently computable binary relation such that $|w| = \mathrm{poly}(|C|)$. Here, C is a statement, and w is a witness. Let $\mathcal{L} = \{C : \exists w, (C, w) \in \mathcal{R}\}$ be an **NP**-language. Let n be some fixed input length $n = |C|$. For fixed n, we have a relation $\mathcal{R}_n$ and a language $\mathcal{L}_n$. A *non-interactive argument* [BFM88] for $\mathcal{R}$ consists of the following PPT algorithms:

a common reference string (CRS) generator $\mathcal{G}_{\mathsf{crs}}$, a prover $\mathcal{P}$, and a verifier $\mathcal{V}$. For $\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$, $\mathcal{P}(\mathsf{crs}; C, w)$ produces an argument π. The verifier $\mathcal{V}(\mathsf{crs}; C, \pi)$ outputs either 1 (accept) or 0 (reject).

A non-interactive argument $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}, \mathcal{V})$ is *perfectly complete*, if $\forall n = \mathrm{poly}(\kappa)$,

$$\Pr[\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n), (C, w) \leftarrow \mathcal{R}_n : \mathcal{V}(\mathsf{crs}; C, \mathcal{P}(\mathsf{crs}; C, w)) = 1] = 1 \ .$$

A non-interactive argument $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}, \mathcal{V})$ is *(adaptively) computationally sound*, if for all non-uniform PPT adversaries $\mathcal{A}$ and all $n = \mathrm{poly}(\kappa)$, the probability

$$\Pr[\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n), (C, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : C \notin \mathcal{L} \wedge \mathcal{V}(\mathsf{crs}; C, \pi) = 1] = \mathrm{negl}(\kappa) \ .$$

The soundness is adaptive, that is, the adversary sees the CRS before producing the statement C.

A non-interactive argument $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}, \mathcal{V})$ is *perfectly witness-indistinguishable*, if for all $n = \mathrm{poly}(\kappa)$, if $\mathsf{crs} \in \mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$ and $((C, w_0), (C, w_1)) \in \mathcal{R}_n^2$, then the distributions $\mathcal{P}(\mathsf{crs}; C, w_0)$ and $\mathcal{P}(\mathsf{crs}; C, w_1)$ are equal. A *zap* [DN00] is a two-message witness-indistinguishable proof, where the first message is sent by the verifier, such that the verifier does not use any private coins, and the verifier's message can be fixed once and then used in arbitrary many proofs.

A non-interactive argument $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}, \mathcal{V})$ is *perfectly zero-knowledge*, if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, such that for all stateful non-uniform PPT adversaries $\mathcal{A}$ and $n = \mathrm{poly}(\kappa)$ (with td_π being the *simulation trapdoor*),

$$\Pr\begin{bmatrix}\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n), (C, w) \leftarrow \mathcal{A}(\mathsf{crs}),\\ \pi \leftarrow \mathcal{P}(\mathsf{crs}; C, w) :\\ (C, w) \in \mathcal{R}_n \wedge \mathcal{A}(\pi) = 1\end{bmatrix} = \Pr\begin{bmatrix}(\mathsf{crs}; \mathsf{td}_\pi) \leftarrow \mathcal{S}_1(1^\kappa, n), (C, w) \leftarrow \mathcal{A}(\mathsf{crs}),\\ \pi \leftarrow \mathcal{S}_2(\mathsf{crs}; C, \mathsf{td}_\pi) :\\ (C, w) \in \mathcal{R}_n \wedge \mathcal{A}(\pi) = 1\end{bmatrix} \ .$$

3 New Commitment Scheme

In this section, we will modify the commitment scheme of [Gro10,Lip12] by defining (see Prot. 1) the (Λ, υ) *trapdoor (knowledge) commitment scheme* in group $\mathbb{G}_t$ for $t \in \{1, 2\}$. Groth [Gro10] proposed a variant of this commitment scheme with $\Lambda = [n]$ and $\upsilon = 0$, while Lipmaa [Lip12] generalized Λ to any set $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ with $0 < \lambda_i < \lambda_{i+1}$ and $\lambda_n = \mathrm{poly}(\kappa)$ (while still letting $\upsilon = 0$).

We use the following security assumptions from [CLZ12]. Let p be as output by $\mathcal{G}_{\mathsf{bp}}$. Let $\Phi \subset \mathbb{Z}_p[X]$, with $d := \max_{\varphi \in \Phi} \deg \varphi$, be a set of linearly independent polynomials, such that $|\Phi|$, all coefficients of all $\varphi \in \Phi$, and d are polynomial in κ. Let 1 be the polynomial with $1(x) = 1$ for all $x \in \mathbb{Z}_p$.

Definition 1 (Φ-PDL and Φ-PSDL assumptions [CLZ12]). *Let Φ and d be as in above. A bilinear group generator $\mathcal{G}_{\mathsf{bp}}$ is* Φ-PDL secure *in $\mathbb{G}_t$, if for any non-uniform PPT adversary $\mathcal{A}$,*

$$\Pr\begin{bmatrix}\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa), g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}, \\ g_2 \leftarrow \mathbb{G}_2 \setminus \{1\}, \sigma \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{parm}; (g_t^{\varphi(\sigma)})_{\varphi \in \{1\} \cup \Phi}) = \sigma\end{bmatrix} = \mathrm{negl}(\kappa) \ .$$

A bilinear group generator $\mathcal{G}_{\mathsf{bp}}$ is Φ-PSDL secure*, if for any non-uniform PPT adversary $\mathcal{A}$,*

$$\Pr\begin{bmatrix}\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa), g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}, \\ g_2 \leftarrow \mathbb{G}_2 \setminus \{1\}, \sigma \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{parm}; (g_1^{\varphi(\sigma)}, g_2^{\varphi(\sigma)})_{\varphi \in \{1\} \cup \Phi}) = \sigma\end{bmatrix} = \mathrm{negl}(\kappa) \ .$$

Assumptions of similar complexity are relatively common in contemporary bilinear-group based cryptography, see for example [Wat12].

Theorem 1. *Let Φ and d be as in above. The Φ-PSDL assumption holds in the generic group model. Any successful generic adversary for Φ-PSDL requires time $\Omega(\sqrt{p/d})$.*

Proof. In the generic group model, an adversary only performs generic group operations (multiplications in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, bilinear pairings, and equality tests). A generic adversary produces an element of $\mathbb{Z}_p$, which depends only on parm and $(g_1^{\phi(\sigma)}, g_2^{\phi(\sigma)})_{\phi \in \{1\} \cup \Phi}$. The only time the adversary gets any information is when an equality (collision) between two previously computed elements of either $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$ occurs.

We prove that finding even a single collision is difficult even if the adversary can compute an arbitrary group element in unit time.

Assume that the adversary can find a collision $y = y^*$ in group $\mathbb{G}_1$. Then it must be the case that $y = \prod_{\phi_\ell \in \{1\}\cup\Phi} g_1^{a_\ell \phi_\ell(\sigma)}$ and $y' = \prod_{\ell\in\{0\}\cup\Lambda} g_1^{a_\ell^* \phi_\ell(\sigma)}$ for some known values of a_ℓ and a_ℓ^*. But then also

$$\sum_{\ell\in\{0\}\cup\Lambda} (a_\ell - a_\ell^*)\phi_\ell(\sigma) \equiv 0 \pmod{p} \ .$$

Since the adversary does not know the actual representations of the group elements, it will perform the same group operations independently of σ. Thus a_ℓ and a_ℓ^* are independent of σ. By the Schwartz-Zippel lemma [Sch80] modulo p, the probability that $\sum_{\ell\in\{0\}\cup\Lambda}(a_\ell - a_\ell^*)\phi_\ell(\sigma) \equiv 0 \pmod{p}$ is equal to d/p for randomly chosen a_ℓ and a_ℓ^*. If the adversary works in polynomial time $\tau = \mathrm{poly}(\kappa)$, it can generate at most τ such group elements. The total probability that there exists a collision between any two generated group elements is thus upper bounded by $\binom{\tau}{2}\cdot d/p$, and thus a successful adversary requires time $\Omega(\sqrt{p/d})$ to produce one collision.

A similar bound $\binom{\tau}{2}\cdot d/p$ holds for collisions in $\mathbb{G}_2$. In the case of $\mathbb{G}_T$, the pairing enables the adversary to compute up to τ different values

$$y = \hat{e}(g_1, g_2)^{\sum_{\phi_{1i}\in\{1\}\cup\Phi}\sum_{\phi_{2j}\in\{1\}\cup\Phi} a_{ij}\phi_{1i}(\sigma_1)\phi_{2j}(\sigma)} \ ,$$

and thus we get an upper bound $\binom{\tau}{2}\cdot 2d/p$, and thus a successful adversary requires time $\Omega(\sqrt{p/d})$ to produce one collision. □

Abe and Fehr showed in [AF07] that no statistically zero-knowledge non-interactive argument for an **NP**-complete language can have a "direct black-box" security reduction to a standard cryptographic assumption unless $\mathbf{NP} \subseteq \mathbf{P}/\mathrm{poly}$. (See also [GW11].) In fact, the soundness of NIZK arguments (for example, of an argument that a perfectly hiding commitment scheme commits to 0) is often unfalsifiable by itself. Similarly to [Gro10,Lip12,CLZ12,LZ12], we will base our NIZK argument for circuit satisfiability on an explicit knowledge assumption. This assumption, originally proposed in [CLZ12], is a generalization of the KEA assumption of Damgård [Dam91], the KEA3 assumption of Bellare and Palacio [BP04], the n-PKE assumption of Groth [Gro10], and the Λ-PKE assumption of Lipmaa [Lip12].

Let $t \in \{1, 2\}$. For algorithms $\mathcal{A}$ and $X_\mathcal{A}$, we write $(y; z) \leftarrow (\mathcal{A}||X_\mathcal{A})(\sigma)$ if $\mathcal{A}$ on input σ outputs y, and $X_\mathcal{A}$ on the same input (including the random tape of $\mathcal{A}$) outputs z.

Definition 2 (Φ-PKE security, [CLZ12]). *The bilinear group generator $\mathcal{G}_{\mathsf{bp}}$ is* Φ-PKE secure in group $\mathbb{G}_t$ *if for any non-uniform PPT adversary $\mathcal{A}$ there exists a non-uniform PPT extractor $X_\mathcal{A}$, such that*

$$\Pr\left[\begin{array}{l} \mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa), g_t \leftarrow \mathbb{G}_t \setminus \{1\}, (\alpha, \sigma) \leftarrow \mathbb{Z}_p^2, \\ \mathsf{crs} \leftarrow (\mathsf{parm}; (g_t^{\phi(\sigma)}, g_t^{\alpha\phi(\sigma)})_{\phi\in\Phi}), (c, \hat{c}; r, (a_\phi)_{\phi\in\Phi}) \leftarrow (\mathcal{A}||X_\mathcal{A})(\mathsf{crs}) : \\ \hat{c} = c^\alpha \wedge c \neq g_t^r \cdot \prod_{\phi\in\Phi} g_t^{a_\ell\phi(\sigma)} \end{array}\right] = \mathrm{negl}(\kappa) \ .$$

One can generalize the proof of Groth [Gro10] to show that the Φ-PKE assumption holds in the generic group model.

Let $t = 1$. Consider a CRS ck that in particular specifies $g_2, \hat{g}_2 \in \mathbb{G}_2$. A commitment $(A, \hat{A}) \in \mathbb{G}_1^2$ is *valid*, if $\hat{e}(A, \hat{g}_2) = \hat{e}(\hat{A}, g_2)$. The case $t = 2$ is dual.

The following theorem generalizes the corresponding theorem from [Gro10,Lip12].

Theorem 2 (Security of commitment scheme). *Let $t = 1$. (The case $t = 2$ is dual.) Let $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ with $\lambda_i < \lambda_{i+1}$ and $\lambda_i = \mathrm{poly}(\kappa)$. Let $\upsilon > \lambda_n$ be linear in $\lambda_n - \lambda_1$. Let $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ be the (Λ, υ) knowledge commitment scheme, see Prot. 1, in $\mathbb{G}_1$.*
(1) Γ is perfectly hiding in $\mathbb{G}_1$, and computationally binding in $\mathbb{G}_1$ under the $(\{X^\upsilon\} \cup \{X^\ell\}_{\ell\in\Lambda})$-PDL assumption in $\mathbb{G}_1$. The reduction time is dominated by the time to factor a degree-$(\upsilon - \lambda_1)$ polynomial in $\mathbb{Z}_p[X]$.
(2) If the $(\{X^\upsilon\}\cup\{X^\ell\}_{\ell\in\Lambda})$-PKE assumption holds in $\mathbb{G}_1$, then for any non-uniform PPT $\mathcal{A}$ that outputs a valid commitment, there exists a non-uniform PPT extractor $X_\mathcal{A}$ that, given the input of $\mathcal{A}$ together with $\mathcal{A}$'s random coins, extracts the contents of these commitments.

System parameters: $n = \text{poly}(\kappa)$, $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ with $\lambda_i \neq \lambda_j$ for $i \neq j$, $\lambda_i = \text{poly}(\kappa)$, and $\upsilon > \max_i \lambda_i$ is an integer. A bilinear group generator $\mathcal{G}_{\mathsf{bp}}$

CRS generation $\mathcal{G}\mathbf{com}(1^\kappa)$: Set $\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_t \leftarrow \mathbb{G}_t \setminus \{1\}$, and $(\sigma, \hat{\alpha}) \leftarrow \mathbb{Z}_p^2$; for each $i \in \{0\} \cup [n]$ do: $g_{t,\lambda_i} \leftarrow g_t^{\sigma^{\lambda_i}}$, $\hat{g}_{t,\lambda_i} \leftarrow g_t^{\hat{\alpha}\sigma^{\lambda_i}}$, $h_t \leftarrow g_t^{\sigma^\upsilon}$, $\hat{h}_t \leftarrow g_t^{\hat{\alpha}\sigma^\upsilon}$; return $\mathsf{ck} \leftarrow (\mathsf{parm}; (g_{t,\lambda_i}, \hat{g}_{t,\lambda_i})_{i\in\{0\}\cup[n]}, h_t, \hat{h}_t)$;

Commitment $\mathcal{C}\mathbf{om}(\mathsf{ck}; \boldsymbol{a}; \cdot)$: $r \leftarrow \mathbb{Z}_p$; return $\mathcal{C}\text{om}(\mathsf{ck}; \boldsymbol{a}; r) := (h_t^r \cdot \prod_{i=1}^n g_{t,\lambda_i}^{a_i}, \hat{h}_t^r \cdot \prod_{i=1}^n \hat{g}_{t,\lambda_i}^{a_i})$;

Trapdoor CRS generation $\mathcal{G}\mathbf{com}_{td}(1^\kappa)$: Execute $\mathcal{G}_{\mathsf{crs}}(1^\kappa)$; return $(\mathsf{ck}_{td} \leftarrow \mathsf{ck}; \mathsf{td} \leftarrow \sigma)$;

Trapdoor commitment $\mathcal{C}\mathbf{om}_{td}(\mathsf{ck}_{td}; \cdot)$: $r \leftarrow \mathbb{Z}_p$; return $\mathcal{C}\text{om}_{td}(\mathsf{ck}_{td}; r) \leftarrow \mathcal{C}\text{om}(\mathsf{ck}_{td}; \mathbf{0}; r) = (h_t^r, \hat{h}_t^r)$;

Trapdoor opening $\mathcal{O}\mathbf{pen}_{td}(\mathsf{ck}_{td}; \mathsf{td}, \boldsymbol{a}, r)$: return $r_{td} \leftarrow r\sigma^\upsilon - \sum_{i=1}^n a_i \sigma^{\lambda_i}$;

Protocol 1: The (Λ, υ) trapdoor commitment scheme. Here, $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$

Proof. PERFECT HIDING: follows from the fact that the output of $\mathcal{C}\text{om}$ is a random element of $\mathbb{G}_1$. COMPUTATIONAL BINDING: Assume that $\mathcal{A}_{\mathcal{C}\text{om}}$ is an adversary that can break the binding property with some non-negligible probability. We construct the following adversary $\mathcal{A}_{pdl}$ against the $(\{X^\upsilon\} \cup \{X^\ell\}_{\ell\in\Lambda})$-PDL assumption in $\mathbb{G}_1$ that works with the same probability. Here, $\mathcal{C}$ is the challenger of the PDL game.

$\mathcal{C}$ sets $\mathsf{parm} \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}$, and $\sigma \leftarrow \mathbb{Z}_p$;
$\mathcal{C}$ sends $(\mathsf{parm}; (g_1^{\sigma^\ell})_{\ell\in\{\upsilon\}\cup\Lambda})$ to $\mathcal{A}_{pdl}$;
$\mathcal{A}_{pdl}$ sets $\hat{\alpha}^* \leftarrow \mathbb{Z}_p$;
$\mathcal{A}_{pdl}$ sets $\mathsf{ck} \leftarrow (\mathsf{parm}; (g_1^{\sigma^\ell}, g_1^{\hat{\alpha}^*\sigma^\ell})_{\ell\in\{\upsilon\}\cup\Lambda})$;
1 $\mathcal{A}_{pdl}$ obtains $(\boldsymbol{a}, r_a, \boldsymbol{b}, r_b) \leftarrow \mathcal{A}_{\mathcal{C}\text{om}}(\mathsf{ck})$;
if $\boldsymbol{a} \notin \mathbb{Z}_p^n \vee \boldsymbol{b} \notin \mathbb{Z}_p^n \vee r_a \notin \mathbb{Z}_p \vee r_b \notin \mathbb{Z}_p \vee (\boldsymbol{a}, r_a) = (\boldsymbol{b}, r_b) \vee \mathcal{C}om(\mathsf{ck}; \boldsymbol{a}, r_a) \neq \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}, r_b)$ **then**
$\mathcal{A}_{pdl}$ aborts;
else
2 $\mathcal{A}_{pdl}$ sets $\delta(X) \leftarrow (r_a - r_b)X^{\upsilon-\lambda_1} + \sum_{i=1}^n (a_i - b_i)X^{\lambda_i-\lambda_1}$.
$\mathcal{A}_{pdl}$ sets $(t_1, \ldots, t_{\upsilon-\lambda_1+1}) \leftarrow \mathsf{PolyFact}(\delta)$;
3 $\mathcal{A}_{pdl}$ finds by an exhaustive search a root $\sigma_0 \in \{t_1, \ldots, t_{\upsilon-\lambda_1+1}\}$, such that $g_1^{\sigma^{\lambda_1}} = g_1^{\sigma_0^{\lambda_1}}$;
$\mathcal{A}_{pdl}$ returns $\sigma \leftarrow \sigma_0$ to the challenger;
end

Let us assume that on step 1, $\mathcal{A}_{\mathcal{C}\text{om}}$ is successful with probability $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}_{\mathcal{C}\text{om}}}(\Gamma)$. Thus, with probability $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}_{\mathcal{C}\text{om}}}(\Gamma)$, $(\boldsymbol{a}, r_a) \neq (\boldsymbol{b}, r_b)$ and

$$g_1^{r_a\sigma^\upsilon} \cdot \prod_{i\in[n]} g_1^{a_i\sigma^{\lambda_i}} = g_1^{r_b\sigma^\upsilon} \cdot \prod_{i\in[n]} g_1^{b_i\sigma^{\lambda_i}} \ .$$

But then

$$g_1^{(r_a-r_b)\sigma^\upsilon + \sum_{i=1}^n (a_i-b_i)\sigma^{\lambda_i}} = 1 \ ,$$

and thus

$$(r_a - r_b)\sigma^\upsilon + \sum_{i=1}^n (a_i - b_i)\sigma^{\lambda_i} \equiv 0 \pmod p \ ,$$

or equivalently,

$$(r_a - r_b)\sigma^{\upsilon-\lambda_1} + \sum_{i=1}^n (a_i - b_i)\sigma^{\lambda_i-\lambda_1} \equiv 0 \pmod p \ .$$

Since $\upsilon > \lambda_n$, $\delta(X)$, as defined on step 2 is a degree-$(\upsilon - \lambda_1)$ non-zero polynomial.

Thus, the adversary has generated a non-trivial degree-$(\upsilon - \lambda_1)$ polynomial $f(X)$ such that $f(\sigma) \equiv 0 \pmod p$. Therefore, $\mathcal{A}_{pdl}$ can use polynomial factorization to find all roots of δ, and one of those roots must be equal to σ. On step 3, $\mathcal{A}_{pdl}$ finds which root is equal to σ by an exhaustive search among all roots returned in the previous step. Thus, clearly $\mathcal{A}_{pdl}$ returns the correct value of sk (and thus violates the $(\{X^\upsilon\} \cup \{X^\ell\}_{\ell\in\Lambda})$-PDL assumption) with probability $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}_{\mathcal{C}\text{om}}}(\Gamma)$. Finally, the execution time of $\mathcal{A}_{pdl}$ is clearly dominated by the execution time of $\mathcal{A}_{\mathcal{C}\text{om}}$ and the time to factor δ.

EXTRACTABILITY: By the $(\{X^{\upsilon}\} \cup \{X^{\ell}\}_{\ell \in \Lambda})$-PKE assumption in group $\mathbb{G}_1$, for every committer $\mathcal{A}$ there exists an extractor $X_{\mathcal{A}}$ that can open the commitment in group $\mathbb{G}_1$, 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. □

We will sometimes use the same commitment scheme in both $\mathbb{G}_1$ and $\mathbb{G}_2$. In such cases, we will emphasize the underlying group by having a different CRS, but we will not change the name of the commitment scheme.

Computational Complexity of Commitment. Assume that $\alpha = ||\boldsymbol{a}||_{\infty} = \max_i a_i$, and $n \geq 2$. By using Pippenger's multi-exponentiation algorithm [Pip80], the computational complexity of the commitment function $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r)$ is dominated by

$$2\log_2 \alpha + (1 + o(1)) \cdot \frac{2n \log_2 \alpha}{\log_2(n \log_2 \alpha)} + O(n)$$

multiplications in $\mathbb{G}_t$. In our applications, $n \gg \log_2 \alpha$ (for example, $\alpha = 2$, $\alpha = n$, or even $\alpha = p$ given that n is reasonably large), and thus we get a simpler bound of

$$(2 + o(1)) \cdot \frac{n}{\log_2 n} \cdot \log_2 \alpha + O(n) \ .$$

multiplications. This can be compared to $3n \log_2 \alpha$ multiplications on average that one would have to execute by using the straightforward square-and-multiply exponentiation algorithm.

4 Improved Hadamard Product Argument

In this section, we propose a version of the product argument of [Lip12] that works together with the (Λ, υ) commitment scheme of Sect. 3. As we will see below (both in this section and in Sect. 5), the value of υ depends on the precise construction of the argument. For example, while the commitment scheme is binding for $\upsilon > \lambda_n$, for the product argument to be (weakly) sound we require that $\upsilon > 2\lambda_n - \lambda_1$. If one uses several such arguments together (for example, to construct a subset sum argument or a range argument), one has to choose a value of υ that is secure for all basic arguments. We also show that one can use FFT and Pippenger's multi-exponentiation algorithm to make the product argument more efficient.

Definition 3 (Hadamard product argument). *Assume that* $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ *is a trapdoor knowledge commitment scheme that commits to elements* $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ *for a prime* p *and integer* $n \geq 1$. *In an* Hadamard product argument*, the prover aims to convince the verifier that given commitments* A*,* B *and* C*, he can open them as* $A = \mathcal{C}om(\mathsf{ck}; \boldsymbol{a}; r_a)$*,* $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$*, and* $C = \mathcal{C}om(\mathsf{ck}; \boldsymbol{c}; r_c)$*, such that* $c_i = a_i b_i$ *for* $i \in [n]$.

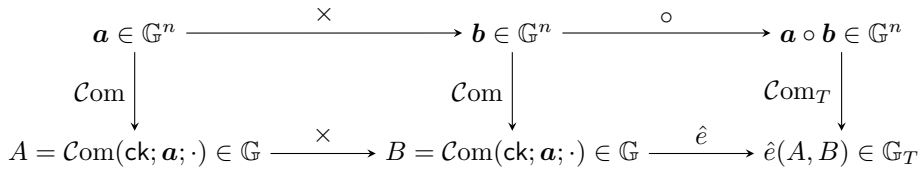
In other words, a product argument has n constraints $c_i = a_i b_i$ for $i \in [n]$.

In [Lip12], Lipmaa constructed an Hadamard product argument for the $(\Lambda, \upsilon = 0)$ commitment scheme with communication of 5 group elements, verifier's computation $\Theta(n)$, prover's computation of $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$, and the CRS of $\Theta(r_3^{-1}(n))$ group elements. We present a more efficient (implementation of this) argument in Prot. 2.

We first recall the basic idea of Lipmaa's Hadamard product argument. Let $\circ$ be the Hadamard product of two vectors, let $\mathbf{1} = (1, \ldots, 1)$, and let $A * B := \hat{e}(A, B)$. The Hadamard product argument of both [Gro10,Lip12] is a vector version of the Groth-Sahai proofs [GS08]. The verification equation

$$\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r_a) * \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{b}; r_b) = (\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{c}; r_c) * \mathcal{C}\mathrm{om}(\mathsf{ck}; \mathbf{1}; 0)) \cdot (g_1 * \pi)$$

"maps" the to-be-verified algebraic property $(\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \circ \mathbf{1})$ to a different algebraic domain. Here, both $\circ$ and $*$ are bilinear operators. We can exemplify this by the following commutative diagram, though here $\mathcal{C}\mathrm{om}_T$ is not a real function since $\hat{e}(A, B)$ depends not only on $\boldsymbol{a} \circ \boldsymbol{b}$ but also on the concrete values $\boldsymbol{a}$ and $\boldsymbol{b}$:



Here, π compensates for the inclusion of the randomizers r_a, r_b and r_c in the "commitment domain".

We now rewrite Lipmaa's argument for the generic (Λ, υ) commitment scheme Γ. Similarly to [Lip12], we will use Γ in both $\mathbb{G}_1$ (to commit to $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$) and $\mathbb{G}_2$ (to commit to $\boldsymbol{b}$ and $\mathbf{1}$). Let $\widehat{\mathsf{ck}}$ be the CRS in group $\mathbb{G}_1$ (see Prot. 2), and $\widehat{\mathsf{ck}}^*$ be the dual CRS in group $\mathbb{G}_2$ (that is, $\widehat{\mathsf{ck}}^*$ is defined as $\widehat{\mathsf{ck}}$, but with g_1 replaced by g_2). Thus, for example, $(B, \hat{B}) = \mathcal{C}\mathrm{om}(\widehat{\mathsf{ck}}; \boldsymbol{b}; r_b)$. Then, we have $\log_{g_1} A = r_a\sigma^\upsilon + \sum_{i=1}^n a_i\sigma^{\lambda_i}$, $\log_{g_1} B = r_b\sigma^\upsilon + \sum_{i=1}^n b_i\sigma^{\lambda_i}$, and $\log_{g_1} C = c_i\sigma^\upsilon + \sum_{i=1}^n r_c\sigma^{\lambda_i}$. We also have an element B_2, such that $\hat{e}(g_1, B_2) = \hat{e}(B, g_2)$. Thus, for $(D, \hat{D}) = \mathcal{C}\mathrm{om}(\widehat{\mathsf{ck}}^*; \mathbf{1}; 0)$ (in group $\mathbb{G}_2$),

$$\log_{\hat{e}(g_1,g_2)}(\hat{e}(A, B_2)/\hat{e}(C, D)) = (r_a\sigma^\upsilon + \sum_{i=1}^n a_i\sigma^{\lambda_i})(r_b\sigma^\upsilon + \sum_{i=1}^n b_i\sigma^{\lambda_i}) - (r_c\sigma^\upsilon + \sum_{i=1}^n c_i\sigma^{\lambda_i})(\sum_{i=1}^n \sigma^{\lambda_i})$$

can be written — after substituting σ with a formal variable X — as a sum of two formal polynomials $F_{con}(X)$ and $F_\pi(X)$, such that $F_{con}(X)$ (the constraint polynomial) as a formal polynomial has one monomial per constraint ($a_ib_i = c_i$) and is zero if the prover is honest, while $F_\pi(X)$ (the *argument polynomial*) has potentially many more monomials. (More precisely, F_π has $\Theta(r_3^{-1}(n))$ monomials, and the CRS has length $\Theta(r_3^{-1}(n))$.) The honest prover has to compute $(\pi, \widehat{\pi}) \leftarrow (g_2^{F_\pi(\sigma)}, \hat{g}_2^{F_\pi(\sigma)})$. The PSDL and the PKE assumption guarantee that he cannot do it if at least one of the n constraints is not satisfied.

In [Lip12], for soundness, one had to assume that the used set Λ is a progression-free set of odd positive integers. By using such Λ, [Lip12] proved that the polynomials $F_{con}(X)$ and $F_\pi(X)$ were spanned by two non-intersecting sets of powers of X. From this, [Lip12] then deduced (weak) soundness.

In what follows, we show that by using the (Λ, υ) commitment scheme (for a well-chosen value of υ), one can — without any loss in efficiency — assume that Λ is just a progression-free set. This makes the product argument slightly more efficient. More importantly, it makes it clear that the property that Λ has to satisfy is really progression-freeness, and not say having only odd integers as its members.

For a set Λ and an integer υ, define

$$\hat{\Lambda} := \{2\upsilon\} \cup (\upsilon + \Lambda) \cup 2\hat{}\Lambda \ . \tag{1}$$

(In [Lip12], this definition was only given for $\upsilon = 0$. Then, $\hat{\Lambda} = \{0\} \cup \Lambda \cup 2\hat{}\Lambda$.)

Lemma 1. *Assume that $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ with $\lambda_i < \lambda_{i+1}$, and $\upsilon > 2\lambda_n - \lambda_1$. Λ is a progression-free set if and only if $2 \cdot \Lambda \cap \hat{\Lambda} = \emptyset$.*

Proof. Assume Λ is progression-free. Then, clearly $2\hat{}\Lambda \cap 2 \cdot \Lambda = \emptyset$. Since $\upsilon > 2\lambda_n - \lambda_1$, we also have $(\{2\upsilon\} \cup (\upsilon + \Lambda)) \cap 2 \cdot \Lambda = \emptyset$. (In [Lip12], $\upsilon = 0$, and $(\{0\} \cup \Lambda) \cap 2 \cdot \Lambda = \emptyset$ was guaranteed by assuming that every integer in Λ is odd and non-zero.) Assume now that $2 \cdot \Lambda \cap \hat{\Lambda} = \emptyset$. In particular, this means that $2 \cdot \Lambda \cap 2\hat{}\Lambda = \emptyset$, and thus Λ is a progression-free set. □

Lemma 2. *For any $n > 0$, there exists a progression-free set $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$, with $\lambda_i < \lambda_{i+1}$ and $\lambda_n = \mathrm{poly}(\kappa)$, and an integer $\upsilon > 2\lambda_n - \lambda_1$, υ linear in $\lambda_n - \lambda_1$, such that $|\hat{\Lambda}| = \Theta(r_3^{-1}(n)) = o(n2^{2\sqrt{2\log_2 n}})$.*

Proof. Let Λ be the progression-free set from [Elk11], seen as a subset of $[\lambda_1, \lambda_n]$ (with λ_1 possibly being negative), with $\lambda_n - \lambda_1 \approx r_3^{-1}(n) = o(n2^{2\sqrt{2\log_2 n}})$. Since $\upsilon > 2\lambda_n - \lambda_1$ is linear in $\lambda_n - \lambda_1$, $\hat{\Lambda} \subset \{2\lambda_1, \ldots, 2\upsilon\}$ and $|\hat{\Lambda}| = \Theta(r_3^{-1}(n))$. □

We can clearly add some constant k to all members of Λ and υ, so that the previous results still hold. In particular, according to the previous two lemmas, the best value (in the sense of efficiency) of λ_n might be 0.

We state and prove the security of the new Hadamard product argument when using the (Λ, υ) knowledge commitment scheme by following the claim and the proof from [Lip12] very closely, mostly

System parameters: Let $n = \text{poly}(\kappa)$. Let $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ be a progression-free set, such that $\lambda_i < \lambda_{i+1}$ and $\lambda_n - \lambda_1 = \text{poly}(\kappa)$. Let $\upsilon > 2\lambda_n - \lambda_1$ be linear in $\lambda_n - \lambda_1$. Let $\hat{\Lambda}$ be as in Eq. (1).
Define $\mathfrak{I}_1(\ell) := \{(i,j) : i,j \in [n] \wedge i \neq j \wedge \lambda_i + \lambda_j = \ell\}$.

CRS generation $\mathcal{G}_{\mathsf{crs}}(1^\kappa)$**:** Set $\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $(g_1, g_2) \leftarrow (\mathbb{G}_1 \setminus \{1\}, \mathbb{G}_2 \setminus \{1\})$, $\sigma, \hat{\alpha} \leftarrow \mathbb{Z}_p$;
for each $\ell \in \{\upsilon\} \cup \Lambda$ do: $g_{1,\ell} \leftarrow g_1^{\sigma^\ell}$, $\hat{g}_{1,\ell} \leftarrow g_1^{\hat{\alpha}\sigma^\ell}$;
for each $\ell \in \{\upsilon\} \cup \hat{\Lambda}$ do: $g_{2,\ell} \leftarrow g_2^{\sigma^\ell}$, $\hat{g}_{2,\ell} \leftarrow g_2^{\hat{\alpha}\sigma^\ell}$;
Set $D \leftarrow \prod_{i=1}^n g_{2,\lambda_i}$, $\widehat{\mathsf{ck}} \leftarrow (\mathsf{parm}; (g_{1,\ell}, \hat{g}_{1,\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$;
Return $\mathsf{crs} \leftarrow (\widehat{\mathsf{ck}}, g_{2,\upsilon}, (g_{2,\ell}, \hat{g}_{2,\ell})_{\ell \in \hat{\Lambda}}, D)$;

Argument generation $\mathcal{P}_\times(\mathsf{crs}; (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C}), (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c))$**:**
for each $\ell \in 2^{\hat{}}\Lambda$ do: $\mu_\ell \leftarrow \sum_{(i,j) \in \mathfrak{I}_1(\ell)} (a_i b_j - c_i)$;
$(\pi, \hat{\pi}) \leftarrow (g_{2,2\upsilon}^{r_a r_b} \cdot \prod_{i=1}^n g_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2^{\hat{}}\Lambda} g_{2,\ell}^{\mu_\ell}, \hat{g}_{2,2\upsilon}^{r_a r_b} \cdot \prod_{i=1}^n \hat{g}_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2^{\hat{}}\Lambda} \hat{g}_{2,\ell}^{\mu_\ell})$;
return $\pi^\times \leftarrow (\pi, \hat{\pi}) \in \mathbb{G}_2^2$;

Verification $\mathcal{V}_\times(\mathsf{crs}; (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C}), \pi^\times)$**:** If $\hat{e}(A, B_2)/\hat{e}(C, D) = \hat{e}(g_1, \pi)$ and $\hat{e}(g_1, \hat{\pi}) = \hat{e}(\hat{g}_1, \pi)$ then accept. Otherwise, reject.

Protocol 2: New Hadamard product argument $[\![(A, \hat{A})]\!] \circ [\![(B, \hat{B}, B_2)]\!] = [\![(C, \hat{C})]\!]$

just to be able to use the result later, in Sect. 6, to construct efficient subset sum, range, and decision knapsack arguments. The (knowledge) commitments are $(A, \hat{A})$, $(B, \hat{B})$ and $(C, \hat{C})$. For efficiency (and backwards compatability) reasons, following [Lip12], we include another element B_2 to the statement of the Hadamard product language.

Since for given $\boldsymbol{a}$ and $\boldsymbol{b}$, $(C, \hat{C})$ is a commitment of $(a_1 b_1, \ldots, a_n b_n)$ for *some* value of r_c, we cannot claim that Prot. 2 is computationally sound (even under a knowledge assumption). Instead, analogously to [Gro10,Lip12], we prove a somewhat weaker version of soundness that is however sufficient to achieve soundness of the subset sum and range arguments. The last statement of Thm. 3 basically says that no efficient adversary can output an input to the Hadamard product argument together with an accepting argument and openings to all commitments and all other pairs of type $(y, \hat{y})$ that are present in the argument, such that $a_i b_i \neq c_i$ for some $i \in [n]$. Intuitively, the theorem statement below, see Thm. 3, includes certain elements f_ℓ^* only for $\ell \in \hat{\Lambda}$ (resp., a_ℓ for $\ell \in \Lambda$ together with r) since $\hat{g}_{2,\ell}$ (resp., $\hat{g}_{1,\ell}$) belongs to the CRS only for $\ell \in \hat{\Lambda}$ (resp., $\ell \in \{\upsilon\} \cup \Lambda$). This "weak" soundness is similar to the co-soundness as defined in [GL07]. However, in the case of co-soundness, the adversary is not be required to open the argument (by presenting values f_ℓ^*, as in the theorem statement). One could define the corresponding formal security notion, but in our opinion, it would not increase readability.

Theorem 3 (Security of product argument). *Let $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ be the (Λ, υ) commitment scheme in group $\mathbb{G}_1$. Then*
(1) Prot. 2 is perfectly complete and perfectly witness-indistinguishable.
(2) If $\mathcal{G}_{\mathsf{bp}}$ is $(\{X^\upsilon\} \cup \{X^\ell\}_{\ell \in \hat{\Lambda}})$-PSDL secure, then a non-uniform PPT adversary against Prot. 2 has negligible chance, given correctly generated CRS crs as an input, of outputting $inp^\times \leftarrow (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C})$ and an accepting argument $\pi^\times \leftarrow (\pi, \hat{\pi})$ together with a witness $w^\times \leftarrow (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c, (f_\ell^*)_{\ell \in \hat{\Lambda}})$, such that

(i) $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{Z}_p^n$, $r_a, r_b, r_c \in \mathbb{Z}_p$, *and* $f_\ell^* \in \mathbb{Z}_p$ *for* $\ell \in \hat{\Lambda}$,
(ii) $(A, \hat{A}) = \mathcal{C}om(\widehat{\mathsf{ck}}; \boldsymbol{a}; r_a)$, $(B, \hat{B}) = \mathcal{C}om(\widehat{\mathsf{ck}}; \boldsymbol{b}; r_b)$, $B_2 = g_{2,\upsilon}^{r_b} \cdot \prod_{i=1}^n g_{2,\lambda_i}^{b_i}$, *and* $(C, \hat{C}) = \mathcal{C}om(\widehat{\mathsf{ck}}; \boldsymbol{c}; r_c)$,
(iii) $\log_{g_2} \pi = \log_{\hat{g}_2} \hat{\pi} = \sum_{\ell \in \hat{\Lambda}} f_\ell^* \sigma^\ell$, *and*
(iv) for some $i \in [n]$, $a_i b_i \neq c_i$.

The reduction time is dominated by the time it takes to factor a degree-$(2\upsilon - 2\lambda_1) = \Theta(r_3^{-1}(n))$ polynomial in $\mathbb{Z}_p[X]$.

Proof. Let $h \leftarrow \hat{e}(g_1, g_2)$ and $F(\sigma) \leftarrow \log_h(\hat{e}(A, B_2)/\hat{e}(C, D))$. Witness-Indistinguishability: since the argument $\pi^\times = (\pi, \hat{\pi})$ that satisfies the verification equations is unique, all witnesses result in the same argument, and therefore the Hadamard product argument is witness-indistinguishable.

$\mathcal{C}$ forms crs as in Prot. 2;
$\mathcal{C}$ sends crs to $\hat{\mathcal{A}}$;
$\hat{\mathcal{A}}$ obtains $(inp^{\times}, w^{\times}, \pi^{\times}) \leftarrow \mathcal{A}_{\times}(\mathsf{crs})$;
if *the conditions (i–iv) in the statement of Thm. 3 do not hold* **then** $\hat{\mathcal{A}}$ aborts;
else
1 $\hat{\mathcal{A}}$ expresses $F(X)$ as a polynomial $f(X) \leftarrow \sum_{\ell \in \hat{\Lambda} \cup 2 \cdot \Lambda} f_\ell X^\ell$;
2 $\hat{\mathcal{A}}$ computes a polynomial $f^*(X) \leftarrow \sum_{\ell \in \hat{\Lambda}} f^*_\ell X^\ell$;
$\hat{\mathcal{A}}$ lets $\delta(X) \leftarrow (f(X) - f^*(X)) \cdot X^{-2\lambda_1}$;
$\hat{\mathcal{A}}$ sets $(t_1, \ldots, t_{2(\upsilon - \lambda_1)}) \leftarrow \mathsf{PolyFact}(\delta)$;
3 $\hat{\mathcal{A}}$ finds by an exhaustive search a root $\sigma_0 \in (t_1, \ldots, t_{2(\upsilon-\lambda_1)})$, such that $g_1^{\sigma^\ell} = g_1^{\sigma_0^\ell}$;
$\hat{\mathcal{A}}$ returns $\sigma \leftarrow \sigma_0$ to the challenger;
end

Algorithm 1: Construction of $\hat{\mathcal{A}}$ in the security reduction of Thm. 3

Perfect completeness. Assume that the prover is honest. The second verification is straightforward. For the first one, note that

$$\begin{aligned}
F(\sigma) =& (r_a\sigma^\upsilon + \sum_{i=1}^n a_i\sigma^{\lambda_i})(r_b\sigma^\upsilon + \sum_{i=1}^n b_i\sigma^{\lambda_i}) - (r_c\sigma^\upsilon + \sum_{i=1}^n c_i\sigma^{\lambda_i})(\sum_{i=1}^n \sigma^{\lambda_i}) \\
=& r_a r_b \sigma^{2\upsilon} + \sum_{i=1}^n (r_a b_i + r_b a_i - r_c)\sigma^{\upsilon+\lambda_i} + \sum_{i=1}^n \sum_{j=1}^n (a_i b_j - c_i)\sigma^{\lambda_i+\lambda_j} \\
=& r_a r_b \sigma^{2\upsilon} + \sum_{i=1}^n (r_a b_i + r_b a_i - r_c)\sigma^{\upsilon+\lambda_i} + \sum_{i=1}^n (a_i b_i - c_i)\sigma^{2\lambda_i} + \sum_{i=1}^n \sum_{\substack{j=1 \\ j\neq i}}^n (a_i b_j - c_i)\sigma^{\lambda_i+\lambda_j} \quad .
\end{aligned}$$

That is, $F(\sigma) = F_{\mathsf{con}}(\sigma) + F_\pi(\sigma)$, where F_{con} and F_π are formal polynomials with

$$\begin{aligned}
F_{\mathsf{con}}(X) =& \sum_{i=1}^n (a_i b_i - c_i) X^{2\lambda_i} \quad , \\
F_\pi(X) =& r_a r_b X^{2\upsilon} + \sum_{i=1}^n (r_a b_i + r_b a_i - r_c) X^{\upsilon+\lambda_i} + \sum_{i=1}^n \sum_{\substack{j=1 \\ j\neq i}}^n (a_i b_j - c_i) X^{\lambda_i+\lambda_j} \quad .
\end{aligned}$$

Here, $F(X)$, $F_{con}(X)$ and $F_\pi(X)$ are formal polynomials of X, and $F(X)$ is spanned by $\left\{X^\ell\right\}_{\ell \in 2\cdot\Lambda \cup \hat{\Lambda}}$. More precisely, $F_{con}(X)$ is the constraint polynomial, that has one monomial per constraint $c_i = a_i b_i$, and $F_\pi(X)$ is the argument polynomial.

If the prover is honest, then $c_i = a_i b_i$ for $i \in [n]$, and thus $F(X) = F_\pi(X)$ is spanned by $\left\{X^\ell\right\}_{\ell \in \hat{\Lambda}}$. Denoting

$$\pi \leftarrow g_{2,\upsilon}^{r_a r_b} \cdot \prod_{i=1}^n g_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{i=1}^n \prod_{\substack{j=1 \\ j\neq i}}^n g_{2,\lambda_i+\lambda_j}^{a_i b_j - c_i} = g_2^{r_a r_b} \cdot \prod_{i=1}^n g_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2\hat{} \Lambda} g_{2,\ell}^{\mu_\ell} \quad ,$$

where μ_ℓ is defined as in Prot. 2, we see that clearly $\hat{e}(g_1, \pi) = h$. Thus, the first verification succeeds.

Weaker version of soundness. Assume that $\mathcal{A}_\times$ is an adversary that can break the last statement of the theorem. We construct an adversary $\hat{\mathcal{A}}$ against the $(\{X^\upsilon\} \cup \left\{X^\ell\right\}_{\ell \in \hat{\Lambda}})$-PSDL assumption, see Prot. 1. Here, $\mathcal{C}$ is the challenger of the PSDL game.

Let us analyse the advantage of $\hat{\mathcal{A}}$. First, clearly crs_{td} has the same distribution as $\mathcal{G}_{\mathsf{crs}}(1^\kappa)$. Thus, $\mathcal{A}_\times$ gets a correct input. She aborts with probability $1 - \mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$. Otherwise, with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$, $inp^\times = (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C})$ and $w^\times = (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c, (f^*_\ell)_{\ell \in \hat{\Lambda}})$, such that the conditions (i–iv) hold.

The steps from step 1 onwards are executed with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$. Since $\mathcal{A}_\times$ succeeds and $2\cdot\Lambda\cap\hat{\Lambda}=\emptyset$, at least for one $\ell\in 2\cdot\Lambda$, $f(X)$ has a non-zero coefficient $a_ib_i-c_i$. $\hat{\mathcal{A}}$ succeeds on step 2, since $\log_{g_2}\pi=\sum_{\ell\in\hat{\Lambda}} f^*_\ell\sigma^\ell$. Moreover, all non-zero coefficients of X^ℓ in $f^*(X)$ correspond to $\ell\in\hat{\Lambda}$. Since Λ is a progression-free set, $\upsilon>2\lambda_n-\lambda_1$, and all elements of $2\cdot\Lambda$ are distinct, then by Lem. 1, $\ell\notin 2\cdot\Lambda$. Thus, all coefficients of $f^*(X)$ corresponding to any X^ℓ, $\ell\in 2\cdot\Lambda$, are equal to 0. Thus, $f(X)=\sum_{\ell\in\hat{\Lambda}\cup(2\cdot\Lambda)} f_\ell X^\ell$ and $f^*(X)=\sum_{\ell\in\hat{\Lambda}} f^*_\ell X^\ell$ are different polynomials with $f(\sigma)=f^*(\sigma)=F(\sigma)$. Note that all coefficients of X^ℓ, for $\ell<2\lambda_1$, of both $f(X)$ and $f^*(X)$ are equal to 0.

Thus, $\delta(X)$ is a non-zero degree-$(2\upsilon-2\lambda_1)$ polynomial, such that

$$\delta(\sigma)=\sum_{\ell\in(\hat{\Lambda}\cup(2\cdot\Lambda))-2\lambda_1}\delta_\ell\sigma^\ell=0\ .$$

Therefore, $\hat{\mathcal{A}}$ can use polynomial factorization to find all $\leq 2(\upsilon-\lambda_1)$ roots of δ, where one of the found roots must be equal to σ. On step 3, $\hat{\mathcal{A}}$ finds which root is equal to σ by an exhaustive search among all roots returned in the previous step. Thus, clearly $\hat{\mathcal{A}}$ returns the correct value of σ (and thus violates the $(\{X^\upsilon\}\cup\{X^\ell\}_{\ell\in\hat{\Lambda}})$-PSDL assumption) with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$. Finally, the execution time of $\hat{\mathcal{A}}$ is clearly dominated by the execution time of $\mathcal{A}_\times$ and the time to factor δ. □

Efficiency. We will show that the product argument of this section (and therefore also the product argument of [Lip12]) is computationally much more efficient than it was claimed in [Lip12]. Namely, in [Lip12], the product argument was said to require the prover to compute $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$ and $\Theta(r_3^{-1}(n))=o(n2^{2\sqrt{2\log_2 n}})$ exponentiations in $\mathbb{G}_2$. We will optimize the prover's computation so that it will require a significantly smaller number of multiplications and no exponentiations at all.

Theorem 4 (Efficiency of product argument). *Let Λ be the progression-free set from [Elk11]. The communication (argument size) of Prot. 2 is 2 elements from $\mathbb{G}_2$. The prover's computational complexity is dominated by $\Theta(r_3^{-1}(n)\cdot\log r_3^{-1}(n))=o(n2^{2\sqrt{2\log_2 n}}\cdot\log n)$ multiplications in $\mathbb{Z}_p$ and two $\Theta(r_3^{-1}(n))=o(n2^{2\sqrt{2\log_2 n}})$-wide multi-exponentiations in $\mathbb{G}_2$. The verifier's computational complexity is dominated by 5 bilinear pairings and 1 bilinear-group multiplication. The CRS consists of $\Theta(r_3^{-1}(n))=o(n2^{2\sqrt{2\log_2 n}})$ group elements.*

Proof. By Lem. 2, the size of the CRS is $\Theta(|\hat{\Lambda}|)=\Theta(r_3^{-1}(n))$. From the CRS, the verifier clearly only needs to access g_1, $\hat{g}_1$, and D. Since $2\hat{}\Lambda\subseteq\hat{\Lambda}$, the statement about the prover's computational complexity follows from Fast Fourier Transform [CT65] based polynomial multiplication [GS66] techniques. More precisely, to compute all the coefficients of the formal polynomial

$$\mu(X):=\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}(a_ib_j-c_i)X^{\lambda_i+\lambda_j}\ ,$$

the prover executes Prot. 3. Here, FFTMult denotes a FFT-based polynomial multiplication algorithm.

After using FFTMult to compute the initial version of $\mu(X)$ and $\nu(X)$,

$$\mu_\ell=\sum_{\substack{(i,j)\in[n]^2\\ \lambda_i+\lambda_j=\ell}}a_ib_j\quad\text{and}\quad\nu_\ell=\sum_{\substack{(i,j)\in[n]^2\\ \lambda_i+\lambda_j=\ell}}c_i\ .$$

Thus, after the penultimate step of Prot. 3, $\mu_\ell=\sum_{(i,j)\in\mathfrak{I}_1(\ell)}a_ib_j$, and after the last step, $\mu_\ell=\sum_{(i,j)\in\mathfrak{I}_1(\ell)}a_ib_j-c_i$, as required by Prot. 2. Since FFT takes time $\Theta(N\log N)$, where $N=r_3^{-1}(n)$ is the input size, then we have shown the part about the prover's computational complexity. The verifier's computational complexity follows from the description of the argument. □

We remark that FFT is not useful to speed up Groth's product argument from [Gro10], since there $\lambda_n=\Theta(n^2)$. Moreover, FFT does not seem to be useful in the case of the permutation argument from [Lip12]. Finally, it may be possible to speed up the described procedure, by taking into account the fact that all $a^\dagger$, $b^\dagger$, $c^\dagger$ and $d^\dagger$ have only n non-zero monomials.

for $i \leftarrow 0$ to λ_n do: $a_i^\dagger \leftarrow 0$, $b_i^\dagger \leftarrow 0$, $c_i^\dagger \leftarrow 0$, $d_i^\dagger \leftarrow 0$;
for $i \leftarrow 1$ to n do: $a_{\lambda_i}^\dagger \leftarrow a_i$, $b_{\lambda_i}^\dagger \leftarrow b_i$, $c_{\lambda_i}^\dagger \leftarrow c_i$, $d_{\lambda_i}^\dagger \leftarrow 0$;
Denote $a^\dagger(X) := \sum_{i=0}^{\lambda_n} a_i^\dagger X^i$, $b^\dagger(X) := \sum_{i=0}^{\lambda_n} b_i^\dagger X^i$, $c^\dagger(X) := \sum_{i=0}^{\lambda_n} c_i^\dagger X^i$ and $d^\dagger(X) := \sum_{i=0}^{\lambda_n} d_i^\dagger X^i$;
Let $\mu(X) \leftarrow \mathsf{FFTMult}(a^\dagger(X), b^\dagger(X))$;
Let $\nu(X) \leftarrow \mathsf{FFTMult}(c^\dagger(X), d^\dagger(X))$;
for $i \leftarrow 1$ to n do: $\mu_{2\lambda_i} \leftarrow \mu_{2\lambda_i} - a_i b_i$;
Let $\mu(X) \leftarrow \mu(X) - \nu(X)$;

Protocol 3: FFT-based prover's computation of $\{\mu_\ell\}$ in the product argument

Using Efficient Multi-Exponentiation. Let $\alpha := \max(||a||_\infty, ||b||_\infty, ||c||_\infty)$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the vectors committed by the prover. (See Sect. 6 for the concrete values of α needed in applications.) The number of bilinear-group operations the prover has to perform (on top of computing the exponents by using the described FFT-based polynomial multiplication technique) to compute π in the product argument is dominated by $L(2,n,p)+L(2,r_3^{-1}(n),\Theta((\alpha n)^2))$. Here, the very conservative value $\Theta((\alpha n)^2)$ follows from

$$|\mu_\ell| = |\sum_{(i,j)\in\mathfrak{I}_1(\ell)} (a_i b_j - c_i)| \leq \sum_{(i,j)\in\mathfrak{I}_1(\ell)} |a_i b_j - c_i| \leq \sum_{(i,j)\in\mathfrak{I}_1(\ell)} (\alpha^2+\alpha) < (n^2-n)(\alpha^2+\alpha) = \Theta((\alpha n)^2) \ .$$

Due to Fact 2, we get that, for $n = \Omega(\log p)$,

$$L(2,n,p) = 2\log_2 p + \frac{2n\log_2(p+1)}{\log_2(2n\log_2(p+1))} \cdot (1+o(1)) + O(n) = (2+o(1)) \cdot \frac{n}{\log_2 n} \cdot \log_2 p \ ,$$

and, since in our applications, $n \gg \log_2 \Theta((\alpha n)^2)$,

$$\begin{aligned} L(2,r_3^{-1}(n),\Theta((\alpha n)^2)) &= 2\log_2(\alpha n^2) + \frac{2r_3^{-1}(n)\log_2\Theta((\alpha n)^2)}{\log_2(2r_3^{-1}(n)\log_2\Theta((\alpha n)^2))} \cdot (1+o(1)) + O(r_3^{-1}(n)) \\ &= (2+o(1)) \cdot \frac{r_3^{-1}(n)}{\log_2 r_3^{-1}(n)} \cdot 2\log_2(\alpha n) \ . \end{aligned}$$

Thus, the prover has to compute

$$(2+o(1)) \cdot \left(\frac{n}{\log_2 n} \cdot \log_2 p + \frac{r_3^{-1}(n)}{\log_2 r_3^{-1}(n)} \cdot 2\log_2(\alpha n) \right) \tag{2}$$

bilinear-group multiplications. We will instantiate α and other values to this in Sect. 6.

5 Shift And Rotation Arguments

Definition 4 (Shift and rotation argument). *In a* shift argument*, the prover aims to convince the verifier that for two commitments A and B, he knows how to open them as $A = \mathcal{C}om(\mathsf{ck}; \boldsymbol{a}; r_a)$ and $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that*

$$a_i = \begin{cases} b_{i+1} \ , & i \in [n-1] \ , \\ 0 \ , & i = n \ . \end{cases}$$

Analogously, in a rotation argument*, the prover aims to convince the verifier that for two commitments A and B, he knows how to open them as $A = \mathcal{C}om(\mathsf{ck}; \boldsymbol{a}; r_a)$ and $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that*

$$a_i = \begin{cases} b_{i+1} \ , & i \in [n-1] \ , \\ b_1 \ , & i = n \ . \end{cases}$$

Groth [Gro10] and Lipmaa [Lip12] defined NIZK arguments for arbitrary permutation ϱ (that is, that $a_{\varrho(i)} = b_i$ for ϱ that is a part of the argument). However, their permutation arguments are quite complex and computationally intensive. Moreover, many applications do not require arbitrary permutations. We give concrete examples of the latter claim in Sect. 6.

CRS generation $\mathcal{G}_{\mathsf{crs}}(1^\kappa)$**:** Set $\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}$, $g_2 \leftarrow \mathbb{G}_2 \setminus \{1\}$, $\sigma, \tilde{\alpha} \leftarrow \mathbb{Z}_p$;
for each $t \in \{1,2\}$ do: $\tilde{g}_t \leftarrow g_t^{\tilde{\alpha}}$;
for each $\ell \in \{\upsilon\} \cup \Lambda$ do: $(g_{1,\ell}, \tilde{g}_{1,\ell}) \leftarrow (g_1^{\sigma^\ell}, \tilde{g}_1^{\sigma^\ell})$;
Set $g_{2,1} \leftarrow g_2^{\sigma}$, $g_{2,\lambda_1} \leftarrow g_2^{\sigma^{\lambda_1}}$;
for each $\ell \in \{\upsilon, \upsilon+1\}$ do: $(g_{2,\ell}, \tilde{g}_{2,\ell}) \leftarrow (g_2^{\sigma^\ell}, \tilde{g}_2^{\sigma^\ell})$;
Set $(h_{2,1}, \tilde{h}_{2,1}) \leftarrow (g_2^{\sigma^{\lambda_n+1}-\sigma^{\lambda_i}}, \tilde{g}_2^{\sigma^{\lambda_n+1}-\sigma^{\lambda_i}})$;
for each $\ell \in [2,n]$ do: $(h_{2,\ell}, \tilde{h}_{2,\ell}) \leftarrow (g_2^{\sigma^{\lambda_{i-1}+1}-\sigma^{\lambda_i}}, \tilde{g}_2^{\sigma^{\lambda_{i-1}+1}-\sigma^{\lambda_i}})$;
Set $\widetilde{\mathsf{ck}} \leftarrow (\mathsf{parm}; (g_{1,\ell}, \tilde{g}_{1,\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$;
return $\mathsf{crs} \leftarrow (\widetilde{\mathsf{ck}}, g_2, g_{2,1}, (g_{2,\ell}, \tilde{g}_{2,\ell})_{\ell \in \{\lambda_1, \upsilon, \upsilon+1\}}, (h_{2,i}, \tilde{h}_{2,i})_{i \in [2,n]})$;

Argument generation $\mathcal{P}_{\mathsf{sft}}(\mathsf{crs}; (A, \tilde{A}, B, \hat{B}, \tilde{B}), (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b))$**:**
set $(\pi, \tilde{\pi}) \leftarrow (g_{2,\upsilon+1}^{r_a} \cdot g_{2,\upsilon}^{-r_b} \cdot g_{2,\lambda_1}^{-b_1} \cdot \prod_{i=2}^n h_{2,i}^{b_i}, \tilde{g}_{2,\upsilon+1}^{r_a} \cdot \tilde{g}_{2,\upsilon}^{-r_b} \cdot \tilde{g}_{2,\lambda_1}^{-b_1} \cdot \prod_{i=2}^n \tilde{h}_{2,i}^{b_i})$;
return $\pi^{\mathsf{sft}} \leftarrow (\pi, \tilde{\pi}) \in \mathbb{G}_2^2$;

Verification $\mathcal{V}_{\mathsf{sft}}(\mathsf{crs}; (A, \tilde{A}, B, \hat{B}, \tilde{B}), \pi^{\mathsf{sft}})$**:** if $\hat{e}(A, g_{2,1})/\hat{e}(B, g_2) = \hat{e}(g_1, \pi) \wedge \hat{e}(g_1, \tilde{\pi}) = \hat{e}(\tilde{g}_1, \pi)$ then $\mathcal{V}_{\mathsf{sft}}$ accepts else $\mathcal{V}_{\mathsf{sft}}$ rejects;

Protocol 4: New shift argument $\mathsf{shift}(\llbracket (A, \tilde{A}) \rrbracket) = \llbracket (B, \tilde{B}) \rrbracket$

We now describe the new shift argument $\mathsf{shift}(\llbracket (A, \tilde{A}) \rrbracket) = \llbracket (B, \tilde{B}) \rrbracket$, that is much simpler and significantly more computation-efficient than the generic permutation arguments of Groth and Lipmaa. One can design a very similar rotation argument; since it will use basically the same underlying ideas, we will only comment on the differences between the new shift argument and the corresponding rotation argument.

Let $\log_{g_1} A = r_a \sigma^\upsilon + \sum_{i=1}^n a_i \sigma^{\lambda_i}$ and $\log_{g_1} B = r_b \sigma^\upsilon + \sum_{i=1}^n b_i \sigma^{\lambda_i}$. Replacing σ with a formal variable X, we get that if the prover is honest (full derivation of this is given in the proof of Thm. 5), then

$$F(X) := X \cdot \log_{g_1} A - \log_{g_1} B = \sum_{i=2}^n b_i (X^{\lambda_{i-1}+1} - X^{\lambda_i}) - b_1 X^{\lambda_1} + r_a X^{\upsilon+1} - r_b X^\upsilon \ .$$

Thus, one can verify that A is a shift of B by just checking that $\hat{e}(A, g_2^\sigma)/\hat{e}(B, g_2) = \hat{e}(g_1, \pi)$, where $\pi = g_2^{F(\sigma)}$ is defined as in Prot. 4.

As seen from the following theorem and its proof, the actual security proof, especially for the (weaker version of) soundness, is somewhat more complicated. Complications arise from the use of polynomials of type $X^i - X^j$ in the verification equation; because of this we have to rely on a less straightforward version of the Φ-PSDL assumption than before. One has also to be careful in the choice of the set Λ: namely, if say $\lambda_n + 1 = \lambda_1$ then some of the monomials of $F(X)$ will collapse, and the security proof will not go through.

Theorem 5 (Security of the shift argument). *Let $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ be the (Λ, υ) commitment scheme in group $\mathbb{G}_1$.*
(1) Prot. 4 is perfectly complete and perfectly witness-indistinguishable.
(2) Let $\Lambda = (\lambda_1, \ldots, \lambda_n)$ be a tuple of integers, such that $\lambda_i + 1 < \lambda_{i+1}$ and $\lambda_i = \mathrm{poly}(\kappa)$. Let $\Phi := \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n$. Let $\upsilon > \lambda_n + 1$. If $\mathcal{G}_{\mathsf{bp}}$ is Φ-PSDL secure, then a non-uniform PPT adversary against Prot. 4 has negligible chance, given a correctly formed CRS crs as an input, of outputting $inp^{\mathsf{sft}} \leftarrow (A, \tilde{A}, B, \tilde{B})$ and an accepting argument $\pi^{\mathsf{sft}} \leftarrow (\pi, \tilde{\pi})$ together with a witness $w^{\mathsf{sft}} \leftarrow (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, (f_\phi^*)_{\phi \in \Phi})$, such that

(i) $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, $r_a, r_b \in \mathbb{Z}_p$, *and* $f_\phi^* \in \mathbb{Z}_p$ *for* $\phi \in \Phi$,
(ii) $(A, \tilde{A}) = \mathcal{C}om(\widetilde{\mathsf{ck}}; \boldsymbol{a}; r_a)$, $(B, \tilde{B}) = \mathcal{C}om(\widetilde{\mathsf{ck}}; \boldsymbol{b}; r_b)$,
(iii) $\log_{g_2} \pi = \log_{\tilde{g}_2} \tilde{\pi} = \sum_{\phi \in \Phi} f_\phi^* \cdot \phi(\sigma)$, *and*
(iv) $(a_n, a_{n-1}, \ldots, a_1) \neq (0, b_n, \ldots, b_2)$.

The reduction time is dominated by the time it takes to factor a degree-$(\upsilon+1)$ polynomial in $\mathbb{Z}_p[X]$.

Proof. Denote $h \leftarrow \hat{e}(g_1, g_2)$ and $F(\sigma) := \log_h(\hat{e}(A, g_{2,1})/\hat{e}(B, g_2))$. Witness-Indistinguishability: since argument π^{sft} that satisfies the verification equations is unique, all witnesses result in the same argument, and therefore the permutation argument is witness-indistinguishable.

$\mathcal{C}$ forms crs as in Prot. 4;
$\mathcal{C}$ sends crs to $\tilde{\mathcal{A}}$;
$\tilde{\mathcal{A}}$ obtains $(inp^{\mathsf{sft}}, w^{\mathsf{sft}}, \pi^{\mathsf{sft}}) \leftarrow \mathcal{A}_{\mathsf{sft}}(\mathsf{crs})$;
if *the conditions (i–iv) in the statement of Thm. 5 do not hold* **then** $\tilde{\mathcal{A}}$ aborts;
else
1 $\tilde{\mathcal{A}}$ expresses $F(X)$ as a polynomial $f(X) = \sum_{\phi \in \Phi^\pi} f_\phi \cdot \phi(X)$;
2 $\tilde{\mathcal{A}}$ computes a polynomial $f^*(X) := \sum_{\phi \in \Phi} f^*_\phi \cdot \phi(X)$;
$\tilde{\mathcal{A}}$ lets $\delta(X) \leftarrow f(X) - f^*(X)$;
$\tilde{\mathcal{A}}$ uses a polynomial factorization algorithm in $\mathbb{Z}_p[X]$ to compute all $\leq (\upsilon + 2)$ roots of $\delta(X)$;
3 $\tilde{\mathcal{A}}$ finds by an exhaustive search a root σ_0, such that $g_1^{\sigma^\ell} = g_1^{\sigma_0^\ell}$;
$\tilde{\mathcal{A}}$ returns $\sigma \leftarrow \sigma_0$;
end

Algorithm 2: Construction of $\tilde{\mathcal{A}}$ in the security reduction of Thm. 5

Perfect completeness. The second verification is straightforward. For the first verification $\hat{e}(A, g_{2,1})/\hat{e}(B, g_2) = \hat{e}(g_1, \pi)$, consider $F(X) := X \cdot \log_{g_1} A - \log_{g_1} B$, where we have replaced σ with a formal variable X. Clearly,

$$\begin{aligned} F(X) =& \sum_{i=1}^{n} a_i X^{\lambda_i+1} - \sum_{i=1}^{n} b_i X^{\lambda_i} + r_a X^{\upsilon+1} - r_b X^{\upsilon} \\ =& \sum_{i=1}^{n-1} a_i X^{\lambda_i+1} + a_n X^{\lambda_n+1} - b_1 X^{\lambda_1} - \sum_{i=2}^{n} b_i X^{\lambda_i} + r_a X^{\upsilon+1} - r_b X^{\upsilon} \\ =& a_n X^{\lambda_n+1} - b_1 X^{\lambda_1} + \sum_{i=2}^{n} (a_{i-1} X^{\lambda_{i-1}+1} - b_i X^{\lambda_i}) + r_a X^{\upsilon+1} - r_b X^{\upsilon} \\ =& \sum_{i=2}^{n} (a_{i-1} - b_i) X^{\lambda_{i-1}+1} + a_n X^{\lambda_n+1} - b_1 X^{\lambda_1} + \sum_{i=2}^{n} b_i (X^{\lambda_{i-1}+1} - X^{\lambda_i}) + r_a X^{\upsilon+1} - r_b X^{\upsilon} \ . \end{aligned} \tag{3}$$

If the prover is honest, then $a_i = b_{i+1}$ for $i \in [n-1]$ and $a_n = 0$, and thus

$$F(X) = -b_1 X^{\lambda_1} + \sum_{i=2}^{n} b_i (X^{\lambda_{i-1}+1} - X^{\lambda_i}) + r_a X^{\upsilon+1} - r_b X^{\upsilon}$$

is spanned by $\{\phi(X)\}_{\phi \in \Phi}$. Defining π as in Prot. 4, we see that the second verification holds.

Weaker version of soundness. Assume that $\mathcal{A}_{\mathsf{sft}}$ is an adversary that can break the last statement of the theorem. We construct an adversary $\tilde{\mathcal{A}}$ against the Φ-PSDL assumption, see Prot. 2. Here, $\mathcal{C}$ is the challenger of the PSDL game, and $\Phi^\pi := \left\{X^\upsilon, X^{\upsilon+1}\right\} \cup \left\{X^{\lambda_i}, X^{\lambda_i+1}\right\}_{i=1}^n$ is defined by following the first line of Eq. (3).

Let us analyse the advantage of $\tilde{\mathcal{A}}$. First, clearly crs_{td} has the same distribution as $\mathcal{G}_{\mathsf{crs}}(1^\kappa)$. Thus, $\mathcal{A}_{\mathsf{sft}}$ gets a correct input, and succeeds with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$. Clearly, $\tilde{\mathcal{A}}$ aborts with probability $1 - \mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$.

Otherwise, with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$, $inp^{\mathsf{sft}} = (A, \tilde{A}, B, \tilde{B})$ and $w^{\mathsf{sft}} = (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, (f^*_\phi)_{\phi \in \Phi})$, such that the conditions (i–iv) hold. In particular, $f(X) = F(X)$ in Eq. (3), and

$$f^*(X) = f^*_{X^{\lambda_1}} \cdot X^{\lambda_1} + \sum_{i=2}^{n} f_{X^{\lambda_{i-1}+1} - X^{\lambda_i}} (X^{\lambda_{i-1}+1} - X^{\lambda_i}) + f^*_{X^{\upsilon+1}} X^{\upsilon+1} + f^*_{X^\upsilon} X^\upsilon \ .$$

Since $(a_n, a_{n-1}, \ldots, a_1) \neq (0, b_n, \ldots, b_2)$, $f(X)$ has at least one more non-zero monomial, either of type X^{λ_n+1} or of type $(a_i - b_j)X^{\lambda_i+1}$, than $f^*(X)$. Since X^{λ_i+1} cannot be represented as a linear combination of polynomials from Φ, $f(X)$ and $f^*(X)$ are different polynomials with $f(\sigma) = f^*(\sigma) = F(\sigma)$.

Thus, $\delta(X)$ is a non-zero degree-$(\upsilon+1)$ polynomial, such that $\delta(\sigma) = 0$. Therefore, $\tilde{\mathcal{A}}$ can use an efficient polynomial factorization algorithm [vHN10] to find all roots of δ, and one of those roots must be

equal to σ. On step 3, $\tilde{\mathcal{A}}$ finds which root is equal to σ by an exhaustive search among all roots returned in the previous step. Thus, clearly $\tilde{\mathcal{A}}$ returns the correct value of σ (and thus violates the Φ-PSDL assumption) with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$. Finally, the execution time of $\tilde{\mathcal{A}}$ is clearly dominated by the execution time of $\mathcal{A}_{\mathsf{sft}}$ and the time to factor δ. □

Note that in an upper level argument, the verifier must check that $\hat{e}(A, \tilde{g}_2) = \hat{e}(\tilde{A}, g_2)$, and $\hat{e}(B, \tilde{g}_2) = \hat{e}(\tilde{B}, g_2)$.

Theorem 6 (Efficiency of shift argument). *Let Λ and υ be as defined in Thm. 5. Let $\beta \leftarrow ||\boldsymbol{b}||_\infty$, $\beta < p$. Assume $n > \log_2 \beta$. The communication (argument size) of Prot. 4 is 2 elements from $\mathbb{G}_2$. The prover's computational complexity is dominated by $\Theta(n)$ multiplications in $\mathbb{Z}_p$ and*

$$(2 + o(1)) \cdot \frac{n \log_2 \beta}{\log_2 n} + O(n)$$

bilinear-group multiplications. The verifier's computational complexity is dominated by 5 bilinear pairings. The CRS consists of $\Theta(n)$ group elements.

Proof. By using Pippenger's algorithm, the prover computes two multi-exponentiations in

$$L(2, n, \beta) = 2\log_2 \beta + (1 + o(1)) \cdot \frac{2n \log_2(\beta + 1)}{\log_2(2n \log_2(\beta + 1))} + O(n) = (2 + o(1)) \cdot \frac{n \log_2 \beta}{\log_2 n} + O(n)$$

bilinear-group multiplications. Other claims are straightforward. □

Rotation Argument. In the rotation argument,

$$\begin{aligned} F(X) =& (a_n - b_1)X^{\lambda_n+1} + \sum_{i=2}^{n}(a_{i-1} - b_i)X^{\lambda_{i-1}+1} + b_1(X^{\lambda_n+1} - X^{\lambda_1}) + \\ & \sum_{i=2}^{n} b_i(X^{\lambda_{i-1}+1} - X^{\lambda_i}) + r_a X^{\upsilon+1} - r_b X^{\upsilon} \ . \end{aligned}$$

Thus, in the case Φ is different, $\Phi = \left\{X^{\upsilon}, X^{\upsilon+1}, X^{\lambda_n+1} - X^{\lambda_1}\right\} \cup \left\{X^{\lambda_{i-1}+1} - X^{\lambda_i}\right\}_{i=2}^{n}$. Given this modification, one can construct a rotation argument that is very similar to Prot. 4.

6 Applications

We will now describe how to use the new product and shift arguments to construct a new subset sum argument, and to improve on the range argument of [CLZ12]. Finally, we show how to combine subset sum and range arguments to construct a decision knapsack argument. In all three cases, the shift argument is mainly used to construct an intermediate scan argument. Recall that vector $\boldsymbol{b}$ is a *scan* [Ble90] of vector a, if $b_i = \sum_{j>i} a_j$. As demonstrated over and over in [Ble90], vector scan (also known as all-prefix-sums) is a powerful operator that can be used to solve many important computational problems. However, in the context of zero knowledge, we only need to be able to verify that one vector is a scan of the second vector.

Definition 5 (Scan argument). *In a* scan argument*, the prover aims to convince the verifier that given two commitments A and B, he knows how to open them as $A = \mathcal{C}om(\mathsf{ck}; \boldsymbol{a}; r_a)$ and $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that $b_i = \sum_{j>i} a_j$.*

A scan argument π^{scan} is just equal to a shift argument $\mathsf{shift}([\![B]\!]) = [\![A \cdot B]\!]$, which proves that $b_i = a_{i+1} + a_{i+1}$, for $i < n$, and $b_n = 0$. Thus, $b_n = 0$, $b_{n-1} = a_n$, $b_{n-2} = a_{n-1} + b_{n-1} = a_{n-1} + a_n$, and in general, $b_i = \sum_{j>i} a_j$.

6.1 Subset Sum Argument

Assume we want to construct an efficient argument for *some* NP-complete problem. Circuit-SAT seems to require the use of product and permutation arguments [Gro10,Lip12], so we will try to find another problem. A simple example is subset sum, where the prover aims to prove that he knows a non-zero subset of the input set S that sums to 0. We assume that $\mathcal{S} = (s_1, \ldots, s_n) \subset \mathbb{Z}_p$, $n \ll p$.

Definition 6 (Subset sum argument). *In a* subset sum argument*, the prover aims to convince the verifier that given $\mathcal{S} = (s_1, \ldots, s_n) \subseteq \mathbb{Z}_p$ and a commitment B, he knows how to open it as $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that $\boldsymbol{b}$ is Boolean and non-zero, and $\sum_{i=0}^{n-1} a_i s_i = 0$.*

That is, $b_i = 1$ iff s_i belongs to the subset of S that sums to 0.

During the new subset sum argument, both parties can compute a commitment S to $\boldsymbol{s}$. The prover commits to a Boolean vector $\boldsymbol{b}$. He computes a commitment C to a vector $\boldsymbol{c}$, such that $c_i = b_i s_i$. He computes a commitment D to the *scan* [Ble90] $\boldsymbol{d}$ of vector $\boldsymbol{c}$. That is, $d_i = \sum_{j<i} c_j$, and in particular, $d_n = \sum_{j<n} c_i$ and $c_n + d_n = \sum_{j \le n} c_j$.

The prover computes the subset sum argument as follows:

Compute a product argument π_1 showing that $\boldsymbol{b}$ is Boolean;
Compute an argument π_2 showing that $\boldsymbol{b} \neq \boldsymbol{0}$;
Compute a product argument π_3 showing that $c_i = b_i \cdot s_i$ for $i \in [n]$;
Compute a scan argument π_4 showing that $\boldsymbol{d}$ is the scan of $\boldsymbol{c}$;
Compute a restriction argument π_5 showing that the last coordinate of $\boldsymbol{c} + \boldsymbol{d}$ is 0;
The subset sum argument is equal to $(B, C, D, \pi_1, \ldots, \pi_5)$;

The subargument π_2 is computed as follows:

Assume $B = g_{1,v}^{r_b} \prod g_{1,\lambda_i}^{b_i}$; `/* we want to show that` $\boldsymbol{b} \neq \boldsymbol{0}$ `*/`
Assume that $\mathring{g}_{1,i} = g_{1i}^{\alpha}$ and $\mathring{g}_2 = g_2^{\mathring{\alpha}}$ for a secret $\mathring{\alpha}$;
Create $\mathring{B} \leftarrow \mathring{g}_{1,v}^{r_b} \cdot \prod_{i=1}^{n} \mathring{g}_{1,\lambda_i}^{b_i}$ and a hybrid $B^* \leftarrow g_{1,v}^{r_b} \cdot \prod \mathring{g}_{1,\lambda_i}^{b_i}$;
`/* Verifier can check that` $\mathring{B}$ `is correct by checking that` $\hat{e}(\mathring{B}, g_2) = \hat{e}(B, \mathring{g}_2)$ `*/`
Show that $\mathring{B}/B^* = (\mathring{g}_{1,v}/g_{1,v})^{r_b}$ commits to zero by using the zero argument from [LZ12];
Verifier checks that $\hat{e}(B, \mathring{g}_2) \neq \hat{e}(B^*, g_2)$;

The correctness of this subargument is self-evident: it shows that $\mathring{B}$ commits to the same value (and uses the same randomizer) as B. It also shows B^* commits to the same value as both B and $\mathring{B}$. More precisely, the zero argument convinces the verifier that B^* is correctly computed from $\mathring{B}$. Therefore the last check shows that B does not commit to 0, since otherwise $\hat{e}(B, \mathring{g}_2) = \hat{e}(B^*, g_2)$.

The subargument π_4 is computed by the prover by creating a shift argument $\mathsf{shift}(\llbracket D \rrbracket) = \llbracket CD \rrbracket$ that proves that $d_i = c_{i+1} + d_{i+1}$ and $d_n = 0$. Thus, $d_n = 0$, $d_{n-1} = c_n$, $d_{n-2} = c_{n-1} + d_{n-1} = c_{n-1} + c_n$, and in general, $d_i = \sum_{j>i} c_j$. Thus, $d_1 = \sum_{j>1} c_j$. Thus the last element of $\boldsymbol{c} + \boldsymbol{d}$ is $\sum a_i b_i$.

Finally, π_5 is computed by using the restriction argument from [Gro10], that adds linear number of elements to CRS, but has a constant complexity otherwise.

The resulting subset sum argument is arguably simpler than the circuit SAT argument of [Gro10,Lip12]. Moreover, instead of the product and permutation arguments it only uses product and a more efficient shift argument (zero argument is trivial).

6.2 Improved Range Argument

Since the used commitment scheme is homomorphic, the generic range argument (prove that the committed value belongs to range $[L, H]$ for $L < H$) is equivalent to proving that the committed value belongs to $[0, H]$ for $H > 0$. In what follows, we will therefore concentrate on this simpler case. In [CLZ12], the authors proposed a new range argument that is based on the product and permutation arguments from [Lip12]. Interestingly enough, [CLZ12] makes use of the permutation argument only to show that a vector is a scan of another vector. More precisely, they first apply a permutation argument, followed by a product argument (meant to modify a rotation to a shift by clearing out one of the elements).

Therefore, we can replace the product and permutation arguments from [Lip12] with the product and shift arguments (or with the product and scan arguments) from the current paper. Thus, we can

Table 2. Comparison of NIZK arguments for range proof. Here, $\mathfrak{m}/\mathfrak{m}_b/\mathfrak{e}/\mathfrak{p}$ means the number of multiplications in $\mathbb{Z}_p$, bilinear-group multiplications, exponentiations and pairings. Communication is given in group elements. Here, $n \approx \log_u H$, $n_v = \lfloor \log_2(u-1) \rfloor$, $h = \log_2 H$, $N = r_3^{-1}(h) = o(h 2^{2\sqrt{2\log_2 h}})$, and $N^* = r_3^{-1}(\sqrt{h}) = o(\sqrt{h} \cdot 2^{2\sqrt{\log_2 h}})$.

	CRS length	Arg. length	Prover comp.	Verifier comp.
[RKP09]	$\Theta(1)$	$\Theta(h)$	$\Theta(h)$	$\Theta(h)$
[RKP09]	$\Theta(\frac{h}{\log h})$	$\Theta(\frac{h}{\log h})$	$\Theta(\frac{h}{\log h})$	$\Theta(\frac{h}{\log h})$
Chaabouni, Lipmaa, and Zhang [CLZ12]				
General	$\Theta(r_3^{-1}(n))$	$5n_v + 40$	$\Theta(n^2 n_v)\mathfrak{m} + \Theta(r_3^{-1}(n) n_v)\mathfrak{e}$	$\Theta(n)\mathfrak{e} + (9n_v + 81)\mathfrak{p}$
$u = 2$	$\Theta(N)$	40	$\Theta(h^2)\mathfrak{m} + \Theta(N)\mathfrak{e}$	$\Theta(h)\mathfrak{e} + 81\mathfrak{p}$
$u = 2^{\sqrt{h}}$	$\Theta(N^*)$	$\approx 5\sqrt{h} + 40$	$\Theta(h^{3/2})\mathfrak{m} + \Theta(\sqrt{h} \cdot N^*)\mathfrak{e}$	$\approx \Theta(\sqrt{h})\mathfrak{e} + (9\sqrt{h} + 81)\mathfrak{p}$
$u = H$	$\Theta(1)$	$\approx 5h + 40$	$\Theta(h)\mathfrak{m} + \Theta(h)\mathfrak{e}$	$\approx \Theta(1)\mathfrak{e} + (9h + 81)\mathfrak{p}$
The current paper				
General	$\Theta(r_3^{-1}(n))$	$5n_v + 31$	$\Theta(r_3^{-1}(n) \log r_3^{-1}(n) \cdot n_v)\mathfrak{m} + \Theta(r_3^{-1}(n) n_v)\mathfrak{m}_b$	$(9n_v + 65)\mathfrak{p}$
$u = 2$	$\Theta(N)$	31	$\Theta(N \cdot \log N)\mathfrak{m} + \Theta(N)\mathfrak{m}_b$	$65\mathfrak{p}$
$u = 2^{\sqrt{h}}$	$\Theta(N^*)$	$\approx 5\sqrt{h} + 31$	$\Theta(\sqrt{h} \cdot N^* \cdot \log N^*)\mathfrak{m} + \Theta(\sqrt{h} \cdot N^*)\mathfrak{m}_b$	$\approx (9\sqrt{h} + 65)\mathfrak{p}$
$u = H$	$\Theta(1)$	$\approx 5h + 31$	$\Theta(h)\mathfrak{m} + \Theta(h)\mathfrak{m}_b$	$\approx (9h + 65)\mathfrak{p}$

base the range argument on a progression-free set Λ, without additionally requiring Λ to consist of odd integers. The resulting range argument will also be shorter by one product argument.

Moreover, the use of new basic arguments will decrease the number of $\mathbb{Z}_p$-multiplications — except the cost of computing the multi-exponentiations — in the main range argument from $\Theta(n^2 n_v)$, where $n_v \approx \log_2 u$, to $\Theta(r_3^{-1}(n) \cdot \log r_3^{-1}(n) \cdot n_v) = o(\log H \cdot 2^{2\sqrt{2\log_2 \log_u H}} \cdot \log \log_u H)$. By using Pippenger's multi-exponentiation algorithm [Pip80], we get the cost of multi-exponentiation down to

$$(2 + o(1)) \cdot \frac{2 r_3^{-1}(n) \log_2(un)}{\log_2 r_3^{-1}(n)}$$

multiplications in bilinear groups. The communication will decrease by $4 + 2 + 3 = 9$ group elements, due to the replacement of the permutation argument with the shift argument (minus 4), having one less product argument (minus 2), and also because one needs to commit to one less element ($(C_{\mathsf{rot}}, \hat{C}_{\mathsf{rot}}, \tilde{C}_{\mathsf{rot}})$ in [CLZ12], minus 3). The verifier also has to perform $7+5+4 = 16$ less pairings, due to the replacement of the permutation argument with the shift argument (minus 7) and one less product argument (minus 5). Also, it is not necessary anymore to verify the correctness of $(C_{\mathsf{rot}}, \hat{C}_{\mathsf{rot}}, \tilde{C}_{\mathsf{rot}})$ (minus 4). One can analogously compute the verifier's computational complexity, see Tbl. 2.

Remark 1. In the permutation argument of [Lip12], the verifier also has to compute certain triple $(T^*, \hat{T}^*, T_2^*)$ by using 3 multi-exponentiations. This is not included in the comparison table (or the claims) in [Lip12], and the same mistake was replicated in [CLZ12]. Tbl. 1 and Tbl. 2 correct this mistake, by giving the correct complexity estimation of the arguments from [Lip12,CLZ12].[2]

Since the non-balanced range argument only uses one permutation argument, the corrected permutation argument of the current paper makes the argument shorter only by 4 group elements, and decreases the verifier's workload by 7 pairings.

One can consider now several settings. The setting $u = 2$ minimizes the communication and the verifier's computational complexity. The setting $u = 2^{\sqrt{\log_2 H}}$ minimizes the summatory length of the CRS and the argument. The setting $u = H$ minimizes the prover's computational complexity. See Tbl. 2.

[2] We note that the range argument from [CLZ12] only uses the permutation argument with one fixed permutation (rotation), and thus the value $(T^*, \hat{T}^*, T_2^*)$, that corresponds to this concrete permutation, can be put to the CRS. Therefore, if one applies this small modification, the verifier's computational complexity in the range argument actually does not increase compared to what was claimed in [CLZ12]. Since [CLZ12] itself did not mention this, we consider it to be an additional small contribution of the current paper.

6.3 Decision Knapsack Argument

Finally, we will construct also an argument for the following problem.

Definition 7 (Decision knapsack problem). *In a decision knapsack problem one has to decide, given a set $\mathcal{S}$, integers W and B, and a benefit value b_i and weight w_i of every item of $\mathcal{S}$, whether there exists a subset $\mathcal{T} \subseteq \mathcal{V}$, such that $\sum_{i\in\mathcal{T}} w_i \leq W$ and $\sum_{i\in\mathcal{T}} b_i \geq B$.*

It is known that the decision knapsack problem is NP-complete, see [BCJ11,DDKS12] for the best known (exponential-time) algorithms. One can obviously combine a version of the subset sum argument of the current section with the range argument of Sect. 6.2 to construct a decision knapsack argument, where the prover convinces the verifier that he knows such a subset $\mathcal{T}$. In a nutshell, the argument is as follows.

Let $t_i = 1$ iff $i \in \mathcal{T}$;
Prover generates a commitment T of $\boldsymbol{t}$;
Prover proves that T is Boolean by using a product argument π_1;
Prover generates a commitment W_T of $(w_1t_1, \ldots, w_nt_n)$;
Prover proves that W_T has been computed correctly by using a product argument π_2;
Prover generates a scan A of W_T, $a_i = \sum_{j>i} w_jt_j$;
Prover proves that A has been computed correctly by using a scan argument π_3;
Prover generates a commitment C of $(0, \ldots, 0, \sum_{i=1}^{n} w_it_i)$;
Prover proves that C has been created correctly by using a product argument π_4;
Analogously, prover generates commitments B_T, D, and E, and arguments π_5, π_6 and π_6 to show that E commits to $(0, \ldots, 0, \sum_{i=1}^{n} b_it_i)$;
Prover proves that the last element of C is $\leq W$ by using a range argument π_7;
Prover proves that the last element of E is $\geq B$ by using a range argument π_8;
The whole argument is $(T, W_T, A, C, B_T, D, E, \pi_1, \ldots, \pi_8)$;

It is clear from the description of this argument that it works correctly.

Acknowledgments. This work was partially done while the second author was working at the University of Tartu, Estonia. The first author was supported by Estonian Science Foundation, grant #9303, and European Union through the European Regional Development Fund.

References

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

BCJ11. Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved Generic Algorithms for Hard Knapsacks. In Kenny Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 364–385, Tallinn, Estonia, May15–19, 2011. Springer, Heidelberg.

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.

BF01. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from The Weil Pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, USA, August 19–23, 2001. Springer, Heidelberg.

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.

Bla02. Matt Blaze, editor. *FC 2002*, volume 2357 of *LNCS*, Southhampton Beach, Bermuda, March 11–14, 2002. Springer, Heidelberg.

Ble90. Guy Blelloch. *Vector Models for Data-Parallel Computing.* MIT Press, 1990.

BN05. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In Bart Preneel and Stafford E. Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, Kingston, ON, Canada, August 11–12, 2005. Springer, Heidelberg.

Bou00. Fabrice Boudot. Efficient Proofs That a Committed Number Lies in an Interval. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 431–444, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg.

BP04. Mihir Bellare and Adriana Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289, Santa Barbara, USA, August 15–19, 2004. Springer, Heidelberg.

CCs08. Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient Protocols for Set Membership and Range Proofs. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 234–252, Melbourne, Australia, December 7–11, 2008. Springer, Heidelberg.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. In Jeffrey Scott Vitter, editor, *STOC 1998*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998.

CGS97. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Fumy [Fum97], pages 103–118.

CLs10. Rafik Chaabouni, Helger Lipmaa, and abhi shelat. Additive Combinatorics and Discrete Logarithm Based Range Protocols. In Ron Steinfeld and Philip Hawkes, editors, *ACISP 2010*, volume 6168 of *LNCS*, pages 336–351, Sydney, Australia, July 5–7, 2010. Springer, Heidelberg.

CLZ12. Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A Non-Interactive Range Proof with Constant Communication. In Angelos Keromytis, editor, *FC 2012*, volume 7397 of *LNCS*, pages 179–199, Bonaire, The Netherlands, February 27–March 2, 2012. Springer, Heidelberg.

CT65. James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19:297–301, 1965.

Dam91. Ivan Damgård. Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In Joan Feigenbaum, editor, *CRYPTO 1991*, volume 576 of *LNCS*, pages 445–456, Santa Barbara, California, USA, August 11–15, 1991. Springer, Heidelberg, 1992.

DDKS12. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. In Safavi-Naini and Canetti [SNC12], pages 719–740.

DJ01. Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136, Cheju Island, Korea, February 13–15, 2001. Springer, Heidelberg.

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.

Elk11. Michael Elkin. An Improved Construction of Progression-Free Sets. *Israeli Journal of Mathematics*, 184:93–128, 2011.

ET36. Paul Erdős and Paul Turán. On Some Sequences of Integers. *Journal of the London Mathematical Society*, 11(4):261–263, 1936.

Fum97. Walter Fumy, editor. *EUROCRYPT 1997*, volume 1233 of *LNCS*, Konstanz, Germany, 11–15 May 1997. Springer, Heidelberg.

GGPR12. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic Span Programs and Succinct NIZKs without PCPs. Technical Report 2012/215, International Association for Cryptologic Research, April 19, 2012. Available at http://eprint.iacr.org/2012/215, last retrieved version from June 18, 2012.

GK03. Shafi Goldwasser and Yael Tauman Kalai. On the (In)security of the Fiat-Shamir Paradigm. In *FOCS 2003*, pages 102–113, Cambridge, MA, USA, October, 11–14 2003. IEEE, IEEE Computer Society Press.

GL07. Jens Groth and Steve Lu. A Non-interactive Shuffle with Pairing Based Verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg.

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

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, Heidelberg.

GS66. W. Morven Gentleman and Gordon Sande. Fast Fourier Transforms — For Fun And Profit. In *Fall Joint Computer Conf. AFIPS Proc.*, volume 29, pages 563–578, Washington, DC, USA, 1966. ACM.

GS08. Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In Nigel Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg.

GW10. Ben Green and Julia Wolf. A Note on Elkin's Improvement of Behrend's Construction. In David Chudnovsky and Gregory Chudnovsky, editors, *Additive Number Theory*, pages 141–144. Springer New York, 2010.

GW11. Craig Gentry and Daniel Wichs. Separating Succinct Non-Interactive Arguments from All Falsifiable Assumptions. In Salil Vadhan, editor, *STOC 2011*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.

HSV06. Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The Eta Pairing Revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

LAN02. Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In Blaze [Bla02], pages 87–101.

Lip03. Helger Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In Chi Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 398–415, Taipei, Taiwan, November 30–December 4, 2003. Springer, Heidelberg.

Lip11. Helger Lipmaa. Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. Technical Report 2011/009, International Association for Cryptologic Research, January 5, 2011. Available at http://eprint.iacr.org/2011/009.

Lip12. Helger Lipmaa. Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Italy, March 18–21, 2012. Springer, Heidelberg.

LZ12. Helger Lipmaa and Bingsheng Zhang. A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In Ivan Visconti and Roberto De Prisco, editors, *SCN 2012*, volume 7485 of *LNCS*, pages 477–502, Amalfi, Italy, September 5–7, 2012. Springer, Heidelberg.

Pip80. Nicholas Pippenger. On the Evaluation of Powers and Monomials. *SIAM Journal of Computing*, 9(2):230–250, 1980.

PSNB11. C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto. A Family of Implementation-Friendly BN Elliptic Curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.

RKP09. Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally Composable Adaptive Priced Oblivious Transfer. In Hovav Shacham and Brent Waters, editors, *Pairing 2009*, volume 5671 of *LNCS*, pages 231–247, Palo Alto, CA, USA, August 12–14, 2009. Springer, Heidelberg.

San11. Tom Sanders. On Roth's Theorem on Progressions. *Annals of Mathematics*, 174(1):619–636, July 2011.

Sch80. Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980.

Sho97. Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In Fumy [Fum97], pages 256–266.

SNC12. Rei Safavi-Naini and Ran Canetti, editors. *CRYPTO 2012*, volume 7417 of *LNCS*, Santa Barbara, California, USA, August 19–23, 2012. Springer, Heidelberg.

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

vHN10. Mark van Hoeij and Andrew Novocin. Gradual Sub-lattice Reduction and a New Complexity for Factoring Polynomials. In Alejandro López-Ortiz, editor, *LATIN 2010*, volume 6034 of *LNCS*, pages 539–553, Oaxaca, Mexico, April 19–23, 2010. Springer, Heidelberg.

Wat12. Brent Waters. Functional Encryption for Regular Languages. In Safavi-Naini and Canetti [SNC12], pages 218–235.