Efficient Zero-Knowledge Arguments in Discrete Logarithm Setting: Sublogarithmic Proof or Sublinear Verifier

Sungwook Kim$^{1[0000-0003-4789-3347]}$, Hyeonbum Lee$^{2[0000-0003-0435-4394]}$, and Jae Hong Seo$^{2\star[0000-0003-0547-5702]}$

^{1} Department of Information Security, Seoul Women's University, Republic of Korea
`kim.sungwook@swu.ac.kr`

^{2} Department of Mathematics & Research Institute for Natural Sciences, Hanyang University, Seoul 04763, Republic of Korea
`{leehb3706, jaehongseo}@hanyang.ac.kr`

Abstract. We propose three interactive zero-knowledge arguments for arithmetic circuit of size N in the common random string model, which can be converted to be non-interactive by Fiat-Shamir heuristics in the random oracle model. First argument features $O(\sqrt{\log N})$ communication and round complexities and $O(N)$ computational complexity for the verifier. Second argument features $O(\log N)$ communication and $O(\sqrt{N})$ computational complexity for the verifier. Third argument features $O(\log N)$ communication and $O(\sqrt{N} \log N)$ computational complexity for the verifier. Contrary to first and second arguments, the third argument is free of reliance on pairing-friendly elliptic curves. The soundness of three arguments is proven under the standard discrete logarithm and/or the double pairing assumption, which is at least as reliable as the decisional Diffie-Hellman assumption.

1 Introduction

A zero-knowledge (ZK) argument is a protocol between two parties, the prover and the verifier, such that the prover can convince the verifier that a particular statement is true without revealing anything else about the statement itself. ZK arguments have been used in numerous applications such as verifiable outsourced computation, anonymous credentials, and cryptocurrencies.

Our goal is to build an efficient ZK argument for arithmetic circuit (AC) in the common random string model that is sound under well-established standard assumptions, such as the discrete logarithm (DL) assumption: Compared to q-type strong assumptions such as q-DLOG [38, 27], the standard assumptions will provide strong security guarantees as well as a good efficiency with smaller group size due to Cheon's attack on q-type assumptions [20]. To this end, we propose three inner-product (IP) arguments with the same properties (standard assumption, common random string model), where an IP argument is

* corresponding author

a proof system that convinces the verifier of an inner-product relation between committed integer vectors. Then, we can apply well-established reductions from IP argument to ZK argument for AC [13, 17, 46, 18].

The first sublinear ZK argument for AC solely based on the hardness of the DL problem is due to Groth [29] and improved by Seo [44]. These works feature constant round complexity as well. Groth [31] gives a ZK argument with a cubic root communication complexity using pairing-based two-tiered homomorphic commitment scheme whose binding property is based on the double pairing (DPair) assumption [1]. The first logarithmic ZK argument for AC solely from the DL assumption is due to Bootle, Cerulli, Chaidos, Groth, and Petit [13] and improved by Bünz, Bootle, Boneh, Poelstra, Wuille, and Maxwell [17], which is called Bulletproofs. Hoffmann, Klooß, and Rupp [34] revisited and improved Bulletproofs by showing that it can cover systems of quadratic equations, of which rank-1 constraint systems is a special case. These logarithmic ZK argument systems [13, 17, 34] have linear verifiers. Other DL-based ZK argument systems with different asymptotic performance, in particular sublinear verifier, have been proposed. e.g., Hyrax [46] and Spartan [45]. Recently, Bünz, Maller, Mishra, Tyagi, and Vesely [19] achieved a logarithmic ZK argument with a sublinear verifier under the DPair assumption.

Focusing on specific languages, there are more researches achieving logarithmic communication complexity [3, 33] prior to Bulletproofs. Logarithmic communication complexity in these works is attained with relatively large round complexity, compared to [29, 44].

Relying on the non-standard but reliable assumptions, there exists a ZK argument system with better asymptotic performance due to Bünz, Fisch, and Szepieniec [18] that achieve logarithmic communications and logarithmic verifier simultaneously, but it relies on a rather stronger assumption such as the strong RSA assumption and the adaptive root assumption. A lot of important research for succinct non-interactive argument (SNARG) [30, 37, 10, 28, 11, 40, 6, 9, 33, 32, 38, 27, 47, 21] have been proposed on the top of bilinear groups, where an argument consists of a constant number of group elements. However, the soundness of these works relies on non-falsifiable knowledge extractor assumptions and/or the structured reference string (SRS) that requires a trusted setup, which is not required in the aforementioned DL-based protocols. There is another important line of works [5, 7, 22, 48] for SNARG without using pairings, but based on interactive oracle proofs [8]. These works are strong candidates for post-quantum ZK arguments and simultaneously minimizing communication cost and verifier computation. However, their communication cost is proportional to $\log^2 N$ for the circuit size N, which is larger than that of the DL based approach [13, 17].

Our Results. We propose three IP arguments between two integer vectors of length N in the common random string model. We refer to [13, 17, 46, 18] or Section 6 for a constant round reduction from ZK arguments for AC of size N with fan-in 2 gates to IP arguments. We summarize our results as follows.

1. We propose the first IP argument with *sublogarithmic communication*. We prove its soundness under the DL assumption and the DPair assumption.

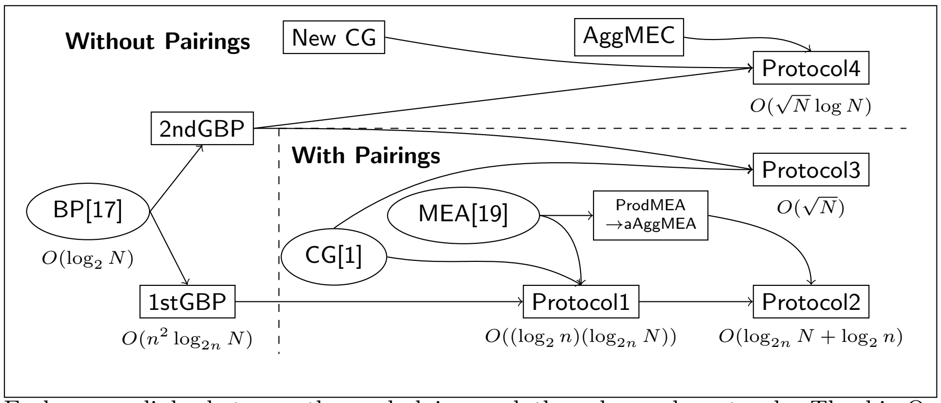
Scheme	Communication	$\mathcal{P}$'s comp.	$\mathcal{V}$'s comp.	Assump.
Groth [29] & Seo [44]	$O(\sqrt{N})\mathbb{G}_1$	$O(N)\tau_1$	$O(N)\tau_1$	DL
Groth [31]	$O(\sqrt[3]{N})\mathbb{G}_1$	$O(N)\tau_1$	$O(\sqrt[3]{N})\tau_1$	DPair
BP [13,17] & HKR [34]	$O(\log N)\mathbb{G}_1$	$O(N)\tau_1$	$O(N)\tau_1$	DL
Hyrax [46]	$O(\sqrt{w} + d\log N)\mathbb{G}_1$	$O(N\log N)\tau_1$	$O(\sqrt{w} + d\log N)\tau_1$	DL
Spartan DL [45]	$O(\sqrt{N})\mathbb{G}_1$	$O(N)\tau_1$	$O(\sqrt{N})\tau_1$	DL
BMMTV [19]	$O(\log N)\mathbb{G}_t$	$O(N)\tau_1$	$O(\sqrt{N})\tau_2$	DPair
Supersonic [18]	$O(\log N)\mathbb{G}_U$	$O(N\log N)\mathsf{u}$	$O(\log N)\mathsf{u}$	UOGroup
Spartan CL [45]	$O(\log^2 N)\mathbb{G}_U$	$O(N\log N)\mathsf{u}$	$O(\log^2 N)\mathsf{u}$	UOGroup
Ligero [2]	$O(\sqrt{N})\mathbb{H}$	$O(N\log N)\mathsf{h}$	$O(N)\mathsf{h}$	CR hash
STARK [5]	$O(\log^2 N)\mathbb{H}$	$O(N\log^2 N)\mathsf{h}$	$O(\log^2 N)\mathsf{h}$	CR hash
Aurora [7]	$O(\log^2 N)\mathbb{H}$	$O(N\log N)\mathsf{h}$	$O(N)\mathsf{h}$	CR hash
Fractal [22]	$O(\log^2 N)\mathbb{H}$	$O(N\log N)\mathsf{h}$	$O(\log^2 N)\mathsf{h}$	CR hash
Virgo [48]	$O(d\log N)\mathbb{H}$	$O(N\log N)\mathsf{h}$	$O(d\log N)\mathsf{h}$	CR hash
BCGGHJ [14]	$O(\sqrt{N})\mathbb{H}$	$O(N)\mathsf{m}$	$O(N)\mathsf{m}$	CR hash
BCL [15]	$\mathsf{polylog}(N)\mathbb{H}$	$O(N)\mathsf{m}$	$\mathsf{polylog}(N)\mathsf{m}$	CR hash
Our IP arguments + Section 6				
Protocol2 (Section 3.2)	$O(\sqrt{\log N})\mathbb{G}_t$	$O(N2^{\sqrt{\log N}})\tau_1$	$O(N)\tau_1$	DL†&DPair
Protocol3 (Section 4.3)	$O(\log N)\mathbb{G}_t$	$O(N)\tau_1$	$O(\sqrt{N})\tau_2$	DL†
Protocol4 (Section 5.3)	$O(\log N)\mathbb{G}_q$	$O(N)\tau_p$	$O(\sqrt{N}\log N)\tau_q$	DL

Table 1. Comparison for transparent ZK arguments
N: circuit size, d: circuit depth, w: input size, $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t)$: bilinear groups, $(\mathbb{G}_p, \mathbb{G}_q)$: elliptic curve groups of order p and q, $\mathbb{G}_U$: group of unknown order, $\mathbb{H}$: hash function, m, p, h, τ_i, u: operation of field, pairing, hash, $\mathbb{G}_i$, $\mathbb{G}_U$,
UOGroup: unknown-order group (strong RSA & adaptive root assumptions), CR hash: collision-resistant hashes, DL†: DL assumption over pairing-friendly elliptic curves
All arguments in the table are public coin (Definition 1), so that they achieve non-interactivity in the random oracle model using the Fiat-Shamir heuristic [25].

2. We present the first IP argument with $O(\log N)$ communication and $O(\sqrt{N})$ verifier computation such that its soundness is *based on the DL assumption.*
3. We introduce a novel method to achieve the IP argument with a similar performance to the second argument, especially *without the reliance of pairings.*

We provide a comparison for transparent ZK arguments in Table 1[3]. Note that there are more efficient arguments in the DL setting [9, 36, 38, 27, 21, 24] if we rely on a trusted setup or non-standard, non-falsifiable assumptions.

[3] We often use a terminology ‘transparent’ in the meaning of ’without trusted setup’.



Each arrow links between the underlying and the advanced protocols. The big-O notation under each protocol indicates communication complexity, except for Protocol3 and Protocol4 that indicate verifier's computational complexity. The oval nodes indicate known results; BP: Bulletproofs [13, 17], MEA: multi-exponentiation argument [19], CG: Commitment to group elements [1]. The rectangle nodes indicate the proposed protocols; New CG: Commitment to Elliptic Curve Points, 1stGBP & 2stGBP: generalizations of Bulletproofs, aAggMEA & AggMEC: aggregations of multi-exponentiations & multi-elliptic curve operations. Protocol1: an intermediate protocol. Protocol2: Sublogarithmic IP argument, Protocol3 & Protocol4: Sublinear Verifier IP arguments. N is the dimension of witness vectors. n is a positive integer parameter for 1stGBP, where $n = 1$ implies the original Bulletproofs.

Fig. 1. Overview of Our Approach toward Sublogarithmic Proofs or Sublinear Verifier

Our starting point is Bünz et al.'s Bulletproofs IP argument (BP-IP) [17] that features $O(\log N)$ communication and $O(N)$ computation in the common random string model and is sound under the DL assumption. For shorter proofs or faster verification, we first generalize BP-IP in two different ways. A pictorial overview of our approach is given in Fig. 1.

Sublogarithmic Communications. BP-IP consists of $\log N$ recursive steps such that the prover sends two group elements per each round. The goal of each recursive step is to halve the length of witness. Our first generalization of BP-IP reduces the length of witness one $2n$-th per each recursive round if N is a power of $2n$ for any positive integer n. If need be, one can easily pad the inputs to ensure that the requirement for the format of N holds, like in BP-IP. Then, the recursive steps are finished in $\log_{2n} N$ rounds and the prover sends $2n(2n-1)$ group elements in each round, so that the overall communication cost is $O((\log_{2n} N) \times n^2)$, which becomes minimal when $n = 1$. That is, this generalization has no advantage over BP-IP in terms of communications.

Nevertheless, we observe that the commit-and-prove approach can reduce transmission overhead; the prover can commit to $2n(2n-1)$ group elements instead of sending them all, and then proves that the openings satisfy what the verifier should have checked with the openings. To this end, we use a pairing-based commitment scheme to group elements (e.g., AFGHO [1]). This process of committing and proving can be achieved using a multi-exponentiation argument (e.g., [19]). Unfortunately, this naïve commit-and-prove approach ends up with asymptotically the same proof size as BP-IP since we must prove several multi-exponentiation arguments for every round. We call this protocol $\mathsf{Protocol1}$.

To further reduce the communication cost, we aggregate multiple multi-exponentiation arguments. Although there are well-known aggregating techniques for multiple arguments with homomorphic commitment scheme (e.g., aggregating range proofs [17], linear combinations of protocols [34]), these aggregating techniques are not well applicable to the case including ours such that bases and exponents are distinct for multiple arguments. We try to reduce multiple relations to a single relation by multiplying all relations and then employ a recursive proof technique like BP-IP. However, we find that this strategy does not work well. The detailed explanation about the difficulty we faced is given in Section 3.2. Instead, we devise a novel aggregating technique using newly proposed *augmented aggregated multi-exponentiation argument* $\mathsf{aAggMEA}$ and product argument $\mathsf{ProdMEA}$. The final protocol, called $\mathsf{Protocol2}$, using $\mathsf{aAggMEA}$ and $\mathsf{ProdMEA}$ achieves sublogarithmic communication overhead.

Sublinear Verifier. The soundness of BP-IP is based on the discrete logarithm relation assumption (DLR), which is equivalent to the DL assumption, such that no adversary can find non-trivial relation among *uniformly* chosen group elements. We observe that the uniform condition in sampling group elements is not necessary in the soundness proof of BP-IP, but the hardness of finding non-trivial relation among the CRS is sufficient. From this observation, we first generalize the DLR assumption by removing the uniform condition and then propose and prove that a new assumption with non-uniform distribution holds. More precisely, we combine this generalization with the AFGHO commitments; Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ be a bilinear map, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are source groups and $\mathbb{G}_t$ is the target group. $g_1, \ldots, g_{\sqrt{N}} \in \mathbb{G}_1$ and $H_1, \ldots, H_{\sqrt{N}} \in \mathbb{G}_2$ are uniformly chosen. We prove that no adversary can find a non-trivial vector $(a_{11}, \ldots, a_{\sqrt{N}\sqrt{N}}) \in \mathbb{Z}_p^N$ satisfying $\prod_{i,j=1}^{\sqrt{N}} e(g_i, H_j)^{a_{ij}}$ if the DL assumptions in the source groups hold. That is, $e(g_i, H_j)$'s are not uniformly distributed but hard to find non-trivial relation among them. Therefore, if we set $e(g_i, H_j)$'s as the CRS of BP-IP, then the actual CRS becomes g_i's and H_j's of $2\sqrt{N}$ size while keeping the soundness proof under the DL assumption in the source group.

Nevertheless, a naïve approach using the above idea will keep linear verifier computation in N since we still keep the same verification process as that of BP-IP. We introduce a trick to track verifier's computation with $O(\sqrt{N})$ computation. For example, in the first recursive step of BP-IP, the verifier should update the public parameter $g_1, \ldots, g_N$ to $g_1^x g_{N/2+1}^{x^{-1}}, \ldots, g_{N/2}^x g_N^{x^{-1}}$ for a challenge integer x, which requires $O(N)$ computation. In our setting, the public parameter $e(g_1, H_1), \ldots, e(g_{\sqrt{N}}, H_{\sqrt{N}})$ can be halved to $e(g_1^x g_{\sqrt{N}/2+1}^{x^{-1}}, H_i), \ldots,$ $e(g_{\sqrt{N}/2}^x g_{\sqrt{N}}^{x^{-1}}, H_i)$ for all $i = 1, \ldots, \sqrt{N}$. Therefore, the verifier can track this computation by computing only $g_i^x g_{\sqrt{N}/2+i}^{x^{-1}}$ for $i = 1, \ldots, \sqrt{N}$, which require $O(\sqrt{N})$ exponentiations in $\mathbb{G}_1$. Note that this trick does not increase the prover's overhead, so that we sacrifice neither the other complexities nor assumptions to achieve sublinear verifier. The resulting protocol with sublinear verifier is called Protocol3.

Sublinear Verifier without Pairings. The core of the above second generalization of BP-IP is to employ *two-tiered homomorphic commitment scheme*: Pedersen commitment scheme to integers in the 1st layer + pairing-based AFGHO commitment scheme to group elements in the 2nd layer. We propose another IP argument with sublinear verifier, particularly not relying on pairing-friendly elliptic curves. To circumvent the use of AGFHO scheme, we propose a new two-tiered commitment scheme built on a usual elliptic curve with a mild condition. Although the proposed two-tiered commitment scheme is not homomorphic, we emphasize that it has a similar-but-weakened property, *friendly to proving homomorphic operations* of the underlying mathematical structure, particularly the group law of elliptic curve over finite fields. Second, we show that this weakened property is sufficient to construct an IP argument protocol with sublinear verifier. After replacing pairing-based two-tiered homomorphic commitment scheme with the new commitment scheme, the prover performs the verifier computation, proves the integrity of the computation, and sends the verifier the computation along with a proof. In order to raise efficiency of this approach, we also bring in the aggregation technique used for the protocol with sublogarithmic communications. The resulting protocol without pairings is called Protocol4.

Related Works and Organization. Additional related works that were not mentioned before are provided in the full version [35]. After providing necessary definitions in the next section, we present our first generalization of BP-IP and then reduce its communication overhead by using the aggregation technique in Section 3. We present another generalization that achieves sublinear CRS size and verifier computation in Section 4 (with Pairings) and Section 5 (without Pairings). In Section 6, we extend our IP arguments to ZK arguments for AC.

2 Definitions

Argument System for Relation $\mathcal{R}$. Let $\mathcal{K}$ be the common reference string (CRS) generator that takes the security parameter as input and outputs the CRS σ. In this paper, the CRS consists of randomly generated group elements, so that indeed we are in the common random string model, where an argument consists of two interactive PPT algorithms $(\mathcal{P}, \mathcal{V})$ such that $\mathcal{P}$ and $\mathcal{V}$ are called the prover and the verifier, respectively. The transcript produced by an interaction between $\mathcal{P}$ and $\mathcal{V}$ on inputs x and y is denoted by $tr \leftarrow \langle \mathcal{P}(x), \mathcal{V}(y) \rangle$. Since we are in the common random string model, for the sake of simplicity, we omit the process of running $\mathcal{K}$ and assume the CRS is given as common input to both $\mathcal{P}$

and $\mathcal{V}$. At the end of transcript, the verifier $\mathcal{V}$ outputs b, which is denoted by $\langle \mathcal{P}(x), \mathcal{V}(y) \rangle = b$, where $b = 1$ if $\mathcal{V}$ accepts and $b = 0$ if $\mathcal{V}$ rejects.

Let $\mathcal{R}$ be a polynomial time verifiable ternary relation consisting of tuples of the CRS σ, a statement x, and a witness w. We define a CRS-dependent language L_σ as the set of statements x that have a witness w such that $(\sigma, x, w) \in \mathcal{R}$. That is, $L_\sigma = \{\ x \mid \exists\ w$ satisfying $(\sigma, x, w) \in \mathcal{R}\ \}$. For a ternary relation $\mathcal{R}$, we use the format $\{(\text{common input}; \text{witness}) : \mathcal{R}\}$ to denote the relation $\mathcal{R}$ using specific common input and witness.

Argument of Knowledge. Informally, the argument of knowledge means the argument system satisfying the completeness and the soundness with extractability. Due to space constraints, we provide definitions in the full version [35].

Transparent and Non-interactive Argument in the Random Oracle Model. A protocol in the common random string model can be converted into a protocol without a trusted setup in the random oracle model [4]; if $\mathcal{K}$ outputs random group elements of an elliptic curve group $\mathbb{G}$ of prime order, then the CRS can be replaced with hash values of a small random seed, where the hash function mapping from $\{0,1\}^*$ to $\mathbb{G}$ is modeled as a random oracle as in [12].

Any public coin interactive argument protocol defined in Definition 1 can be converted into a non-interactive one by applying the Fiat-Shamir heuristic [25] in the random oracle model; all $\mathcal{V}$'s challenges can be replaced with hash values of the transcript up to that point.

Definition 1. *An argument $(\mathcal{P}, \mathcal{V})$ is called* public coin *if all $\mathcal{V}$'s challenges are chosen uniformly at random and independently of $\mathcal{P}$'s messages.*

All interactive arguments proposed in this paper can be converted to transparent non-interactive arguments in the random oracle model.

Assumptions Let $\mathcal{G}$ be a group generator such that $\mathcal{G}$ takes 1^λ as input and outputs $(p, \mathbb{G}, g)$, where λ is the security parameter, $\mathbb{G}$ is the description of a group of order p, and g is a generator of G, which is used to sample an element of $\mathbb{G}$ with uniform distribution. Let $negl(\lambda)$ be a negligible function in λ.

Definition 2 (Discrete Logarithm (DL) Assumption). *We say that the group generator $\mathcal{G}$ satisfies the discrete logarithm assumption if for all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

$$\Pr\left[g^a = h \;\middle|\; (p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda),\quad h \xleftarrow{\$} \mathbb{G};\ a \leftarrow \mathcal{A}(p, g, h, \mathbb{G}) \right] < negl(\lambda).$$

Definition 3 (Discrete Logarithm Relation (DLR) Assumption). *We say that the group generator $\mathcal{G}$ satisfies the discrete logarithm relation assumption if for all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

$$\Pr\left[\boldsymbol{a} \neq \boldsymbol{0} \wedge \boldsymbol{g}^{\boldsymbol{a}} = 1_{\mathbb{G}} \;\middle|\; (p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda), \boldsymbol{g} \xleftarrow{\$} \mathbb{G}^n; \boldsymbol{a} \leftarrow \mathcal{A}(p, \mathbb{G}, g, \boldsymbol{g}) \right] < negl(\lambda),$$

where $1_{\mathbb{G}}$ *is the identity of* $\mathbb{G}$.

Although the equivalence between DLR and DL assumptions is well-known, to be self-contained, we provide the complete reductions in the full version [35].

Let $\mathcal{G}_b$ be an asymmetric bilinear group generator such that $\mathcal{G}_b$ takes 1^λ as input and outputs $(p, g, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are the descriptions of distinct cyclic groups of order p of length λ, g and H are generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively, and e is a non-degenerate bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_t$.

Definition 4 (Double Pairing Assumption). *We say that the asymmetric bilinear group generator* $\mathcal{G}_b$ *satisfies the double pairing assumption if for all non-uniform polynomial-time adversaries* $\mathcal{A}$*, the following inequality holds.*

$$\Pr\left[\begin{array}{c} e(g', G) = e(g'', G^a) \\ \wedge\ (g', g'') \neq (1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}) \end{array}\middle|\begin{array}{c} (p, g, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e) \leftarrow \mathcal{G}_b(1^\lambda), \\ G \xleftarrow{\$} \mathbb{G}_2,\ a \xleftarrow{\$} \mathbb{Z}_p; \\ (g', g'') \leftarrow \mathcal{A}((G, G^a), (p, g, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)) \end{array}\right] < negl(\lambda)$$

Abe et al. [1] proved that the double pairing assumption is as reliable as the decisional Diffie-Hellman assumption in $\mathbb{G}_2$.

Groups, Vectors, and Operations. We introduce some notations for succinct description of protocols. $[m]$ denotes a set of continuous integers from 1 to m, $\{1, \ldots, m\}$. For elements in groups $\mathbb{G}_1$ and $\mathbb{G}_2$ obtained by $\mathcal{G}_b$, we separately use lower case letters for $\mathbb{G}_1$ and upper case letters for $\mathbb{G}_2$. A vector is denoted by a bold letter, e,g., $\boldsymbol{g} = (g_1, ..., g_m) \in \mathbb{G}_1^m$ and $\boldsymbol{a} = (a_1, ..., a_m) \in \mathbb{Z}_p^m$. For a vector $\boldsymbol{a} \in \mathbb{Z}_p^m$, its separation to the left half vector $\boldsymbol{a}_1 \in \mathbb{Z}_p^{m/2}$ and the right half vector $\boldsymbol{a}_{-1} \in \mathbb{Z}_p^{m/2}$ is denoted by $\boldsymbol{a} = \boldsymbol{a}_1 \| \boldsymbol{a}_{-1}$. Equivalently, the notation $\|$ is used when sticking two vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_{-1}$ to $\boldsymbol{a}$ and can be sequentially used when sticking several vectors.[4] We use several vector operations denoted as follows.

Component-wise Operations. The component-wise multiplication between several vectors is denoted by $\circ$ e.g., for $\boldsymbol{g}_k = (g_{k,1}, \ldots, g_{k,n}) \in \mathbb{G}_i^n$, $i \in \{1, 2, t\}$, and $k \in [m]$, $\circ_{k\in[m]}\boldsymbol{g}_k = (\prod_{k\in[m]} g_{k,1} \ldots, \prod_{k\in[m]} g_{k,n})$. If $k = 2$, we simply denote it by $\boldsymbol{g}_1 \circ \boldsymbol{g}_2$.

Bilinear Functions & Scalar-Vector Operations.

1. The standard inner-product in $\mathbb{Z}_p^n$ is denoted by $\langle\ ,\ \rangle$ and it satisfies the following bilinearity. $\langle \sum_{k\in[m]} \boldsymbol{a}_k, \sum_{j\in[n]} \boldsymbol{b}_j \rangle = \sum_{k\in[m]} \sum_{j\in[n]} \langle \boldsymbol{a}_k, \boldsymbol{b}_j \rangle \in \mathbb{Z}_p$.
2. For $\boldsymbol{g} = (g_1, \ldots, g_n) \in \mathbb{G}_i^n$, $i \in \{1, 2, t\}$ and $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$, the multi-exponentiation is denoted by $\boldsymbol{g}^{\boldsymbol{a}} := \prod_{k\in[n]} g_k^{a_k} \in \mathbb{G}_i$ and it satisfies the following bilinearity. $(\circ_{k\in[m]}\boldsymbol{g}_k)^{\sum_{j\in[\ell]} \boldsymbol{z}_j} = \prod_{k\in[m]} \prod_{j\in[\ell]} \boldsymbol{g}_k^{\boldsymbol{z}_j} \in \mathbb{G}_i$.

[4] Note that we use the indices $(1, -1)$ instead of $(1, 2)$ since it harmonizes well with the usage of the challenges in Bulletproofs and our generalization of Bulletproofs. e.g., let $\boldsymbol{a} = \boldsymbol{a}_1 \| \boldsymbol{a}_{-1}$ be a witness and x be a challenge, and then $\boldsymbol{a}$ is updated to $\sum_{i=\pm1} \boldsymbol{a}_i x^i$, a witness for the next recursive round.

3. For $\boldsymbol{g} = (g_1, \dots, g_n) \in \mathbb{G}_1^n, \boldsymbol{H} = (H_1, \dots, H_n) \in \mathbb{G}_2^n$, the inner-pairing product is denoted by $\boldsymbol{E}(\boldsymbol{g}, \boldsymbol{H}) := \prod_{k \in [n]} e(g_k, H_k) \in \mathbb{G}_t$ and it satisfies the following bilinearity. $\boldsymbol{E}(\circ_{k\in[m]}\boldsymbol{g}_k, \circ_{j\in[\ell]}\boldsymbol{H}_j) = \prod_{k\in[m]} \prod_{j\in[\ell]} \boldsymbol{E}(\boldsymbol{g}_k, \boldsymbol{H}_j) \in \mathbb{G}_t$.
4. For $c \in \mathbb{Z}_p$ and $\boldsymbol{a} \in \mathbb{Z}_p^m$, the scalar multiplication is denoted by $c \cdot \boldsymbol{a} := (c \cdot a_1, \dots, c \cdot a_n) \in \mathbb{Z}_p^m$.
5. For $c \in \mathbb{Z}_p$ and $\boldsymbol{g} \in \mathbb{G}_i^m$, $i \in \{1, 2, t\}$ the scalar exponentiation is denoted by $\boldsymbol{g}^c := (g_1^c, \dots, g_m^c) \in \mathbb{G}_i^m$.
6. For $\boldsymbol{c} \in \mathbb{Z}_p^m$ and $g \in \mathbb{G}_i$, $i \in \{1, 2, t\}$ the vector exponentiation is denoted by $g^{\boldsymbol{c}} := (g^{c_1}, \dots, g^{c_m}) \in \mathbb{G}_i^m$.

3 Sublogarithmic Proofs via Generalization of BP-IP

In this section, we present our first generalization of BP-IP for the following IP relation $\mathcal{R}_{\mathsf{IP}}$ and then reduce its communication cost using the newly proposed aggregation technique.

$$\mathcal{R}_{\mathsf{IP}} = \left\{ (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, u, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b}) : \; P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G} \right\} \tag{1}$$

where $\mathbb{G}$ is an arbitrary cyclic group of order p satisfying the DL assumption, and $\boldsymbol{g}, \boldsymbol{h}$, and u are uniformly selected common inputs.

The BP-IP consists of $\log N$ recursive steps that halves the size of witness and the parameters. In each recursive round of BP-IP, each vector in the CRS and a witness are split into two equal-length subvectors. We generalize BP-IP by splitting a vector of length N into $2n$ subvectors of length $N/2n$ in each round, where $n = 1$ implies the original BP-IP. Similar to BP-IP, we assume N is a power of $2n$ for the sake of simplicity. Let $\widehat{N} = \frac{N}{2n}$ and the prover begins with parsing the witness $\boldsymbol{a}, \boldsymbol{b}$ and the parameter $\boldsymbol{g}, \boldsymbol{h}$ to

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{a}_1 \| \boldsymbol{a}_{-1} \| \cdots \boldsymbol{a}_{2n-1} \| \boldsymbol{a}_{-2n+1}, & \boldsymbol{b} &= \boldsymbol{b}_1 \| \boldsymbol{b}_{-1} \| \cdots \boldsymbol{b}_{2n-1} \| \boldsymbol{b}_{-2n+1}, \\ \boldsymbol{g} &= \boldsymbol{g}_1 \| \boldsymbol{g}_{-1} \| \cdots \boldsymbol{g}_{2n-1} \| \boldsymbol{g}_{-2n+1}, & \text{and } \boldsymbol{h} &= \boldsymbol{h}_1 \| \boldsymbol{h}_{-1} \| \cdots \boldsymbol{h}_{2n-1} \| \boldsymbol{h}_{-2n+1}. \end{aligned}$$

Let $I_n = \{\pm 1, \pm 3, \dots, \pm(2n-1)\}$ be a $2n$-size index set. In each recursive round of BP-IP, the prover computes and sends two group elements L and R. In our generalization, instead of L and R, $\mathcal{P}$ calculates $v_{i,j} = \boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j \boldsymbol{b}_i \rangle} \in \mathbb{G}$ for all distinct $i, j \in I_n$, and then sends $\{v_{i,j}\}_{\substack{i,j \in I_n \\ i \neq j}}$ to $\mathcal{V}$. Note that if $n = 1$, then $v_{1,-1}$ and $v_{-1,1}$ are equal to L and R in BP-IP, respectively. $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$. Finally, both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\widehat{\boldsymbol{g}} = \circ_{i \in I_n} \boldsymbol{g}_i^{x^{-i}} \in \mathbb{G}^{\widehat{N}}, \; \widehat{\boldsymbol{h}} = \circ_{i \in I_n} \boldsymbol{h}_i^{x^i} \in \mathbb{G}^{\widehat{N}}, \text{ and } \widehat{P} = P \cdot \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x^{j-i}} \in \mathbb{G}$$

and $\mathcal{P}$ additionally computes a witness for the next round argument

$$\widehat{\boldsymbol{a}} = \sum_{i \in I_n} \boldsymbol{a}_i x^i \in \mathbb{Z}_p^{\widehat{N}} \text{ and } \widehat{\boldsymbol{b}} = \sum_{i \in I_n} \boldsymbol{b}_i x^{-i} \in \mathbb{Z}_p^{\widehat{N}}.$$

We can verify that this process is a reduction to a one $2n$-th length IP argument by checking $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$ satisfies the relation $\mathcal{R}_{\mathsf{IP}}$. The concrete descriptions of BP-IP and the above generalized BP-IP and their proofs for the perfect completeness and the soundness are relegated to the full version [35].

Efficiency Analysis The prover repeats the $(N > 1)$ case $\log_{2n} N$ times and then runs the $(N = 1)$ case. For each $(N > 1)$ case, $\mathcal{P}$ sends $v_{i,j}$'s of size $2n(2n-1)$ and two integers in the $(N = 1)$ case, so that the communication overhead sent by $\mathcal{P}$ is $2n(2n-1)\log_{2n} N$ group elements and 2 integers. The verifier updates $\widehat{\boldsymbol{g}}$, $\widehat{\boldsymbol{h}}$ and $\widehat{P}$ that cost $O(N + n^2 \log_{2n} N)$ group exponentiation. For sufficiently small $n < \sqrt{N}$, it becomes $O(N)$. The prover should compute $v_{i,j}$ for all i, j for each round, so that the prover's computation overhead is $O(Nn^2)$. The overall complexities are minimized when n has the smallest positive integer (that is, $n = 1$), which is identical to the BP-IP protocol.

3.1 Proof Size Reduction using Multi-Exponentiation Argument

We improve our generalization of BP-IP by using the pairing-based homomorphic commitment scheme to group elements [1]. We first slightly extend our target relation by adding the commitment key of [1] into the common random string in our argument as follows.

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^N, u \in \mathbb{G}_1, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_m \in \mathbb{G}_2^{2n(2n-1)}, \boldsymbol{H} \in \mathbb{G}_2^m, P \in \mathbb{G}_1; \boldsymbol{a}, \boldsymbol{b}) \\ : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G}_1 \end{array} \right\} \tag{2}$$

where $\boldsymbol{g}, \boldsymbol{h}$,$u$,$\boldsymbol{F}_k$, and $\boldsymbol{H}$ are the common random string. Here, $\boldsymbol{F}_k$ and $\boldsymbol{H}$ are not necessary to define the relation $P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}$. However, our IP protocols will use them to run a subprotocol for multi-exponentiation arguments given in the following subsections.

The generalized BP-IP with $n > 1$ carries larger communication overhead than that of BP-IP. In order to reduce the communication cost in each round, we can use a commitment to group elements. That is, the prover sends a commitment to group elements $v_{i,j}$'s instead of sending all $v_{i,j}$'s. This approach will reduce communication cost in each round. Then, however, the verifier cannot directly compute the update $\widehat{P}$ of P, $\prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x^{j-i}}$, by himself, and thus the prover sends it along with its proof of validity, which is exactly a multi-exponentiation argument proving the following relation.

$$\left\{ (\boldsymbol{F} \in \mathbb{G}_2^N, \boldsymbol{z} \in \mathbb{Z}_p^N, P \in \mathbb{G}_t, q \in \mathbb{G}_1; \boldsymbol{v} \in \mathbb{G}_1^N) : P = \boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}) \wedge q = \boldsymbol{v}^{\boldsymbol{z}} \right\}, \tag{3}$$

where $\boldsymbol{F}$ is the common random string such that their discrete logarithm relation is unknown to both $\mathcal{P}$ and $\mathcal{V}$ and $\boldsymbol{z}$ is an arbitrary public vector.

We will omit the detailed description for the multi-exponentiation argument for the relation in (3), but provide an intuitive idea for it. In fact, BP-IP argument can be naturally extended to this proof system due to the resemblance between the standard IP and the inner-pairing product. More precisely, the additive homomorphic binding commitment to an integer vector (e.g., $\boldsymbol{g}^{\boldsymbol{a}}$) is changed with the multiplicative homomorphic commitment to a group element vector (e.g., $\boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F})$) and the standard IP between two integer vectors (e.g., $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$) can be substituted with multi-exponentiation (e.g., $\boldsymbol{v}^{\boldsymbol{z}}$).[5] This type of extension is well formalized by Bünz, Maller, Mishra, Tyagi, and Vesely [19] in terms of two-tiered homomorphic commitment scheme [31]. The multi-exponentiation argument in [19] costs the same complexities as those of BP-IP; $O(\log N)$ communication overhead and $O(N)$ computational costs for the prover and the verifier.

For our purpose, we can use the commitment scheme to group elements [31] and the multi-exponentiation argument in [19] so that we can construct a protocol with shorter communications, denoted by Protocol1. We provide full description of our generalized BP-IP with Multi-Exponentiation Argument, denoted by Protocol1 in Fig. 2. In the protocol, we add the state information for the prover and the verifier, denoted by st_P and st_V, respectively. Both st_P and st_V are initialized as empty lists and used to stack the inputs of the multi-exponentiation argument for each recursive round. At the final stage, the prover and the verifier can run several multi-exponentiation argument protocols in parallel.

Efficiency Analysis Although this approach reduces communication overheads, compared to the generalized BP-IP, it is not quite beneficial for our purpose. More precisely, the communication overhead $O(n^2 \log_{2n} N)$ of the generalized BP-IP is reduced to $O((\log n) \cdot (\log_{2n} N))$ since the communication overhead per round $O(n^2)$ is reduced to its logarithm $O(\log n)$ by the multi-exponentiation argument. Although the communication overhead is reduced to $O((\log n) \cdot (\log_{2n} N))$ compared with the generalized BP-IP ($n > 1$), the resulting complexity is equal to $O(\log N)$, which is asymptotically the same as the communication overhead of BP-IP. Therefore, this protocol is no better than BP-IP, at least in terms of communication complexity. Nevertheless, this protocol is a good basis for our sublogarithmic protocol presented in the next subsection.

3.2 Sublogarithmic Protocol from Aggregated Multi-Exponentiation Arguments

We build a protocol, denoted by Protocol2, for sublogarithmic transparent IP arguments on the basis of Protocol1 described in the previous subsection. To this end, we develop an aggregation technique to prove multiple multi-exponentiation arguments at once, which proves the following aggregated relation.

$$\mathcal{R}_{AggMEA} = \left\{ \begin{array}{l} \left(\begin{array}{l} \boldsymbol{F}_k \in \mathbb{G}_2^{2n(2n-1)}, \boldsymbol{z}_k \in \mathbb{Z}_p^{2n(2n-1)}, P_k \in \mathbb{G}_t, q_k \in \mathbb{G}_1 \\ ; \boldsymbol{v}_k \in \mathbb{G}_1^{2n(2n-1)} \text{ for } k \in [m] \end{array} \right) \\ : \bigwedge_{k \in [m]} \left(P_k = \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k) \wedge q_k = \boldsymbol{v}_k^{\boldsymbol{z}_k} \right) \end{array} \right\}$$

[5] The BP-IP is about two witness vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and it can be easily modified with one witness vector $\boldsymbol{a}$ and a public $\boldsymbol{b}$. e.g., [46]. Our multi-exponentiation argument corresponds to this variant.

```
$\boxed{\mathsf{Protocol1}(\boldsymbol{g}, \boldsymbol{h}, u, \boldsymbol{F}_k \text{ for } k \in [m], P \in \mathbb{G}_1, st_V; \boldsymbol{a}, \boldsymbol{b}, st_P), \text{ where } m = \log_{2n} N}$
If $N = 1$:
    Step 1: $\mathcal{P}$ sends $\mathcal{V}$ $a$ and $b$.
    Step 2: $\mathcal{V}$ proceeds the next step if $P = g^a h^b u^{a \cdot b}$ holds.
        Otherwise, $\mathcal{V}$ outputs Reject.
    Step 3: If $st_P$ is empty, then $\mathcal{V}$ outputs Accept.
        Otherwise, let $(u_k, v_k, \boldsymbol{x}_k; \boldsymbol{v}_k)$ be the $k$-th row in $st_P$.
    Step 4: $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{MEA}(\boldsymbol{F}_k, \boldsymbol{x}_k, u_k, v_k; \boldsymbol{v}_k)$ for $k \in [m]$.
Else ($N > 1$): Let $\widehat{N} = \frac{N}{2n}$ and parse $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to
        $\boldsymbol{a} = \boldsymbol{a}_1 \| \boldsymbol{a}_{-1} \| \cdots \boldsymbol{a}_{2n-1} \| \boldsymbol{a}_{-2n+1}$,    $\boldsymbol{b} = \boldsymbol{b}_1 \| \boldsymbol{b}_{-1} \| \cdots \boldsymbol{b}_{2n-1} \| \boldsymbol{b}_{-2n+1}$,
        $\boldsymbol{g} = \boldsymbol{g}_1 \| \boldsymbol{g}_{-1} \| \cdots \boldsymbol{g}_{2n-1} \| \boldsymbol{g}_{-2n+1}$,    and $\boldsymbol{h} = \boldsymbol{h}_1 \| \boldsymbol{h}_{-1} \| \cdots \boldsymbol{h}_{2n-1} \| \boldsymbol{h}_{-2n+1}$.
    Step 1: $\mathcal{P}$ calculates for all distinct $i \neq j \in I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$,
            $v_{i,j} = \boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \in \mathbb{G}_1$
        sets $\boldsymbol{v} = (v_{i,j}) \in \mathbb{G}_1^{2n(2n-1)}$ in the lexicographic order and sends $\mathcal{V}$ $\boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}_m)$.
    Step 2: $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.
    Step 3: $\mathcal{P}$ computes $v = \boldsymbol{v}^{\boldsymbol{x}} = \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x^{j-i}} \in \mathbb{G}_1$, where $\boldsymbol{x}$ is the vector
        consisting of $x^{j-i}$, and then sends it to $\mathcal{V}$.
    Step 4: Both $\mathcal{P}$ and $\mathcal{V}$ compute
            $\widehat{\boldsymbol{g}} = \circ_{i \in I_n} \boldsymbol{g}_i^{x^{-i}} \in \mathbb{G}_1^{\widehat{N}}$,   $\widehat{\boldsymbol{h}} = \circ_{i \in I_n} \boldsymbol{h}_i^{x^i} \in \mathbb{G}_1^{\widehat{N}}$   and   $\widehat{P} = P \cdot v \in \mathbb{G}_1$.
        Additionally, $\mathcal{P}$ computes $\widehat{\boldsymbol{a}} = \sum_{i \in I_n} \boldsymbol{a}_i x^i \in \mathbb{Z}_p^{\widehat{N}}$ and $\widehat{\boldsymbol{b}} = \sum_{i \in I_n} \boldsymbol{b}_i x^{-i} \in \mathbb{Z}_p^{\widehat{N}}$.
    Step 5: $\mathcal{V}$ updates $st_V$ by adding a tuple $(\boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}_m), v, \boldsymbol{x})$ into the bottom. $\mathcal{P}$
        updates $st_P$ by adding a tuple $(\boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}_m), v, \boldsymbol{x}; \boldsymbol{v})$ into the bottom. Both $\mathcal{P}$
        and $\mathcal{V}$ run $\mathsf{Protocol1}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \boldsymbol{F}_k \text{ for } k \in [m-1], \widehat{P}, st_V; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, st_P)$.
```

Fig. 2. Protocol1

Failed naïve approach: linear combination. One may try to employ a random linear combination technique, which is widely used to aggregate multiple relations using homomorphic commitment schemes. For example, it is called *linear combination of protocols* in [34]. To this end, one may also try to use one $\boldsymbol{F}$ instead of distinct $\boldsymbol{F}_k$'s for every pairing equation and employ homomorphic property of pairings and multi-exponentiations to apply a random linear combination technique. Unfortunately, however, the relation $\mathcal{R}_{AggMEA}$ consists of two distinct types of equations P_k and q_k containing *distinct* $\boldsymbol{z}_k$'s, so that such a random linear combination technique is not directly applicable to $\mathcal{R}_{AggMEA}$ even with one $\boldsymbol{F}$.

Why we use distinct $\boldsymbol{F}_k$'s? Our basic strategy for aggregation is to merge multiple equations into a single equation by product. Later, we will present a reduction for it (Theorem 2). To this end, it is necessary to use distinct $\boldsymbol{F}_k$'s for each equation since it prevents the prover from changing opening vectors between committed vectors in the product.

A difficulty when we use several F_k's. As we mentioned, we use different $\boldsymbol{F}_k$'s for each commitment P_k. In this case, it is not easy to efficiently prove that the equation that $P_k = \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)$ holds. The CRS contains all $\boldsymbol{F}_k$'s, and thus, in order to prove $P_k = \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)$, we have to prove that only one $\boldsymbol{F}_k$ is used and the others are not used in the equation. Proving unusedness of the other $\boldsymbol{F}_j$ for $j \neq k$ with high performance is rather challenging. Let us consider the following simplified aggregation relation to clarify this difficulty.

$$\mathcal{R}_{2agg} = \left\{(\boldsymbol{F}_k, P_k; \boldsymbol{v}_{k,j} \text{ for } k \in [2]) : \wedge_{k\in[2]} P_k = \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)\right\}$$

In order to merge two equations into a single equation by product, we might construct a reduction as follows; The verifier chooses a challenge y, both the players set $\tilde{\boldsymbol{F}}_1 = \boldsymbol{F}_1, \tilde{\boldsymbol{F}}_2 = \boldsymbol{F}_2^y$, and $\tilde{P} = P_1 P_2^y$, and then run a product argument convincing that the knowledge of $\tilde{\boldsymbol{v}}_k$ satisfying

$$\tilde{P} = \boldsymbol{E}(\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{F}}_1)\boldsymbol{E}(\tilde{\boldsymbol{v}}_2, \tilde{\boldsymbol{F}}_2). \tag{4}$$

Here, one may expect that an equality Eq. (4) guarantees two equalities in $\mathcal{R}_{2agg}$ by a random challenge y. Unfortunately, this is not true since fake P'_k passing the protocol can be created by $\boldsymbol{E}(\boldsymbol{v}_{k1}, \boldsymbol{F}_1)\boldsymbol{E}(\boldsymbol{v}_{k2}, \boldsymbol{F}_2)$ for $k = 1, 2$. That is, this reduction failed to show the unusedness of $\boldsymbol{F}_2$ in P_1 and $\boldsymbol{F}_1$ in P_2.

Our solution: augmented aggregate multi-exponentiation argument. Although the above approach is failed to prove the unusedness of $\boldsymbol{F}_2$ in P_1 and $\boldsymbol{F}_1$ in P_2, it can be still used to prove that P_k's are of the form $\boldsymbol{E}(\boldsymbol{v}_1, \boldsymbol{F}_1)\boldsymbol{E}(\boldsymbol{v}_2, \boldsymbol{F}_2)$ for some witness $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Therefore, instead of devising a protocol for the unusedness, we keep using the above approach of reducing to a product equation but change the target relation; we add redundant relations so that the final relation contains our target relation, multiple multi-exponentiations. That is, by adding some redundant values, we can further generalize the relation $\mathcal{R}_{AggMEA}$ and obtained the following relation $\mathcal{R}_{aAggMEA}$ for *augmented* aggregation of multi-exponentiations.

$$\left\{\begin{array}{l} \left(\begin{array}{l} \boldsymbol{F}_k \in \mathbb{G}_2^{2n(2n-1)}, \boldsymbol{z}_k \in \mathbb{Z}_p^{2n(2n-1)}, H_k \in \mathbb{G}_2, P_k \in \mathbb{G}_t, q_k \in \mathbb{G}_1 \\ ; \boldsymbol{v}_{k,j} \in \mathbb{G}^{2n(2n-1)} \text{ for } k, j \in [m] \end{array}\right) \\ : \bigwedge_{k\in[m]} \left(P_k = \prod_{j\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,j}, \boldsymbol{F}_j) \wedge q_k = \boldsymbol{v}_{k,k}^{\boldsymbol{z}_k} \wedge (\boldsymbol{v}_{k,j}^{\boldsymbol{z}_j} = 1_{\mathbb{G}_1} \text{ for } j \neq k)\right) \end{array}\right\} \tag{5}$$

Here, P_k is a commitment to $\boldsymbol{v}_{k,j}$'s and q_k is a multi-exponentiation of the committed value $\boldsymbol{v}_{k,k}$ and a public vector $\boldsymbol{z}$. In particular, P_k is defined by using all $\boldsymbol{F}_k$'s to avoid the difficulty of proving unusedness. Although there are redundant $\boldsymbol{v}_{k,j}$'s in P_k $(j \neq k)$, the above relation is sufficient to guarantee q_k is a multi-exponentiation of a committed value $\boldsymbol{v}_{k,k}$. In addition, H_k's are not necessary in the above relation, but we will use H_k's in the product argument, where we reduce from the augmented aggregation multi-exponentiation protocol.

The full description of Protocol2 using aAggMEA is given in Fig. 3.

$\boxed{\mathsf{Protocol2}(\boldsymbol{g},\boldsymbol{h},u,\boldsymbol{F}_k \text{ for } k\in[m],\boldsymbol{H},P\in\mathbb{G}_1,st_V;\boldsymbol{a},\boldsymbol{b},st_P)\text{, where } m=\log_{2n}N}$

If $N=1$:

Step 1: $\mathcal{P}$ sends $\mathcal{V}$ a and b.

Step 2: $\mathcal{V}$ proceeds the next step if $P=g^a h^b u^{a\cdot b}$ holds. Otherwise, $\mathcal{V}$ outputs *Reject*.

Step 3: If st_P is empty, then $\mathcal{V}$ outputs *Accept*. Otherwise, let $(u_k,v_k,\boldsymbol{x}_k;\boldsymbol{v}_k)$ be the k-th row in st_P and

$$\boldsymbol{v}_{k,j}=\begin{cases}\boldsymbol{1}_{\mathbb{G}_1} & \text{if } j\neq k\\ \boldsymbol{v}_k & \text{if } j=k\end{cases}$$

Step 4: $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{aAggMEA}(\boldsymbol{F}_k,\boldsymbol{x}_k,H_k,u_k,v_k;\boldsymbol{v}_{k,j}\text{ for }k,j\in[m])$.

Else $(N>1)$: Let $\widehat{N}=\frac{N}{2n}$ and parse $\boldsymbol{a},\boldsymbol{b},\boldsymbol{g}$, and $\boldsymbol{h}$ to

$$\boldsymbol{a}=\boldsymbol{a}_1\|\boldsymbol{a}_{-1}\|\cdots\boldsymbol{a}_{2n-1}\|\boldsymbol{a}_{-2n+1},\qquad \boldsymbol{b}=\boldsymbol{b}_1\|\boldsymbol{b}_{-1}\|\cdots\boldsymbol{b}_{2n-1}\|\boldsymbol{b}_{-2n+1},$$
$$\boldsymbol{g}=\boldsymbol{g}_1\|\boldsymbol{g}_{-1}\|\cdots\boldsymbol{g}_{2n-1}\|\boldsymbol{g}_{-2n+1},\qquad \text{and } \boldsymbol{h}=\boldsymbol{h}_1\|\boldsymbol{h}_{-1}\|\cdots\boldsymbol{h}_{2n-1}\|\boldsymbol{h}_{-2n+1}.$$

Step 1: $\mathcal{P}$ calculates for all distinct $i,j\in I_n=\{\pm1,\pm3,\ldots,\pm(2n-1)\}$,

$$v_{i,j}=\ \boldsymbol{g}_i^{\boldsymbol{a}_j}\boldsymbol{h}_j^{\boldsymbol{b}_i}u^{\langle\boldsymbol{a}_j,\boldsymbol{b}_i\rangle}\in\mathbb{G}_1$$

sets $\boldsymbol{v}=(v_{i,j})\in\mathbb{G}_1^{2n(2n-1)}$ in the lexicographic order and sends $\mathcal{V}$ $\boldsymbol{E}(\boldsymbol{v},\boldsymbol{F}_m)$.

Step 2: $\mathcal{V}$ chooses $x\xleftarrow{\$}\mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.

Step 3: $\mathcal{P}$ computes $v=\boldsymbol{v}^{\boldsymbol{x}}=\prod_{\substack{i,j\in I_n\\ i\neq j}}v_{i,j}^{x^{j-i}}\in\mathbb{G}_1$, where $\boldsymbol{x}$ is the vector consisting of x^{j-i}, and then send v to $\mathcal{V}$.

Step 4: Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\widehat{\boldsymbol{g}}=\circ_{i\in I_n}\boldsymbol{g}_i^{x^{-i}}\in\mathbb{G}_1^{\widehat{N}},\quad \widehat{\boldsymbol{h}}=\circ_{i\in I_n}\boldsymbol{h}_i^{x^{i}}\in\mathbb{G}_1^{\widehat{N}},\quad\text{and}\quad \widehat{P}=P\cdot v\in\mathbb{G}_1.$$

In addition, $\mathcal{P}$ computes

$$\widehat{\boldsymbol{a}}=\sum_{i\in I_n}\boldsymbol{a}_i x^i\in\mathbb{Z}_p^{\widehat{N}}\qquad\text{and}\qquad \widehat{\boldsymbol{b}}=\sum_{i\in I_n}\boldsymbol{b}_i x^{-i}\in\mathbb{Z}_p^{\widehat{N}}.$$

Step 5: $\mathcal{V}$ updates st_V by adding a tuple $(\boldsymbol{E}(\boldsymbol{v},\boldsymbol{F}_m),v,\boldsymbol{x})$ into the bottom. $\mathcal{P}$ updates st_P by adding a tuple $(\boldsymbol{E}(\boldsymbol{v},\boldsymbol{F}_m),v,\boldsymbol{x};\boldsymbol{v})$ into the bottom. Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{Protocol2}(\widehat{\boldsymbol{g}},\widehat{\boldsymbol{h}},u,\boldsymbol{F}_k\text{ for }k\in[m-1],\boldsymbol{H},\widehat{P},st_V;\widehat{\boldsymbol{a}},\widehat{\boldsymbol{b}},st_P)$.

Fig. 3. Protocol2: Sublogarithmic IP Argument

Theorem 1. *The IP argument in Fig. 3 has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption in* $\mathbb{G}_1$ *and the double pairing assumption.*

Due to space constraints, the proof is provided in the full version [35].

Efficiency Analysis A main difference between Protocol1 and Protocol2 is the aggregating process for $\log_{2n} N$ multi-exponentiation arguments; Our proposal for aAggMEA in the next subsection has logarithmic communication overhead in the size of witness. For each round of Protocol2, $2n(2n-1)$ group elements are committed and thus total $\log_{2n} N \times 2n(2n-1)$ group elements are committed. Therefore, the overall communication overhead is $O(\log_{2n} N)$ for the main recursive rounds and $O(\log(\log_{2n} N \times 2n(2n-1))) = O(\log n + \log(\log_{2n} N))$ for aAggMEA. That is, $O(\log n + \log_{2n} N)$. If n satisfies $O(\log_{2n} N) = O(\log n)$, then the communication complexity becomes $O(\log n + \log_{2n} N) = O(\sqrt{\log N})$.

As for the computational overhead, compared to generalized BP-IP, only a run of aAggMEA protocol is imposed. Our proposal for the aAggMEA protocol is an extended variant of BP-IP, so that its computational complexity is still linear in the length of witness vector that is $O(n^2 \log_{2n} N)$. Therefore, for sufficiently small $n < \sqrt{N}$, this does not affect on the overall complexity, so that the total prover's computational overhead is $O(Nn^2)$ and the verifier's computational overhead is $O(N + n^2 \log_{2n} N)$ that are equal to those of general BP-IP.[6]

3.3 Aggregating Multi-Exponentiation Argument

In this section, we propose an augmented aggregation of multi-exponentiation arguments aAggMEA for the relation in Eq. (5). Vectors in Eq. (5) are of dimension $2n(2n-1)$. For the sake of simplicity, we set the dimension of vectors N in this section and, by introducing dummy components, we can without loss of generality assume that N is a power of 2. The proposed protocol consists of two parts.

First, the aAggMEA is reduced to a proof system, denoted by ProdMEA, for the following relation $\mathcal{R}_{PMEA}$ for a product of multi-exponentiation.

$$\mathcal{R}_{PMEA} = \left\{ \begin{array}{l} (\boldsymbol{F}_k \in \mathbb{G}_2^N, \boldsymbol{z}_k \in \mathbb{Z}_p^N, H_k \in \mathbb{G}_2, P \in \mathbb{G}_t; \boldsymbol{v}_k \in \mathbb{G}_1^N \text{ for } k \in [m]) \\ : P = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k) e(\boldsymbol{v}_k^{\boldsymbol{z}_k}, H_k) \end{array} \right\}$$

The reduction is provided in Fig. 4 and its security property is given in the following theorem.

Theorem 2. *The* aAggMEA *protocol in Fig. 4 has perfect completeness and computational witness-extended-emulation if the* ProdMEA *protocol used in Fig. 4 has perfect completeness and computational witness-extended-emulation and the double pairing assumption holds.*

Due to space constraints, the proof is relegated to the full version [35].

Second part of aAggMEA is to run ProdMEA. The idea for the construction of ProdMEA is to use the resemblance between $\mathcal{R}_{PMEA}$ and $\mathcal{R}_{IP}$; $\mathcal{R}_{IP}$ is the relation about the inner-product between integer vectors, that is, a sum

[6] Note that when the communication complexity is evaluated, we set $n = 2^{\sqrt{\log N}}$ that is much smaller than $\sqrt{N} = 2^{\frac{1}{2}\log N}$, and thus our estimation for computational cost makes sense.

$\boxed{\mathsf{aAggMEA}(\boldsymbol{F}_k \in \mathbb{G}_2^N, \boldsymbol{z}_k \in \mathbb{Z}_p^N, H_k \in \mathbb{G}_2, P_k \in \mathbb{G}_t, q_k \in \mathbb{G}_1; \boldsymbol{v}_{k,j} \in \mathbb{G}_1^N \text{ for } k, j \in [m])}$

Step 1: $\mathcal{V}$ chooses and sends $y \overset{\$}{\leftarrow} \mathbb{Z}_p$ to $\mathcal{P}$.
Step 2: Both $\mathcal{P}$ and $\mathcal{V}$ set

$$\tilde{\boldsymbol{z}}_k = y^{k-1}\boldsymbol{z}_k,\ \tilde{\boldsymbol{F}}_k = \boldsymbol{F}_k^{y^{k-1}},\ \tilde{H}_k = H_k^{y^m},\ \text{ and } \tilde{P} = \prod_{k\in[m]} \Big(P_k^{y^{k-1}} \cdot e(q_k^{y^{k-1}}, \tilde{H}_k)\Big),$$

and $\mathcal{P}$ additionally sets $\tilde{\boldsymbol{v}}_k = \circ_{j\in[m]} \boldsymbol{v}_{j,k}^{y^{j-k}}$.
Step 3: $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{ProdMEA}(\tilde{\boldsymbol{F}}_k, \tilde{\boldsymbol{z}}_k, \tilde{H}_k, \tilde{P}; \tilde{\boldsymbol{v}}_k \text{ for } k \in [m])$

Fig. 4. Reduction from aAggMEA to ProdMEA

of component-wise products. $\mathcal{R}_{PMEA}$ is the relation about a product of exponentiation between a vector of group element $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ and an integer vector $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m)$. In particular, P is a product of $\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)$ a commitment to $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$, and $\prod_{k\in[m]} e(\boldsymbol{v}_k^{\boldsymbol{z}_k}, H_k)$ a commitment to the product of component-wise exponentiation between $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ and $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m)$. The homomorphic property of commitment to group elements enables us to construct ProdMEA similarly to BP-IP using the homomorphic Pedersen commitment to integers. We provide the construction and the detailed explanation of the protocol ProdMEA in the full version [35].

The computational costs of ProdMEA for the prover and the verifier are linear and the communication cost is logarithmic in the size of witness, like BP-IP. The reduction from aAggMEA to ProdMEA requires a constant communication cost and linear computational costs for both prover and verifier in the size of witness. Therefore, aAggMEA requires linear computational complexity and logarithmic communication complexity in the size of witness.

4 Sublinear Verifier via Second Generalization

In this section, we propose an IP argument with logarithmic communication and sublinear verifier computation, solely based on the DL assumption.

4.1 Matrices and Operations

For succinct exposition, we additionally define notations using matrices. Similar to a vector, a matrix is denoted by a bold letter and a vector is considered a row matrix. For a matrix $\boldsymbol{a} \in \mathbb{Z}_p^{m\times n}$, its separation to the upper half matrix $\boldsymbol{a}_1 \in \mathbb{Z}_p^{m/2\times n}$ and the lower half matrix $\boldsymbol{a}_{-1} \in \mathbb{Z}_p^{m/2\times n}$ is denoted by $\boldsymbol{a} = [\![\boldsymbol{a}_1 \| \boldsymbol{a}_{-1}]\!]$. We define three matrix operations as follows.

Inner-Product. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m\times n}$, the inner-product between $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as $\langle \boldsymbol{a}, \boldsymbol{b}\rangle := \sum_{r\in[m],s\in[n]} a_{r,s}b_{r,s} \in \mathbb{Z}_p$.

Multi-Exponentiation. For $\boldsymbol{g} \in \mathbb{G}_i^{m\times n}$, $i \in \{1,2,t\}$ and $\boldsymbol{a} \in \mathbb{Z}_p^{m\times n}$, the multi-exponentiation is defined as $\boldsymbol{g}^{\boldsymbol{a}} := \prod_{r\in[m],s\in[n]} g_{r,s}^{a_{r,s}} \in \mathbb{G}_i$.

Outer-Pairing Product. For $\boldsymbol{g} \in \mathbb{G}_1^m$ and $\boldsymbol{H} \in \mathbb{G}_2^n$, the outer-pairing product[7] is defined as

$$\boldsymbol{g} \otimes \boldsymbol{H} := \begin{bmatrix} e(g_1,H_1) & \dots & e(g_1,H_n) \\ \vdots & \ddots & \vdots \\ e(g_m,H_1) & \dots & e(g_m,H_n) \end{bmatrix} \in \mathbb{G}_t^{m\times n}.$$

Note that we set the output of the outer-pairing product to be a matrix instead of a vector, unlike an usual vector-representation of a tensor product since the matrix-representation is useful when separating it into two parts.

4.2 General Discrete Logarithm Relation Assumption

We restate the DLR assumption in terms of problem instance sampler to generalize it. Let GDLRsp be a sampler that takes the security parameter λ as input and outputs $(p, g_1, \dots, g_n, \mathbb{G})$, where $\mathbb{G}$ is a group $\mathbb{G}$ of λ-bit prime-order p and $g_1, \dots, g_n$ are generators of $\mathbb{G}$.

Definition 5 (General Discrete Logarithm Relation Assumption). *Let GDLRsp be a sampler. We say that GDLRsp satisfies the general discrete logarithm relation (GDLR) assumption if all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

$$\Pr\left[\boldsymbol{a} \neq \boldsymbol{0} \ \wedge \ \boldsymbol{g}^{\boldsymbol{a}} = 1_{\mathbb{G}} \middle| \begin{array}{c} (p, \boldsymbol{g} \in \mathbb{G}^n, \mathbb{G}) \leftarrow \mathsf{GDLRsp}(1^\lambda) \\ \boldsymbol{a} \leftarrow \mathcal{A}(p, \boldsymbol{g}, \mathbb{G}) \end{array}\right] < negl(\lambda),$$

where $1_{\mathbb{G}}$ is the identity of $\mathbb{G}$ and $negl(\lambda)$ is a negligible function in λ.

Definition 6. *For a fixed integer N, the sampler GDLRsp_{Rand} is defined as follows.*

$$\begin{array}{rl} \mathsf{GDLRsp}_{Rand}(1^\lambda): & \textit{Choose a group } \mathbb{G} \textit{ of } \lambda\textit{-bit prime-order } p;\ \boldsymbol{g} \xleftarrow{\$} \mathbb{G}^N; \\ & \textit{Output } (p, \boldsymbol{g}, \mathbb{G}). \end{array}$$

Theorem 3. *GDLRsp_{Rand} satisfies the GDLR assumption if the DL assumption holds for the same underlying group $\mathbb{G}$.*

The soundness theorem of BP-IP holds under the GDLR assumption; it uses only the fact that no adversary can find a non-trivial relation, regardless of the distribution of generators $\boldsymbol{g}$. We restate the soundness theorem of BP-IP below.

Theorem 4 ([17]). *The BP-IP has perfect completeness and computational witness-extended-emulation under the GDLR assumption.*

[7] Note that this operation is also called "projecting bilinear map" in the context of converting composite-order bilinear groups to prime-order bilinear groups [26].

We propose another sampler that satisfies the GDLR assumption.

Definition 7. *For $m, n \in \mathbb{N}$, the sampler* GDLRsp_{BM} *is defined as follows.*

$$\begin{aligned}\mathsf{GDLRsp}_{BM}(1^\lambda) : (p, g, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e) \leftarrow \mathcal{G}(1^\lambda); \boldsymbol{g} \xleftarrow{\$} \mathbb{G}_1^m; \boldsymbol{H} \xleftarrow{\$} \mathbb{G}_2^n, u \xleftarrow{\$} \mathbb{G}_t;\\ \textit{Output } (p, \boldsymbol{g} \otimes \boldsymbol{H}, u, \mathbb{G}_t).\end{aligned}$$

Theorem 5. GDLRsp_{BM} *satisfies the GDLR assumption if the DL assumption holds on $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Proof. Suppose that there exists a non-uniform polynomial-time adversary $\mathcal{A}$ breaking the GDLR assumption with non-negligible probability. That is, with non-negligible probability, $\mathcal{A}$ outputs a matrix $\boldsymbol{a} \in \mathbb{Z}_p^{m \times n}$ and an integer $c \in \mathbb{Z}_p$ such that $(\boldsymbol{g} \otimes \boldsymbol{H})^{\boldsymbol{a}} u^c = 1_{\mathbb{G}_t}$ and $\boldsymbol{a}$, c are not all zeros, where $1_{\mathbb{G}_t}$ is the identity of $\mathbb{G}_t$. We separate the adversarial types according to the output distribution. Let $\boldsymbol{a}_i \in \mathbb{Z}_p^n$ be the i-th row vector of $\boldsymbol{a}$ for $i \in [m]$.

– (Type 1) $c \neq 0$
– (Type 2) Not Type-1. $\forall i \in [m]$, $\boldsymbol{H}^{\boldsymbol{a}_i} = 1_{\mathbb{G}_2}$.
– (Type 3) Neither Type-1 or Type-2.

It is straightforward that $\mathcal{A}$ should be at least one of the above 3 types. For each adversary, we show how to break one of the DL assumption on $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$.[8]

Type-1 adversary. Given a DL instance $h_t \in \mathbb{G}_t$, we construct a simulator finding $Dlog_{e(g,H)} h_t$. First, choose $\boldsymbol{x}$ and $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$ and set $\boldsymbol{g} = g^{\boldsymbol{x}}$, $\boldsymbol{H} = H^{\boldsymbol{z}}$, and $u = h_t$. Then, the distribution of $(\boldsymbol{g}, \boldsymbol{H}, u)$ is identical to the real GDLR instance. The type-1 adversary outputs $\boldsymbol{a}$ and c such that $c \neq 0$ and $\boldsymbol{a} \neq \boldsymbol{0}$. From the necessary condition for $\boldsymbol{a}$ and c, we know the following equality holds.

$$\langle \boldsymbol{x} \otimes \boldsymbol{z}, \boldsymbol{a} \rangle + c \cdot Dlog_{e(g,H)} h_t = 0 \pmod{p}$$

Since we know all components except for $Dlog_{e(g,H)} h_t$ and $c \neq 0$, we can find $Dlog_{e(g,H)} h_t$ by solving the above modular equation.

Type-2 adversary. This type of adversary can be used as an attacker breaking the GDLR assumption on $\mathbb{G}_2$ with a sampler GDLRsp_{Rand}. Theorem 3 guarantees that there is no type-2 adversary breaking the GDLR assumption with GDLRsp_{BM} under the DL assumption on $\mathbb{G}_2$.

Type-3 adversary. Given a DL instance $\hat{g} \in \mathbb{G}_1$, we construct a simulator finding $DL_g \hat{g}$. First, choose an index $k \xleftarrow{\$} [m]$, integer vectors $\boldsymbol{x} = (x_1, \ldots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$, $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$, and $w \xleftarrow{\$} \mathbb{Z}_p$, and set $\boldsymbol{g} = (g^{x_1}, \ldots, g^{x_{k-1}}, \hat{g}, g^{x_{k+1}}, \ldots, g^{x_m})$, $\boldsymbol{H} = H^{\boldsymbol{z}}$, and $u = e(g, H)^w$. Then, the distribution of $(\boldsymbol{g}, \boldsymbol{H}, u)$ is identical to the real GDLR instance. Let $\hat{\boldsymbol{x}} = (x_1, \ldots, x_{k-1}, Dlog_g \hat{g}, x_{k+1}, \ldots, x_m)$. Then, $\boldsymbol{g} = g^{\hat{\boldsymbol{x}}}$.

[8] Note that the DL assumption on $\mathbb{G}_1$ implies the DL assumption on $\mathbb{G}_t$ by the MOV attack [39].

The type-3 adversary outputs $\boldsymbol{a}$ and c such that $c = 0$ and $\boldsymbol{H}^{\boldsymbol{a}_i} \neq 1_{\mathbb{G}_2}$ for some $i \in [n]$. From the necessary condition for $\boldsymbol{a}$ and c, we know the following equality holds.

$$\begin{aligned}\langle \hat{\boldsymbol{x}} \otimes \boldsymbol{z}, \boldsymbol{a} \rangle + c \cdot w &= x_1 \langle \boldsymbol{z}, \boldsymbol{a}_1 \rangle + \cdots + (Dlog_g \hat{g}) \langle \boldsymbol{z}, \boldsymbol{a}_k \rangle + \cdots + x_m \langle \boldsymbol{z}, \boldsymbol{a}_m \rangle + c \cdot w \\ &= 0 \pmod p\end{aligned}$$

Since the index k is completely hidden from the viewpoint of $\mathcal{A}$, $i = k$ with non-negligible $1/m$ probability. If $i = k$, then $\langle \boldsymbol{z}, \boldsymbol{a}_k \rangle \neq 0$, so that we can recover $(Dlog_g \hat{g})$ by solving the above modular equation, since we know all components except for $Dlog_g \hat{g}$. □

4.3 Another Generalization of BP-IP with Sublinear Verifier

In BP-IP, most of the common input for $\mathcal{P}$ and $\mathcal{V}$ are uniformly selected group elements, which is the common random string. What we expect from these group elements is that their discrete logarithms are unknown, so that the DLR assumption holds. The DL assumption implies the GDLR assumption with uniform sampler and this assumption is the root of the soundness of BP-IP. We can generalize BP-IP while keeping the soundness proof by using an arbitrary sampler satisfying the GDLR assumption, instead of GDLRsp_{Rand} to create the CRS.

Sublinear Common Inputs. We uniformly generate $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m$ and $\boldsymbol{H} \in \mathbb{G}_2^n$ and use $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H} \in \mathbb{G}_t^{m \times n}$ instead of the CRS in BP-IP. That is, we construct a proof system for the following relation.

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m, \boldsymbol{H} \in \mathbb{G}_2^n, u, P \in \mathbb{G}_t;\ \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}) \\ :\ P = (\boldsymbol{g} \otimes \boldsymbol{H})^{\boldsymbol{a}} (\boldsymbol{h} \otimes \boldsymbol{H})^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G}_t \end{array} \right\} \quad (6)$$

Note that this modification does not require the structured reference string since $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$ are publicly computable from the common random string $\boldsymbol{g}$, $\boldsymbol{h}$ and $\boldsymbol{H}$. Furthermore, the proof system is still sound since, like the CRS in BP-IP, $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$ hold the GDLR assumption under the DL assumption on $\mathbb{G}_1$ and $\mathbb{G}_2$ by Theorem 5.

Sublinear Verification. If we set $m = n = \sqrt{N}$, the above modification can reduce the CRS size to be a square root of BP-IP. Nevertheless, computing $\boldsymbol{g} \otimes \boldsymbol{H}$ requires linear computation in N so that the verification cost is still linear in N. We arrange the order of witness $\boldsymbol{a}$ and $\boldsymbol{b}$ in each round, and thus we can go through the process without exactly computing $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$. We explain how to avoid a full computation of $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$. Without loss of generality, we assume that m and n are powers of 2.[9] If $m > 1$, then let $\widehat{m} = \frac{m}{2}$ and parse $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}, \boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m$ to

[9] If needed, we can appropriately pad zeros in the vectors since zeros do not affect the result of inner-product.

$$\boldsymbol{a} = [\![\boldsymbol{a}_1 \| \boldsymbol{a}_{-1}]\!] \quad \boldsymbol{b} = [\![\boldsymbol{b}_1 \| \boldsymbol{b}_{-1}]\!], \quad \boldsymbol{g} = \boldsymbol{g}_1 \| \boldsymbol{g}_{-1}, \text{ and } \boldsymbol{h} = \boldsymbol{h}_1 \| \boldsymbol{h}_{-1}.$$

Then, the bases $\boldsymbol{g} \otimes \boldsymbol{H} \in \mathbb{G}_t^{m \times n}$ and $\boldsymbol{h} \otimes \boldsymbol{H} \in \mathbb{G}_t^{m \times n}$ are able to be implicitly parsed to $[\![\boldsymbol{g}_1 \otimes \boldsymbol{H} \| \boldsymbol{g}_{-1} \otimes \boldsymbol{H}]\!]$ and $[\![\boldsymbol{h}_1 \otimes \boldsymbol{H} \| \boldsymbol{h}_{-1} \otimes \boldsymbol{H}]\!]$, respectively. Let $\tilde{\boldsymbol{g}}_i = \boldsymbol{g}_i \otimes \boldsymbol{H} \in \mathbb{G}_t^{\widehat{m} \times n}$ and $\tilde{\boldsymbol{h}}_i = \boldsymbol{h}_i \otimes \boldsymbol{H} \in \mathbb{G}_t^{\widehat{m} \times n}$ for $i \in \{1, -1\}$. Next, $\mathcal{P}$ calculates

$$L = \tilde{\boldsymbol{g}}_{-1}^{\boldsymbol{a}_1} \ \tilde{\boldsymbol{h}}_1^{\boldsymbol{b}_{-1}} u^{\langle \boldsymbol{a}_1, \boldsymbol{b}_{-1} \rangle} \text{ and } R = \tilde{\boldsymbol{g}}_1^{\boldsymbol{a}_{-1}} \tilde{\boldsymbol{h}}_{-1}^{\boldsymbol{b}_1} \ u^{\langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle} \in \mathbb{G}_t$$

and sends them to $\mathcal{V}$. This computation of $\mathcal{P}$ is equivalent to BP-IP with CRS $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$. $\mathcal{V}$ returns a random challenge $x \xleftarrow{\$} \mathbb{Z}_p^*$ to $\mathcal{P}$. Finally, both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\widehat{\boldsymbol{g}} = \boldsymbol{g}_1^{x^{-1}} \circ \boldsymbol{g}_{-1}^{x} \in \mathbb{G}_1^{\widehat{m}}, \quad \widehat{\boldsymbol{h}} = \boldsymbol{h}_1^{x} \circ \boldsymbol{h}_{-1}^{x^{-1}} \in \mathbb{G}_1^{\widehat{m}}, \quad \text{and } \widehat{P} = L^{x^2} P \ R^{x^{-2}} \in \mathbb{G}_t$$

and $\mathcal{P}$ additionally computes $\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1}$ and $\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{m} \times n}$. Then, $\widehat{P}$ is well computed since L and R are equivalent to those in BP-IP. In BP-IP, however, $\tilde{\boldsymbol{g}}_1^{x^{-1}} \circ \tilde{\boldsymbol{g}}_{-1}^{x}$ and $\tilde{\boldsymbol{h}}_1^{x} \circ \tilde{\boldsymbol{h}}_{-1}^{x^{-1}}$ should be computed as the new bases for the next round argument with witness $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$. Instead, in Protocol3, we use the equality $\widehat{\boldsymbol{g}} \otimes \boldsymbol{H} = \tilde{\boldsymbol{g}}_1^{x^{-1}} \circ \tilde{\boldsymbol{g}}_{-1}^{x}$ and $\widehat{\boldsymbol{h}} \otimes \boldsymbol{H} = \tilde{\boldsymbol{h}}_1^{x} \circ \tilde{\boldsymbol{h}}_{-1}^{x^{-1}}$ such that $\widehat{\boldsymbol{g}}$ and $\widehat{\boldsymbol{h}}$ are the bases for the next argument with $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$. Therefore, both $\mathcal{P}$ and $\mathcal{V}$ can run the protocol with $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \boldsymbol{H}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$. If $m = 1$, the CRS is of the form $e(g, \boldsymbol{H})$ and $e(h, \boldsymbol{H})$, which is uniform in $\mathbb{G}_t$, so that we can directly run BP-IP over $\mathbb{G}_t$. We present the full description of our protocol, denoted by Protocol3, in Fig. 5. The number of rounds and the communication cost in Protocol3 are the same as those of BP-IP over $\mathbb{G}_t$. The verification cost is $O(\sqrt{N})$ when setting $m = n$. Note that a naïve verification in the $(m = 1)$ case requires $O(\sqrt{N})$ expensive pairing computation for calculating $e(g, \boldsymbol{H})$ and $e(h, \boldsymbol{H})$, but using a similar trick in the case $(m > 1)$, the verifier can update $\boldsymbol{H}$ only instead of $e(g, \boldsymbol{H})$ and $e(h, \boldsymbol{H})$ and then perform constant pairing operations only at the final stage.

Linear Prover and Logarithmic Communication. In terms of the prover's computation and communication overheads, Protocol3 is asymptotically the same as BP-IP since we can consider Protocol3 as BP-IP with CRS $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$. That is, $O(N)$ and $O(\log_2 N)$ for computation and communication, respectively.

Theorem 6. *The argument presented in Fig. 5 for the relation* (6) *has perfect completeness and computational witness-extended-emulation under the GDLR assumption with the sampler* GDLRsp_{BM}.

Proof. Although the verification cost in Protocol3 is reduced compared with BP-IP, both players' computation in Protocol3 is equivalent to that of BP-IP with the CRS $\boldsymbol{g} \otimes \boldsymbol{H}$ and $\boldsymbol{h} \otimes \boldsymbol{H}$. Therefore, the proof of this theorem should be exactly the same as the proof of BP-IP [35], except that the GDLR assumption is guaranteed by Theorem 5 instead of Theorem 3. $\square$

$$\boxed{\mathsf{Protocol3}(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m, \boldsymbol{H} \in \mathbb{G}_2^n, u, P; \boldsymbol{a}, \boldsymbol{b})}$$

If $m = 1$: $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP}_{\mathsf{IP}}(e(g, \boldsymbol{H}), e(h, \boldsymbol{H}), u, P; \boldsymbol{a}, \boldsymbol{b})$.
Else $(m > 1)$: Let $\widehat{m} = \frac{m}{2}$. Parse $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to

$$\boldsymbol{a} = [\![\boldsymbol{a}_1 \| \boldsymbol{a}_{-1}]\!] \quad \boldsymbol{b} = [\![\boldsymbol{b}_1 \| \boldsymbol{b}_{-1}]\!], \quad \boldsymbol{g} = \boldsymbol{g}_1 \| \boldsymbol{g}_{-1}, \text{ and } \boldsymbol{h} = \boldsymbol{h}_1 \| \boldsymbol{h}_{-1}.$$

Step 1: $\mathcal{P}$ calculates

$$L = (\boldsymbol{g}_{-1} \otimes \boldsymbol{H})^{\boldsymbol{a}_1} \ (\boldsymbol{h}_1 \ \otimes \boldsymbol{H})^{\boldsymbol{b}_{-1}} u^{\langle \boldsymbol{a}_1, \boldsymbol{b}_{-1} \rangle} \in \mathbb{G}_t$$

$$\text{and } R = (\boldsymbol{g}_1 \ \otimes \boldsymbol{H})^{\boldsymbol{a}_{-1}} (\boldsymbol{h}_{-1} \otimes \boldsymbol{H})^{\boldsymbol{b}_1} \ u^{\langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle} \in \mathbb{G}_t$$

and sends them to $\mathcal{V}$.
Step 2: $\mathcal{V}$ chooses $x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.
Step 3: Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\widehat{\boldsymbol{g}} = \boldsymbol{g}_1^{x^{-1}} \circ \boldsymbol{g}_{-1}^{x} \in \mathbb{G}_1^{\widehat{m}}, \quad \widehat{\boldsymbol{h}} = \boldsymbol{h}_1^{x} \circ \boldsymbol{h}_{-1}^{x^{-1}} \ \in \mathbb{G}_1^{\widehat{m}}, \quad \text{and } \widehat{P} = L^{x^2} P \ R^{x^{-2}} \in \mathbb{G}_t.$$

Additionally, $\mathcal{P}$ computes $\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1}$ and $\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{m}}$.
Step 4: Both $\mathcal{P}$ and $\mathcal{V}$ run the protocol with $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \boldsymbol{H}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$.

Fig. 5. Protocol3: Another Generalization of BP-IP

4.4 Practical verification of Protocol 3

When it comes to asymptotic complexity, Protocol3 is definitely better than BP-IP. However, for practical performance, we need to consider the computation time of group operations which depends on the choice of elliptic curves. Actually, BP-IP and Protocol3 are built on different elliptic curves. Current implementations of BP-IP use two curves, i.e., secp256k1 and ed25519 curves. The dalek project has reported that the use of ed25519 provides approximately 2x speepup [23]. However, Protocol3 cannot use ed25519 because it requires pairing operations. Therefore, we take ed25519 for BP-IP and BLS12-381 for Protocol3 in the below estimation.

We consider a typical parameter setting $N = 2^{20}$ in 128-bits security which both secp256k1 and ed25519 curves provide. BP-IP requires 2×2^{20} group operations for verification. Protocol3 requires 2×2^{10} $\mathbb{G}_1$ operations and 2×2^{10} $\mathbb{G}_2$ operations for verification. According to the implementation results from [43], the execution times of operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ of BLS12-381 are roughly $5\times$ and $10\times$ slower than that of ed25519, respectively. Thus, we expect that Protocol3's verifier is significantly faster (approximately $70\times$) than that of BP-IP.

5 Sublinear Verifier without Pairing

We propose another IP argument with sublinear verifier, particularly without pairings. The crucial ingredient for Protocol3 is pairing-based homomorphic commitments to group elements [1], which is employed as the second layer scheme of the two-tiered commitment scheme. For example, L in **Step 1** of Protocol3 contains a factor $(\boldsymbol{g}_{-1} \otimes \boldsymbol{H})^{\boldsymbol{a}_1}$, which can be considered as a vector of homomorphic commitments to $\boldsymbol{g}_{-1}^{\boldsymbol{a}_1} \in \mathbb{G}_1^n$, where $\boldsymbol{g}_{-1}^{\boldsymbol{a}_1}$ is a vector of the first layer commitments to columns of $\boldsymbol{a}_1$ and $\boldsymbol{g}_{-1} \in \mathbb{G}_1^m$ and $\boldsymbol{H} \in \mathbb{G}_2^n$ are the commitment keys of first and second layer schemes, respectively. When the verifier checks $\widehat{P} = L^{x^2} P\, R^{x^{-2}} \in \mathbb{G}_t$ in **Step 3** of Protocol3, the homomorphic property of the second layer scheme guarantees a vector of linear group equations $(\boldsymbol{g}_{-1}^{\boldsymbol{a}_1})^{x^2} \cdot (\boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}}) \cdot (\boldsymbol{h}_1^{\boldsymbol{b}_{-1}})^{x^{-2}} \in \mathbb{G}_1^n$ holds, where $(\boldsymbol{g}_{-1}^{\boldsymbol{a}_1})$, $(\boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}})$, and $(\boldsymbol{h}_1^{\boldsymbol{b}_{-1}})$ are second layer openings of L, R, and P. Since the first layer scheme is homomorphic, these n equations in $\mathbb{G}_1^n$ similarly guarantee that mn linear equations hold.

In order to circumvent the necessity of using the pairing-based primitive, we propose a new two-tiered commitment scheme such that the first layer scheme is still Pedersen commitment scheme mapping from integers to group elements and the second layer scheme for committing to group elements is replaced with the new one. We show that although the new second layer scheme is not homomorphic in group operations, it facilitates efficient proving group operations.

Indeed the integrity of homomorphic operation is sufficient to build an argument system. For example, if the prover computes L and R in **Step 1** by using the new two-tiered commitments, the verifier cannot compute $\widehat{P}$ by herself in **Step 3**, so that the prover should send $\widehat{P}$ along with its integrity proof. As mentioned above, the relation for the integrity proof is exactly a vector of linear group equations between the second layer openings. Since the new commitment scheme facilitates proving this type of relation, the new argument system still has the benefit of sublinear verifier.

Unfortunately, this approach increases the proof size due to additional integrity proofs for each round. Finally, we bring in the aggregation technique used for the sublogarithmic proofs in Section 3, so that we can simultaneously attain both logarithmic proof size and sublinear verifier.

Notation. We use a pair of elliptic curve groups, denoted by $(\mathbb{G}_p, \mathbb{G}_q)$, of distinct prime order p and q such that $\mathbb{G}_p := E(\mathbb{Z}_q)$. In order to avoid confusion, we use lower case letters to denote elements in $\mathbb{G}_p$ and upper case letters to denote elements in $\mathbb{G}_q$. For example, $g \in \mathbb{G}_p$ and $G \in \mathbb{G}_q$. In our protocol, we repeatedly use parallel multi-exponentiations with the same base $\boldsymbol{g} \in \mathbb{G}_p^m$. For example, given an integer matrix $\boldsymbol{a} \in \mathbb{Z}_p^{m \times n}$, we often compute $\boldsymbol{g}^{\boldsymbol{a}_i}$ for $i \in [n]$, that are n multi-exponentiations, where $\boldsymbol{a}_i$ is the i-th column of $\boldsymbol{a}$. This computation is compactly denoted by $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} := (\boldsymbol{g}^{\boldsymbol{a}_1}, \ldots, \boldsymbol{g}^{\boldsymbol{a}_n})$.

5.1 Projective Presentation for Elliptic Curve Group

Affine coordinates are the conventional way of expressing elliptic curve points. However, there is no complete addition formula in affine coordinates, i.e., affine coordinates require special addition formulas for exceptional cases such as doubling and operations with the point at infinity or the inverse point. In our construction, it is desirable to have an arithmetic circuit which correctly computes the operation between any two points in the elliptic curve group. Thus, we make use of complete addition formulas for prime order elliptic curves in projective coordinates, which have been proposed by Renes et al. [41] based on the work of Bosma and Lenstra [16].

Let $E(\mathbb{Z}_q)$ with $q \geq 5$ be a prime order elliptic curve group given by the short Weierstrass equation in two-dimensional projective space $\mathbb{P}^2(\mathbb{Z}_q)$, i.e.,

$$\{(X, Y, Z) \in \mathbb{Z}_q^3 | Y^2 Z = X^3 + aXZ^2 + bZ^3\}.$$

Two points (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) are equal in $\mathbb{P}^2(\mathbb{Z}_q)$ if and only if $(X_2, Y_2, Z_2) = (\lambda X_1, \lambda Y_1, \lambda Z_1)$ for some $\lambda \in \mathbb{Z}_q^*$. The point at infinity is equal to $(0, 1, 0)$. Because $E(\mathbb{Z}_q)$ has prime order, there is no $\mathbb{Z}_q$-rational point of order 2. In this setting, for any two pair of points (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2), Bosma and Lenstra gave the complete formulas to compute $(X_3, Y_3, Z_3) = (X_1, Y_1, Z_1) + (X_2, Y_2, Z_2)$ where X_3, Y_3, and Z_3 are expressed as polynomials in X_1, Y_1, Z_1, X_2, Y_2, and Z_2. Later, Renes et al. presented the algorithm [41, Algorithm 1] for the optimized version of Bosma and Lenstra' addition formula. The algorithm covers both doubling and addition operations without exceptional cases using 12 multiplications, 5 multiplications by constant, and 23 additions over $\mathbb{Z}_q$. Thus, we consider the arithmetic circuit from this formula for group operations of $E(\mathbb{Z}_q)$ in our construction. For the convenience of readers, we provide the algorithm given by Renes et al. in the full version [35].

5.2 Two-Tiered Commitment Scheme and Proof for Second Layer

We introduce a two-tiered commitment scheme for handing columns of a matrix $\boldsymbol{a} \in \mathbb{Z}_p^{m \times n}$. The first layer commitment is for committing to a vector in $\mathbb{Z}_p^m$. The second layer commitment is for committing to the multiple, say n, first layer commitments. Therefore, the final two-tiered commitment scheme is for committing to a matrix $\boldsymbol{a} \in \mathbb{Z}_p^{m \times n}$.

We begin with a pair of elliptic curve groups $(\mathbb{G}_p = E(\mathbb{Z}_q), \mathbb{G}_q)$ of respective order p and q such that the discrete logarithm assumption holds in both $\mathbb{G}_p$ and $\mathbb{G}_q$. Note that there are efficient methods to generate such a pair of prime order elliptic curves $(\mathbb{G}_p = E(\mathbb{Z}_q), \mathbb{G}_q)$ of given primes p and q whose sizes are both 2λ for the security parameter λ [42]. In the first layer, we use the Pedersen commitment scheme with commitment key $\boldsymbol{g} \in \mathbb{G}_p^m$ to commit to columns of $\boldsymbol{a}$.[10] That is, the commitment is $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \in \mathbb{G}_p^n$, which is an n-tuple of Pedersen commitments to columns of $\boldsymbol{a}$. Since it consists of elliptic curve group elements, it can be represented by n sequences of 3-element tuples $(X_i, Y_i, Z_i)_{i=1}^n \in \mathbb{Z}_q^{3n}$, where (X_i, Y_i, Z_i) is the projective representation of the i-th component of $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}}$. For the second layer, we again use the Pedersen commitment with a *different*

[10] More precisely, we use a slightly modified Pedersen commitment scheme in the sense that (1) opening is not an integer but a vector and (2) the random element is always set to be zero since the hiding property is not required.

commitment key $\boldsymbol{G} = (G_1, \ldots, G_{3n}) \in \mathbb{G}_q^{3n}$ so that the commitment to $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} = (X_i, Y_i, Z_i)_{i=1}^n$ is defined as $\prod_{i=1}^n G_{3i-2}^{X_i} G_{3i-1}^{Y_i} G_{3i}^{Z_i}$, denoted by $\mathsf{Com}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}}; \mathbf{G})$.

Note that we often consider $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}}$ as an element in $\mathbb{Z}_q^{3n}$ since we always use the projective representation for $\mathbb{G}_p = E(\mathbb{Z}_q)$ throughout the paper. The binding property of the proposed commitment scheme holds under the discrete logarithm assumption in $\mathbb{G}_p$ and $\mathbb{G}_q$.

Proving for Relation between Second Layer Opening. The second layer opening is $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \in \mathbb{G}_p^n$, a vector of group elements, which can be considered as a vector of $\mathbb{Z}_q^{3n}$. As aforementioned in the first part of this section, we should prove a relation among the second layer openings that consist of a vector of group operations. As shown in Section 5.1, the group law of $E(\mathbb{Z}_q)$ can be represented by an arithmetic circuit over $\mathbb{Z}_q$ of constant size. Therefore, we eventually need a proof system for arithmetic circuits over $\mathbb{Z}_q$ such that the input of the circuit is given as commitments. In fact, the bulletproofs for arithmetic circuit (BP-AC) [17] allows to take Pedersen commitments as input. However, BP-AC uses the ordinary Pedersen commitment to an integer, so that it is not directly applicable with the generalized Pedersen commitment to a vector of integers. We generalize BP-AC for handling the general Pedersen commitments and provide the protocol, denoted by $\mathsf{Comp.BP}_{AC}$, and the security and efficiency analysis in the full version [35]. If we prove $O(\ell)$ group operations, then the circuit size is $O(\ell \cdot n)$, so that both the computational cost for the prover and the verifier are $O(\ell \cdot n)$ and the cost for round and communication is $O(\log n + \log \ell)$.

In fact, the new commitment scheme can take any sequence of 3-integer tuples $(X_i, Y_i, Z_i) \in \mathbb{Z}_q^3$ as input. Although we normally take (X_i, Y_i, Z_i) from $\mathbb{G}_p = E(\mathbb{Z}_q)$, to prevent abnormal usages, we need a proof that $(X_i, Y_i, Z_i) \in \mathbb{Z}_q^3$ is on the elliptic curve, equivalently, it satisfies $Y^2 Z = Z^3 + aXZ^2 + bZ^3$ for some $a, b \in \mathbb{Z}_q$. Since the relation for the membership proof consists of low degree polynomials, it can be performed by $\mathsf{Comp.BP}_{AC}$ whose cost is cheaper than that for elliptic curve operations.

5.3 Sublinear Verifier from New Two-tiered Commitment Scheme

We propose a new IP argument with the sublinear verifier, denoted by $\mathsf{Protocol4}$, that proves the following IP relation.

$$\mathcal{R}_{\mathsf{IP}}^{m,n} = \left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_p^m, \boldsymbol{F} \in \mathbb{G}_q^{6n}, P \in \mathbb{G}_q, c \in \mathbb{Z}_p; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}) : \\ P = \mathsf{Com}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \parallel \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}; \boldsymbol{F}) \wedge c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle, \end{array} \right\} \quad (7)$$

where $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ is the Frobenius inner product between matrices $\boldsymbol{a}$ and $\boldsymbol{b}$. Similarly to $\mathsf{Protocol3}$, $\mathsf{Protocol4}$ consists of two parts, the row-reduction and the column-reduction. The row-reduction part is denoted by $\mathsf{Protocol4.Row}$ and reduces from the relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$ to $\mathcal{R}_{\mathsf{IP}}^{1,n}$. The column-reduction part is denoted by $\mathsf{Protocol4.Col}$ and reduces from the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ to $\mathcal{R}_{\mathsf{IP}}^{1,1}$.

Let $\ell = \log m$. For each $(\ell+1-k)$-th row-reduction[11] round in $\mathsf{Protocol4.Row}$ the prover sends the verifier a commitment S_k by using the new commitment scheme in Section 5.2. However, contrary to $\mathsf{Protocol3}$, the verifier cannot compute a valid instance P_k for the next round by himself, due to lack of homomorphic property. Instead, the prover sends a new instance for the next round along with a proof for its integrity. For the column-relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$, both the prover and the verifier can similarly perform a column-reduction protocol $\mathsf{Protocol4.Col}$ and the corresponding integrity proof at the final step of the protocol. In a nutshell, $\mathsf{Protocol4}$ resembles $\mathsf{Protocol3}$ except that $\mathsf{Protocol4}$ uses a different commitment scheme and additionally requires the integrity proof. The full description of $\mathsf{Protocol4.Row}$ is provided in the full version [35].

In general, this *commit-first-and-prove-later* approach indeed ends up with low efficiency if the relation is not algebraic (e.g., non-polynomial relations) or we do not use homomorphic commitment scheme (e.g., collision-resistant hash functions). Our new two-tiered commitment scheme helps to circumvent such efficiency degradation since it is friendly to proving homomorphic operations and the prover's computation in $\mathsf{Protocol4}$ exactly consists of elliptic curve operations that can be represented by polynomials as we already investigated in Section 5.1.

Although the new two-tiered commitment scheme contributes for the sub-linear verifier, the naïve approach for the integrity proof increases the proof size $O(\log(N)^2)$, which is larger than $O(\log N)$ of $\mathsf{Protocol3}$, where $N = mn$. Therefore, we bring in another technique to make the proof size compact. We apply the aggregation techniques as in Section 3.3 such that the integrity of the prover's computation in all reduction rounds is relegated to the final round and then proven in aggregate. More concretely, the integrity proof should guarantee that the openings $\boldsymbol{p}_{k+1} \in \mathbb{G}_p^{2n}$, $\boldsymbol{l}_k \| \boldsymbol{r}_k \in \mathbb{G}^{4n}$, and $\boldsymbol{p}_k \in \mathbb{G}_p^{2n}$ of P_{k+1}, S_k, and P_k satisfies $\boldsymbol{p}_k = \boldsymbol{l}_k^{x^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k^{x^{-2}}$, which is essentially equivalent to the relation between openings of $\widehat{P} = L^{x^2} P R^{x^{-2}}$ in **Step 3** of $\mathsf{Protocol3}$. The formal relation for the aggregated integrity proof is given in Eq. (8) (for $\mathsf{Protocol4.row}$) and Eq. (9) (for $\mathsf{Protocol4.col}$), where x_k is a challenge chosen by the verifier and the others are the common random strings. Using the protocol for $\mathcal{R}_{\mathsf{AggMEC.Row}}$ ($\mathcal{R}_{\mathsf{AggMEC.Col}}$, resp.), denoted by $\mathsf{AggMEC.Row}$ ($\mathsf{AggMEC.Col}$, resp.), $\mathsf{Protocol4.Row}$ ($\mathsf{Protocol4.Col}$, resp.) reduces from the relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$ ($\mathcal{R}_{\mathsf{IP}}^{1,n}$, resp.) to the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ ($\mathcal{R}_{\mathsf{IP}}^{1,1}$, resp.).

$$\mathcal{R}_{\mathsf{AggMEC.Row}} = \left\{ \begin{array}{l} \left(\begin{bmatrix} (\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k) \\ (\ \cdot\ , \boldsymbol{F}_{\ell+1}, \cdot\ , P_{\ell+1}, \cdot\ ) \end{bmatrix} ; \begin{bmatrix} (\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k) \\ (\ \cdot,\ \cdot, \boldsymbol{p}_{\ell+1}) \end{bmatrix} \text{ for } k \in [\ell] \right) : \\ \wedge_{j=1}^{\ell+1} \left(P_j = \mathsf{Com}(\boldsymbol{p}_j; \boldsymbol{F}_j) \right) \\ \wedge_{k=1}^{\ell} \left(S_k = \mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k; \boldsymbol{S}_k) \wedge \boldsymbol{p}_k = \boldsymbol{l}_k^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k^{x_k^{-2}} \right) \\ \boxed{\wedge_{k=1}^{\ell} \boldsymbol{l}_k, \boldsymbol{r}_k \in \mathbb{G}_p^{2n} \wedge \boldsymbol{p}_{\ell+1} \in \mathbb{G}_p^{2n}} \end{array} \right\} \tag{8}$$

[11] Notice that we use a subscript k in reverse order from $k = \ell$ to $k = 1$. That is, $\mathsf{Protocol4.Row}$ reduces an instance from P_{k+1} to P_k.

where $\big((\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k); (\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k)\big) \in \big((\mathbb{G}_q^{12n} \times \mathbb{G}_q^{6n} \times \mathbb{G}_q \times \mathbb{G}_q \times \mathbb{Z}_p) \times (\mathbb{Z}_q^{6n} \times \mathbb{Z}_q^{6n} \times \mathbb{Z}_q^{6n})\big)$.

$$\mathcal{R}_{\mathsf{AggMEC.Col}} = \left\{ \begin{array}{l} \left(\begin{bmatrix} (\boldsymbol{D}_k, P_k, x_k) \\ (\boldsymbol{D}_{\ell+1}, P_{\ell+1}, \ \cdot \ ) \end{bmatrix}; \begin{bmatrix} (\boldsymbol{p}_k) \text{ for } k \in [\ell] \\ (\boldsymbol{p}_{\ell+1}) \end{bmatrix} \right) : \\ \wedge_{j=1}^{\ell+1} \big(P_j = \mathsf{Com}(\boldsymbol{p}_j; \boldsymbol{D}_j)\big) \wedge \boxed{\boldsymbol{p}_{\ell+1} \in \mathbb{G}_p^{2^{\ell+1}}} \\ \wedge_{k=1}^{\ell} \big(\boldsymbol{p}_k = (\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}}\big) \\ \text{where } \boldsymbol{p}_{k+1} = \boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1} \parallel \boldsymbol{p}_{4,k+1} \end{array} \right\} \quad (9)$$

where $\big((\boldsymbol{D}_k, P_k, x_k); (\boldsymbol{p}_k)\big) \in \big((\mathbb{G}_q^{3\cdot 2^k} \times \mathbb{G}_q \times \mathbb{Z}_p) \times (\mathbb{Z}_q^{3\cdot 2^k})\big)$.

The concrete descriptions of the four protocols Protocol4.Row, Protocol4.Col, AggMEC.Row, and AggMEC.Col and the proofs for proving argument systems are given in the full version [35].

We remark that $\mathcal{R}_{\mathsf{AggMEC.Row}}$ and $\mathcal{R}_{\mathsf{AggMEC.Col}}$ contain the group membership relations of the openings, which are marked with the block boxes. As for the group membership proof, it is sufficient to prove only memberships of $\boldsymbol{l}_k, \boldsymbol{r}_k$ for $k \in [\ell]$ and $\boldsymbol{p}_{\ell+1}$ since $\boldsymbol{p}_k$ for $k \in [\ell]$ are defined as a result of the group operations among $\boldsymbol{l}_k, \boldsymbol{r}_k$ for $k \in [\ell]$ and $\boldsymbol{p}_{\ell+1}$.

Efficiency Analysis We analyze the efficiency of Protocol4 at a high level. The detailed analysis is given in the full version [35]. Below, we denote group operations in a group G by G-operations. The efficiency of Protocol4 is basically equivalent to that of Protocol3 except for using a different commitment scheme and the most computational cost of $\mathcal{V}$ shifts to the column reduction part (Protocol4.Col). In the row-reduction part (Protocol4.Row), the computation cost for $\mathcal{P}$ is dominated by $O(mn \log p)$ $\mathbb{G}_p$-operations for computing two-tiered commitments with $N = mn$ integers, the computation cost for $\mathcal{V}$ is $O(m \log p)$ $\mathbb{G}_p$-operations, and $\mathcal{P}$ and $\mathcal{V}$ communicate with $O(\log m)$ $\mathbb{G}_q$-elements. The complexity of the column-reduction part is dominated by proving the following relations, which can be represented by small-degree polynomials, by running the arithmetic circuit argument $\mathsf{Comp.BP}_{AC}$ given in Section 5.2:

$$\boldsymbol{l}_k^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k^{x_k^{-2}} - \boldsymbol{p}_k = \boldsymbol{0} \text{ for } k \in [\ell] \quad (10)$$

$$(\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} - \boldsymbol{p}_k = \boldsymbol{0} \in \mathbb{G}_p^{2k} \text{ for } k \in [\ell]. \quad (11)$$

Arithmetic circuits for computing Eq.(10) and Eq.(11) consist of $O(n\ell \log p)$ and $O(2^\ell \log p)$ $\mathbb{G}_p$-operations, respectively. Finally, $\mathsf{Comp.BP}_{AC}$ for the above arithmetic circuits cost $O((n\ell + 2^\ell) \log p \log q)$ $\mathbb{G}_q$-operations for each $\mathcal{P}$ and $\mathcal{V}$ and transmissions of $O(\log n + \ell + \log\log p)$ $\mathbb{G}_q$-elements. Setting $\ell \leftarrow \log m$, $\mathcal{P}$'s computation complexity is $O(mn \log p)$ $\mathbb{G}_p$-operations, $\mathcal{V}$'s computation complexity is $O(m \log p)$ $\mathbb{G}_p$-operations and $O(n \log m \log p \log q)$ $\mathbb{G}_q$-operations, and the communication complexity is $O(\log n + \log m + \log\log p)$ $\mathbb{G}_q$-elements.

6 Extensions

6.1 Transparent Polynomial Commitment Scheme

Informally, using the polynomial commitment scheme (PCS), a committer first commits to a polynomial $f(X)$, and then later opens $f(x)$ at some point x (mostly chosen by a verifier) and convinces a verifier of correctness of $f(x)$. Due to space constraint, we provide the definition of the PCS, a way to use the proposed IP arguments as PCS, and a comparison table in the full version [35].

6.2 Zero-Knowledge Argument for Arithmetic Circuits

There is a well-established approach toward the argument for arithmetic circuits via polynomial commitment scheme; an IP argument is firstly reduced to polynomial commitment schemes as in 6.1 and then combined with polynomial IOPs [18]. This reduction increases constant times the complexity, where linear preprocessing is required for the verifier. Therefore, the final argument for the arithmetic circuit of size N has the same complexity as those of our IP arguments between vectors of length N, where the online verifier's complexity is unchanged, but the offline verifier's complexity is linear in N.

The perfect special honest verifier zero-knowledge (SHVZK) means that given the challenge values, it is possible to simulate the whole transcript even without knowing the witness. If the polynomial commitment scheme is hiding and the proof of evaluation is SHVZK, then the resulting argument for arithmetic circuit is SHVZK as well. Although the proposed IP protocols do not have these properties yet, there is a simple method to add ZK into IP arguments [46, 18]. For example, we can extend commitment schemes used in the paper to have hiding factors like the original Pedersen commitment scheme.

There is another approach for converting from an IP argument without SHVZK to the SHVZK argument for arithmetic circuit [13, 17]. We can apply this reduction to our IP arguments. We provide the details in the full version [35].

7 Discussion on Best of Two Generalizations

It would be interesting to devise a technique for combining two generalizations.

First, we find that naïve combining $\mathsf{Protocol2}$ and $\mathsf{Protocol3}$ is difficult because each of them uses a bilinear map for a different purpose. In $\mathsf{Protocol2}$, the bilinear map is used in the first step for compressing multiple group elements by sending a commitment instead of multiple group elements. In the first step of $\mathsf{Protocol3}$, the $\mathcal{P}$ sends L and R to the verifier, where L and R are elements in $\mathbb{G}_t$. We can generalize $\mathsf{Protocol3}$ like $\mathsf{Protocol1}$, but we cannot put L and R into a homomorphic commitment scheme directly since L and R are already in the target group of the bilinear map.

Although $\mathsf{Protocol4}$ does not use the bilinear map, combining $\mathsf{Protocol2}$ and $\mathsf{Protocol4}$ will be challenging as well. Since both protocols use two-tier commitment schemes, we may need three-tier commitment scheme such as $\mathsf{C}_3 \circ \mathsf{C}_2 \circ \mathsf{C}_1$, where C_3 is pairing-based AFGHO scheme, C_2 is a commitment to elliptic curve point, and C_1 is Pedersen commitment scheme. $\mathsf{Protocol4}$ requires to prove small-degree polynomial relations over C_1 and C_2 supports an efficient protocol for it. C_3 may support to prove a small-degree polynomial relation over C_2. However, since C_1 is an opening of an opening of C_3, the small-degree polynomial relation over C_1 might be represented as a complicated relation over C_2 of higher-degree. We leave achieving the best of both generalizations as an open problem.

Acknowledgement

We thank Taechan Kim for discussion on complete addition formulas for elliptic curves. This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (A Study on Cryptographic Primitives for SNARK, 50%) under Grant 20210007270012002, and in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT), 50%, under Grant 2020R1C1C1A0100696812.

References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. *Journal of Cryptology*, 29(2):363–421, 2016.
2. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *ACM CCS 2017*, pages 2087–2104. ACM, 2017.
3. S. Bayer and J. Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 646–663. Springer, 2013.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 1993*, pages 62–73. ACM, 1993.
5. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.
6. E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.
7. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for r1cs. In *EUROCRYPT 2019*, volume 11476 of *LNCS*, pages 103–128. Springer, 2019.
8. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016*, volume 9986 of *LNCS*, pages 31–60. Springer, 2016.
9. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von Neumann architecture. In *USENIX Security 2014*, pages 781–796, 2014.
10. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS 2012*, pages 326–349. Springer, 2012.

11. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *Symposium on Theory of Computing Conference, STOC 2013*, pages 111–120. ACM, 2013.
12. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.
13. J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 327–357. Springer, 2016.
14. J. Bootle, A. Cerulli, E. Ghadafi, J. Groth, M. Hajiabadi, and S. K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *ASIACRYPT 2017*, volume 10626 of *LNCS*, pages 336–365. Springer, 2017.
15. J. Bootle, A. Chiesa, and S. Liu. Zero-knowledge IOPs with linear-time prover and polylogarithmic-time verifier. In *EUROCRYPT 2022*, volume 13276 of *LNCS*, pages 275–304. Springer, 2022.
16. W. Bosma and H. W. Lenstra. Complete systems of two addition laws for elliptic curves. *Journal of Number Theory*, 53:229–240, 1995.
17. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy 2018*, pages 315–334. IEEE Computer Society, 2018.
18. B. Bünz, B. Fisch, and A. Szepieniec. Transparent snarks from dark compilers. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 677–706. Springer, 2020.
19. B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely. Proofs for inner pairing products and applications. In *ASIACRYPT 2021*, volume 13092 of *LNCS*, pages 65–97. Springer, 2021.
20. J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, 2010.
21. A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 738–768. Springer, 2020.
22. A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 769–793. Springer, 2020.
23. dalek cryptography:Bulletproofs. https://github.com/dalek-cryptography/bulletproofs, 2018.
24. V. Daza, C. Ràfols, and A. Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In *PKC 2020*, volume 12110 of *LNCS*, pages 527–557. Springer, 2020.
25. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.
26. D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.
27. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.pdf.
28. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, 2013.
29. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 192–208. Springer, 2009.

30. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, 2010.
31. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 431–448. Springer, 2011.
32. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016.
33. J. Groth and M. Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 253–280, 2015.
34. M. Hoffmann, M. Klooß, and A. Rupp. Efficient zero-knowledge arguments in the discrete log setting, revisited. In *ACM CCS 2019*, pages 2093–2110, 2019.
35. S. Kim, H. Lee, and J. H. Seo. Efficient zero-knowledge argument in discrete logarithm setting: Sublogarithmic proof or sublinear verifier. Cryptology ePrint Archive, Paper 2021/1450, 2021. https://eprint.iacr.org/2021/1450.
36. libsnark. https://github.com/scipr-lab/libsnark, 2017.
37. H. Lipmaa. Progression-free sets and sublinear pairing-based noninteractive zero-knowledge arguments. In *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, 2012.
38. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *ACM CCS 2019*, pages 2111–2128. Association for Computing Machinery, 2019.
39. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on information Theory*, 39(5):1639–1646, 1993.
40. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy 2013*, pages 238–252. IEEE, 2013.
41. J. Renes, C. Costello, and L. Batina. Complete addition formulas for prime order elliptic curves. In *EUROCRYPT 2016*, volume 9665 of *LNCS*, pages 403–428. Springer, 2016.
42. E. Savas, T. A. Schmidt, and Ç. K. Koç. Generating elliptic curves of prime order. In *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *LNCS*, pages 142–158. Springer, 2001.
43. M. Scott. On the deployment of curve based cryptography for the internet of things. Cryptology ePrint Archive, Report 2020/514, 2020. https://eprint.iacr.org/2020/514.
44. J. H. Seo. Round-efficient sub-linear zero-knowledge arguments for linear algebra. In *PKC 2011*, volume 6571 of *LNCS*, pages 387–402. Springer, 2011.
45. S. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *CRYPTO 2020*, volume 12172 of *LNCS*, pages 704–737. Springer, 2020.
46. R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. Doubly-efficient zkSNARKs without trusted setup. In *IEEE Symposium on Security and Privacy 2018*, pages 926–943. IEEE, 2018.
47. T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO 2019*, volume 11694 of *LNCS*, pages 733–764. Springer, 2019.
48. J. Zhang, T. Xie, Y. Zhang, and D. Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *IEEE Symposium on Security and Privacy 2020*, pages 859–876. IEEE, 2019.