# Efficient Modular NIZK Arguments from Shift and Product

Prastudy Fauzi[1], Helger Lipmaa[1], and Bingsheng Zhang[2]

[1] University of Tartu, Estonia
[2] National and Kapodistrian University of Athens, Greece

**Abstract.** We propose a non-interactive product argument, that is more efficient than the one by Groth and Lipmaa, and a novel shift argument. We then use them to design several novel non-interactive zero-knowledge (NIZK) arguments. We obtain the first range proof with constant communication and subquadratic prover's computation. We construct NIZK arguments for **NP**-complete languages, Set-Partition, Subset-Sum and Decision-Knapsack, with constant communication, subquadratic prover's computation and linear verifier's computation.

**Keywords:** FFT, multi-exponentiation, non-interactive zero knowledge, product argument, range argument, shift argument.

## 1 Introduction

By using a zero knowledge proof [20], a prover can prove the correctness of a statement without leaking any side information. Efficient non-interactive zero knowledge (NIZK, [4]) proofs are crucial in the design of cryptographic protocols. A typical application is e-voting, where the voters must prove the correctness of encrypted ballots and the servers must prove the correctness of the tallying process. Since the voters may not be available every time when one verifies the ballots, one cannot rely on interactive zero knowledge. Moreover, it is important to have succinct (e.g., logarithmic in input size) NIZK proofs with efficient verification. For example, in e-voting, correctness proofs are collected and stored, and then verified by many independent observers.

It is well-known that NIZK proofs are impossible in the standard model without any trust assumptions. One usually constructs NIZK proofs in the common reference string (CRS, [4]) model, where all parties have access to a CRS generated by a trusted third party. (We do not consider the random oracle model, since random oracles cannot always be instantiated [9,19].) Moreover, one can only construct succinct *computationally* sound proofs, also known as arguments [7].

Only a few generic techniques of constructing succinct NIZK arguments for non-trivial languages are known, unless $\mathbf{P} = \mathbf{NP}$. In [21], Groth constructed non-interactive witness-indistinguishable (and weakly sound, see [21]) product and permutation arguments. He used them, together with some other arguments, to construct the first succinct NIZK argument for an **NP**-complete language, Circuit-SAT. The latter argument is modular, i.e., it is in a black-box way based on a small number of basic arguments. Let $n = |C|$ be the circuit size. Groth's product and permutation arguments have CRS length and prover's computation $\Theta(n^2)$, while the communication and verifier's computation are constant. (The communication is given in group elements, and the computation is in group operations.) His CIRCUIT-SAT argument has the same complexity parameters, except that the verifier's computation is $\Theta(n)$, see Table 1. (The verifier's computation in Lipmaa's argument in Table 1 differs from what was claimed in [27]. The slightly incorrect claim from [27] was also replicated in [11]. See Remark 1 on page 108.)

Lipmaa [27] improved Groth's basic arguments. Let $r_3(N) = N^{1-o(1)}$ be the size of the largest known progression-free subset [14] of $[N] = \{1, \ldots, N\}$. (See Sect. 2.) Lipmaa's basic arguments have CRS length $\Theta(r_3^{-1}(n)) = n^{1+o(1)}$, and slightly better prover's computation. This results straightforwardly in a more efficient modular CIRCUIT-SAT argument. Another important property of Lipmaa's arguments is the flexibility in choosing the progression-free set. For small values of $N$ the value $r_3(N)$ is much smaller than predicted by Elkin, [13,15]. For practically all interesting values of $n$, one should choose the Erdős-Turán progression-free set [15], which results in the CRS length $\Theta(n^{\log_2 3})$, with a very small constant. Given any progress in the theory of progression-free sets, Lipmaa's arguments can become even more efficient. Thus, Groth's and Lipmaa's basic arguments offer essentially optimal communication and verifier's computation, but they are quite inefficient in other parameters. We estimate that due to quadratic prover's computation, they can only handle circuits of size $\leq 2^{10}$.

The basic arguments of [21,27] can be used to construct other modular arguments. E.g., a modular range argument was constructed in [11]. As shown in [30], following the same framework, one can construct other basic arguments — for example, 1-sparsity in [30] — and use them to construct efficient modular arguments (shuffle in [30]). It is an important open problem to increase the library of basic arguments even further, and to investigate for which (complex) languages one can construct efficient arguments by using the basic arguments in a modular manner. Moreover, the basic arguments of Groth and Lipmaa are computationally intensive for the prover. It is desirable that the new basic arguments (that at the same time have meaningful applications) were more efficient.

We construct a more efficient variant of Lipmaa's product argument, and we propose a new efficient shift-by-$\xi$ argument. We then demonstrate the power of the modular approach, by using the product argument and the shift argument — together with some other, even simpler, arguments — to make the modular range argument of [11] more efficient, and then to construct efficient modular arguments for SET-PARTITION, SUBSET-SUM and DECISION-KNAPSACK (all **NP**-complete languages). All new arguments have constant communication, and significantly improved prover's computation ($\Theta(r_3^{-1}(n) \log n)$ versus $\Theta(n^2)$ in previous work). By using the same basic arguments, one can clearly construct modular NIZK arguments for other languages.

More precisely, we first modify the commitment scheme from [27]. In that commitment scheme (and thus also in all related NIZK arguments), one uses a progression-free set $\Lambda$ of odd positive integers. When the new commitment scheme is used, $\Lambda$ can be an arbitrary progression-free set. This is important conceptually, making it clear that one requires progression-freeness of $\Lambda$ (and nothing else) in similar arguments.

We then construct a more efficient product argument by applying two unrelated algorithmic techniques to the product argument of [27]. While not new, these techniques help us to significantly speed up the product argument, and thus also other arguments that we build on top of it. First, we use Fast Fourier Transform (FFT, [12]) based polynomial multiplication [17] to reduce the prover's computation from $\Theta(n^2)$ to $n^{1+o(1)}$ $\mathbb{Z}_p$-multiplications. 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 prover's computation and CRS length $n^{1+o(1)}$. We note that FFT-based techniques are not applicable to optimize the arguments of Groth [21], since there the largest element of $\Lambda$ is $\Theta(n^2)$.

Second, we use Pippenger's [31] algorithm to speed up multi-exponentiations. More precisely, the prover must perform $\Theta(r_3^{-1}(n))$ bilinear-group multiplications to evaluate two $\Theta(r_3^{-1}(n))$-wide bilinear-group multi-exponentiations needed in Lipmaa's product argument. This is smaller than the number of $\mathbb{Z}_p$-multiplications but since bilinear-group multiplications are more expensive, we will count them separately.

We were unable to apply FFT to the permutation argument from [27]; this is since Lipmaa's product argument has an FFT-friendly construction while the permutation argument has a more complex structure. (Thus, the idea of using FFT is not as straightforward as it may seem initially.) Instead, we propose shift-by-$\xi$ and rotation-by-$\xi$ arguments that have constant communication and verifier's computation, and linear prover's computation and CRS length. None of these complexities depends on $\xi$. Thus, the new shift and rotation arguments are (in some parameters) $\Theta(n)$ times more efficient than Groth's permutation argument. As a drawback, we prove their security only by reduction to the $\Phi$-PSDL assumption [11] (see Sect. 3), which is a generalization of the $\Lambda$-PSDL assumption from [27]. To show that the $\Phi$-PSDL assumption is reasonable, we prove that the $\Phi$-PSDL assumption is secure in the generic group model.

We show that based on the product and shift arguments, one can build efficient modular arguments for several important languages. All our applications use an intermediate scan argument that verifies that one vector is the scan [3] (or sum-of-all-prefixes) of another vector. While the scan argument can be straightforwardly constructed from the shift argument, it serves as a very useful intermediate building block.

In a range argument (or a range proof, see [6,29,26,8,10]), the prover aims to convince the verifier that the committed value belongs to an integer range $[L, H]$. Range arguments are needed in many cryptographic applications, typically in cases where for the security of the master protocol (e.g., e-voting or e-auctions) it is necessary to show that the encrypted or committed values come from a correct range. Construction of non-interactive range arguments has only taken off

**Table 1.** Comparison of modular NIZK arguments for **NP**-complete languages with (worst-case) sublinear argument size. Here, $n$ is the size of circuit, $N = r_3^{-1}(n)$, and $N^* = r_3^{-1}(\sqrt{n})$, and $m$ is the balancing parameter. Moreover, $\mathfrak{g}$ corresponds to 1 group element and $\mathfrak{a}/\mathfrak{m}/\mathfrak{m}_b/\mathfrak{e}/\mathfrak{p}$ correspond to 1 addition/$\mathbb{Z}_p$-multiplication/bilinear-group multiplication/exponentiation/pairing.

| $m$ | $\lvert\text{CRS}\rvert$ | $\lvert\text{Argument}\rvert$ | Prover comp. | Verifier comp. |
|---|---|---|---|---|
| | | Circuit-SAT arguments from [21] | | |
| $1$ | $\Theta(n^2)\mathfrak{g}$ | $\Theta(1)\mathfrak{g}$ | $\Theta(n^2)\mathfrak{e}$ | $\Theta(n)\mathfrak{m}_b + \Theta(1)\mathfrak{p}$ |
| $n^{1/3}$ | $\Theta(n^{\frac{2}{3}})\mathfrak{g}$ | $\Theta(n^{\frac{2}{3}})\mathfrak{g}$ | $\Theta(n^{4/3})\mathfrak{e}$ | $\Theta(n)\mathfrak{m}_b + \Theta(n^{\frac{2}{3}})\mathfrak{p}$ |
| | | Circuit-SAT arguments from [27] | | |
| $1$ | $\Theta(N)\mathfrak{g}$ | $\Theta(1)\mathfrak{g}$ | $\Theta(n^2)\mathfrak{a} + \Theta(N)\mathfrak{e}$ | $\Theta(n)\mathfrak{e} + 62\mathfrak{p}$ |
| $\sqrt{n}$ | $\Theta(N^*)\mathfrak{g}$ | $\Theta(\sqrt{n})\mathfrak{g}$ | $\Theta(n^{3/2})\mathfrak{a} + \Theta(\sqrt{n} \cdot N^*)\mathfrak{e}$ | $\Theta(n)\mathfrak{e} + \Theta(\sqrt{n})\mathfrak{p}$ |
| | | Set-Partition, Subset-Sum and Decision-Knapsack arguments from the current paper | | |
| $1$ | $\Theta(N)\mathfrak{g}$ | $\Theta(1)\mathfrak{g}$ | $\Theta(N \log n)\mathfrak{m} + \Theta(N)\mathfrak{m}_b$ | $\Theta(n)\mathfrak{m}_b + \Theta(1)\mathfrak{p}$ |
| $\sqrt{n}$ | $\Theta(N^*)\mathfrak{g}$ | $\Theta(\sqrt{n})\mathfrak{g}$ | $\Theta(\sqrt{n} \cdot N^* \log n)\mathfrak{m} + \Theta(\sqrt{n} \cdot N^*)\mathfrak{m}_b$ | $\Theta(n)\mathfrak{m}_b + \Theta(\sqrt{n})\mathfrak{p}$ |

during the last few years [32,11]. In [11], Chaabouni, Lipmaa and Zhang used the product and permutation arguments of [27] to construct the first known constant-communication (interactive or non-interactive) range argument over prime-order groups. They achieved this by combining the basic arguments of [21,27] with several different (and unrelated) techniques that were developed specifically for range arguments in [29,10].

We use the new basic arguments to optimize the range argument from [11], reducing the prover's computation from $\Theta(h^2)$ to $\Theta(r_3^{-1}(h) \cdot \log r_3^{-1}(h))$ multiplications in $\mathbb{Z}_p$, and from $\Theta(r_3^{-1}(h))$ bilinear-group exponentiations to $\Theta(r_3^{-1}(h))$ bilinear-group multiplications. Here, $h = \log_2(H - L)$. The new argument is the first range argument at all (i.e., not only in prime-order groups) that has constant-length arguments and subquadratic-in-$h$ prover's computation. See Sect. 6. We also note that [11] replicated the small mistake of [27] (see Remark 1) and thus the computational complexity of the argument of [11] is larger than claimed in [11]. We propose another modification of the range argument of [11] to make it even more efficient. We also discuss balanced versions of the new range argument with better prover's computation but larger communication.

We then proceed to demonstrate the power of the "shift-and-multiply" modular approach. We also construct an efficient NIZK argument for the **NP**-complete language Set-Partition (the prover knows a partition of the given set of integers to two sets that have the same sum), where the communication and computation are dominated by two product arguments and one shift argument. The new argument has parameters outlined in Table 1. In this case, $n$ denotes the cardinality of the public set. We also construct an NIZK argument for the **NP**-complete language Subset-Sum (the prover knows a non-zero subset of the given set of integers that sums to 0), with parameters outlined in Table 1. In this case, $n$ denotes the size of the input domain, that is, the public set $S$ is known to belong to $[n]$. As the final example, we show that one can combine Subset-Sum and range arguments to construct an argument for Decision-Knapsack, another **NP**-complete language.

When using the balancing techniques of [21,27] (briefly, instead of applying the basic arguments to length $n$-vectors, apply them in parallel to $m$ length-$(n/m)$ vectors), if $m = \sqrt{n}$, we obtain balanced NIZK arguments with the parameters, given in the last row of Table 1. (This means that by using the techniques of [21], one can construct a perfect zap with the same complexity.)

Gennaro et al [16] recently proposed an efficient Circuit-SAT argument based either on quadratic span programs or quadratic arithmetic programs. Their argument has prover's computation $\Theta(n \log^3 n)$, which is larger than $\Theta(n^{\log_2 3} \log n)$ for all practical values of $n$. Subsequently, Lipmaa [28] improved the prover's computation to $\Theta(n \log^2 n)$. See also [2,1]. Since these arguments explicitly rely on the efficient arithmetic-circuit representation of the underlying language, it is unclear if they can be used to construct arguments with subquadratic prover's computation for other **NP**-complete languages. (Using polynomial-time reductions between **NP**-complete languages is usually not an option since we are interested in subquadratic complexity.) Since we are considering different **NP**-complete languages, direct efficiency comparison between [16,28] and the current work is not possible. Moreover, our approach seems to be more flexible, enabling one to construct direct NIZK arguments without a reduction to Circuit-SAT.

## 2 Preliminaries

Let $[L, H] = \{L, L + 1, \dots, H\}$ and $[H] = [1, H]$. By $\boldsymbol{a}$, we denote the vector $\boldsymbol{a} = (a_1, \dots, a_n)$. Since for groups $G$ and $H$, their direct product $G \times H$ is also a group, we use freely notation like $(g, h)^a = (g^a, h^a)$. If $y = h^x$, then $\log_h y := x$. Let $\kappa$ be the security parameter. We abbreviate probabilistic polynomial-time as PPT, non-uniform PPT by NUPPT. Let $\mathrm{poly}(\kappa)/\mathrm{negl}(\kappa)$ be an arbitrary polynomial/negligible function.

If $\Lambda_1$ and $\Lambda_2$ 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 $\Lambda$ 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 $\Lambda$, and $2\widehat{\ }\Lambda = \{\lambda_1 + \lambda_2 : \lambda_1 \in \Lambda \wedge \lambda_2 \in \Lambda \wedge \lambda_1 \neq \lambda_2\} \subseteq \Lambda + \Lambda$ is a *restricted sumset*. See [35].

A set $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ is *progression-free* (or non-averaging, [15,35]), if no three elements of $\Lambda$ are in arithmetic progression, that is, $\lambda_i + \lambda_j = 2\lambda_k$ only if $i = j = k$. That is, $2\widehat{\ }\Lambda \cap 2 \cdot \Lambda = \emptyset$. Let $r_3(N)$ be the cardinality of the largest progression-free set $\Lambda \subseteq [N]$. Recently, Elkin [14] proved that

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

Thus, for any $n > 0$, there exists $N = o(n 2^{2\sqrt{2 \log_2 n}})$, such that $[N]$ contains an $n$-element progression-free subset. However, for say $N \leq 2^{25}$, the Erdős-Turán progression-free subset [15], of size $\approx N^{\log_3 2}$, is larger. For $N \leq 123$, the optimal values of $r_3(N)$ were recently computed in [13]. For any $N$, the currently best upper bound was proven by Sanders [34].

Polynomial factorization in $\mathbb{Z}_p[X]$ can be done in polynomial time [25,23]. Let $\mathsf{PolyFact}$ be an efficient polynomial factorization algorithm that on input a degree-$d$ polynomial $f$ outputs all $d+1$ roots of $f$.

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 [31], Pippenger proved that

**Fact 1.** *Assume that* $h = MN \cdot \log(B+1) \to \infty$. *Then* $L(M, N, B) = \min\{M, N\} \log B + h/\log h \cdot (1 + O((\log\log h/\log h)^{1/2})) + O(\max\{M, N\})$.

A bilinear group generator $\mathcal{G}_{\mathsf{bp}}$ outputs a description of a bilinear group [33,24,5] $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, s.t. $p$ is a $\kappa$-bit prime, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of order $p$ (with identity elements 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 $g_z$ generates $\mathbb{G}_z$ for $z \in \{1, 2\}$, then $\hat{e}(g_1, g_2)$ generates $\mathbb{G}_T$. Also, it is efficient to decide membership in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, group operations and the pairing are efficiently computable, generators are efficiently sampleable, and the descriptions of the groups and group elements each are $O(\kappa)$ bits long. An optimal Ate pairing [22] over a subclass of Barreto-Naehrig curves can be implemented efficiently. Then, 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.

A trapdoor commitment scheme [7] $\Gamma$ consists of five PPT algorithms: a randomized common reference string (CRS) generation algorithm $\mathcal{G}_{\mathsf{com}}$, 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 $\mathcal{O}\mathrm{pen}_{td}$. Here, (1) the CRS generation algorithm $\mathcal{G}_{\mathsf{com}}(1^\kappa)$ produces a CRS $\mathsf{ck}$, (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 $\mathsf{ck}_{td}$, which has the same distribution as $\mathcal{G}_{\mathsf{com}}(1^\kappa)$, and a trapdoor $\mathsf{td}$, (4) the randomized trapdoor commitment algorithm $\mathcal{C}\mathrm{om}_{td}(\mathsf{ck}_{td}; r)$ takes $\mathsf{ck}_{td}$ and a randomizer $r$ as inputs, and outputs $\mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \boldsymbol{0}; r)$, 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 $\mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \boldsymbol{0}; r) = \mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \boldsymbol{a}; r_{td})$.

$\Gamma$ is *computationally binding*, if no NUPPT adversary can open a commitment to two different values. That is, for every NUPPT $\mathcal{A}$,

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

is negligible in $\kappa$. $\Gamma$ is *perfectly hiding*, if the commitments of any two messages have the same distribution. That is, for any $\mathsf{ck} \in \mathcal{G}_{\mathsf{com}}(1^\kappa)$ and any $\boldsymbol{a_1}, \boldsymbol{a_2}$, 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.

The new commitment scheme allows committing to vectors of predetermined length $n$. Thus, one must input $n$ (or a reasonable upper bound on $n$) as an additional parameter for the (trapdoor) CRS generation algorithms. We assume that the value of $n$ is implicitly obvious while committing and trapdoor opening.

Let $\mathcal{R} = \{(C, w)\}$ be an efficiently computable binary relation with $|w| = \mathrm{poly}(|C|)$. 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 $n = |C|$ be the input length. For fixed $n$, we have a relation $\mathcal{R}_n$ and a language $\mathcal{L}_n$. A *non-interactive argument* for $\mathcal{R}$ consists of three 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 $\pi$, and $\mathcal{V}(\mathsf{crs}; C, \pi)$ outputs either 1 (accept) or 0 (reject).

$\Pi$ is *perfectly complete*, if for all $n = \mathrm{poly}(\kappa)$,

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

$\Pi$ is *computationally sound*, if for all $n = \mathrm{poly}(\kappa)$ and NUPPT $\mathcal{A}$,

$$\Pr[\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n), (C, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) : C \not\in \mathcal{L} \wedge \mathcal{V}(\mathsf{crs}; C, \pi) = 1] = \mathrm{negl}(\kappa) \enspace .$$

$\Pi$ 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. $\Pi$ is *perfectly zero-knowledge*, if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, such that for all stateful NUPPT adversaries $\mathcal{A}$ and $n = \mathrm{poly}(\kappa)$ (with $\mathsf{td}_\pi$ being the *simulation trapdoor*),

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

## 3 New Commitment Scheme

Let $\Lambda = (\lambda_1, \dots, \lambda_n) \in \mathbb{Z}^n$ and $v \in \mathbb{Z}$. Next, we define the $(\Lambda, v)$ *trapdoor commitment scheme* in group $\mathbb{G}_z$, $z \in \{1, 2\}$. See Prot. 1. Intuitively, $\boldsymbol{a} = (a_1, \dots, a_n)$ is committed to as $g_z^{r\sigma^v + \sum a_i \sigma^{\lambda_i}}$, where $r$ is the randomness, $g_z$ is a generator of $\mathbb{G}_z$, and $\sigma$ is the secret key. Groth [21] proposed a variant of this commitment scheme with $\Lambda = [n]$ and $v = 0$, while Lipmaa [27] generalized $\Lambda$ to any set $\Lambda$ with $0 < \lambda_i < \lambda_{i+1}$ and $\lambda_n = \mathrm{poly}(\kappa)$ (while still letting $v = 0$).

We use the following security assumptions from [11]. 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 $\kappa$. Let 1 be the polynomial with $1(x) = 1$ for all $x \in \mathbb{Z}_p$.

**Definition 1.** *$\mathcal{G}_{\mathsf{bp}}$ is $\Phi$-*PDL secure in $\mathbb{G}_z$*, if for any NUPPT $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa), g_z \leftarrow \mathbb{G}_z \setminus \{1\}, \sigma \leftarrow \mathbb{Z}_p :\\ \mathcal{A}(\mathsf{gk}; (g_z^{\varphi(\sigma)})_{\varphi \in \{1\} \cup \Phi}) = \sigma\end{array}\right] = \mathrm{negl}(\kappa) \enspace .$$

$\mathcal{G}_{\mathsf{bp}}$ *is* $\varPhi$-PSDL secure*, if for any NUPPT* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa), g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}, \\ g_2 \leftarrow \mathbb{G}_2 \setminus \{1\}, \sigma \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{gk}; (g_1^{\varphi(\sigma)}, g_2^{\varphi(\sigma)})_{\varphi \in \{1\} \cup \varPhi}) = \sigma\end{array}\right] = \operatorname{negl}(\kappa) \enspace .$$

A much stronger version of the P(S)DL assumption was recently used in [2].

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

(See App. A for a proof.) As shown in [18], sublinear NIZK proofs are only possible under non-standard (e.g., knowledge) assumptions. We use the following knowledge assumption from [11]. For algorithms $\mathcal{A}$ and $X_{\mathcal{A}}$, we write $(y; y_X) \leftarrow (\mathcal{A}||X_{\mathcal{A}})(\sigma)$ if $\mathcal{A}$ on input $\sigma$ outputs $y$, and $X_{\mathcal{A}}$ on the same input (including the random tape of $\mathcal{A}$) outputs $y_X$.

**Definition 2.** *Let* $z \in \{1, 2\}$*.* $\mathcal{G}_{\mathsf{bp}}$ *is* $\varPhi$-PKE secure in $\mathbb{G}_z$ *if for any NUPPT* $\mathcal{A}$ *there exists an NUPPT extractor* $X_{\mathcal{A}}$*, s.t. the following probability is negligible:*

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

One can generalize the proof from [21] to show that $\varPhi$-PKE holds in the generic group model. Let $z = 1$. Consider a CRS $\mathsf{ck}$ 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 $z = 2$ is dual. The following theorem generalizes the corresponding results from [21,27].

**Theorem 2.** *Let* $z \in \{1, 2\}$*. Let* $\varLambda = (\lambda_1, \ldots, \lambda_n)$ *with* $\lambda_i < \lambda_{i+1}$ *and* $\lambda_i = \operatorname{poly}(\kappa)$*. Let* $\upsilon > \lambda_n$ *be linear in* $\lambda_n - \lambda_1$*. Let* $\varGamma$ *be the* $(\varLambda, \upsilon)$ *knowledge commitment scheme in* $\mathbb{G}_z$ *of Prot. 1. Let*

$$\varPhi_\varGamma := \{X^\upsilon\} \cup \{X^\ell\}_{\ell \in \varLambda} \enspace .$$

*Then*

**System parameters:** $\mathcal{G}_{\mathsf{bp}}$, $n = \operatorname{poly}(\kappa)$, $\varLambda = \{\lambda_1, \ldots, \lambda_n\}$ with $\lambda_i < \lambda_{i+1}$, $\lambda_i = \operatorname{poly}(\kappa)$, and $\upsilon > \max_i \lambda_i$;

$\mathcal{G}\mathbf{com}_{td}(1^\kappa, n)$**:** Set $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_z \leftarrow \mathbb{G}_z \setminus \{1\}$, $(\sigma, \hat{\alpha}) \leftarrow \mathbb{Z}_p^2$; For $i \in [n]$ do: $(g_{z,\lambda_i}, \hat{g}_{z,\lambda_i}) \leftarrow (g_z, g_z^{\hat{\alpha}})^{\sigma^{\lambda_i}}$; Set $(h_z, \hat{h}_z) \leftarrow (g_z, g_z^{\hat{\alpha}})^{\sigma^\upsilon}$; Let $\mathsf{ck} \leftarrow (\mathsf{gk}; (g_{z,\lambda_i} \hat{g}_{z,\lambda_i})_{i \in [n]}, h_z, \hat{h}_z)$; Return $(\mathsf{ck}; \mathsf{td} \leftarrow \sigma)$;

$\mathcal{G}_{\mathsf{com}}(1^\kappa, n)$**:** $(\mathsf{ck}; \mathsf{td}) \leftarrow \mathcal{G}_{\mathsf{com}}(1^\kappa)$; return $\mathsf{ck}$;

$\mathcal{C}\mathbf{om}(\mathsf{ck}; \boldsymbol{a}; \cdot)$**,** $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$**:** $r \leftarrow \mathbb{Z}_p$; return $(h_z, \hat{h}_z)^r \cdot \prod_{i=1}^n (g_{z,\lambda_i}, \hat{g}_{z,\lambda_i})^{a_i}$;

$\mathcal{C}\mathbf{om}_{td}(\mathsf{ck}_{td}; \cdot)$**:** $r \leftarrow \mathbb{Z}_p$; return $(h_z, \hat{h}_z)^r$;

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

**Protocol 1:** The $(\varLambda, \upsilon)$ trapdoor commitment scheme in $\mathbb{G}_z$ for $z \in \{1, 2\}$

*(a) $\varGamma$ is perfectly hiding, and computationally binding under the $\varPhi_\varGamma$-PDL assumption in $\mathbb{G}_z$. The reduction overhead is dominated by the time to factor a degree-$(\upsilon - \lambda_1)$ polynomial in $\mathbb{Z}_p[X]$.*

*(b) If $\varPhi_\varGamma$-PKE holds in $\mathbb{G}_z$, then for any NUPPT $\mathcal{A}$ that outputs a valid commitment $C$, there exists an NUPPT extractor $X_\mathcal{A}$ that, given the input of $\mathcal{A}$ together with $\mathcal{A}$'s random coins, extracts the contents of $C$.*

*Proof.* Perfect hiding: follows since the output of $\mathcal{C}om$ is a random element of $\mathbb{G}_1$. Computational binding: Assume $\mathcal{A}_{\mathcal{C}om}$ is an adversary that can break the binding property with non-negligible probability. We construct the following adversary $\mathcal{A}_{pdl}$, see Prot. 1, against the $\varPhi_\varGamma$-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 $\mathsf{gk} \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_z \leftarrow \mathbb{G}_z \setminus \{1\}$, and $\sigma \leftarrow \mathbb{Z}_p$;
  $\mathcal{C}$ sends $(\mathsf{gk}; (g_z^{\sigma^\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$ to $\mathcal{A}_{pdl}$;
  $\mathcal{A}_{pdl}$ sets $\hat{\alpha}^* \leftarrow \mathbb{Z}_p$;
  $\mathcal{A}_{pdl}$ sets $\mathsf{ck} \leftarrow (\mathsf{gk}; ((g_z, g_z^{\hat{\alpha}^*})^{\sigma^\ell})_{\ell \in \Lambda}, (g_z, g_z^{\hat{\alpha}^*})^{\sigma^\upsilon})$;
1 $\mathcal{A}_{pdl}$ obtains $(\boldsymbol{a}, r_a, \boldsymbol{b}, r_b) \leftarrow \mathcal{A}_{\mathcal{C}om}(\mathsf{ck})$;
  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 $(t_1, \dots, t_{\upsilon - \lambda_1 + 1}) \leftarrow \mathsf{PolyFact}(\delta)$;
3     $\mathcal{A}_{pdl}$ finds by an exhaustive search a root $\sigma_0 \in \{t_i\}_{i=1}^{\upsilon - \lambda_1 + 1}$, s.t. $g_z^{\sigma^{\lambda_1}} = g_z^{\sigma_0^{\lambda_1}}$;
      $\mathcal{A}_{pdl}$ returns $\sigma \leftarrow \sigma_0$ to the challenger;
  end
```

**Algorithm 1.** Adversary in Thm. 2

Assume that on step 1, $\mathcal{A}_{\mathcal{C}om}$ is successful with some probability $\varepsilon_c$. Thus, with probability $\varepsilon_c$, $(\boldsymbol{a}, r_a) \neq (\boldsymbol{b}, r_b)$ and

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

But then

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

and thus

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

or equivalently,

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

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 $f(\sigma) \equiv 0 \pmod{p}$. Hence, $\mathcal{A}_{pdl}$ can use polynomial factorization to find all roots of $\delta$, and one of those roots must be equal to $\sigma$. On step 3, $\mathcal{A}_{pdl}$ finds the correct root by an exhaustive search among all roots returned in the previous step. Thus, clearly $\mathcal{A}_{pdl}$ returns the correct value of $\mathsf{sk}$ (and thus violates the $\Phi_\Gamma$-PDL assumption) with probability $\varepsilon_c$. Finally, the time of $\mathcal{A}_{pdl}$ is clearly dominated by the execution time of $\mathcal{A}_{\mathcal{C}\mathrm{om}}$ and the time to factor $\delta$.

Extractability: By the $\Phi_\Gamma$-PKE assumption in group $\mathbb{G}_z$, for every committer $\mathcal{A}$ there exists an extractor $X_\mathcal{A}$ that can open the commitment in group $\mathbb{G}_z$, given access to $\mathcal{A}$'s inputs and random tape. Since $\Gamma$ is computationally binding, then the extracted opening has to be the same that $\mathcal{A}$ used. □

Sometimes, we 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.

Let $\alpha = ||\boldsymbol{a}||_\infty = \max_i a_i$, and $n \geq 2$. When using Pippenger's algorithm, the computation of $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r)$ is dominated by $L(2, n, \alpha) = 2 \log_2 \alpha + (2 + o(1)) \cdot n \log_2 \alpha / \log_2(n \log_2 \alpha) + O(n)$ $\mathbb{G}_z$-multiplications. In our applications, $n \gg \log_2 \alpha$ (e.g., $\alpha = 2$, $\alpha = n$, or even $\alpha = p$ given that $n$ is reasonably large), and thus we get a simpler bound of $(2 + o(1)) \log_2 \alpha \cdot n / \log_2 n + O(n)$ multiplications. This can be compared to $3n \log_2 \alpha$ multiplications on average when using the square-and-multiply exponentiation algorithm.

## 4 Improved Hadamard Product Argument

Next, we propose a version of the product argument of [27] with respect to the $(\Lambda, \upsilon)$ commitment scheme of Sect. 3. As we will see (both in this section and in Sect. 5), the value of $\upsilon$ depends on the construction of the argument. E.g., while the commitment scheme is binding for $\upsilon > \lambda_n$, for the product argument to be (weakly[1]) sound we need $\upsilon > 2\lambda_n - \lambda_1$. If one uses several such arguments together (e.g., to construct a range argument or a Subset-Sum argument), one has to choose a value of $\upsilon$ 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.

Assume that $\Gamma$ is a trapdoor commitment scheme that commits to $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ for $n \geq 1$. In an *Hadamard product argument*, the prover aims to convince the verifier that given commitments $A$, $B$ and $C$, he can open them as $A = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r_a)$, $B = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{b}; r_b)$, and $C = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{c}; r_c)$, s.t. $c_i = a_i b_i$ for $i \in [n]$. A product argument has $n$ constraints $c_i = a_i b_i$ for $i \in [n]$.

Lipmaa [27] constructed a product argument for the $(\Lambda, 0)$ 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. Prot. 2 presents a more efficient variant of this argument for

[1] For an explanation and motivation of weak soundness, we refer the reader to [21,27].

the $(\Lambda, \upsilon)$ commitment scheme $\Gamma$. Similarly to [27], we use $\Gamma$ 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 $\mathbb{G}_1$, and $\widehat{\mathsf{ck}}^*$ be the dual CRS in group $\mathbb{G}_2$ (i.e., $\widehat{\mathsf{ck}}^*$ is defined as $\widehat{\mathsf{ck}}$, but with $g_1$ replaced by $g_2$). Thus, e.g., $(B, \hat{B}) = \mathcal{C}\mathrm{om}(\widehat{\mathsf{ck}}; \boldsymbol{b}; r_b)$. Then, $\log_{g_1} A = r_a \sigma^\upsilon + \sum_{i=1}^n a_i \sigma^{\lambda_i}$, $\log_{g_1} B = r_b \sigma^\upsilon + \sum_{i=1}^n b_i \sigma^{\lambda_i}$, and $\log_{g_1} C = c_i \sigma^\upsilon + \sum_{i=1}^n r_c \sigma^{\lambda_i}$. The prover also computes an element $B_2$, s.t. $\hat{e}(g_1, B_2) = \hat{e}(B, g_2)$. Thus, for $(D, \hat{D}) = \mathcal{C}\mathrm{om}(\widehat{\mathsf{ck}}^*; \mathbf{1}; 0)$ (in $\mathbb{G}_2$), $\log_{\hat{e}(g_1, g_2)}(\hat{e}(A, B_2)/\hat{e}(C, D)) = (r_a \sigma^\upsilon + \sum_{i=1}^n a_i \sigma^{\lambda_i})(r_b \sigma^\upsilon + \sum_{i=1}^n b_i \sigma^{\lambda_i}) - (r_c \sigma^\upsilon + \sum_{i=1}^n c_i \sigma^{\lambda_i})(\sum_{i=1}^n \sigma^{\lambda_i})$ can be written — after substituting $\sigma$ with a formal variable $X$ — as a sum of two formal polynomials $F_{con}(X)$ and $F_\pi(X)$, s.t. $F_{con}(X)$ (the constraint polynomial) has one monomial per constraint ($a_i b_i = c_i$) and is 0 if the prover is honest, while $F_\pi(X)$ has many more monomials. More precisely, $F_\pi$ 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 assumptions guarantee that he cannot do it if at least one of the $n$ constraints is not satisfied.

In [27], for soundness, one had to assume that the used set $\Lambda$ is a progression-free set of odd positive integers. By using such $\Lambda$, [27] proved that the polynomials $F_{con}(X)$ and $F_\pi(X)$ were spanned by two non-intersecting sets of powers of $X$. From this, [27] then deduced (weak) soundness.

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

For a set $\Lambda$ and an integer $\upsilon$, define

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

(In [27], this definition was only given for $\upsilon = 0$. Then, $\hat{\Lambda} = \{0\} \cup \Lambda \cup 2\hat{\ }\Lambda$.)

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

*Proof.* Assume $\Lambda$ is progression-free. Then $2\hat{\ }\Lambda \cap 2 \cdot \Lambda = \emptyset$. Since $\upsilon > 2\lambda_n - \lambda_1$, $(\{2\upsilon\} \cup (\upsilon + \Lambda)) \cap 2 \cdot \Lambda = \emptyset$. (In [27], $\upsilon = 0$, and $(\{0\} \cup \Lambda) \cap 2 \cdot \Lambda = \emptyset$ was guaranteed by assuming that every integer in $\Lambda$ is odd and non-zero.) Assume now that $2 \cdot \Lambda \cap \hat{\Lambda} = \emptyset$. Thus, $2 \cdot \Lambda \cap 2\hat{\ }\Lambda = \emptyset$, and $\Lambda$ is a progression-free set. □

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

*Proof.* Let $\Lambda$ be the progression-free set from [14], seen as a subset of $[\lambda_1, \lambda_n]$ (with $\lambda_1$ possibly being negative), with $\lambda_n - \lambda_1 \approx r_3^{-1}(n)$. Since $\upsilon > 2\lambda_n - \lambda_1$ is linear in $\lambda_n - \lambda_1$, $\hat{\Lambda} \subset [2\lambda_1, 2\upsilon]$ and $|\hat{\Lambda}| = \Theta(r_3^{-1}(n))$. □

```
CRS generation $\mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$:
    Set $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $(g_1, g_2) \leftarrow (\mathbb{G}_1 \setminus \{1\}, \mathbb{G}_2 \setminus \{1\})$;
    Set $\sigma, \hat{\alpha} \leftarrow \mathbb{Z}_p$, $\hat{g}_1 \leftarrow g_1^{\hat{\alpha}}$;
    For each $\ell \in \{\upsilon\} \cup \Lambda$ do: $(g_{1,\ell}, \hat{g}_{1,\ell}) \leftarrow (g_1, \hat{g}_1)^{\sigma^\ell}$;
    For each $\ell \in \{\upsilon\} \cup \hat{\Lambda}$ do: $(g_{2,\ell}, \hat{g}_{2,\ell}) \leftarrow (g_2, g_2^{\hat{\alpha}})^{\sigma^\ell}$;
    Set $D \leftarrow \prod_{i=1}^n g_{2,\lambda_i}$, $\widehat{\mathsf{ck}} \leftarrow (\mathsf{gk}; (g_{1,\ell}, \hat{g}_{1,\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$;
    Return $\mathsf{crs} \leftarrow (\widehat{\mathsf{ck}}, g_1, \hat{g}_1, (g_{2,\ell}, \hat{g}_{2,\ell})_{\ell \in \hat{\Lambda}}, D)$;
Argument generation $\mathcal{P}_\times(\mathsf{crs}; (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C}), (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c))$:
    Define $\mathfrak{I}_1(\ell) := \{(i,j) : i, j \in [n] \wedge i \neq j \wedge \lambda_i + \lambda_j = \ell\}$;
    For each $\ell \in 2\hat{~}\Lambda$ do: $\mu_\ell \leftarrow \sum_{(i,j) \in \mathfrak{I}_1(\ell)} (a_i b_j - c_i)$;
    $(\pi, \hat{\pi}) \leftarrow (g_{2,2\upsilon}, \hat{g}_{2,2\upsilon})^{r_a r_b} \cdot \prod_{i=1}^n (g_{2,\upsilon+\lambda_i}, \hat{g}_{2,\upsilon+\lambda_i})^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2\hat{~}\Lambda} (g_{2,\ell}, \hat{g}_{2,\ell})^{\mu_\ell}$;
    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, else reject.
```

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

One can add any constant to all members of $\Lambda$ and $\upsilon$, so that the previous results still hold. In particular, according to the previous two lemmas, the best value (in the sense of efficiency) of $\lambda_n$ might be 0.

We state and prove the security of the new product argument when using the $(\Lambda, \upsilon)$ knowledge commitment scheme by closely following the claim and the proof from [27]. The (knowledge) commitments are $(A, \hat{A})$, $(B, \hat{B})$ and $(C, \hat{C})$. For efficiency (and backwards compatibility) reasons, following [27], we include another element $B_2$ to the statement of the Hadamard product language.

Since for any $\boldsymbol{a}$ and $\boldsymbol{b}$, $(C, \hat{C})$ is a commitment of $(a_1 b_1, \ldots, a_n b_n)$ for *some* value of $r_c$, Prot. 2 cannot be computationally sound (even under a knowledge assumption). Instead, as in [21,27], we prove a weaker version of soundness that is sufficient to achieve soundness of the more complex arguments. The last statement of Thm. 3 basically says that no efficient adversary can output an input to the 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, s.t. $a_i b_i \neq c_i$ for some $i \in [n]$. See App:prodsec for the proof.

**Theorem 3.** *Let $n = \mathrm{poly}(\kappa)$. Let $\Lambda = (\lambda_1, \ldots, \lambda_n)$ be a progression-free set with $\lambda_{i+1} > \lambda_i$, $\lambda_i = \mathrm{poly}(\kappa)$, $\upsilon > 2\lambda_n - \lambda_1$, and $\upsilon = \mathrm{poly}(\kappa)$. Let $\Gamma$ be the $(\Lambda, \upsilon)$ commitment scheme in $\mathbb{G}_1$. Let $\Phi_\times := \{X^\upsilon\} \cup \{X^\ell\}_{\ell \in \hat{\Lambda}}$.*

1. *Prot. 2 is perfectly complete and perfectly witness-indistinguishable.*
2. *If $\mathcal{G}_{\mathsf{bp}}$ is $\Phi_\times$-PSDL secure, then an NUPPT adversary against Prot. 2 has negligible chance, given $\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$ 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*
   *(a) $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{Z}_p^n$, $r_a, r_b, r_c \in \mathbb{Z}_p$, and $f_\ell^* \in \mathbb{Z}_p$ for $\ell \in \hat{\Lambda}$,*
   *(b) $(A, \hat{A}) = \mathcal{C}om(\widehat{\mathsf{ck}}; \boldsymbol{a}; r_a)$, $(B, \hat{B}) = \mathcal{C}om(\widehat{\mathsf{ck}}; \boldsymbol{b}; r_b)$, $B_2 = g_{2,\upsilon}^{r_b} \cdot \prod_{i=1}^n g_{2,\lambda_i}^{b_i}$, and $(C, \hat{C}) = \mathcal{C}om(\widehat{\mathsf{ck}}; \boldsymbol{c}; r_c)$,*

*(c)* $\log_{g_2} \pi = \log_{\hat{g}_2} \hat{\pi} = \sum_{\ell \in \hat{\Lambda}} f_\ell^* \sigma^\ell$*, where* $\hat{g}_2 = g_2^{\hat{\alpha}}$*, and*
*(d) for some* $i \in [n]$*,* $a_i b_i \neq c_i$*.*
*The reduction overhead is dominated by the time to factor a degree-*$(2v - 2\lambda_1)$ *polynomial in* $\mathbb{Z}_p[X]$*.*

Next, we will show that the product argument of this section (and also the product argument of [27]) is computationally much more efficient than it was claimed in [27]. In [27], the prover was said to require computing $\Theta(n^2)$ multiplications in $\mathbb{Z}_p$ and $\Theta(r_3^{-1}(n))$ exponentiations in $\mathbb{G}_2$. We optimize the prover's computation so that it will require a significantly smaller number of multiplications and no exponentiations at all.

**Theorem 4.** *The communication (argument size) of Prot. 2 is* $2$ *elements from* $\mathbb{G}_2$*. The prover's computation is dominated by* $\Theta(r_3^{-1}(n) \cdot \log r_3^{-1}(n))$ *multiplications in* $\mathbb{Z}_p$ *and two* $\Theta(r_3^{-1}(n))$*-wide multi-exponentiations in* $\mathbb{G}_2$*. The verifier's computation is dominated by* $5$ *bilinear pairings and* $1$ *bilinear-group multiplication. The CRS consists of* $\Theta(r_3^{-1}(n))$ *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 only needs to access $g_1$, $\hat{g}_1$, and $D$. Since $2 \frown \Lambda \subseteq \hat{\Lambda}$, the statement about the prover's computation follows from Fast Fourier Transform [12] based polynomial multiplication [17] techniques. To compute all the coefficients of the polynomial $\mu(X) := \sum_{i=1}^n \sum_{j=1:j\neq i}^n (a_i b_j - c_i) X^{\lambda_i + \lambda_j}$, the prover executes Alg. 2. Here, FFTMult denotes an FFT-based polynomial multiplication algorithm.

```
For $i \leftarrow 0$ to $\lambda_n$ 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 $a^\dagger(X) := \sum_{i=0}^{\lambda_n} a_i^\dagger X^i$ and $b^\dagger(X) := \sum_{i=0}^{\lambda_n} b_i^\dagger X^i$;
Denote $c^\dagger(X) := \sum_{i=0}^{\lambda_n} c_i^\dagger X^i$ 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 $\mu(X) \leftarrow \mu(X) - \nu(X)$;
```

**Algorithm 2.** FFT-based prover's computation of $\{\mu_\ell\}$

After using FFTMult to compute the initial version of $\mu(X)$ and $\nu(X)$, $\mu_\ell = \sum_{(i,j)\in[n]^2:\lambda_i+\lambda_j=\ell} a_i b_j$ and $\nu_\ell = \sum_{(i,j)\in[n]^2:\lambda_i+\lambda_j=\ell} c_i$. Thus, after the penultimate step of Alg. 2, $\mu_\ell = \sum_{(i,j)\in\mathfrak{I}_1(\ell)} a_i b_j$, and after the last step, $\mu_\ell = \sum_{(i,j)\in\mathfrak{I}_1(\ell)} a_i b_j - c_i$, as required by Prot. 2. Since FFT takes time $\Theta(N \log N)$, where $N = r_3^{-1}(n)$ is the input size, we have shown the part about the prover's computational complexity. The verifier's computational complexity follows from the description of the argument. □

FFT does not help to speed up Groth's product argument [21], since there $\lambda_n = \Theta(n^2)$. FFT does also not seem to be useful in the case of the permutation argument from [27]. Finally, it may be possible to speed up Alg. 2, by taking into account the fact that all $a^\dagger$, $b^\dagger$, $c^\dagger$ and $d^\dagger$ have only $n$ non-zero monomials.

Next, we use efficient multi-exponentiation for additional speed-up. Let $\alpha := \max(||a||_\infty, ||b||_\infty, ||c||_\infty)$, where the prover has committed to $\boldsymbol{a}$ and $\boldsymbol{b}$. (See Sect. 6 for the concrete values of $\alpha$.) The number of bilinear-group operations the prover has to perform (on top of computing the exponents by using the FFT-based polynomial multiplication) to compute $\pi$ is dominated by $L(2, n, p) + L(2, r_3^{-1}(n), \Theta((\alpha n)^2))$. 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 1, for $n = \Omega(\log p)$, $L(2, n, p) = 2\log_2 p + (2 + o(1)) \cdot n \log_2(p + 1)/(\log_2(2n \log_2(p + 1))) + O(n) = (2 + o(1)) \cdot \log_2 p \cdot n/\log_2 n$, and, since in our applications, $n \gg \log_2 \Theta((\alpha n)^2)$, $L(2, r_3^{-1}(n), \Theta((\alpha n)^2)) = 2\log_2(\alpha n^2) + \frac{(2+o(1))r_3^{-1}(n)\log_2 \Theta((\alpha n)^2)}{(\log_2(2r_3^{-1}(n)\log_2 \Theta((\alpha n)^2)))} + O(r_3^{-1}(n)) = \frac{(2+o(1))r_3^{-1}(n)}{(\log_2 r_3^{-1}(n))} \cdot 2\log_2(\alpha n)$. Thus, the prover has to compute $(2 + o(1)) \cdot (\frac{n}{\log_2 n} \cdot \log_2 p + \frac{r_3^{-1}(n)}{\log_2 r_3^{-1}(n)} \cdot 2\log_2(\alpha n))$ bilinear-group multiplications. We will instantiate $\alpha$ and other values in Sect. 6.

## 5 Shift and Rotation Arguments

In a *right shift-by-$\xi$ argument* (resp., right rotation-by-$\xi$ argument), the prover aims to convince the verifier that for two commitments $A$ and $B$, he knows how to open them as $A = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r_a)$ and $B = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that $a_i = b_{i+\xi}$ for $i \in [n - \xi]$ and $a_{n-\xi+1} = \cdots = a_n = 0$ (resp., $a_{n-\xi+1} = b_1$, $\ldots a_n = b_\xi$). That is, $(a_n, \ldots, a_1) = (0, \ldots, 0, b_n, \ldots, b_{\xi+1})$ (resp., $(a_n, \ldots, a_1) = (b_\xi, \ldots, b_1, b_n, \ldots, b_{\xi+1})$). Left shift and left rotation arguments are defined dually, we omit their descriptions.

Groth [21] and Lipmaa [27] defined NIZK arguments for arbitrary permutation $\varrho$ (i.e., $a_{\varrho(i)} = b_i$ for public $\varrho$). However, their permutation arguments are quite complex and computationally intensive. Moreover, many applications do not require arbitrary permutations. We give examples of the latter in Sect. 6.

We now describe the new right shift-by-$\xi$ argument $\mathsf{rsft}_\xi(\llbracket(A, \tilde{A})\rrbracket) = \llbracket(B, \tilde{B})\rrbracket$, 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, see App. D. Let $\log_{g_1} A = r_a\sigma^v + \sum_{i=1}^n a_i\sigma^{\lambda_i}$ and $\log_{g_1} B = r_b\sigma^v + \sum_{i=1}^n b_i\sigma^{\lambda_i}$. We replace $\sigma$ with a formal variable $X$. If the prover is honest (full derivation of this is given in the proof of Thm. 5), then $F(X) := X^\xi \cdot \log_{g_1} A - \log_{g_1} B = -\sum_{i=1}^{\xi} b_i X^{\lambda_i} + \sum_{i=\xi+1}^n b_i(X^{\lambda_{i-\xi}+\xi} - X^{\lambda_i}) + r_a X^{v+\xi} - r_b X^v$. Thus, one can verify that $A$ is a right shift-by-$\xi$ of $B$ by checking that $\hat{e}(A, g_2^{\sigma^\xi})/\hat{e}(B, g_2) = \hat{e}(g_1, \pi)$, where $\pi = g_2^{F(\sigma)}$ is defined as in Prot. 3. As seen from the proof of the following theorem, the actual security proof, especially for the (weaker version of) soundness, is more complicated. Complications arise from the use of polynomials of type $X^i - X^j$ in the verification equation; because of this we must rely on a less straightforward variant of the PSDL assumption than before. One has also to be careful in the choice of the set $\Lambda$: if say $\lambda_{n-\xi} + \xi = \lambda_n$, then some of the monomials of $F(X)$ will collapse, and the security proof will not go through.

```
CRS generation $\mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$:
    Set $\mathsf{gk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}$, $g_2 \leftarrow \mathbb{G}_2 \setminus \{1\}$, $\sigma, \tilde{\alpha} \leftarrow \mathbb{Z}_p$;
    For each $z \in \{1, 2\}$ do: $\tilde{g}_z \leftarrow g_z^{\tilde{\alpha}}$;
    For each $\ell \in \{\upsilon\} \cup \Lambda$ do: $(g_{1,\ell}, \tilde{g}_{1,\ell}) \leftarrow (g_1, \tilde{g}_1)^{\sigma^\ell}$;
    Set $g_{2,\xi} \leftarrow g_2^{\sigma^\xi}$;
    For each $i \in \{\lambda_1, , \ldots, \lambda_\xi, \upsilon, \upsilon + \xi\}$ do: $(g_{2,i}, \tilde{g}_{2,i}) \leftarrow (g_2, \tilde{g}_2)^{\sigma^i}$;
    For each $i \in [1, n - \xi]$ do: $(h_{2,i}, \tilde{h}_{2,i}) \leftarrow (g_2, \tilde{g}_2)^{\sigma^{\lambda_i + \xi} - \sigma^{\lambda_{i+\xi}}}$;
    Set $\widetilde{\mathsf{ck}} \leftarrow (\mathsf{gk}; (g_{1,\ell}, \tilde{g}_{1,\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$;
    Return $\mathsf{crs} \leftarrow (\widetilde{\mathsf{ck}}, g_1, \tilde{g}_1, g_2, g_{2,\xi}, (g_{2,i}, \tilde{g}_{2,i})_{i \in \{\lambda_1, \ldots, \lambda_\xi, \upsilon, \upsilon+\xi\}}, (h_{2,i}, \tilde{h}_{2,i})_{i \in [1, n-\xi]})$;
Argument generation $\mathcal{P}_{\mathsf{rsft}}(\mathsf{crs}; (A, \tilde{A}, B, \hat{B}, \tilde{B}), (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b))$:
    $(\pi, \tilde{\pi}) \leftarrow \prod_{i=1}^{n-\xi} (h_{2,i}, \tilde{h}_{2,i})^{b_{i+\xi}} \prod_{i=1}^{\xi} (g_{2,\lambda_i}, \tilde{g}_{2,\lambda_i})^{-b_i} (g_{2,\upsilon+\xi}, \tilde{g}_{2,\upsilon+\xi})^{r_a} \cdot (g_{2,\upsilon}, \tilde{g}_{2,\upsilon})^{-r_b}$;
    Return $\pi^{\mathsf{rsft}} \leftarrow (\pi, \tilde{\pi}) \in \mathbb{G}_2^2$;
Verification $\mathcal{V}_{\mathsf{rsft}}(\mathsf{crs}; (A, \tilde{A}, B, \hat{B}, \tilde{B}), \pi^{\mathsf{rsft}})$:
    If $\hat{e}(A, g_{2,\xi}) / \hat{e}(B, g_2) = \hat{e}(g_1, \pi)$ and $\hat{e}(g_1, \tilde{\pi}) = \hat{e}(\tilde{g}_1, \pi)$ then accept, else reject;
```

**Protocol 3:** New right shift-by-$\xi$ argument $\mathsf{rsft}_\xi(\llbracket(A, \tilde{A})\rrbracket) = \llbracket(B, \tilde{B})\rrbracket$

**Theorem 5.** *Let $n = \mathrm{poly}(\kappa)$. Let $\Lambda = (\lambda_1, \ldots, \lambda_n) \subset \mathbb{Z}$, s.t. $\lambda_{i+1} > \lambda_i$, $\lambda_i \neq \lambda_j + \xi$ for $i \neq j$, and $\lambda_i = \mathrm{poly}(\kappa)$. Let $\upsilon > \lambda_n + \xi$ be an integer, s.t. $\upsilon = \mathrm{poly}(\kappa)$. Let $\Gamma$ be the $(\Lambda, \upsilon)$ commitment scheme in $\mathbb{G}_1$.*

*(1) Prot. 3 is perfectly complete and perfectly witness-indistinguishable.*

*(2) Let*

$$\Phi_{\mathsf{rsft}}^{\xi} := \{X^\upsilon, X^{\upsilon+\xi}\} \cup \{X^{\lambda_i}\}_{i=1}^{\xi} \cup \{X^{\lambda_i+\xi} - X^{\lambda_{i+\xi}}\}_{i=1}^{n-\xi} \enspace .$$

*If $\mathcal{G}_{\mathsf{bp}}$ is $\Phi_{\mathsf{rsft}}^{\xi}$-PSDL secure, then an NUPPT adversary against Prot. 3 has negligible chance, given $\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$ as an input, of outputting $inp^{\mathsf{rsft}} \leftarrow (A, \tilde{A}, B, \tilde{B})$ and an accepting argument $\pi^{\mathsf{rsft}} \leftarrow (\pi, \tilde{\pi})$ together with a witness $w^{\mathsf{rsft}} \leftarrow (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, (f_\phi^*)_{\phi \in \Phi_{\mathsf{rsft}}^{\xi}})$, such that*

*(a) $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, $r_a, r_b \in \mathbb{Z}_p$, and $f_\phi^* \in \mathbb{Z}_p$ for $\phi \in \Phi_{\mathsf{rsft}}^{\xi}$,*

*(b) $(A, \tilde{A}) = \mathcal{C}om(\widetilde{\mathsf{ck}}; \boldsymbol{a}; r_a)$, $(B, \tilde{B}) = \mathcal{C}om(\widetilde{\mathsf{ck}}; \boldsymbol{b}; r_b)$,*

*(c) $\log_{g_2} \pi = \log_{\tilde{g}_2} \tilde{\pi} = \sum_{\phi \in \Phi_{\mathsf{rsft}}^{\xi}} f_\phi^* \cdot \phi(\sigma)$, and*

*(d) $(a_n, a_{n-1}, \ldots, a_1) \neq (0, \ldots, 0, b_n, \ldots, b_{\xi+1})$.*

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

(See App. C for a proof.) In an upper level argument, the verifier must check that $\hat{e}(A, \tilde{g}_2) = \hat{e}(\tilde{A}, g_2)$, and $\hat{e}(B, \tilde{g}_2) = \hat{e}(\tilde{B}, g_2)$. A simple valid choice of $\Lambda$ is the initial segment of $\mathbb{Z}_\xi \cup (\mathbb{Z}_\xi + 2\xi) \cup (\mathbb{Z}_\xi + 4\xi) \cup \cdots$.

**Theorem 6.** *Let $\Lambda$ and $\upsilon$ be as defined in Thm. 5. Let $\beta \leftarrow ||\boldsymbol{b}||_\infty$, $\beta < p$. Assume $n > \log_2 \beta$. The argument size of Prot. 3 is 2 elements from $\mathbb{G}_2$. The prover's computation is dominated by $\Theta(n)$ $\mathbb{Z}_p$-multiplications and $(2 + o(1)) \cdot \log_2 \beta \cdot n / \log_2 n + O(n)$ bilinear-group multiplications. The verifier's computation is dominated by 5 bilinear pairings. The CRS consists of $\Theta(n)$ group elements.*

*Proof.* 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. □

## 6 Applications

We will now describe how to use the new product and shift arguments to improve on the range argument of [11], and to construct new SET-PARTITION and SUBSET-SUM arguments. Then, we combine the 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* [3] of vector $a$, if $b_i = \sum_{j>i} a_j$. As abundantly demonstrated in [3], vector scan (also known as all-prefix-sums) is a powerful operator that can be used to solve many important computational problems. In the context of zero knowledge, we will only need to be able to *verify* that one vector is a scan of the second vector.

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 $A = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r_a)$ and $B = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{b}; r_b)$, s.t. $b_i = \sum_{j>i} a_j$. A scan argument is just equal to a right shift-by-1 argument $\mathsf{rsft}_1(\llbracket B \rrbracket) = \llbracket A + B \rrbracket$, that proves that $b_i = a_{i+1} + b_{i+1}$, 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 Improved Range Argument

Since the used commitment scheme is homomorphic, the generic range argument (prove that the committed value $x$ belongs to the interval $[L, H]$ for $L < H$) is equivalent to proving that the committed value $y = x - L$ belongs to the interval $[0, H - L]$. In what follows, we will therefore concentrate on this simpler case.

In [11], the authors proposed a range argument that is based on the product and permutation arguments from [27]. Interestingly, [11] 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 right shift-by-1 by clearing out one of the elements). Hence, we can replace the permutation and product arguments from [27] with the right shift-by-1 (or scan) and product arguments from the current paper. Thus, it suffices for $\Lambda$ to be an arbitrary progression-free set. The resulting range argument is also shorter by one product argument. The security proof does not change significantly. To show that the range argument is computationally sound, one has to assume that the product argument and the right shift-by-1 argument are weakly sound (and that the PKE assumption holds).

The use of the new basic arguments will decrease the number of $\mathbb{Z}_p$-multiplications — except when 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 algorithm [31], the cost of the multi-exponentiation decreases to $(2 + o(1)) \cdot 2r_3^{-1}(n) \log_2(un)/\log_2 r_3^{-1}(n)$ bilinear-group multiplications. The communication decreases by $4 + 2 + 3 = 9$ group elements, due to the replacement of the permutation argument with the right shift-by-1 argument (minus 4), having one less product argument (minus 2), and also because one needs to commit to one less element ($(C_{\mathsf{rrot}}, \hat{C}_{\mathsf{rrot}}, \tilde{C}_{\mathsf{rrot}})$ in [11], minus 3). The verifier also has to perform $7+5+4 = 16$ less pairings, due to the replacement of the permutation argument with the right shift-by-1 argument (minus 7) and one less product argument (minus 5). Also, it is not necessary to verify the correctness of $(C_{\mathsf{rrot}}, \hat{C}_{\mathsf{rrot}}, \tilde{C}_{\mathsf{rrot}})$ (minus 4). One can analogously compute the verifier's computation, see Table 2.

*Remark 1.* In the permutation argument of [27], the verifier also has to compute a certain triple $(T^*, \hat{T}^*, T_2^*)$ by using 3 multi-exponentiations. This is not included in the comparison table (or the claims) in [27], and the same mistake was replicated in [11]. Table 1 and Table 2 correct this mistake, by giving the correct complexity estimation of the arguments from [27,11]. The range argument from [11] 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. After this modification, the verifier's computational complexity actually does not increase compared to what was claimed in [11]. Since [11] itself did not mention this, we consider it to be an additional small contribution.

Since the non-balanced range argument only uses one permutation argument, the corrected permutation argument of this paper makes the argument shorter 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 total length of the CRS and the argument. The setting $u = H$ minimizes the prover's computational complexity. See Table 2. 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 $N^* = r_3^{-1}(\sqrt{h}) = o(\sqrt{h} \cdot 2^{2\sqrt{\log_2 h}})$. The rest of the notation is as in Table 1.

**Theorem 7.** *Let $\Gamma$ be the $(\Lambda, \upsilon)$ commitment scheme in group $\mathbb{G}_1$. Let $\Lambda = (\lambda_1, \dots, \lambda_n) \in \mathbb{Z}^n$ be progression-free, s.t. $\lambda_{i+1} > \lambda_i + 1$ and $\lambda_i = \mathrm{poly}(\kappa)$. Let*

$$\Phi := \Phi_\times \cup \Phi^1_{\mathsf{rsft}} = \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n \cup \{X^\ell\}_{\ell \in \hat{\Lambda}} \quad . \qquad (2)$$

*Let $\upsilon > \max(2\lambda_n - \lambda_1, \lambda_n + 1)$ be linear in $\lambda_n - \lambda_1$. The modified range argument is complete and computationally zero knowledge. Also, if $\mathcal{G}_{\mathsf{bp}}$ is $\Phi$-PSDL secure and the $\Phi$-PKE assumption holds in $\mathbb{G}_1$ and the $\Phi$-PKE assumption holds in $\mathbb{G}_2$, then the range argument is computationally sound.*

The proof is similar to [11]. Note that $\lambda_{i+1} > \lambda_i + 1$ guarantees that both $\lambda_{i+1} > \lambda_i$ and $\lambda_j \neq \lambda_i + 1$ for $i \neq j$.

**Table 2.** Comparison of NIZK range arguments

| | $\lvert\mathrm{CRS}\rvert$ | $\lvert\mathrm{Argument}\rvert$ | Prover comp. | Verifier comp. |
|---|---|---|---|---|
| [32] | $\Theta(1)\mathfrak{g}$ | $\Theta(h)\mathfrak{g}$ | $\Theta(h)$ | $\Theta(h)$ |
| [32] | $\Theta(h/\log h)\mathfrak{g}$ | $\Theta(h/\log h)\mathfrak{g}$ | $\Theta(h/\log h)$ | $\Theta(h/\log h)$ |
| Chaabouni, Lipmaa, and Zhang [11] | | | | |
| General | $\Theta(r_3^{-1}(n))\mathfrak{g}$ | $(5n_v+40)\mathfrak{g}$ | $\Theta(n^2 n_v)\mathfrak{m}+\Theta(r_3^{-1}(n)n_v)\mathfrak{e}$ | $\Theta(n)\mathfrak{e}+(9n_v+81)\mathfrak{p}$ |
| $u=2$ | $\Theta(N)\mathfrak{g}$ | $40\mathfrak{g}$ | $\Theta(h^2)\mathfrak{m}+\Theta(N)\mathfrak{e}$ | $\Theta(h)\mathfrak{e}+81\mathfrak{p}$ |
| $u=2^{\sqrt{h}}$ | $\Theta(N^*)\mathfrak{g}$ | $\approx(5\sqrt{h}+40)\mathfrak{g}$ | $\Theta(h^{3/2})\mathfrak{m}+\Theta(\sqrt{h}\cdot N^*)\mathfrak{e}$ | $\Theta(\sqrt{h})\mathfrak{e}+(9\sqrt{h}+81)\mathfrak{p}$ |
| $u=H$ | $\Theta(1)\mathfrak{g}$ | $\approx(5h+40)\mathfrak{g}$ | $\Theta(h)\mathfrak{m}+\Theta(h)\mathfrak{e}$ | $\Theta(1)\mathfrak{e}+(9h+81)\mathfrak{p}$ |
| The current paper | | | | |
| General | $\Theta(r_3^{-1}(n))\mathfrak{g}$ | $(5n_v+31)\mathfrak{g}$ | $\Theta(r_3^{-1}(n)\log r_3^{-1}(n)\cdot n_v)\mathfrak{m}+\Theta(r_3^{-1}(n)n_v)\mathfrak{m}_b$ | $(9n_v+65)\mathfrak{p}$ |
| $u=2$ | $\Theta(N)\mathfrak{g}$ | $31\mathfrak{g}$ | $\Theta(N\cdot\log N)\mathfrak{m}+\Theta(N)\mathfrak{m}_b$ | $65\mathfrak{p}$ |
| $u=2^{\sqrt{h}}$ | $\Theta(N^*)\mathfrak{g}$ | $\approx(5\sqrt{h}+31)\mathfrak{g}$ | $\Theta(\sqrt{h}\cdot N^*\cdot\log N^*)\mathfrak{m}+\Theta(\sqrt{h}\cdot N^*)\mathfrak{m}_b$ | $\approx(9\sqrt{h}+65)\mathfrak{p}$ |
| $u=H$ | $\Theta(1)\mathfrak{g}$ | $(\approx 5h+31)\mathfrak{g}$ | $\Theta(h)\mathfrak{m}+\Theta(h)\mathfrak{m}_b$ | $\approx(9h+65)\mathfrak{p}$ |

## 6.2 Arguments for NP-Complete Languages

Finally, we construct efficient modular arguments, that only use product and shift arguments, for some **NP**-complete languages. CIRCUIT-SAT seems to require the use of permutation arguments [21,27], so we will find other problems.

**Set-Partition.** Let $n \ll p$. Given a multiset $\mathcal{S}=(s_1,\dots,s_n)$, with $s_i\in\mathbb{Z}_p$, and a commitment $B$, in the SET-PARTITION *argument*, the prover has to convince the verifier that he knows how to open the commitment as $B=\mathcal{C}om(\mathsf{ck};\boldsymbol{b};r_b)$, such that $b_i\in\{-1,1\}$, and $\sum_{i=1}^n b_i s_i=0$. If we define $\mathcal{V}=\{i: b_i=1\}$, then $\sum_{i=1}^n b_i s_i=0$ is equivalent to $\sum_{i\in\mathcal{V}} s_i=\sum_{i\in\mathcal{S}\setminus\mathcal{V}} s_i$. The prover computes the SET-PARTITION argument as follows.

```
Compute a product argument $\pi_1$ for $b_i \cdot b_i = 1$, showing that $b_i \in \{-1,1\}$;
Compute a product argument $\pi_2$ for $c_i = b_i \cdot s_i$;
Compute a scan argument $\pi_3$ showing that $\boldsymbol{d}$ is the scan of $\boldsymbol{c}$;
Compute a restriction argument $\pi_4$ showing the first coordinate of $\boldsymbol{c}+\boldsymbol{d}$ is 0;
The SET-PARTITION argument is equal to $(B, C, D, \pi_1, \dots, \pi_4)$;
```

Here, $C$ commits to $\boldsymbol{c}=(b_1s_1,\dots,b_ns_n)$, $S$ commits to $\boldsymbol{s}$, and $D$ commits to $\boldsymbol{d}$, the scan of $\boldsymbol{c}$. That is, $d_i=\sum_{j>i}c_j$, and in particular, $d_1=\sum_{j>1}c_i$ and $c_1+d_1=\sum_{j\geq 1}c_j$. We omit the security proof of this argument since it is similar to the proof of the SUBSET-SUM argument.

**Subset-Sum.** Another 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. In a SUBSET-SUM *argument*, the prover aims to convince the verifier that given $\mathcal{S}=(s_1,\dots,s_n)\subseteq\mathbb{Z}_p$, $n\ll p$, and a commitment $B$, he knows how to open it as $B=\mathcal{C}om(\mathsf{ck};\boldsymbol{b};r_b)$, s.t. $\boldsymbol{b}$ is non-zero and Boolean, and $\sum_{i=1}^n b_is_i=0$. That is, $b_i=1$ iff $s_i$ belongs to the subset of $S$ that sums to 0. (As always, the committed elements belong to $\mathbb{Z}_p$. Thus, $\sum_{i=1}^n b_is_i=0$ holds modulo $p$.)

In the new SUBSET-SUM argument, both parties compute a commitment $S$ to $\boldsymbol{s}$. The prover commits to a Boolean vector $\boldsymbol{b}$ and to a vector $\boldsymbol{c}$, s.t. $c_i=b_is_i$.

He computes a commitment $D$ to the *scan* $\boldsymbol{d}$ of vector $\boldsymbol{c}$. I.e., $d_i = \sum_{j>i} c_j$, and in particular, $d_1 = \sum_{j>1} c_i$ and $c_1 + d_1 = \sum_{j\geq 1} c_j$. The resulting Subset-Sum argument can be seen as a slight modification of the Set-Partition argument. The main conceptual difference is that we also need to prove $\boldsymbol{b} \neq \mathbf{0}$ (not necessary in the Set-Partition argument).

```
Compute a product argument $\pi_1$ for $b_i^2 = b_i$, showing that $\boldsymbol{b}$ is Boolean;
Compute an argument $\pi_2$ showing that $\boldsymbol{b} \neq \mathbf{0}$;
Compute a product argument $\pi_3$ showing that $c_i = b_i \cdot s_i$ for $i \in [n]$;
Compute a scan argument $\pi_4$ showing that $\boldsymbol{d}$ is the scan of $\boldsymbol{c}$;
Compute a restriction argument $\pi_5$ showing the first coordinate of $\boldsymbol{c} + \boldsymbol{d}$ is 0;
The Subset-Sum argument is equal to $(B, C, D, \pi_1, \dots, \pi_5)$;
```

Here, $\pi_5$ is computed by using the restriction argument from [21], which adds linear number of elements to the CRS, but has a constant complexity otherwise. The subargument $\pi_2$ is computed as in Alg. 3.

Note that the verifier can check that $\mathring{B}$ is correct by checking that $\hat{e}(\mathring{B}, g_2) = \hat{e}(B, \mathring{g}_2)$. It is straightforward to prove that the new Subset-Sum argument is complete and perfectly zero-knowledge. It is also computationally sound under appropriate assumptions. See App. E for a proof.

The resulting Subset-Sum argument is simpler than the Circuit-SAT arguments of [21,27] that consist of $\geq 7$ product and permutation arguments. Moreover, instead of the product and permutation arguments it only uses product and a more efficient right shift-by-1 argument (zero argument is trivial).

```
Assume $B = g_{1,\upsilon}^{r_b} \prod g_{1,\lambda_i}^{b_i}$ ;            /* we want to show that $\boldsymbol{b} \neq \mathbf{0}$ */
Assume that $\mathring{g}_{1,i} = g_{1i}^{\mathring{\alpha}}$ and $\mathring{g}_2 = g_2^{\mathring{\alpha}}$ for a secret $\mathring{\alpha}$;
Create $\mathring{B} \leftarrow \mathring{g}_{1,\upsilon}^{r_b} \cdot \prod_{i=1}^{n} \mathring{g}_{1,\lambda_i}^{b_i}$ and a hybrid $B^* \leftarrow g_{1,\upsilon}^{r_b} \cdot \prod \mathring{g}_{1,\lambda_i}^{b_i}$;
Show $\mathring{B}/B^* = (\mathring{g}_{1,\upsilon}/g_{1,\upsilon})^{r_b}$ commits to $\mathbf{0}$ by using the zero argument [30];
Verifier checks that $\hat{e}(B, \mathring{g}_2) \neq \hat{e}(B^*, g_2)$;
```

**Algorithm 3.** Argument $\pi_2$

**Decision-Knapsack.** In the **NP**-complete 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 $\mathcal{T} \subseteq \mathcal{S}$, such that $\sum_{i\in\mathcal{T}} w_i \leq W$ and $\sum_{i\in\mathcal{T}} b_i \geq B$. One can combine a version of the Subset-Sum argument of the current section with the range argument of Sect. 6.1 to construct a Decision-Knapsack argument, where the prover convinces the verifier that he knows such a subset $\mathcal{T}$. See Alg. 6 in App. F.

**Acknowledgments.** The first two authors were supported from research theme IUT2-1 and European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS. The third author is supported by Project FINER, Greek Secretariat of Research and Technology. The work was partially done while the third author was working in University of Tartu.

## References

1. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (2013)
2. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct Non-interactive Arguments via Linear Interactive Proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (2013)
3. Blelloch, G.: Vector Models for Data-Parallel Computing. MIT Press (1990)
4. Blum, M., Feldman, P., Micali, S.: Non-Interactive Zero-Knowledge and Its Applications. In: STOC 1988, May 2-4, pp. 103–112. ACM Press, Chicago (1988)
5. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boudot, F.: Efficient Proofs That a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
7. Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. Journal of Computer and System Sciences 37(2), 156–189 (1988)
8. Camenisch, J.L., Chaabouni, R., Shelat, A.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
9. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. In: Vitter, J.S. (ed.) STOC 1998, Dallas, Texas, USA, May 23-26, pp. 209–218 (1998)
10. Chaabouni, R., Lipmaa, H., Shelat, A.: Additive Combinatorics and Discrete Logarithm Based Range Protocols. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 336–351. Springer, Heidelberg (2010)
11. Chaabouni, R., Lipmaa, H., Zhang, B.: A Non-interactive Range Proof with Constant Communication. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 179–199. Springer, Heidelberg (2012)
12. Cooley, J.W., Tukey, J.W.: An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation 19, 297–301 (1965)
13. Dybizbański, J.: Sequences Containing No 3-Term Arithmetic Progressions. Electron. J. of Combin. 19(2), P15 (2012)
14. Elkin, M.: An Improved Construction of Progression-Free Sets. Israel J. of Math. 184, 93–128 (2011)
15. Erdős, P., Turán, P.: On Some Sequences of Integers. J. London Math. Soc. 11(4), 261–263 (1936)
16. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013)
17. Gentleman, W.M., Sande, G.: Fast Fourier Transforms — For Fun and Profit. In: Fall Joint Computer Conf. AFIPS Proc., vol. 29, pp. 563–578. ACM, Washington, DC (1966)

18. Gentry, C., Wichs, D.: Separating Succinct Non-Interactive Arguments from All Falsifiable Assumptions. In: Vadhan, S. (ed.) STOC 2011, June 6-8, pp. 99–108. ACM Press, San Jose (2011)
19. Goldwasser, S., Kalai, Y.T.: On the (In)security of the Fiat-Shamir Paradigm. In: FOCS 2003, October 11–14, pp. 102–113. IEEE (2003)
20. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: Sedgewick, R. (ed.) STOC 1985, May 6-8, pp. 291–304. ACM Press, Providence (1985)
21. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
22. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
23. van Hoeij, M., Novocin, A.: Gradual Sub-lattice Reduction and a New Complexity for Factoring Polynomials. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 539–553. Springer, Heidelberg (2010)
24. Joux, A.: A One-Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–393. Springer, Heidelberg (2000)
25. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring Polynomials with Rational Coefficients. Mathematische Annalen 261, 513–534 (1982)
26. Lipmaa, H.: On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
27. Lipmaa, H.: Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (2012)
28. Lipmaa, H.: Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013 Part I. LNCS, vol. 8269, pp. 41–60. Springer, Heidelberg (2013)
29. Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey Auctions without Threshold Trust. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 87–101. Springer, Heidelberg (2003)
30. Lipmaa, H., Zhang, B.: A More Efficient Computationally Sound Non-Interactive Zero-Knowledge Shuffle Argument. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 477–502. Springer, Heidelberg (2012)
31. Pippenger, N.: On the Evaluation of Powers and Monomials. SIAM J. Comput. 9(2), 230–250 (1980)
32. Rial, A., Kohlweiss, M., Preneel, B.: Universally Composable Adaptive Priced Oblivious Transfer. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 231–247. Springer, Heidelberg (2009)
33. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems Based on Pairing. In: SCIS 2000, Okinawa, Japan (2000)
34. Sanders, T.: On Roth's Theorem on Progressions. Ann. of Math. 174(1), 619–636 (2011)
35. Tao, T., Vu, V.: Additive Combinatorics. Cambridge Studies in Advanced Mathematics. Cambridge University Press (2006)

## A Proof of Thm. 1

*Proof.* In the generic group model, an adversary $\mathcal{A}$ 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 $\mathsf{gk}$ and $((g_1, g_2)^{\phi(\sigma)})_{\phi \in \{1\} \cup \Phi}$. The only time $\mathcal{A}$ 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 $\mathcal{A}$ can compute an arbitrary group element in unit time.

Assume that $\mathcal{A}$ 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 $a_\ell$ and $a^*_\ell$. But then also

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

Since $\mathcal{A}$ does not know the actual representations of the group elements, it will perform the same group operations independently of $\sigma$. Thus $a_\ell$ and $a^*_\ell$ are independent of $\sigma$. By the Schwartz-Zippel lemma 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 $a_\ell$ and $a^*_\ell$. If $\mathcal{A}$ works in polynomial time $\tau = \text{poly}(\kappa)$, it can generate at most $\tau$ 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 $\mathcal{A}$ 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 $\mathcal{A}$ to compute up to $\tau$ different values

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

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

## B Proof of Thm. 3 (Product Argument Security)

*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.

Perfect completeness. Assume that the prover is honest. The second verification is straightforward. For the first one, note that (after replacing $\sigma$ with a formal variable $X$)

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

Thus, $F(X) = F_{con}(X) + F_{\pi}(X)$, where

$$F_{con}(X) = \sum_{i=1}^{n}(a_i b_i - c_i) X^{2\lambda_i}$$

and

$$F_{\pi}(X) = r_a r_b X^{2\upsilon} + \sum_{i=1}^{n}(r_a b_i + r_b a_i - r_c) X^{\upsilon+\lambda_i} + \sum_{i=1}^{n}\sum_{j=1:j\neq i}^{n}(a_i b_j - c_i) X^{\lambda_i+\lambda_j} \enspace .$$

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

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

$$\begin{aligned}\pi \leftarrow& g_{2,\upsilon}^{r_a r_b} \cdot \prod_{i=1}^{n} g_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{i=1}^{n}\prod_{j=1:j\neq i}^{n} g_{2,\lambda_i+\lambda_j}^{a_i b_j - c_i}\\ =& g_{2,\upsilon}^{r_a r_b} \cdot \prod_{i=1}^{n} g_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2^{\frown}\Lambda} g_{2,\ell}^{\mu_\ell} \enspace ,\end{aligned}$$

where $\mu_\ell$ is as in Prot. 2, we have $\hat{e}(g_1, \pi) = \hat{e}(g_1, g_2^{F(\sigma)}) = h^{F(\sigma)} = \hat{e}(A, B_2)/\hat{e}(C, D)$. Thus, the 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 the following adversary $\hat{\mathcal{A}}$ against the $\Phi_{\times}$-PSDL assumption, see Prot. 4.

Here, $\mathcal{C}$ is the challenger of the PSDL game. Let us analyse the advantage of $\hat{\mathcal{A}}$. First, clearly $\mathsf{crs}_{td}$ has the same distribution as $\mathcal{G}_{\mathsf{crs}}(1^{\kappa})$. Thus, $\mathcal{A}_{\times}$ gets a correct input. She aborts with some probability $1-\varepsilon$. Otherwise, with probability $\varepsilon$, $inp^{\times} = (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C})$ 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 (2a–2d) hold.

The steps from step 1 onwards are executed with probability $\varepsilon$. 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 $a_\ell b_\ell - c_\ell$. $\hat{\mathcal{A}}$ succeeds on step 2, since $\log_{g_2} \pi = \sum_{\ell \in \hat{\Lambda}} f_{\ell}^{*} \sigma^{\ell}$. All non-zero coefficients of $X^{\ell}$ in $f^{*}(X)$ correspond to $\ell \in \hat{\Lambda}$. Since $\Lambda$ is progression-free,

```
$\mathcal{C}$ 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 (2a–2d) in 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 $(t_1, \dots, t_{2(\upsilon - \lambda_1)}) \leftarrow \mathsf{PolyFact}(\delta)$;
3     $\hat{\mathcal{A}}$ finds by an exhaustive search a root $\sigma_0 \in (t_1, \dots, t_{2(\upsilon - \lambda_1)})$, s.t. $g_2^{\sigma^\upsilon} = g_2^{\sigma_0^\upsilon}$;
      $\hat{\mathcal{A}}$ returns $\sigma \leftarrow \sigma_0$ to the challenger;
end
```

**Algorithm 4.** Construction of $\hat{\mathcal{A}}$ in the security reduction of Thm. 3

$\upsilon > 2\lambda_n - \lambda_1$, and all elements of $2 \cdot \Lambda$ are distinct, then by Lem. 1, $\ell \not\in 2 \cdot \Lambda$. Thus, all coefficients of $f^*(X)$ corresponding to any $X^\ell$, $\ell \in 2 \cdot \Lambda$, are 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 $f(\sigma) = f^*(\sigma) = F(\sigma)$. All coefficients of $X^\ell$, for $\ell < 2\lambda_1$, of both $f(X)$ and $f^*(X)$ are equal to 0.

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

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

$\hat{\mathcal{A}}$ uses polynomial factorization to find all $\leq 2(\upsilon - \lambda_1)$ roots of $\delta$. One of the roots must be equal to $\sigma$. On step 3, $\hat{\mathcal{A}}$ finds which root is equal to $\sigma$ by an exhaustive search among all roots returned in the previous step. Clearly $\hat{\mathcal{A}}$ returns the correct value of $\sigma$ (and thus violates the $\Phi_\times$-PSDL assumption) with probability $\varepsilon$. The execution time of $\hat{\mathcal{A}}$ is clearly dominated by the execution time of $\mathcal{A}_\times$ and the time to factor $\delta$. □

## C Proof of Thm. 5 (Shift Argument Security)

*Proof.* Denote $h \leftarrow \hat{e}(g_1, g_2)$ and

$$F(\sigma) := \log_h(\hat{e}(A, g_{2,\xi}) / \hat{e}(B, g_2)) \ .$$

Witness-Indistinguishability: since argument $\pi^{\mathsf{rsft}}$ 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 $\mathsf{crs}$ as in Prot. 3;
$\mathcal{C}$ sends $\mathsf{crs}$ to $\tilde{\mathcal{A}}$;
$\tilde{\mathcal{A}}$ obtains $(inp^{\mathsf{rsft}}, w^{\mathsf{rsft}}, \pi^{\mathsf{rsft}}) \leftarrow \mathcal{A}_{\mathsf{rsft}}(\mathsf{crs})$;
if the conditions (2a–2d) 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 $f^*(X) := \sum_{\phi \in \Phi^{\xi}_{\mathsf{rsft}}} f^*_\phi \cdot \phi(X)$;
      $\tilde{\mathcal{A}}$ lets $\delta(X) \leftarrow f(X) - f^*(X)$;
      $\tilde{\mathcal{A}}$ uses a polynomial factorization algorithm in $\mathbb{Z}_p[X]$ to compute all
      $\le (\upsilon + 2)$ roots of $\delta(X)$;
3     $\tilde{\mathcal{A}}$ finds by an exhaustive search a root $\sigma_0$, such that $g_1^{\sigma^\ell} = g_1^{\sigma_0^\ell}$;
      $\tilde{\mathcal{A}}$ returns $\sigma \leftarrow \sigma_0$;
end
```

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

PERFECT COMPLETENESS. The second verification is straightforward. For the first verification $\hat{e}(A, g_{2,\xi})/\hat{e}(B, g_2) = \hat{e}(g_1, \pi)$, consider

$$F(X) := X^\xi \cdot \log_{g_1} A - \log_{g_1} B \enspace ,$$

where we have replaced $\sigma$ with a formal variable $X$. Clearly,

$$\begin{aligned}
F(X) =& \sum_{i=1}^{n} a_i X^{\lambda_i+\xi} - \sum_{i=1}^{n} b_i X^{\lambda_i} + r_a X^{\upsilon+\xi} - r_b X^{\upsilon} \\
=& \sum_{i=n-\xi+1}^{n} a_i X^{\lambda_i+\xi} + \sum_{i=1}^{n-\xi} a_i X^{\lambda_i+\xi} - \sum_{i=1}^{\xi} b_i X^{\lambda_i} - \sum_{i=\xi+1}^{n} b_i X^{\lambda_i} + \\
& r_a X^{\upsilon+\xi} - r_b X^{\upsilon} \\
=& \sum_{i=1}^{\xi} a_{n-\xi+i} X^{\lambda_{n-\xi+i}+\xi} + \sum_{i=1}^{n-\xi} a_i X^{\lambda_i+\xi} - \sum_{i=1}^{\xi} b_i X^{\lambda_i} - \sum_{i=1}^{n-\xi} b_{i+\xi} X^{\lambda_{i+\xi}} + \\
& r_a X^{\upsilon+\xi} - r_b X^{\upsilon} \\
=& \underbrace{\sum_{i=1}^{n-\xi} (a_i - b_{i+\xi}) X^{\lambda_i+\xi} + \sum_{i=1}^{\xi} a_{n-\xi+i} X^{\lambda_{n-\xi+i}+\xi}}_{=: F_{con}(X)} + \\
& \underbrace{\sum_{i=1}^{n-\xi} b_{i+\xi} (X^{\lambda_i+\xi} - X^{\lambda_{i+\xi}}) - \sum_{i=1}^{\xi} b_i X^{\lambda_i} + r_a X^{\upsilon+\xi} - r_b X^{\upsilon}}_{=: F_{\pi}(X)} \enspace .
\end{aligned} \tag{3}$$

If the prover is honest, then $a_i = b_{i+\xi}$ for $i \in [n-\xi]$ and $a_i = 0$ for $i \in [n-\xi+1, n]$, and thus $F(X) = F_\pi(X)$ is spanned by $\{\phi(X)\}_{\phi \in \varPhi^\xi_{\mathsf{rsft}}}$. With $\pi$ as defined in Prot. 3, the second verification holds as

$$\hat{e}(g_1, \pi) = \hat{e}(g_1, \pi^{F(\sigma)}) = h^{F(\sigma)} = \hat{e}(A, g_{2,1}) / \hat{e}(B, g_2) \ .$$

Weaker version of soundness. Assume that $\mathcal{A}_{\mathsf{rsft}}$ is an adversary that can break the last statement of the theorem. We construct an adversary $\tilde{\mathcal{A}}$ against the $\varPhi^\xi_{\mathsf{rsft}}$-PSDL assumption, see Prot. 5. Here, $\mathcal{C}$ is the challenger of the PSDL game, and

$$\varPhi^\pi := \{X^{\lambda_i+\xi}, X^{\lambda_i}\}_{i=1}^n \cup \{X^{\upsilon+\xi}, X^\upsilon\}$$

is defined by following the first line of Eq. (3). Let us analyse the advantage of $\tilde{\mathcal{A}}$. First, clearly $\mathsf{crs}_{td}$ has the same distribution as $\mathcal{G}_{\mathsf{crs}}(1^\kappa)$. Thus, $\mathcal{A}_{\mathsf{rsft}}$ gets a correct input, and succeeds with some probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{rsft}}}(\varPi_{\mathsf{rsft}})$. Clearly, $\tilde{\mathcal{A}}$ aborts with probability $1 - \mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{rsft}}}(\varPi_{\mathsf{rsft}})$.

Otherwise, with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{rsft}}}(\varPi_{\mathsf{rsft}})$, $inp^{\mathsf{rsft}} = (A, \tilde{A}, B, \tilde{B})$ and $w^{\mathsf{rsft}} = (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, (f^*_\phi)_{\phi \in \varPhi^\xi_{\mathsf{rsft}}})$, such that the conditions (2a–2d) hold. In particular, $f(X) = F(X)$ in Eq. (3), and

$$\begin{aligned} f^*(X) = &\sum_{i=1}^{\xi} f^*_{X^{\lambda_i}} \cdot X^{\lambda_i} + \sum_{i=\xi+1}^{n} f_{X^{\lambda_{i-\xi}+\xi} - X^{\lambda_i}} (X^{\lambda_{i-\xi}+\xi} - X^{\lambda_i}) + \\ &f^*_{X^{\upsilon+\xi}} X^{\upsilon+\xi} + f^*_{X^\upsilon} X^\upsilon \ . \end{aligned}$$

For the rest of the proof to go through, we need that all polynomials that are present in monomials $F_{con}(X)$ ($\varPhi^* := \{X^{\lambda_i+\xi} : i \in [n-\xi]\} \cup \{X^{\lambda_{n-\xi+i}+\xi} : i \in [\xi]\} = \{X^{\lambda_i+\xi} : i \in [1, n-\xi]\} \cup \{X^{\lambda_i+\xi} : i \in [n-\xi+1, n]\} = \{X^{\lambda_i+\xi} : i \in [n]\}$) are different from each other and from all polynomials in $\varPhi^\xi_{\mathsf{rsft}}$. This follows from the conditions (i) $\lambda_j \neq \lambda_i$, (ii) $\lambda_j + \xi \neq \lambda_i$, (iii) $\lambda_i \neq \upsilon$, and (iv) $\lambda_i + \xi \neq \upsilon$, for $i, j \in [n]$, $i \neq j$.

Since $(a_n, a_{n-1}, \dots, a_1) \neq (0, \dots, 0, b_n, \dots, b_{\xi+1})$, $f(X)$ has at least one more non-zero monomial, either of type $a_i X^{\lambda_i+\xi}$ or of type $(a_{i-\xi} - b_i) X^{\lambda_{i-\xi}+\xi}$, than $f^*(X)$. Since $X^{\lambda_{i-\xi}+\xi}$ cannot be represented as a linear combination of polynomials from $\varPhi^\xi_{\mathsf{rsft}}$, $f(X)$ 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 to find all roots of $\delta$, and one of those roots must be equal to $\sigma$. On step 3, $\tilde{\mathcal{A}}$ finds which root is equal to $\sigma$ by an exhaustive search among all roots returned in the previous step. Thus, clearly $\tilde{\mathcal{A}}$ returns the correct value of $\sigma$ (and thus violates the $\varPhi^\xi_{\mathsf{rsft}}$-PSDL assumption) with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{rsft}}}(\varPi_{\mathsf{rsft}})$. Finally, the execution time of $\tilde{\mathcal{A}}$ is clearly dominated by the execution time of $\mathcal{A}_{\mathsf{rsft}}$ and the time to factor $\delta$. □

## D Rotation Argument

Since the rotation argument uses basically the same underlying ideas as the shift argument of Sect. 5, we will only comment on the differences between the new shift argument and the corresponding rotation argument.

In the right rotation-by-$\xi$ argument,

$$\begin{aligned}
F(X) =& \sum_{i=1}^{n} a_i X^{\lambda_i+\xi} - \sum_{i=1}^{n} b_i X^{\lambda_i} + r_a X^{\upsilon+\xi} - r_b X^{\upsilon} \\
=& \sum_{i=n-\xi+1}^{n} a_i X^{\lambda_i+\xi} + \sum_{i=1}^{n-\xi} a_i X^{\lambda_i+\xi} - \sum_{i=\xi+1}^{n} b_i X^{\lambda_i} - \sum_{i=1}^{\xi} b_i X^{\lambda_i} + \\
& r_a X^{\upsilon+\xi} - r_b X^{\upsilon} \\
=& \sum_{i=1}^{\xi} a_{n-\xi+i} X^{\lambda_{n-\xi+i}+\xi} + \sum_{i=1}^{n-\xi} a_i X^{\lambda_i+\xi} - \sum_{i=1}^{n-\xi} b_{\xi+i} X^{\lambda_{\xi+i}} - \sum_{i=1}^{\xi} b_i X^{\lambda_i} + \\
& r_a X^{\upsilon+\xi} - r_b X^{\upsilon} \\
=& \underbrace{\sum_{i=1}^{\xi} (a_{n-\xi+i} - b_i) X^{\lambda_{n-\xi+i}+\xi} + \sum_{i=1}^{n-\xi} (a_i - b_{\xi+i}) X^{\lambda_i+\xi} +}_{=:F_{con}(X)} \\
& \underbrace{\sum_{i=1}^{\xi} b_i (X^{\lambda_{n-\xi+i}+\xi} - X^{\lambda_i}) + \sum_{i=1}^{n-\xi} b_{\xi+i} (X^{\lambda_i+\xi} - X^{\lambda_{\xi+i}}) + r_a X^{\upsilon+\xi} - r_b X^{\upsilon}}_{F_{\pi}(X)}
\end{aligned}$$

Thus, if the prover is honest then $F(X) = F_{\pi}(X)$.

Here, $\Phi$ is different,

$$\Phi_{\mathsf{rot}}^{\xi} = \{X^{\upsilon}, X^{\upsilon+\xi}\} \cup \{X^{\lambda_{n-\xi+1}+\xi} - X^{\lambda_i}\}_{i=1}^{\xi} \cup \{X^{\lambda_i+\xi} - X^{\lambda_{i+\xi}}\}_{i=1}^{n-\xi} \ .$$

Moreover, for the proof of soundness to go through, it is necessary that all polynomials that are present in $F_{con}(X)$ (i.e., from the set $\Phi^* := \{X^{\lambda_{n-\xi+i}+\xi} : i \in [\xi]\} \cup \{X^{\lambda_i+\xi} : i \in [n-\xi]\} = \{X^{\lambda_i+\xi} : i \in [n]\}$), are mutually different and also different from every polynomial in $\Phi_{\mathsf{rot}}^{\xi}$. For this it is sufficient that exactly the same conditions hold as in the case of the right shift-by-$\xi$ argument, i.e., $\lambda_{i+1} > \lambda_i$, $\lambda_j \neq \lambda_i + \xi$ for $i \neq j$, and $\upsilon > \lambda_n + \xi$.

With this modification, one can construct a rotation argument that is very similar to Prot. 3.

## E Subset-Sum

Recall $\Phi_{\Gamma} = (\{X^{\upsilon}\} \cup (X^{\lambda_i})_{i=1}^{n})$. We will need $\Phi_{res}$-PKE assumptions to guarantee soundness of the restriction argument from [21], where $\Phi_{res}$ depends concretely on the restricted coordinates. Since $\Phi_{res} \subseteq \Phi_{\Gamma}$ (for example, in the following theorem, $\varPhi_{res} := \{X^{\upsilon}\} \cup \{X^{\lambda_i}\}_{i=2}^{n}$), we will not have to explicitly mention it.

**Theorem 8.** *Let $\varGamma = (\mathcal{G}_{\mathsf{com}}, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ be the be the $(\varLambda, \upsilon)$ commitment scheme in group $\mathbb{G}_1$. Let $\varLambda = (\lambda_1, \dots, \lambda_n)$ be a progression-free tuple of integers, such that $\lambda_{i+1} > \lambda_i + 1$ and $\lambda_i = \mathrm{poly}(\kappa)$. Let $\varPhi$ be as in Eq. (2). Let $\upsilon > \max(2\lambda_n - \lambda_1, \lambda_n + 1)$ be linear in $\lambda_n - \lambda_1$. The new* Subset-Sum *argument is perfectly complete and perfectly zero-knowledge. Also, $\mathcal{G}_{\mathsf{bp}}$ is $\varPhi$-PSDL secure and the $\varPhi_{\varGamma}$-PKE assumption holds in $\mathbb{G}_1$ and the $\varPhi$-PKE assumption holds in $\mathbb{G}_2$, then the* Subset-Sum *argument is computationally sound.*

*Proof.* Perfect completeness: Assume the prover is honest. The product arguments $\pi_1$ and $\pi_3$ will correctly verify due to Theorem 3 and replacing $(A, B, C)$ in the theorem respectively to $(B, B, B)$ and $(B, S, C)$ in the Subset-Sum protocol. The correctness of the non-zero argument $\pi_2$ can be seen as follows: $\pi_2$ shows that $\mathring{B}$ commits to the same value (and uses the same randomizer) as $B$. It also shows that $B^*$ 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 right shift-by-1 argument $\pi_4$ will also be correctly verified due to Theorem 5. Finally, $\pi_5$ correctly verifies that the first element of $\boldsymbol{c} + \boldsymbol{d}$ is 0 due to the completeness of the restriction argument [Gro10].

Adaptive computational soundness: Let $A$ be an NUPPT adversary that produces commitments $B, C, D$ and an accepting argument $(B, C, D, \pi_1, \dots, \pi_5)$. By the $\varPhi$-PKE assumption in $\mathbb{G}_2$ and by Thm. 3 and Thm. 5, the product and shift arguments are weakly sound according to the statements of corresponding theorems. (I.e., the extractor can open the inputs to the arguments to values that satisfy required restrictions.)

By the $\varPhi_{\varGamma}$-PKE assumption in $\mathbb{G}_1$, there exists a non-uniform PPT extractor $X_A$ that, given $A$'s input and access to $A$'s random coins, extracts all openings of $B, C$, and $D$. From the weaker version of soundness of the product and shift arguments (Theorem 3 and Theorem 5), and the soundness of the non-zero argument, we have that if $\mathcal{G}_{\mathsf{bp}}$ is $\varPhi$-PSDL secure then the following relations hold:

1. $B$ commits to $\boldsymbol{b}$ such that $b_i^2 = b_i \iff b_i \in \{0, 1\}$
2. $\boldsymbol{b} \neq 0$, so at least one of the $b_i$'s is 1.
3. $C$ commits to $\boldsymbol{c}$ such that $c_i = b_i s_i$.
4. $D$ commits to $\boldsymbol{d}$ such that $d_i = \sum_{j>i} c_j$.

Up to this point, it has been verified that $B$ is a commitment of a non-zero vector of boolean elements, and hence $C$ is a commitment of $\boldsymbol{c} = (b_i s_i)$ where each element is either 0 or $s_i$, and at least one of the elements is $c_i = s_i$. Now since $D$ is verified to be the scan of $c$, we have that the first element of $\boldsymbol{c} + \boldsymbol{d}$ is a sum $\sum_{i \geq 1} b_i s_i$. From the $\varPhi_{res}$-PKE assumption that guarantees the soundness of the restriction argument (Theorem 1 and Theorem 2 of [21]), we have that a correct verification implies that $(\boldsymbol{c} + \boldsymbol{d})_1 = 0$, so $A$ has indeed committed to a correct solution of Subset-Sum.

Perfect zero knowledge: We construct a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$. $\mathcal{S}_1$ will create a correctly formed CRS together with a simulation trapdoor $td = \sigma$. The adversary then outputs a correct statement $C_S$ together with a witness $w_S$. The simulator $\mathcal{S}_2$ creates a commitment to $\boldsymbol{b} = (1, 1, \ldots, 1)$ and commitments to the corresponding vectors $\boldsymbol{c}, \boldsymbol{d}$. Due to the knowledge of trapdoor $td$ and the commitment scheme being computationally (not perfect) binding, all the product, scan, non-zero and restriction arguments can be simulated correctly. This simulated NIZK argument $\psi'$ is perfectly indistinguishable from the real argument $\psi$. □

## F Decision-Knapsack

It is clear from the description of this argument that it works correctly. The Decision-Knapsack argument is clearly perfectly zero knowledge and computationally sound under appropriate assumptions, see App. F. The concrete complexity of the Decision-Knapsack argument depends on both how one defines $m$ in Groth's balancing technique and $u$ in the range argument.

**Theorem 9.** *Let $\Gamma = (\mathcal{G}_{\mathsf{com}}, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ be the be the $(\Lambda, \upsilon)$ commitment scheme in group $\mathbb{G}_1$. Let $\Lambda = (\lambda_1, \ldots, \lambda_n)$ be a progression-free tuple of integers, such that $\lambda_{i+1} > \lambda_i + 1$ and $\lambda_i = \mathrm{poly}(\kappa)$. Let $\Phi$ be as in Eq. (2). Let $\upsilon > \max(2\lambda_n - \lambda_1, \lambda_n + 1)$ be linear in $\lambda_n - \lambda_1$. The* Decision-Knapsack *protocol described by Alg. 6 is perfectly complete and perfectly zero-knowledge. Also, if $\mathcal{G}_{\mathsf{bp}}$ is $\Phi$-PSDL secure and the $\Phi_\Gamma$-PKE assumption holds in $\mathbb{G}_1$ and the $\Phi$-PKE assumption holds in $\mathbb{G}_2$, then the* Decision-Knapsack *protocol is computationally sound.*

*Proof.* Perfect completeness: Assume the prover is honest. The product arguments $\pi_1, \pi_2, \pi_4, \pi_5, \pi_7$ will correctly verify due to Theorem 3 and replacing $(A, B, C)$ in the theorem respectively to $(T, T, T)$, $(T, \boldsymbol{W}, W_T)$, $(A, F, C)$, $(T, \boldsymbol{B}, B_T)$ and $(D, F, E)$ in the Decision-Knapsack protocol. Here, $F = \{1, 0, \cdots, 0\}$. The right shift-by-1 arguments $\pi_3, \pi_6$ will also be correctly verified due to Theorem 5. Finally, $\pi_8$ and $\pi_9$ correctly verifies from the completeness of the range argument.

Adaptive computational soundness: Let $A$ be a non-uniform PPT adversary that produces commitments $B, C, D$ and an accepting NIZK argument $(T, W_T, A, C, B_T, D, E, \pi_1, \cdots, \pi_9)$. By the $\Phi$-PKE assumption in $\mathbb{G}_2$ and by Thm. 3 and Thm. 5, the product and shift arguments are weakly sound according to the statements of corresponding theorems. (That is, the extractor can open the inputs to the arguments to values that satisfy required restrictions.) By Thm. 7, the range argument is computationally sound.

By the $\Phi_\Gamma$-PKE assumption in $\mathbb{G}_1$, there exists a non-uniform PPT extractor $X_A$ that, given $A$'s input and access to $A$'s random coins, extracts all openings of $T, W_T, A, C, B_T, D, E$, and $F$. From the weaker version of soundness of the product and shift arguments (Thm. 3 and Thm. 5), and the soundness of the non-zero argument (Thm. 7), we have that the following relations hold:

```
Let $F$ be a commitment of $\boldsymbol{f} = (1, 0, \ldots, 0, 0)$ with randomness 0;
Let $t_i = 1$ iff $i \in \mathcal{T}$;
Generate a commitment $T$ of $\boldsymbol{t}$;
Prove that $T$ is Boolean by using a product argument $\pi_1$;
Generate a commitment $W_T$ of $\boldsymbol{w_T} = (w_1 t_1, \ldots, w_n t_n)$;
Prove that $W_T$ was computed correctly by using a product argument $\pi_2$;
Generate a scan $A$ of $W_T$, $a_i = \sum_{j>i} w_j t_j$;
Prove that $A$ was computed correctly by using a scan argument $\pi_3$;
Generate a commitment $C$ of $(\sum_{i=1}^{n} w_i t_i, 0, \ldots, 0)$;
Prove that $C$ was created correctly ($\boldsymbol{c}$ is a Hadamard product of $\boldsymbol{f}$ and
$\boldsymbol{w_T} + \boldsymbol{a}$) by using a product argument $\pi_4$;
Generate a commitment $B_T$ of $\boldsymbol{b_T} = (b_1 t_1, \ldots, b_n t_n)$;
Prove that $B_T$ was computed correctly by using a product argument $\pi_5$;
Generate a scan $D$ of $B_T$, $d_i = \sum_{j>i} b_j t_j$;
Prove that $D$ was computed correctly by using a scan argument $\pi_6$;
Generate a commitment $E$ of $(\sum_{i=1}^{n} b_i t_i, 0, \ldots, 0)$;
Prove that $E$ was created correctly ($\boldsymbol{e}$ is a Hadamard product of $\boldsymbol{f}$ and $\boldsymbol{b_T} + \boldsymbol{d}$)
by using a product argument $\pi_7$;
Prove that the first element of $C$ is $\leq W$ by using a range argument $\pi_8$;
Prove that the first element of $E$ is $\geq B$ by using a range argument $\pi_9$;
The whole argument is $(T, W_T, A, C, B_T, D, E, \pi_1, \ldots, \pi_9)$;
```

**Algorithm 6.** The Decision-Knapsack argument

1. $T$ commits to $\boldsymbol{t}$ such that $t_i^2 = t_i \iff t_i \in \{0, 1\}$,
2. $W_T$ commits to $\boldsymbol{w_T}$ such that $(w_T)_i = w_i t_i$,
3. $A$ commits to $\boldsymbol{a}$ such that $a_i = \sum_{j>i} w_j t_j$,
4. $C$ commits to the Hadamard product $\boldsymbol{c}$ of $\boldsymbol{f}$ and $\boldsymbol{w_T} + \boldsymbol{a}$, so $\boldsymbol{c} = (1 \cdot (\boldsymbol{w_T} + \boldsymbol{a})_1, 0 \cdot (\boldsymbol{w_T} + \boldsymbol{a})_2, \ldots, 0 \cdot (\boldsymbol{w_T} + \boldsymbol{a})_n) = (\sum_{i=1}^{n} w_i t_i, 0, \cdots, 0)$,
5. $B_T$ commits to $\boldsymbol{b_T}$ such that $(b_T)_i = b_i t_i$,
6. $D$ commits to $\boldsymbol{d}$ such that $d_i = \sum_{j>i} b_j t_j$,
7. $E$ commits to the Hadamard product $\boldsymbol{e}$ of $\boldsymbol{f}$ and $\boldsymbol{b_T} + \boldsymbol{d}$, so $\boldsymbol{e} = (1 \cdot (\boldsymbol{b_T} + \boldsymbol{d})_1, 0 \cdot (\boldsymbol{b_T} + \boldsymbol{d})_2, \cdots, 0 \cdot (\boldsymbol{b_T} + \boldsymbol{d})_n) = (\sum_{i=1}^{n} b_i t_i, 0, \cdots, 0)$.

From the soundness of the range argument, a correct verification of $\pi_8$ will imply that $\boldsymbol{c}_1 \in [0, B]$ while a correct verification of $\pi_9$ will imply that $\boldsymbol{e}_1 \in [E, 2^\kappa]$ for some $\kappa$.

Perfect zero knowledge: We can construct a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ analogous to the simulator for Subset-Sum. $\square$