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

## Draft, February 13, 2013


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

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



**Abstract.** We propose several new efficient non-interactive zero knowledge (NIZK) arguments in the common reference string model. The final arguments are based on two building blocks, a more efficient version of Lipmaa's Hadamard product argument from TCC 2012, and a novel shift argument. Based on these two arguments, we speed up the recent range argument by Chaabouni, Lipmaa and Zhang (FC 2012). We also propose efficient arguments for two NP-complete problems, subset sum and decision knapsack, with constant communication, quasilinear prover's computation and linear verifier's computation.


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


# 1 Introduction

Zero knowledge proofs make it possible for a prover to prove that he has correctly followed some protocol without violating his privacy. Efficient non-interactive zero knowledge (NIZK) proofs and arguments (that is, computationally sound proofs) play an important role in the design of cryptographic protocols. A typical such 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 it is unreasonable to expect that voters are available every time when one verifies the cast ballots, one cannot really rely on interactive zero knowledge. Moreover, one is interested in succinct (e.g., polylogarithmic in input size) NIZK arguments with efficient verification. This is since in applications like e-voting, correctness proofs should be collected and stored and then verified by potentially many independent observers.

Only a few generic techniques[1] of constructing succinct NIZK arguments are known. In [Gro10], Groth constructed the first succinct NIZK argument for Circuit-SAT, an NP-complete language. His Circuit-SAT argument is based on two basic arguments, for Hadamard product and for permutation. Let $n = |C|$ be the circuit size. Groth's basic arguments both have quadratic common reference string (CRS) length and prover's computation, while the communication and verifier's computation are constant; see Tbl. 1. Thus, Groth's arguments offer essentially optimal communication and verifier's computational complexity, but they are quite inefficient in other parameters. In particular, they will not be able to handle circuits of size $2^{10}$ or more.

Subsequently, Lipmaa [Lip12] improved Groth's basic arguments — and thus also the Circuit-SAT argument — by using the theory of progression-free sets. Namely, let $r_3(N) = N^{1-o(1)}$ be the size of the largest known progression-free subset of $[N] = \{1, \ldots, N\}$. Lipmaa's argument has the CRS size to $n^{1+o(1)}$ group elements, and also somewhat more efficient prover's computational complexity. Lipmaa's product and permutation arguments can be used to construct a Circuit-SAT argument with similar asymptotic complexity, see Tbl. 1. (The verifier's computation in Lipmaa's argument in Tbl. 1 differs from what was claimed in [Lip12]. The slightly incorrect claim from [Lip12] was also replicated in [CLZ12]. See Remark 1 on page 11 for a clarification.)

The product and permutation arguments (together with the trivial sum argument that is based on the homomorphic property of the commitment scheme) of Groth and Lipmaa [Gro10,Lip12] can be used to construct other complex arguments (e.g., range arguments [CLZ12]), though the full power of the "NIZK programming language" that consists of these 3 arguments is yet unknown. In addition, as shown in [LZ12], following the same framework, one can construct other basic arguments — for example, 1-sparsity in [LZ12] — and use them to construct efficient complex arguments (shuffle in [LZ12]). It is an important open problem to increase the library of efficient basic arguments even further, and to investigate which (more complex) arguments can be solved by using the new basic

[1] We only mention NIZK proofs that work in the common reference string (CRS) model and not in the random oracle model, since random oracles cannot always be instantiated [CGH98,GK03]. As shown in [GW11], sublinear NIZK proofs are only possible under non-standard (for example, knowledge) assumptions.

**Table 1.** Comparison of knowledge-assumption based adaptive 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 $N^* = r_3^{-1}(\sqrt{n})$, $m$ is the balancing parameter, $\mathfrak{g}$ corresponds to 1 group element and $\mathfrak{a}/\mathfrak{m}/\mathfrak{m}_b/\mathfrak{e}/\mathfrak{p}$ correspond to 1 addition/multiplication in $\mathbb{Z}_p$/multiplication in bilinear group/exponentiation/pairing

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

arguments. Moreover, the basic arguments of Groth and Lipmaa are still computationally intensive for the prover, and the construction of more efficient basic arguments (that at the same time have meaningful applications) is an important open problem.

**Our Contributions.** We make Lipmaa's product argument more efficient, and we propose a new efficient shift argument. We then show how to use the simpler programming language that consists of the product argument and the shift argument — together with some other trivial arguments — to construct an efficient argument for subset sum (another NP-complete language), to make the range argument of [CLZ12] more efficient, and finally to construct an efficient argument for decision knapsack (also NP-complete). We hope that by using the same techniques, it will be possible to use NIZK arguments for other (possibly more interesting in practice) languages.

We first modify the commitment scheme from [Lip12]. Recall that in that commitment scheme (and thus also in all related NIZK arguments), one has to use a progression-free set $\Lambda$ of odd positive integers. When the new commitment scheme is used, $\Lambda$ does not have to consist of positive odd integers. This is important conceptually, making it clear that one requires progression-freeness of $\Lambda$ (and nothing else) in similar arguments.

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

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

Fourth, we propose new shift and rotation arguments that have constant communication and verifier's computational complexity, and linear prover's computational complexity and CRS length, and can work with a large choice of sets $\Lambda$. 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 [CLZ12] (see also Sect. 3), which is a non-trivial generalization of the $\Lambda$-PSDL assumption from [Lip12]. To show that the $\Phi$-PSDL assumption is reasonable, we prove that the $\Phi$-PSDL assumption is secure in the generic group model.

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

When using the balancing techniques of [Gro10,Lip12] (where, instead of applying the arguments to length $n$-vectors, one applies them in parallel to $m$ length-$(n/m)$ vectors), if $m = \sqrt{n}$, we obtain a balanced subset sum NIZK argument with the parameters, given in the last row of Tbl. 1. See Tbl. 1 for more comparison with previous work, and [Gro10,Lip12] for more background about the balancing techniques.

**Efficient Range Argument.** In a range argument [Bou00,LAN02,Lip03,CCs08,CLs10], the prover aims to convince the prover 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 NIZK range arguments has only taken off during the last few years [RKP09,CLZ12]. In [CLZ12], Chaabouni, Lipmaa and Zhang used the product and permutation arguments of [Lip12] to construct the first known constant-communication (interactive or non-interactive) range argument that works in prime-order groups. They achieved this by combining the basic arguments of [Gro10,Lip12] with several different (and unrelated) techniques that have been developed specifically for range proofs in [LAN02,CLs10].

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

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

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

One can use the techniques of [BCCT12] to further decrease the length of the CRS, see [BCCT12] for more details.

Due to the lack of space, practically all proofs are given in appendices.

## 2 Preliminaries

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

**Additive Combinatorics.** 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 $k\Lambda = \{\lambda_1 + \dots + \lambda_k : \lambda_i \in \Lambda\}$ 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 [TV06] for more notation and background.)

**Progression-Free Sets.** A set $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ is *progression-free* [ET36,TV06], 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 [Elk11] proved that $r_3(N) = \Omega((N \cdot \log^{1/4} N)/2^{2\sqrt{2 \log_2 N}})$. Thus, for any fixed $n > 0$, there exists $N = o(n 2^{2\sqrt{2 \log_2 n}})$, such that $[N]$ contains an $n$-element progression-free subset.

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

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

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

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

**Commitment Schemes.** A trapdoor commitment scheme $\Gamma$ consists of five PPT algorithms: a randomized common reference string (CRS) generation algorithm $\mathcal{G}\mathrm{com}$, a randomized commitment algorithm $\mathcal{C}\mathrm{om}$, a randomized trapdoor CRS generation algorithm $\mathcal{G}\mathrm{com}_{td}$, a randomized trapdoor commitment algorithm $\mathcal{C}\mathrm{om}_{td}$, and a trapdoor opening algorithm $\mathcal{O}\mathrm{pen}_{td}$. More precisely, (a) the CRS generation algorithm $\mathcal{G}\mathrm{com}(1^\kappa)$ produces a CRS $\mathsf{ck}$, (b) 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 $(\boldsymbol{a}, r)$, (c) the trapdoor CRS generation algorithm $\mathcal{G}\mathrm{com}_{td}(1^\kappa)$ outputs a CRS $\mathsf{ck}_{td}$, which has the same distribution as $\mathcal{G}\mathrm{com}(1^\kappa)$, and a trapdoor $\mathsf{td}$, (d) 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 $\mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \mathbf{0}; r)$, and (e) the trapdoor opening algorithm $\mathcal{O}\mathrm{pen}_{td}(\mathsf{ck}_{td}; \mathsf{td}, \boldsymbol{a}; r)$ outputs an $r_{td}$, such that $\mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \mathbf{0}; r) = \mathcal{C}\mathrm{om}(\mathsf{ck}_{td}; \boldsymbol{a}; r_{td})$.

A commitment scheme $\Gamma$ is *computationally binding*, if for every NUPPT adversary $\mathcal{A}$, $\Pr[\mathsf{ck} \leftarrow \mathcal{G}\mathrm{com}(1^\kappa), (\boldsymbol{a_1}, r_1, \boldsymbol{a_2}, r_2) \leftarrow \mathcal{A}(\mathsf{ck}) : (\boldsymbol{a_1}, r_1) \neq (\boldsymbol{a_2}, r_2) \wedge \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_1}; r_1) = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_2}; r_2)] = \mathrm{negl}(\kappa)$. A commitment scheme $\Gamma$ is *perfectly hiding*, if for any $\mathsf{ck} \in \mathcal{G}\mathrm{com}(1^\kappa)$ and any two messages $\boldsymbol{a_1}, \boldsymbol{a_2}$, the distributions $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_1}; \cdot)$ and $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a_2}; \cdot)$ are equal.

The commitment scheme of this paper allow committing to vectors $\boldsymbol{a} = (a_1, \dots, a_n)$. Thus, one must input $n$ (or a reasonable upper bound on $n$) as an additional parameter for the CRS and trapdoor CRS generation algorithms. It is assumed that the value of $n$ is implicitly obvious while committing and trapdoor opening.

**Non-Interactive Zero-Knowledge.** 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 $\mathbf{NP}$-language. Let $n$ be some fixed input length $n = |C|$. For fixed $n$, we have a relation $\mathcal{R}_n$ and a language $\mathcal{L}_n$. A *non-interactive argument* for $\mathcal{R}$ consists of the following PPT algorithms: a common reference string (CRS) generator $\mathcal{G}_{\mathsf{crs}}$, a prover $\mathcal{P}$, and a verifier $\mathcal{V}$. For $\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$, $\mathcal{P}(\mathsf{crs}; C, w)$ produces an argument $\pi$. The verifier $\mathcal{V}(\mathsf{crs}; C, \pi)$ outputs either 1 (accept) or 0 (reject).

A non-interactive argument $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}, \mathcal{V})$ is *perfectly complete*, if $\forall n = \mathrm{poly}(\kappa)$, $\Pr[\mathsf{crs} \leftarrow \mathcal{G}_{\mathsf{crs}}(1^\kappa, n), (C, w) \leftarrow \mathcal{R}_n : \mathcal{V}(\mathsf{crs}; C, \mathcal{P}(\mathsf{crs}; C, w)) = 1] = 1$.

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

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

A non-interactive argument $(\mathcal{G}_{\mathsf{crs}}, \mathcal{P}, \mathcal{V})$ is *perfectly witness-indistinguishable*, if for all $n = \mathrm{poly}(\kappa)$, if $\mathsf{crs} \in \mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$ and $((C, w_0), (C, w_1)) \in \mathcal{R}_n^2$, then the distributions $\mathcal{P}(\mathsf{crs}; C, w_0)$ and $\mathcal{P}(\mathsf{crs}; C, w_1)$ are equal.

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

*door*),

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

## 3 New Commitment Scheme

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

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 $x \in \mathbb{Z}_p$. We use the following security assumptions from [CLZ12].

**Definition 1 ($\Phi$-PDL and $\Phi$-PSDL assumptions).** *A bilinear group generator $\mathcal{G}_{\mathsf{bp}}$ is $\Phi$-*PDL *secure in $\mathbb{G}_t$, if for any NUPPT adversary $\mathcal{A}$,* $\Pr[\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa), g_t \leftarrow \mathbb{G}_t \setminus \{1\}, \sigma \leftarrow \mathbb{Z}_p : \mathcal{A}(\mathsf{parm}; (g_t^{\varphi(\sigma)})_{\varphi \in \{1\} \cup \Phi}) = \sigma] = \mathrm{negl}(\kappa)$. $\mathcal{G}_{\mathsf{bp}}$ *is $\Phi$-*PSDL *secure, if for any NUPPT adversary $\mathcal{A}$,*

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

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

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

Similarly to [Gro10,Lip12,CLZ12,LZ12], we will base our NIZK arguments on an explicit knowledge assumption. The concrete assumption, originally proposed in [CLZ12], is a generalization of the $q$-PKE assumption of Groth [Gro10] and the $\Lambda$-PKE assumption of Lipmaa [Lip12]. Let $t \in \{1, 2\}$. For algorithms $\mathcal{A}$ and $X_{\mathcal{A}}$, we write $(y; z) \leftarrow (\mathcal{A}||X_{\mathcal{A}})(\sigma)$ if $\mathcal{A}$ on input $\sigma$ outputs $y$, and $X_{\mathcal{A}}$ on the same input (including the random tape of $\mathcal{A}$) outputs $z$.

**Definition 2 ($\Phi$-PKE security, [CLZ12]).** *Let $t \in \{1, 2\}$. The bilinear group generator $\mathcal{G}_{\mathsf{bp}}$ is $\Phi$-*PKE *secure in $\mathbb{G}_t$ if for any NUPPT adversary $\mathcal{A}$ there exists a NUPPT extractor $X_{\mathcal{A}}$, such that the following probability is negligible in $\kappa$:*

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

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

Let $t = 1$. 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 $t = 2$ is dual.

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

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

**System parameters:** $n = \mathrm{poly}(\kappa)$, $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ with $\lambda_i < \lambda_{i+1}$, $\lambda_i = \mathrm{poly}(\kappa)$, and $\upsilon > \max_i \lambda_i$. A bilinear group generator $\mathcal{G}_{\mathsf{bp}}$;
**Trapdoor CRS generation** $\mathcal{G}\mathbf{com}_{td}(1^\kappa, n)$**:** Set $\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_t \leftarrow \mathbb{G}_t \setminus \{1\}$, and $(\sigma, \hat{\alpha}) \leftarrow \mathbb{Z}_p^2$; For each $i \in [n]$ do: $g_{t,\lambda_i} \leftarrow g_t^{\sigma^{\lambda_i}}$, $\hat{g}_{t,\lambda_i} \leftarrow g_t^{\hat{\alpha}\sigma^{\lambda_i}}$; Set $h_t \leftarrow g_t^{\sigma^\upsilon}$, $\hat{h}_t \leftarrow g_t^{\hat{\alpha}\sigma^\upsilon}$; Let $\mathsf{ck} \leftarrow (\mathsf{parm}; (g_{t,\lambda_i} \hat{g}_{t,\lambda_i})_{i \in [n]}, h_t, \hat{h}_t)$; Return $(\mathsf{ck}; \mathsf{td} \leftarrow \sigma)$;
**CRS generation** $\mathcal{G}\mathbf{com}(1^\kappa, n)$**:** $(\mathsf{ck}; \mathsf{td}) \leftarrow \mathcal{G}\mathrm{com}(1^\kappa)$; return $\mathsf{ck}$;
**Commitment** $\mathcal{C}\mathbf{om}(\mathsf{ck}; \boldsymbol{a}; \cdot)$**:** $r \leftarrow \mathbb{Z}_p$; return $(h_t^r \cdot \prod_{i=1}^n g_{t,\lambda_i}^{a_i}, \hat{h}_t^r \cdot \prod_{i=1}^n \hat{g}_{t,\lambda_i}^{a_i})$;
**Trapdoor commitment** $\mathcal{C}\mathbf{om}_{td}(\mathsf{ck}_{td}; \cdot)$**:** $r \leftarrow \mathbb{Z}_p$; return $(h_t^r, \hat{h}_t^r)$;
**Trapdoor opening** $\mathcal{O}\mathbf{pen}_{td}(\mathsf{ck}_{td}; \mathsf{td}, \boldsymbol{a}, r)$**:** return $r_{td} \leftarrow r\sigma^\upsilon - \sum_{i=1}^n a_i \sigma^{\lambda_i}$;

**Protocol 1:** The $(\Lambda, \upsilon)$ trapdoor commitment scheme for $t \in \{1, 2\}$. Here, $\boldsymbol{a} = (a_1, \dots, a_n) \in \mathbb{Z}_p^n$

We will sometimes use the same commitment scheme in both $\mathbb{G}_1$ and $\mathbb{G}_2$. In such cases, we will emphasize the underlying group by having a different CRS, but we will not change the name of the commitment scheme.
**Computational Complexity.** Assume that $\alpha = ||\boldsymbol{a}||_\infty = \max_i a_i$, and $n \geq 2$. When using Pippenger's multi-exponentiation algorithm [Pip80], the computational complexity of the commitment function $\mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r)$ is dominated by $2 \log_2 \alpha + (2 + o(1)) \cdot n \log_2 \alpha / \log_2(n \log_2 \alpha) + O(n)$ multiplications in $\mathbb{G}_t$. In our applications, $n \gg \log_2 \alpha$ (for example, $\alpha = 2$, $\alpha = n$, or even $\alpha = p$ given that $n$ is reasonably large), and thus we get a simpler bound of $(2 + o(1)) \cdot \frac{n}{\log_2 n} \cdot \log_2 \alpha + O(n)$ multiplications. This can be compared to $3n \log_2 \alpha$ multiplications on average that one would have to execute by using the straightforward square-and-multiply exponentiation algorithm.

## 4 Improved Hadamard Product Argument

In this section, we propose a version of the product argument of [Lip12] that works together with the $(\Lambda, \upsilon)$ commitment scheme of Sect. 3. As we will see below (both in this section and in Sect. 5), the value of $\upsilon$ depends on the precise construction of the argument. For example, while the commitment scheme is binding for $\upsilon > \lambda_n$, for the product argument to be (weakly[2]) sound we require that $\upsilon > 2\lambda_n - \lambda_1$. If one uses several such arguments together (for example, to construct a subset sum argument or a range argument), one has to choose a value of $\upsilon$ 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 $\Gamma = (\mathcal{G}\mathrm{com}, \mathcal{C}\mathrm{om}, \mathcal{G}\mathrm{com}_{td}, \mathcal{C}\mathrm{om}_{td}, \mathcal{O}\mathrm{pen}_{td})$ is a trapdoor commitment scheme that commits to elements $\boldsymbol{a} = (a_1, \dots, a_n) \in \mathbb{Z}_p^n$ for a prime $p$ and integer $n \geq 1$. In an *Hadamard product argument*, the prover aims to convince the verifier that given commitments $A$, $B$ and $C$, he can open them as $A = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{a}; r_a)$, $B = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{b}; r_b)$, and $C = \mathcal{C}\mathrm{om}(\mathsf{ck}; \boldsymbol{c}; r_c)$, such that $c_i = a_i b_i$ for $i \in [n]$. In other words, a product argument has $n$ constraints $c_i = a_i b_i$ for $i \in [n]$.

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

We rewrite Lipmaa's argument for the $(\Lambda, \upsilon)$ commitment scheme $\Gamma$. Similarly to [Lip12], 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 $\mathbb{G}_1$ (see Prot. 2), and $\widehat{\mathsf{ck}}^*$ be the dual CRS in group $\mathbb{G}_2$ (that is, $\widehat{\mathsf{ck}}^*$ is defined as $\widehat{\mathsf{ck}}$, but with $g_1$ replaced by $g_2$). Thus, for example, $(B, \hat{B}) = \mathcal{C}\mathrm{om}(\widehat{\mathsf{ck}}; \boldsymbol{b}; r_b)$. Then, we have $\log_{g_1} A = r_a \sigma^\upsilon + \sum_{i=1}^n a_i \sigma^{\lambda_i}$, $\log_{g_1} B = r_b \sigma^\upsilon + \sum_{i=1}^n b_i \sigma^{\lambda_i}$, and $\log_{g_1} C = c_i \sigma^\upsilon + \sum_{i=1}^n r_c \sigma^{\lambda_i}$. The prover has also computed an element $B_2$, such that $\hat{e}(g_1, B_2) = \hat{e}(B, g_2)$. Thus, for $(D, \hat{D}) = \mathcal{C}\mathrm{om}(\widehat{\mathsf{ck}}^*; \mathbf{1}; 0)$ (in group $\mathbb{G}_2$), $\log_{\hat{e}(g_1, g_2)}(\hat{e}(A, B_2)/\hat{e}(C, D)) = (r_a \sigma^\upsilon + \sum_{i=1}^n a_i \sigma^{\lambda_i})(r_b \sigma^\upsilon + \sum_{i=1}^n b_i \sigma^{\lambda_i}) - (r_c \sigma^\upsilon + \sum_{i=1}^n c_i \sigma^{\lambda_i})(\sum_{i=1}^n \sigma^{\lambda_i})$ can be written — after substituting $\sigma$ with a formal variable $X$ — as a sum of two formal polynomials $F_{con}(X)$ and $F_\pi(X)$, such that $F_{con}(X)$ (the constraint polynomial) as a formal polynomial has one monomial per constraint ($a_i b_i = c_i$) and is zero if the prover is honest, while $F_\pi(X)$ (the *argument polynomial*) has potentially many more monomials. (More precisely, $F_\pi$ 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 assumption guarantee that he cannot do it if at least one of the $n$ constraints is not satisfied.

[2] For an explanation and motivation of weak soundness, we refer the reader to [Gro10,Lip12]

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

**CRS generation** $\mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$**:** Set parm $:= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $(g_1, g_2) \leftarrow (\mathbb{G}_1 \setminus \{1\}, \mathbb{G}_2 \setminus \{1\})$, $\sigma, \hat{\alpha} \leftarrow \mathbb{Z}_p$;

For each $\ell \in \{\upsilon\} \cup \Lambda$ do: $g_{1,\ell} \leftarrow g_1^{\sigma^\ell}$, $\hat{g}_{1,\ell} \leftarrow g_1^{\hat{\alpha}\sigma^\ell}$;

For each $\ell \in \{\upsilon\} \cup \hat{\Lambda}$ do: $g_{2,\ell} \leftarrow g_2^{\sigma^\ell}$, $\hat{g}_{2,\ell} \leftarrow g_2^{\hat{\alpha}\sigma^\ell}$;

Set $D \leftarrow \prod_{i=1}^n g_{2,\lambda_i}$, $\widehat{\mathsf{ck}} \leftarrow (\mathsf{parm}; (g_{1,\ell}, \hat{g}_{1,\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$;

Return $\mathsf{crs} \leftarrow (\widehat{\mathsf{ck}}, g_{2,\upsilon}, (g_{2,\ell}, \hat{g}_{2,\ell})_{\ell \in \hat{\Lambda}}, D)$;

**Argument generation** $\mathcal{P}_\times(\mathsf{crs}; (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C}), (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c))$**:**

For each $\ell \in 2\widehat{\ }\Lambda$ do: $\mu_\ell \leftarrow \sum_{(i,j) \in \mathfrak{I}_1(\ell)} (a_i b_j - c_i)$;

$\pi \leftarrow g_{2,2\upsilon}^{r_a r_b} \cdot \prod_{i=1}^n g_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2\widehat{\ }\Lambda} g_{2,\ell}^{\mu_\ell}$;

$\hat{\pi} \leftarrow \hat{g}_{2,2\upsilon}^{r_a r_b} \cdot \prod_{i=1}^n \hat{g}_{2,\upsilon+\lambda_i}^{r_a b_i + r_b a_i - r_c} \cdot \prod_{\ell \in 2\widehat{\ }\Lambda} \hat{g}_{2,\ell}^{\mu_\ell}$;

return $\pi^\times \leftarrow (\pi, \hat{\pi}) \in \mathbb{G}_2^2$;

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

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

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

In what follows, we show that by using the $(\Lambda, \upsilon)$ 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

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

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

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

**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 $\lambda_n = \text{poly}(\kappa)$, and an integer $\upsilon > 2\lambda_n - \lambda_1$, $\upsilon$ linear in $\lambda_n - \lambda_1$, such that $|\hat{\Lambda}| = \Theta(r_3^{-1}(n)) = o(n 2^{2\sqrt{2\log_2 n}})$.*

We can clearly add some constant $k$ 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 Hadamard product argument when using the $(\Lambda, \upsilon)$ knowledge commitment scheme by closely following the claim and the proof from [Lip12]. The (knowledge) commitments are $(A, \hat{A})$, $(B, \hat{B})$ and $(C, \hat{C})$. For efficiency (and backwards compatibility) reasons, following [Lip12], 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 $(a_1 b_1, \dots, a_n b_n)$ for *some* value of $r_c$, Prot. 2 cannot be computationally sound (even under a knowledge assumption). Instead, analogously to [Gro10,Lip12], we prove a somewhat weaker version of soundness that is however sufficient to achieve soundness of the subset sum and range arguments. The last statement of Thm. 3 basically says that no efficient adversary can output an input to the Hadamard product argument together with an accepting argument and openings to all commitments and all other pairs of type $(y, \hat{y})$ that are present in the argument, such that $a_i b_i \neq c_i$ for some $i \in [n]$. This "weak" soundness is similar to the co-soundness as defined in [GL07]. However, in the case of co-soundness, the adversary is not be required to open the argument (by presenting values $f_\ell^*$, as in the theorem statement).

**Theorem 3 (Security of product argument).** *Let $\Gamma$ be the $(\Lambda, \upsilon)$ commitment scheme in group $\mathbb{G}_1$. Let $\Phi_\times := \{X^\upsilon\} \cup \{X^\ell\}_{\ell \in \hat{\Lambda}}$. Then*
*(1) Prot. 2 is perfectly complete and perfectly witness-indistinguishable.*
*(2) If $\mathcal{G}_{\mathsf{bp}}$ is $\Phi_\times$-PSDL secure, $\Lambda$ is progression-free, and $\upsilon > 2\lambda_n - \lambda_1$, then a NUPPT adversary against Prot. 2 has negligible chance, given correctly generated CRS* crs *as an input, of outputting $inp^\times \leftarrow (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C})$ and an accepting argument $\pi^\times \leftarrow (\pi, \hat{\pi})$ together with a witness $w^\times \leftarrow (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c, (f_\ell^*)_{\ell \in \hat{\Lambda}})$, such that*

```
for i ← 0 to λ_n do: a†_i ← 0, b†_i ← 0, c†_i ← 0, d†_i ← 0;
for i ← 1 to n do: a†_{λ_i} ← a_i, b†_{λ_i} ← b_i, c†_{λ_i} ← c_i, d†_{λ_i} ← 0;
Denote a†(X) := Σ_{i=0}^{λ_n} a†_i X^i, b†(X) := Σ_{i=0}^{λ_n} b†_i X^i, c†(X) := Σ_{i=0}^{λ_n} c†_i X^i and d†(X) := Σ_{i=0}^{λ_n} d†_i X^i;
Let μ(X) ← FFTMult(a†(X), b†(X)); Let ν(X) ← FFTMult(c†(X), d†(X));
for i ← 1 to n do: μ_{2λ_i} ← μ_{2λ_i} − a_i b_i;
Let μ(X) ← μ(X) − ν(X);
```

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

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

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

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

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

*Proof.* By Lem. 2, the size of the CRS is $\Theta(|\hat{\Lambda}|) = \Theta(r_3^{-1}(n))$. From the CRS, the verifier clearly only needs to access $g_1, \hat{g}_1$, and $D$. Since $2\hat{\ }\Lambda \subseteq \hat{\Lambda}$, the statement about the prover's computational complexity follows from Fast Fourier Transform [CT65] based polynomial multiplication [GS66] techniques. To compute all the coefficients of the formal 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 Prot. 3. Here, $\mathsf{FFTMult}$ denotes a FFT-based polynomial multiplication algorithm.

After using $\mathsf{FFTMult}$ 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 Prot. 3, $\mu_\ell = \sum_{(i,j) \in \mathfrak{I}_1(\ell)} a_i b_j$, 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. ◻

We remark that FFT does not help to speed up Groth's product argument from [Gro10], since there $\lambda_n = \Theta(n^2)$. FFT also does not seem to be useful in the case of the permutation argument from [Lip12]. Finally, it may be possible to speed up Prot. 3, by taking into account the fact that all $a^\dagger$, $b^\dagger$, $c^\dagger$ and $d^\dagger$ have only $n$ non-zero monomials.

**Using Efficient Multi-Exponentiation.** Let $\alpha := \max(||a||_\infty, ||b||_\infty, ||c||_\infty)$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are the vectors committed by the prover. (See Sect. 6 for the concrete values of $\alpha$ needed in applications.) The number of bilinear-group operations the prover has to perform (on top of computing the exponents by using the described FFT-based polynomial multiplication technique) to compute $\pi$ in the product argument is dominated by $L(2, n, p) + L(2, r_3^{-1}(n), \Theta((\alpha n)^2))$. Here, the very conservative value $\Theta((\alpha n)^2)$ follows from $|\mu_\ell| = |\sum_{(i,j) \in \mathfrak{I}_1(\ell)} (a_i b_j - c_i)| \leq \sum_{(i,j) \in \mathfrak{I}_1(\ell)} |a_i b_j - c_i| \leq \sum_{(i,j) \in \mathfrak{I}_1(\ell)} (\alpha^2 + \alpha) < (n^2 - n)(\alpha^2 + \alpha) = \Theta((\alpha n)^2)$.

Due to Fact 1, we get that, for $n = \Omega(\log p)$, $L(2, n, p) = 2 \log_2 p + \frac{2n \log_2(p+1)}{\log_2(2n \log_2(p+1))} \cdot (1 + o(1)) + O(n) = (2 + o(1)) \cdot \frac{n}{\log_2 n} \cdot \log_2 p$, and, since in our applications, $n \gg \log_2 \Theta((\alpha n)^2)$, $L(2, r_3^{-1}(n), \Theta((\alpha n)^2)) = 2 \log_2(\alpha n^2) + \frac{2 r_3^{-1}(n) \log_2 \Theta((\alpha n)^2)}{\log_2(2 r_3^{-1}(n) \log_2 \Theta((\alpha n)^2))} \cdot (1 + o(1)) + O(r_3^{-1}(n)) = (2 + o(1)) \cdot \frac{r_3^{-1}(n)}{\log_2 r_3^{-1}(n)} \cdot 2 \log_2(\alpha n)$. Thus, the prover has to compute

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

**CRS generation** $\mathcal{G}_{\mathsf{crs}}(1^\kappa, n)$**:** Set $\mathsf{parm} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}) \leftarrow \mathcal{G}_{\mathsf{bp}}(1^\kappa)$, $g_1 \leftarrow \mathbb{G}_1 \setminus \{1\}$, $g_2 \leftarrow \mathbb{G}_2 \setminus \{1\}$, $\sigma, \tilde{\alpha} \leftarrow \mathbb{Z}_p$;
For each $t \in \{1, 2\}$ do: $\tilde{g}_t \leftarrow g_t^{\tilde{\alpha}}$;
For each $\ell \in \{\upsilon\} \cup \Lambda$ do: $(g_{1,\ell}, \tilde{g}_{1,\ell}) \leftarrow (g_1^{\sigma^\ell}, \tilde{g}_1^{\sigma^\ell})$;
Set $g_{2,1} \leftarrow g_2^\sigma$;
For each $\ell \in \{\lambda_1, \upsilon, \upsilon + 1\}$ do: $(g_{2,\ell}, \tilde{g}_{2,\ell}) \leftarrow (g_2^{\sigma^\ell}, \tilde{g}_2^{\sigma^\ell})$;
For each $\ell \in [2, n]$ do: $(h_{2,\ell}, \tilde{h}_{2,\ell}) \leftarrow (g_2^{\sigma^{\lambda_{i-1}+1} - \sigma^{\lambda_i}}, \tilde{g}_2^{\sigma^{\lambda_{i-1}+1} - \sigma^{\lambda_i}})$;
Set $\widetilde{\mathsf{ck}} \leftarrow (\mathsf{parm}; (g_{1,\ell}, \tilde{g}_{1,\ell})_{\ell \in \{\upsilon\} \cup \Lambda})$;
Return $\mathsf{crs} \leftarrow (\widetilde{\mathsf{ck}}, g_2, g_{2,1}, (g_{2,\ell}, \tilde{g}_{2,\ell})_{\ell \in \{\lambda_1, \upsilon, \upsilon+1\}}, (h_{2,i}, \tilde{h}_{2,i})_{i \in [2,n]})$;
**Argument generation** $\mathcal{P}_{\mathsf{sft}}(\mathsf{crs}; (A, \tilde{A}, B, \hat{B}, \tilde{B}), (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b))$**:**
Set $(\pi, \tilde{\pi}) \leftarrow (g_{2,\upsilon+1}^{r_a} \cdot g_{2,\upsilon}^{-r_b} \cdot g_{2,\lambda_1}^{-b_1} \cdot \prod_{i=2}^n h_{2,i}^{b_i}, \tilde{g}_{2,\upsilon+1}^{r_a} \cdot \tilde{g}_{2,\upsilon}^{-r_b} \cdot \tilde{g}_{2,\lambda_1}^{-b_1} \cdot \prod_{i=2}^n \tilde{h}_{2,i}^{b_i})$;
Return $\pi^{\mathsf{sft}} \leftarrow (\pi, \tilde{\pi}) \in \mathbb{G}_2^2$;
**Verification** $\mathcal{V}_{\mathsf{sft}}(\mathsf{crs}; (A, \tilde{A}, B, \hat{B}, \tilde{B}), \pi^{\mathsf{sft}})$**:** if $\hat{e}(A, g_{2,1})/\hat{e}(B, g_2) = \hat{e}(g_1, \pi)$ and $\hat{e}(g_1, \tilde{\pi}) = \hat{e}(\tilde{g}_1, \pi)$ then $\mathcal{V}_{\mathsf{sft}}$ accepts else $\mathcal{V}_{\mathsf{sft}}$ rejects;

**Protocol 4:** New shift argument $\mathsf{shift}([\![(A, \tilde{A})]\!]) = [\![(B, \tilde{B})]\!]$

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

## 5 Shift And Rotation Arguments

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

$$a_i = \begin{cases} b_{i+1} \ , & i \in [n-1] \ , \\ 0 \ , & i = n \ , \end{cases} \qquad \text{resp., } a_i = \begin{cases} b_{i+1} \ , & i \in [n-1] \ , \\ b_1 \ , & i = n \ . \end{cases}$$

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

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

Let $\log_{g_1} A = r_a \sigma^\upsilon + \sum_{i=1}^n a_i \sigma^{\lambda_i}$ and $\log_{g_1} B = r_b \sigma^\upsilon + \sum_{i=1}^n b_i \sigma^{\lambda_i}$. We replace $\sigma$ with a formal variable $X$. Then, if the prover is honest (full derivation of this is given in the proof of Thm. 5), then $F(X) := X \cdot \log_{g_1} A - \log_{g_1} B = \sum_{i=2}^n b_i (X^{\lambda_{i-1}+1} - X^{\lambda_i}) - b_1 X^{\lambda_1} + r_a X^{\upsilon+1} - r_b X^\upsilon$. Thus, one can verify that $A$ is a shift of $B$ by just checking that $\hat{e}(A, g_2^\sigma)/\hat{e}(B, g_2) = \hat{e}(g_1, \pi)$, where $\pi = g_2^{F(\sigma)}$ is defined as in Prot. 4.

As seen from the 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 $\lambda_n + 1 = \lambda_1$ then some of the monomials of $F(X)$ will collapse, and the security proof will not go through.

**Theorem 5 (Security of the shift argument).** *Let $\Lambda = (\lambda_1, \dots, \lambda_n)$ be a tuple of integers, such that $\lambda_i + 1 < \lambda_{i+1}$ and $\lambda_i = \mathrm{poly}(\kappa)$. Let $\upsilon > \lambda_n + 1$. Let $\Gamma$ be the $(\Lambda, \upsilon)$ commitment scheme in group $\mathbb{G}_1$.*
*(1) Prot. 4 is perfectly complete and perfectly witness-indistinguishable.*
*(2) Let*

$$\Phi_{\mathsf{sft}} := \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n \ .$$

*If $\mathcal{G}_{\mathsf{bp}}$ is $\Phi_{\mathsf{sft}}$-PSDL secure, then a NUPPT adversary against Prot. 4 has negligible chance, given a correctly formed CRS $\mathsf{crs}$ as an input, of outputting $inp^{\mathsf{sft}} \leftarrow (A, \tilde{A}, B, \tilde{B})$ and an accepting argument $\pi^{\mathsf{sft}} \leftarrow (\pi, \tilde{\pi})$ together with a witness $w^{\mathsf{sft}} \leftarrow (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, (f_\phi^*)_{\phi \in \Phi_{\mathsf{sft}}})$, such that*
*(i) $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, $r_a, r_b \in \mathbb{Z}_p$, and $f_\phi^* \in \mathbb{Z}_p$ for $\phi \in \Phi_{\mathsf{sft}}$,*
*(ii) $(A, \tilde{A}) = \mathcal{C}om(\widetilde{\mathsf{ck}}; \boldsymbol{a}; r_a)$, $(B, \tilde{B}) = \mathcal{C}om(\widetilde{\mathsf{ck}}; \boldsymbol{b}; r_b)$,*
*(iii) $\log_{g_2} \pi = \log_{\tilde{g}_2} \tilde{\pi} = \sum_{\phi \in \Phi_{\mathsf{sft}}} f_\phi^* \cdot \phi(\sigma)$, and*

*(iv)* $(a_n, a_{n-1}, \dots, a_1) \neq (0, b_n, \dots, b_2)$.
*The reduction time is dominated by the time it takes to factor a degree-$(\upsilon + 1)$ polynomial in $\mathbb{Z}_p[X]$.*

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

**Theorem 6 (Efficiency of shift argument).** *Let $\Lambda$ and $\upsilon$ be as defined in Thm. 5. Let $\beta \leftarrow ||\boldsymbol{b}||_\infty$, $\beta < p$. Assume $n > \log_2 \beta$. The communication (argument size) of Prot. 4 is 2 elements from $\mathbb{G}_2$. The prover's computational complexity is dominated by $\Theta(n)$ multiplications in $\mathbb{Z}_p$ and $(2 + o(1)) \cdot \frac{n \log_2 \beta}{\log_2 n} + O(n)$ bilinear-group multiplications. The verifier's computational complexity is dominated by 5 bilinear pairings. The CRS consists of $\Theta(n)$ group elements.*

*Proof.* By using Pippenger's algorithm, the prover computes two multi-exponentiations in $L(2, n, \beta) = 2 \log_2 \beta + (1 + o(1)) \cdot \frac{2n \log_2(\beta+1)}{\log_2(2n \log_2(\beta+1))} + O(n) = (2 + o(1)) \cdot \frac{n \log_2 \beta}{\log_2 n} + O(n)$ bilinear-group multiplications. Other claims are straightforward. □

**Rotation Argument.** In the rotation argument, $F(X) = (a_n - b_1)X^{\lambda_n+1} + \sum_{i=2}^n (a_{i-1} - b_i)X^{\lambda_{i-1}+1} + b_1(X^{\lambda_n+1} - X^{\lambda_1}) + \sum_{i=2}^n b_i(X^{\lambda_{i-1}+1} - X^{\lambda_i}) + r_a X^{\upsilon+1} - r_b X^\upsilon$. Thus, in the case $\Phi$ is different, $\Phi_{\mathsf{rot}} = \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_n+1} - X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n$. Given this modification, one can construct a rotation argument that is very similar to Prot. 4.

## 6 Applications

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

**Definition 3 (Scan argument).** *In a* scan argument*, the prover aims to convince the verifier that given two commitments $A$ and $B$, he knows how to open them as $A = \mathcal{C}om(\mathsf{ck}; \boldsymbol{a}; r_a)$ and $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that $b_i = \sum_{j>i} a_j$.*

A scan argument $\pi^{scan}$ is just equal to a shift argument $\mathsf{shift}([\![B]\!]) = [\![A \cdot B]\!]$, which 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 Subset Sum Argument

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

**Definition 4 (Subset sum argument).** *In a* subset sum argument*, the prover aims to convince the verifier that given $\mathcal{S} = (s_1, \dots, s_n) \subseteq \mathbb{Z}_p$ and a commitment $B$, he knows how to open it as $B = \mathcal{C}om(\mathsf{ck}; \boldsymbol{b}; r_b)$, such that $\boldsymbol{b}$ is non-zero and has Boolean elements, and $\sum_{i=0}^{n-1} a_i s_i = 0$.*

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

During the new subset sum argument, both parties can compute a commitment $S$ to $\boldsymbol{s}$. The prover commits to a Boolean vector $\boldsymbol{b}$. He computes a commitment $C$ to a vector $\boldsymbol{c}$, such that $c_i = b_i s_i$. He computes a commitment $D$ to the *scan* [Ble90] $\boldsymbol{d}$ of vector $\boldsymbol{c}$. That is, $d_i = \sum_{j>i} c_j$, and in particular, $d_1 = \sum_{j>1} c_i$ and $c_1 + d_1 = \sum_{j\geq 1} c_j$.

The prover computes the subset sum argument as in Alg. 1. The subargument $\pi_2$ is computed as in Alg. 2.

Here, $\pi_5$ is computed by using the restriction argument from [Gro10], which adds linear number of elements to CRS, but has a constant complexity otherwise.

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 $\boldsymbol{b} \neq \boldsymbol{0}$;
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 that the first coordinate of $\boldsymbol{c} + \boldsymbol{d}$ is 0;
The subset sum argument is equal to $(B, C, D, \pi_1, \dots, \pi_5)$;

**Algorithm 1:** Subset sum argument

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

**Algorithm 2:** Argument $\pi_2$

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. G for a proof.

The resulting subset sum argument is arguably simpler than the Circuit-SAT arguments of [Gro10,Lip12] that consists of (at least) 7 product and permutation arguments. Moreover, instead of the product and permutation arguments it only uses product and a more efficient shift argument (zero argument is trivial).

### 6.2 Improved Range Argument

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

Therefore, we can replace the product and permutation arguments from [Lip12] with the product and shift arguments (or with the product and scan arguments) from the current paper. Thus, 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 show that the range argument is computationally sound, one has to assume that the product argument and the shift argument are weakly sound (and that the PKE assumption holds).

Moreover, the use of new basic arguments will decrease the number of $\mathbb{Z}_p$-multiplications — except the cost of computing the multi-exponentiations — in the main range argument from $\Theta(n^2 n_v)$, where $n_v \approx \log_2 u$, to $\Theta(r_3^{-1}(n) \cdot \log r_3^{-1}(n) \cdot n_v) = o(\log H \cdot 2^{2\sqrt{2 \log_2 \log_u H}} \cdot \log \log_u H)$. By using Pippenger's multi-exponentiation algorithm [Pip80], we get the cost of multi-exponentiation down to $(2 + o(1)) \cdot 2r_3^{-1}(n) \log_2(un)/\log_2 r_3^{-1}(n)$ multiplications in bilinear groups. The communication will decrease by $4 + 2 + 3 = 9$ group elements, due to the replacement of the permutation argument with the shift argument (minus 4), having one less product argument (minus 2), and also because one needs to commit to one less element ($(C_{\mathsf{rot}}, \hat{C}_{\mathsf{rot}}, \tilde{C}_{\mathsf{rot}})$ in [CLZ12], minus 3). The verifier also has to perform $7 + 5 + 4 = 16$ less pairings, due to the replacement of the permutation argument with the shift argument (minus 7) and one less product argument (minus 5). Also, it is not necessary anymore to verify the correctness of $(C_{\mathsf{rot}}, \hat{C}_{\mathsf{rot}}, \tilde{C}_{\mathsf{rot}})$ (minus 4). One can analogously compute the verifier's computational complexity, see Tbl. 2.

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

[3] The range argument from [CLZ12] only uses the permutation argument with one fixed permutation (rotation), and thus the value $(T^*, \hat{T}^*, T_2^*)$, that corresponds to this concrete permutation, can be put to the CRS. After this small modification, the

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

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

### 6.3 Decision Knapsack Argument

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

**Definition 5 (Decision knapsack problem).** *In a* decision knapsack problem *one has to decide, given a set $\mathcal{S}$, integers $W$ and $B$, and a benefit value $b_i$ and weight $w_i$ of every item of $\mathcal{S}$, whether there exists a subset $\mathcal{T} \subseteq \mathcal{S}$, such that $\sum_{i \in \mathcal{T}} w_i \leq W$ and $\sum_{i \in \mathcal{T}} b_i \geq B$.*

It is well-known that the decision knapsack problem is NP-complete, see [BCJ11,DDKS12] for the best known (exponential-time) algorithms. One can obviously combine a version of the subset sum argument of the current section with the range argument of Sect. 6.2 to construct a decision knapsack argument, where the prover convinces the verifier that he knows such a subset $\mathcal{T}$. See Alg. 3.

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

**Algorithm 3:** The decision knapsack argument

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. I. 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.

**Acknowledgments.** This work was partially done while the third author was working at the University of Tartu, Estonia. The first two authors were supported by Estonian Research Council, and European Union through the European Regional Development Fund.

verifier's computational complexity actually does not increase compared to what was claimed in [CLZ12]. Since [CLZ12] itself did not mention this, we consider it to be an additional small contribution of the current paper.

## References


BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data. Technical Report 2012/095, International Association for Cryptologic Research, April 3, 2012. Available at http://eprint.iacr.org/2012/095.

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

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

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

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

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

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

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

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

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

DDKS12. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. In Safavi-Naini and Canetti [SC12], pages 719–740.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SC12. Rei Safavi-Naini and Ran Canetti, editors. *CRYPTO 2012*, volume 7417 of *LNCS*, Santa Barbara, California, USA, August 19–23, 2012. Springer, Heidelberg.

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

Wat12. Brent Waters. Functional Encryption for Regular Languages. In Safavi-Naini and Canetti [SC12], pages 218–235.

## A Proof of Thm. 1

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

Assume that the adversary can find a collision $y = y^*$ in group $\mathbb{G}_1$. Then it must be the case that $y = \prod_{\phi_\ell \in \{1\} \cup \Phi} g_1^{a_\ell \phi_\ell(\sigma)}$ and $y^* = \prod_{\ell \in \{0\} \cup \Lambda} g_1^{a_\ell^* \phi_\ell(\sigma)}$ for some known values of $a_\ell$ and $a_\ell^*$. But then also $\sum_{\ell \in \{0\} \cup \Lambda} (a_\ell - a_\ell^*) \phi_\ell(\sigma) \equiv 0 \pmod{p}$. Since the adversary does not know the actual representations of the group elements, it will perform the same group operations independently of $\sigma$. 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 the adversary works in polynomial time $\tau = \mathrm{poly}(\kappa)$, 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 adversary requires time $\Omega(\sqrt{p/d})$ to produce one collision.

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

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

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

## B Proof of Thm. 2

*Proof.* PERFECT HIDING: follows from the fact that the output of $\mathcal{C}\mathsf{om}$ is a random element of $\mathbb{G}_1$. COMPUTATIONAL BINDING: Assume that $\mathcal{A}_{\mathcal{C}\mathsf{om}}$ is an adversary that can break the binding property with some non-negligible probability. We construct the following adversary $\mathcal{A}_{pdl}$, see Prot. 4, against the $\Phi_\Gamma$-PDL assumption in $\mathbb{G}_1$ that works with the same probability. Here, $\mathcal{C}$ is the challenger of the PDL game.

Let us assume that on step 1, $\mathcal{A}_{\mathcal{C}\mathsf{om}}$ is successful with probability $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}_{\mathcal{C}\mathsf{om}}}(\Gamma)$. Thus, with probability $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}_{\mathcal{C}\mathsf{om}}}(\Gamma)$, $(\boldsymbol{a}, r_a) \neq (\boldsymbol{b}, r_b)$ and $g_1^{r_a \sigma^\upsilon} \cdot \prod_{i \in [n]} g_1^{a_i \sigma^{\lambda_i}} = g_1^{r_b \sigma^\upsilon} \cdot \prod_{i \in [n]} g_1^{b_i \sigma^{\lambda_i}}$. But then

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

and thus

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

or equivalently,

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

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

Thus, the adversary has generated a non-trivial degree-$(\upsilon - \lambda_1)$ polynomial $f(X)$ such that $f(\sigma) \equiv 0 \pmod{p}$. Therefore, $\mathcal{A}_{pdl}$ can use polynomial factorization to find all roots of $\delta$, and one of those roots must

```
  𝒞 sets parm ← 𝒢_bp(1^κ), g_1 ← 𝔾_1 \ {1}, and σ ← ℤ_p;
  𝒞 sends (parm; (g_1^{σ^ℓ})_{ℓ∈{υ}∪Λ}) to 𝒜_pdl;
  𝒜_pdl sets α̂* ← ℤ_p;
  𝒜_pdl sets ck ← (parm; (g_1^{σ^ℓ}, g_1^{α̂* σ^ℓ})_{ℓ∈Λ}, g_1^{σ^υ}, g_1^{α̂* σ^υ});
1 𝒜_pdl obtains (a, r_a, b, r_b) ← 𝒜_com(ck);
  if a ∉ ℤ_p^n ∨ b ∉ ℤ_p^n ∨ r_a ∉ ℤ_p ∨ r_b ∉ ℤ_p ∨ (a, r_a) = (b, r_b) ∨ Com(ck; a, r_a) ≠ Com(ck; b, r_b) then 𝒜_pdl aborts;
  ;
  else
2 |  𝒜_pdl sets δ(X) ← (r_a − r_b)X^{υ−λ_1} + Σ_{i=1}^n (a_i − b_i)X^{λ_i−λ_1}.
  |  𝒜_pdl sets (t_1, ..., t_{υ−λ_1+1}) ← PolyFact(δ);
3 |  𝒜_pdl finds by an exhaustive search a root σ_0 ∈ {t_1, ..., t_{υ−λ_1+1}}, such that g_1^{σ^{λ_1}} = g_1^{σ_0^{λ_1}};
  |  𝒜_pdl returns σ ← σ_0 to the challenger;
  end
```

**Algorithm 4:** Adversary in Thm. 2

be equal to $\sigma$. On step 3, $\mathcal{A}_{pdl}$ finds which root is equal to $\sigma$ 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 $\varPhi_{\varGamma}$-PDL assumption) with probability $\mathsf{Succ}^{\mathsf{binding}}_{\mathcal{A}_{\mathcal{C}om}}(\varGamma)$. Finally, the execution time of $\mathcal{A}_{pdl}$ is clearly dominated by the execution time of $\mathcal{A}_{\mathcal{C}om}$ and the time to factor $\delta$.

EXTRACTABILITY: By the $\varPhi_{\varGamma}$-PKE assumption in group $\mathbb{G}_1$, for every committer $\mathcal{A}$ there exists an extractor $X_{\mathcal{A}}$ that can open the commitment in group $\mathbb{G}_1$, given access to $\mathcal{A}$'s inputs and random tape. Since the commitment scheme is computationally binding, then the extracted opening has to be the same that $\mathcal{A}$ used. □

## C Proof of Lem. 1

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

## D Proof of Lem. 2

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

## E Proof of Thm. 3

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

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

**Algorithm 5:** Construction of $\hat{\mathcal{A}}$ in the security reduction of Thm. 3

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

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

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

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

and

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

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 $c_i = a_i b_i$, and $F_\pi(X)$ is the argument polynomial.

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

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

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

WEAKER VERSION OF SOUNDNESS. Assume that $\mathcal{A}_\times$ is an adversary that can break the last statement of the theorem. We construct an adversary $\hat{\mathcal{A}}$ against the $\Phi_\times$-PSDL assumption, see Prot. 5. 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 probability $1 - \mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$. Otherwise, with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$, $inp^\times = (A, \hat{A}, B, \hat{B}, B_2, C, \hat{C})$ and $w^\times = (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, \boldsymbol{c}, r_c, (f^*_\ell)_{\ell \in \hat{\Lambda}})$, such that the conditions (i–iv) hold.

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

Thus, $\delta(X)$ is a non-zero degree-$(2\upsilon - 2\lambda_1)$ polynomial, such that $\delta(\sigma) = \sum_{\ell \in (\hat{\Lambda} \cup (2 \cdot \Lambda)) - 2\lambda_1} \delta_\ell \sigma^\ell = 0$. Therefore, $\hat{\mathcal{A}}$ can use polynomial factorization to find all $\leq 2(\upsilon - \lambda_1)$ roots of $\delta$, where one of the found 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. Thus, clearly $\hat{\mathcal{A}}$ returns the correct value of $\sigma$ (and thus violates the $\Phi_\times$-PSDL assumption) with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_\times}(\Pi_\times)$. Finally, the execution time of $\hat{\mathcal{A}}$ is clearly dominated by the execution time of $\mathcal{A}_\times$ and the time to factor $\delta$. □

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

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

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

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

If the prover is honest, then $a_i = b_{i+1}$ for $i \in [n-1]$ and $a_n = 0$, and thus $F(X) = -b_1 X^{\lambda_1} + \sum_{i=2}^{n} b_i (X^{\lambda_{i-1}+1} - X^{\lambda_i}) + r_a X^{\upsilon+1} - r_b X^\upsilon$ is spanned by $\{\phi(X)\}_{\phi \in \Phi_{\mathsf{sft}}}$. Defining $\pi$ as in Prot. 4, we see that the second verification holds.

WEAKER VERSION OF SOUNDNESS. Assume that $\mathcal{A}_{\mathsf{sft}}$ is an adversary that can break the last statement of the theorem. We construct an adversary $\tilde{\mathcal{A}}$ against the $\Phi_{\mathsf{sft}}$-PSDL assumption, see Prot. 6. Here, $\mathcal{C}$ is the challenger of the PSDL game, and $\Phi^\pi := \{X^\upsilon, X^{\upsilon+1}\} \cup \{X^{\lambda_i}, X^{\lambda_i + 1}\}_{i=1}^n$ 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, $\mathcal{A}_{\mathsf{sft}}$ gets a correct input, and succeeds with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$. Clearly, $\tilde{\mathcal{A}}$ aborts with probability $1 - \mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$.

Otherwise, with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$, $inp^{\mathsf{sft}} = (A, \tilde{A}, B, \tilde{B})$ and $w^{\mathsf{sft}} = (\boldsymbol{a}, r_a, \boldsymbol{b}, r_b, (f^*_\phi)_{\phi \in \Phi_{\mathsf{sft}}})$, such that the conditions (i–iv) hold. In particular, $f(X) = F(X)$ in Eq. (3), and $f^*(X) = f^*_{X^{\lambda_1}} \cdot X^{\lambda_1} + \sum_{i=2}^{n} f_{X^{\lambda_{i-1}+1} - X^{\lambda_i}} (X^{\lambda_{i-1}+1} - X^{\lambda_i}) + f^*_{X^{\upsilon+1}} X^{\upsilon+1} + f^*_{X^\upsilon} X^\upsilon$.

Since $(a_n, a_{n-1}, \ldots, a_1) \neq (0, b_n, \ldots, b_2)$, $f(X)$ has at least one more non-zero monomial, either of type $a_n X^{\lambda_n + 1}$ or of type $(a_{i-1} - b_i) X^{\lambda_{i-1}+1}$, than $f^*(X)$. Since $X^{\lambda_{i-1}+1}$ cannot be represented as a linear combination of polynomials from $\Phi_{\mathsf{sft}}$, $f(X)$ 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, $\tilde{\mathcal{A}}$

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

**Algorithm 6:** Construction of $\tilde{\mathcal{A}}$ in the security reduction of Thm. 5

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 $\Phi_{\mathsf{sft}}$-PSDL assumption) with probability $\mathsf{Succ}^{\mathsf{sound}}_{\mathcal{A}_{\mathsf{sft}}}(\Pi_{\mathsf{sft}})$. Finally, the execution time of $\tilde{\mathcal{A}}$ is clearly dominated by the execution time of $\mathcal{A}_{\mathsf{sft}}$ and the time to factor $\delta$. □

## G Subset Sum

Recall $\Phi_\Gamma = (\{X^\upsilon\} \cup (X^{\lambda_i})_{i=1}^n)$. We will also need $\Phi_{res}$-PKE assumptions to guarantee soundness of the restriction argument from [Gro10], where $\Phi_{res}$ depends concretely on the restricted coordinates. Since $\Phi_{res} \subseteq \Phi_\Gamma$ (for example, in the following theorem, $\Phi_{res} := \{X^\upsilon\} \cup \{X^{\lambda_i}\}_{i=2}^n$), we will not have to explicitly mention it.

**Theorem 7.** *Let $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ 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 $\Phi := \Phi_\times \cup \Phi_{\mathsf{sft}} = \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n \cup \{X^\ell\}_{\ell \in \hat{\Lambda}}$. Let $\upsilon > \max(2\lambda_n - \lambda_1, \lambda_n + 1)$ be linear in $\lambda_n - \lambda_1$. The subset sum protocol described by Alg. 1 is perfectly complete and perfectly zero-knowledge. Also, $\mathcal{G}_{\mathsf{bp}}$ 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 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 shift 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 a non-uniform PPT adversary that produces commitments $B, C, D$ and an accepting NIZK argument $(B, C, D, \pi_1, \ldots, \pi_5)$. 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 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 $B, C$, 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 $\Phi$-PSDL secure 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 $d_i = \sum_{j>i} c_j$.

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

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

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 $\Phi_{res}$-PKE assumption that guarantees the soundness of the restriction argument (Theorem 1 and Theorem 2 of [Gro10]), 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 $\psi$. ◻

## H Range Argument

**Theorem 8.** *Let $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ 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 $\Phi := \Phi_\times \cup \Phi_{\mathsf{sft}} = \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n \cup \{X^\ell\}_{\ell \in \hat{\Lambda}}$. Let $\upsilon > \max(2\lambda_n - \lambda_1, \lambda_n + 1)$ be linear in $\lambda_n - \lambda_1$. The range argument of [CLZ12], when modified as in Sect. 6.2, 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 proof is computationally sound.*

*Proof.* Similar to [CLZ12]. ◻

## I Decision Knapsack

**Theorem 9.** *Let $\Gamma = (\mathcal{G}com, \mathcal{C}om, \mathcal{G}com_{td}, \mathcal{C}om_{td}, \mathcal{O}pen_{td})$ 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 $\Phi := \Phi_\times \cup \Phi_{\mathsf{sft}} = \{X^\upsilon, X^{\upsilon+1}, X^{\lambda_1}\} \cup \{X^{\lambda_{i-1}+1} - X^{\lambda_i}\}_{i=2}^n \cup \{X^\ell\}_{\ell \in \hat{\Lambda}}$. 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. 3 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 shift 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 $\varPhi$-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. (That is, the extractor can open the inputs to the arguments to values that satisfy required restrictions.) By Thm. 8, the range argument is computationally sound.

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 $T, W_T, A, C, B_T, D, E$, 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. 8), we have that the following relations hold:

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. □