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

Sungwook Kim[1], Hyeonbum Lee[2], and Jae Hong Seo[2]**

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

[2] Department of Mathematics, Hanyang University, Republic of Korea
`{leehb3706, jaehongseo}@hanyang.ac.kr`

**Abstract.** We propose three zero-knowledge arguments for arithmetic circuit of size $N$. 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, 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 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 [44, 31], 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 [23]. To this end, we propose three inner-product (IP) arguments with the same properties (standard assumption, common random string model), and then apply well-established reductions from IP argument to ZK argument for arithmetic circuits [15, 19, 52, 20].

The first sublinear ZK argument for arithmetic circuits solely based on the hardness of the DL problem is due to Groth [33] and improved by Seo [50].

** Corresponding Author

These works feature constant round complexity as well. Groth [35] 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, 2]. The first logarithmic ZK argument for arithmetic circuits solely from the DL assumption is due to Bootle, Cerulli, Chaidos, Groth, and Petit [15] and improved by Bünz, Bootle, Boneh, Poelstra, Wuille, and Maxwell [19], which is called Bulletproofs. Hoffmann, Klooß, and Rupp [39] 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 [15, 19, 39] have linear verifiers. Other DL-based ZK argument systems with different asymptotic performance, in particular sublinear verifier, have been proposed. e.g., Hyrax [52] and Spartan [51]. Recently, Bünz, Maller, Mishra, and Vesely [21] achieved a logarithmic ZK argument with a sublinear verifier whose soundness is proved under the DPair assumption.

Focusing on specific languages, there are more researches achieving logarithmic communication complexity [5, 37] prior to Bulletproofs. Logarithmic communication complexity in these works is attained with relatively large round complexity, compared to [33, 50].

Relying on the non-standard but reliable assumptions, there exists a ZK argument system with better asymptotic performance due to Bünz, Fisch, and Szepieniec [20] 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) [34, 43, 12, 32, 13, 46, 8, 11, 37, 36, 44, 31, 53, 24] 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 [7, 9, 25, 55] for SNARG that are not based on bilinear groups, but based on interactive oracle proofs [10]. 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 [15, 19].

## 1.1 Our Results

We propose three inner-product (IP) arguments between two integer vectors of length $N$ in the common random string model that directly yield ZK arguments for arithmetic circuits of size $N$ with fan-in 2 gates. We refer to [15, 19, 20] or Section 6 for a constant round reduction from ZK arguments 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.

2. We present the first IP argument with logarithmic communication and sublinear verifier computation such that its soundness is *solely based on the DL assumption.*
3. We propose the first IP argument with logarithmic communication and sublinear verifier computation *without relying on pairings* such that its soundness is solely based on the DL assumption.

The main ingredient of our arguments is two different generalizations of Bünz et al.'s Bulletproofs IP argument (BP-IP) [19]. The first generalization is a trade off round complexity against communication cost per round. Then, we devise a novel method to aggregate multiple arguments, so that our first argument achieves sublogarithmic rounds and communications simultaneously. The second generalization substitutes Pedersen commitments with a pairing-based two-tiered commitments (Pedersen commitment to integers + AFGHO commitments to group elements [2]), so that our second argument achieves sublinear verification. Moreover, we develop a new proof method solely relying on DL for the soundness of the second argument. Finally, to move away from reliance on pairings, we propose a new two-tiered commitment scheme without pairings. Although the new commitment scheme is not homomorphic, we show that it can be well harmonized with the second generalization so that we achieve the first IP argument with sublinear verifier, without relying on pairings. We believe that all these techniques applied to three arguments are of independent interest.

We provide a comparison for transparent ZK argument systems in Table 1. Note that there are much more efficient argument systems in the discrete logarithm setting [11, 42, 44, 31, 24, 27] if we rely on a trusted setup and non-standard, non-falsifiable assumptions.

## 1.2 Overview of Our Approach

In this subsection, we briefly overview our approach for IP arguments with improved complexity, sublogarithmic communications or sublinear verifier. Having IP arguments, we can follow well-established methods reducing from IP arguments to ZK arguments for arithmetic circuit (refer to [15, 19, 20] or Section 6).

**Sublogarithmic Communications.** We consider the IP relation $\{(\boldsymbol{g}, \boldsymbol{h}, u, P; \boldsymbol{a}, \boldsymbol{b}) : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}\}$, where $(\boldsymbol{a}, \boldsymbol{b})$ is a witness and $(\boldsymbol{g}, \boldsymbol{h}, u, P)$ composes a statement. First, we generalize BP-IP to trade-off the round complexity against the communication complexity. More precisely, we observe that the length of witness is halved for each recursive step in BP-IP and it can be generalized by reducing the length of witness one $2n$-th 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, in our generalization, 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 $n = 1$.

| Scheme | Communication | $\mathcal{P}$'s comp. | $\mathcal{V}$'s comp. | Assump. |
|---|---|---|---|---|
| Groth [33] & Seo [50] | $O(\sqrt{N})\mathbb{G}_1$ | $O(N)\tau_1$ | $O(N)\tau_1$ | DL |
| Groth [35] | $O(\sqrt[3]{N})\mathbb{G}_1$ | $O(N)\tau_1$ | $O(\sqrt[3]{N})\tau_1$ | DPair |
| BP [15, 19] & HKR [39] | $O(\log N)\mathbb{G}_1$ | $O(N)\tau_1$ | $O(N)\tau_1$ | DL |
| Hyrax [52] | $O(\sqrt{w} + d \log N)\mathbb{G}_1$ | $O(N \log N)\tau_1$ | $O(\sqrt{w} + d \log N)\tau_1$ | DL |
| Spartan DL [51] | $O(\sqrt{N})\mathbb{G}_1$ | $O(N)\tau_1$ | $O(\sqrt{N})\tau_1$ | DL |
| BMMV [21] | $O(\log N)\mathbb{G}_t$ | $O(N)\tau_1$ | $O(\sqrt{N})\tau_2$ | DPair |
| Supersonic [20] | $O(\log N)\mathbb{G}_U$ | $O(N \log N)\mathsf{u}$ | $O(\log N)\mathsf{u}$ | UOGroup |
| Spartan CL [51] | $O(\log^2 N)\mathbb{G}_U$ | $O(N \log N)\mathsf{u}$ | $O(\log^2 N)\mathsf{u}$ | UOGroup |
| Ligero [3] | $O(\sqrt{N})\mathbb{H}$ | $O(N \log N)\mathsf{h}$ | $O(N)\mathsf{h}$ | CR hash |
| STARK [7] | $O(\log^2 N)\mathbb{H}$ | $O(N \log^2 N)\mathsf{h}$ | $O(\log^2 N)\mathsf{h}$ | CR hash |
| Aurora [9] | $O(\log^2 N)\mathbb{H}$ | $O(N \log N)\mathsf{h}$ | $O(N)\mathsf{h}$ | CR hash |
| Fractal [25] | $O(\log^2 N)\mathbb{H}$ | $O(N \log N)\mathsf{h}$ | $O(\log^2 N)\mathsf{h}$ | CR hash |
| Virgo [55] | $O(d \log N)\mathbb{H}$ | $O(N \log N)\mathsf{h}$ | $O(d \log N)\mathsf{h}$ | CR hash |
| BCGGHJ [16] | $O(\sqrt{N})\mathbb{H}$ | $O(N)\mathsf{m}$ | $O(N)\mathsf{m}$ | CR hash |
| **Our IP arguments + Section 6** | | | | |
| Protocol2 (Section 3.2) | $O(\sqrt{\log N})\mathbb{G}_t$ | $O(N 2^{\sqrt{\log N}})\tau_1$ | $O(N)\tau_1$ | 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, $\tau_i$: group operations in $\mathbb{G}_i$, $\mathsf{u}$: group operation in $\mathbb{G}_U$, $\mathsf{p}$: pairing operation, $\mathsf{h}$: hash operation, $\mathsf{m}$: field operation
UOGroup: unknown-order group (strong RSA assumption & adaptive root assumption), CR hash: collision-resistant hashes, DL[†]: DL assumption over pairing-friendly elliptic curves
All arguments in the table are public coin (Definition 4), so that they achieve non-interactivity in the random oracle model using the Fiat-Shamir heuristic [28].

Second, we apply the commit-and-prove approach inside the our generalization to reduce transmission overhead. More precisely, instead of sending $2n(2n-1)$ group elements, the prover sends only a commitment to a vector of $2n(2n-1)$ group elements (e.g., [1, 2]). Then, the verifier cannot prepare input for the next recursive step without openings. Instead, the prover computes on behalf of the verifier and sends the result along with its proof to the verifier. This process of committing and proving can be achieved using a multi-exponentiation argument (e.g., [21]). 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. More precisely, the communication cost $O(n^2 \log_{2n} N)$ of the recursive step in the generalized BP-IP is reduced to $O(\log_{2n} N)$ when committing a value instead of sending $O(n^2)$ group elements in each recursive round, but it additionally requires proving the $\log_{2n} N$ multi-exponentiation arguments with $n^2$-length vectors costing $O\big((\log_{2n} N)\cdot(\log n^2)\big)$. Overall, its communication cost is $O\big((\log_{2n} N)\cdot(\log n^2)\big) = O(\log N)$, which is asymptotically the same as that of BP-IP.

Lastly, to further reduce the communication cost, we devise a novel technique for aggregating multiple multi-exponentiation arguments. As aforementioned, we use commit-and-prove approach, in particular, using pairing-based homomorphic commitment scheme [1, 2]. There exists a well-known aggregating technique for multiple arguments using homomorphic commitment scheme that essentially uses homomorphic property of commitment scheme (e.g., aggregating range proofs [19], linear combinations of protocols [39]). Unfortunately, this aggregating technique is not well applied to our case since each multi-exponentiation argument uses *distinct* base $\boldsymbol{v}_k$ and exponent $\boldsymbol{z}_k$.

Our strategy for aggregating multiple multi-exponentiations is 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 difficulty we faced is given in Section 3.2. Instead, we devise an *augmented aggregated multi-exponentiation argument* called $\mathsf{aAggMEA}$; we add redundant values in the target relation for $\mathsf{aAggMEA}$ such that the final relation can cover the desired aggregated multi-exponentiation relation, and then we reduce from $\mathsf{aAggMEA}$ to a product argument called $\mathsf{ProdMEA}$ that proves a single product relation containing multiple multi-exponentiation relations. After changing to a single relation, we can employ a recursive proof technique used in BP-IP and achieve $O\big((\log_{2n} N) + (\log n^2)\big)$ communication overhead. Using $\mathsf{aAggMEA}$, we finally achieve $O(\sqrt{\log N})$ communication cost when setting $n$ to satisfy $O(\log_{2n} N) = O(\log n^2)$. For example, if $N = (2n)^m$ and $n = 2^m$, then the communication cost is $O(m) = O(\sqrt{\log N})$.

**Sublinear Verifier from Discrete Logarithms.** We show that another generalization of BP-IP with careful analysis enables a sublinear verifier while keeping benefits of BP-IP such as logarithmic communications and soundness proof from the discrete logarithm assumption.

First we start from an observation such that indeed the soundness of BP-IP is based on the *discrete logarithm relation assumption* that no adversary can find non-trivial relation among given group elements. In particular, if all group elements are chosen at random, the discrete logarithm assumption directly guarantees the discrete logarithm relation assumption.

Interestingly, we find that even non-uniform group elements can guarantee that no adversary can find non-trivial relation among given group elements. Note that we do not rely on the structured reference string, but use bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ to generate structured group elements in $\mathbb{G}_t$ from uniformly selected group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$. We formalize our observation by generalizing the discrete logarithm relation assumption (Definition 7) and then prove that the discrete logarithm assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$ is sufficient to guarantee the hardness of finding non-trivial relation among structured group elements in $\mathbb{G}_t$ (Theorem 5). This result is of independent interest and can be used to prove the security of other protocols beyond proof systems. Using this result, we can reduce the CRS size to be a square root of the length $N$ of witness vector whiling keeping the soundness proof under the discrete logarithm assumption.

Nevertheless, a naïve approach using the above idea will keep linear verifier computation in $N$ since we reduce the CRS size only but keep the same verification process as that of Bulletproofs. Using the CRS of $O(\sqrt{N})$ size, we introduce a trick to track verifier's necessary computation with only $O(\sqrt{N})$ computation without performing $O(N)$ computation like Bulletproofs. This trick does not increase the prover's computation and communication overhead, so that we can keep linear prover complexity and logarithmic communication complexity while achieving the sublinear verifier under the discrete logarithm assumption.

**Sublinear Verifier without Pairings.** The sublinear verifier in the above second generalization of BP-IP can be accounted as an outcome of employing the *two-tiered homomorphic commitment scheme*, though its soundness proof under the DL assumption needs careful analysis: First, an integer vector of length $N$ is rearranged as an $m \times n$ matrix $\boldsymbol{a}$, where $N = mn$. Second, columns of $\boldsymbol{a}$ are first committed by using the Pedersen commitment scheme and then the resulting $n$ committed columns are secondly committed by using a commitment scheme to group elements (e.g., AFGHO [2]). Given two commitments to $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}$, the homomorphic property enables anyone to compute a commitment of $\boldsymbol{a} + \boldsymbol{b}$ without having any secrets. To the best of our knowledge, practical homomorphic commitment scheme to group elements (e.g., AFGHO [2]) is built on pairing-friendly elliptic curves.

We propose another IP argument with sublinear verifier, particularly not relying on pairing-friendly elliptic curves. To this end, we devise a new approach to circumvent the necessity of using the two-tiered homomorphic commitment scheme. First, we propose a 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 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.

### 1.3 Related Work

*Argument for Algebraic Relations.* Groth [33] proposed a sublinear ZK argument for linear algebra relations. In particular, he proposed an argument for bilinear maps such as inner-products and then showed that many linear algebra relations, such as trace, can be reduced to that argument for bilinear maps. Bootle et al. reduced communication overhead of IP argument to be logarithmic in the dimension of witness [15, 19]. There are proposals for other algebraic relations. For example, Groth and Sahai [38] proposed a non-interactive ZK proof system for pairing-based relations, such as pairing-product, without relying on NP reduction. Lai, Malavolta, and Ronge proposed a logarithmic argument for pairing-based relations [41] and Bünz, Maller, Mishra, Vesely [21] further improved it.

*Polynomial Commitment Scheme.* Kate, Zaverucha and Goldberg introduced polynomial commitment scheme such that 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 $f(x)$. Polynomial commitments play an important building block not only for constructing ZK arguments for arithmetic circuits [35, 52, 44, 31, 53, 18, 24, 51, 55] but also for constructing many other crypto primitives such as verifiable secret sharing [40, 4], anonymous credentials [22, 30], and proofs of storage and replication [54].

Although Kate et al.'s polynomial commitment scheme achieves succinct opening and verification cost, it requires structured reference string that requires a trusted setup. Polynomial commitment schemes without a trusted setup can be achieved through a transparent IP argument [15, 19, 52, 18]. Bünz, Fisch and Szepieniec formalize a framework of compiling polynomial IOP to ZK argument using polynomial commitment scheme [20].

### 1.4 Organization

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 arithmetic circuits.

## 2 Definitions

**Arguments of Knowledge.** Let $\mathcal{K}$ be the common reference string (CRS) generator that takes the security parameter as input and outputs the CRS $\sigma$. 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 $\sigma$, a statement $x$, and a witness $w$. We define a CRS-dependent language $L_\sigma$ 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. Let $negl(\lambda)$ be a negligible function in $\lambda$.

**Definition 1.** *Let $\mathcal{K}$ be the CRS generator and $(\mathcal{P}, \mathcal{V})$ be an argument. We say that the argument $(\mathcal{P}, \mathcal{V})$ has* perfect completeness *if for all non-uniform polynomial-time interactive adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x) \rangle = 1 \\ \vee\ (\sigma, x, w) \notin \mathcal{R} \end{array} \middle| \begin{array}{c} \sigma \leftarrow \mathcal{K}(1^\lambda); \\ (x, w) \leftarrow \mathcal{A}(\sigma) \end{array}\right] = 1.$$

**Definition 2.** *Let $\mathcal{K}$ be the CRS generator and $(\mathcal{P}, \mathcal{V})$ be an argument. We say that the argument $(\mathcal{P}, \mathcal{V})$ has* witness-extended emulation *if for every deterministic polynomial prover $\mathcal{P}^*$ there exists an expected polynomial time emulator $\mathcal{E}$ such that for all non-uniform polynomial time interactive adversaries $\mathcal{A}$, the following inequality holds.*

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}(tr) = 1 \middle| \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda); (x, s) \leftarrow \mathcal{A}(\sigma); \\ tr \leftarrow \langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle \end{array}\right] \\ - \Pr\left[\begin{array}{l} \mathcal{A}(tr) = 1\ \wedge \\ \textit{if } tr \textit{ is accepting}, \\ \textit{then } (\sigma, x, w) \in \mathcal{R} \end{array} \middle| \begin{array}{l} \sigma \leftarrow \mathcal{K}(1^\lambda); (x, s) \leftarrow \mathcal{A}(\sigma); \\ (tr, w) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle}(\sigma, x) \end{array}\right] \end{array} \right| < negl(\lambda),$$

*$\mathcal{E}$ has access to the oracle $\langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle$ that permits rewinding $\mathcal{P}^*$ to a specific round and rerunning $\mathcal{V}$ using fresh randomness.*

In Definition 2, the value $s$ can be regarded to be the state of $\mathcal{P}^*$, including the randomness. Therefore, whenever $\mathcal{P}^*$ can make a convincing argument when in state $s$, $\mathcal{E}$ can extract a witness. Therefore, we call such an argument $(\mathcal{P}, \mathcal{V})$ satisfying Definition 1 and Definition 2 argument of knowledge and the argument is formalized in Definition 3.

**Definition 3.** *The argument $(\mathcal{P}, \mathcal{V})$ is called an* argument of knowledge *for relation $\mathcal{R}$ if the argument has (perfect) completeness and (computational) witness-extended emulation.*

**Transparent Setup 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 [6]; 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 [14].

Any public coin interactive argument protocol defined in Definition 4 can be converted into a non-interactive one by applying the Fiat-Shamir heuristic [28] in the random oracle model; all $\mathcal{V}$'s challenges can be replaced with hash values of the transcript up to that point.

**Definition 4.** *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 $\mathcal{G}_1$ be a group generator such that $\mathcal{G}_1$ takes $1^\lambda$ as input and outputs $\mathbb{G}$, the description of a group of order $p$.

**Definition 5 (Discrete Logarithm Assumption).** *We say that $\mathbb{G}$ satisfies the discrete logarithm assumption if for all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

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

**Definition 6 (Double Pairing Assumption).** *We say that the asymmetric bilinear group generator $\mathcal{G}$ satisfies the double pairing assumption if for all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

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

Abe et al. [2] 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 $m$, $\{1, \dots, m\}$. Let $\mathcal{G}$ be an asymmetric bilinear group generator that takes the security parameter $\lambda$ 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 distinct (multiplicative) cyclic groups of order $p$ of length $\lambda$, $g$ and $H$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ is a non-degenerate bilinear map. In this paper, we preferably use lower case letters for elements in $\mathbb{G}_1$ and upper case letters for elements in $\mathbb{G}_2$. 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 it can be sequentially used when sticking several vectors.[3]

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.*

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. $(\circ_{k\in[m]}\boldsymbol{g}_k)^{\sum_{j\in[\ell]} \boldsymbol{z}_j} = \prod_{k\in[m]} \prod_{j\in[\ell]} \boldsymbol{g}_k^{\boldsymbol{z}_j} \in \mathbb{G}_i$.
3. For $\boldsymbol{g} = (g_1, \ldots, g_n) \in \mathbb{G}_1^n, \boldsymbol{H} = (H_1, \ldots, H_n) \in \mathbb{G}_2^n$, 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. $\boldsymbol{E}(\circ_{k\in[m]}\boldsymbol{g}_k, \circ_{j\in[\ell]}\boldsymbol{H}_j) = \prod_{k\in[m]} \prod_{j\in[\ell]} \boldsymbol{E}(\boldsymbol{g}_k, \boldsymbol{H}_j) \in \mathbb{G}_t$.

*Scalar-Vector Operations.*

1. For $c \in \mathbb{Z}_p$ and $\boldsymbol{a} \in \mathbb{Z}_p^m$, the scalar multiplication is denoted by $c \cdot \boldsymbol{a} := (c \cdot a_1, \ldots, c \cdot a_n) \in \mathbb{Z}_p^m$.
2. 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 $\boldsymbol{g}^c := (g_1^c, \ldots, g_n^c) \in \mathbb{G}_i^m$.
3. 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 $g^{\boldsymbol{c}} := (g^{c_1}, \ldots, g^{c_m}) \in \mathbb{G}_i^m$.

## 3 Sublogarithmic Proofs via Generalization of BP-IP

In this section, we present our first generalization of BP-IP for the following relation $\mathcal{R}_{\mathsf{IP}}$ and then reduce its communication cost using the newly proposed aggregation technique, where BP-IP is given in Supplementary Material A.1.

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

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

In each 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

[3] 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.

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 $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to

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

Let $I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$ 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 description of the above generalized BP-IP and the proof for the perfect completeness and the soundness are relegated to Supplementary Material B.

**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, 1), which is identical the BP-IP protocol. Therefore, the generalization in this section does not have any benefit, at least compared with the original BP-IP.

## 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, 2]. We first slightly extend our target relation by adding the commitment key of [1, 2] into the common random string in our argument as follows.

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

where $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$ is an asymmetric bilinear group satisfying the discrete logarithm relation assumption in $\mathbb{G}_1$ and the double pairing assumption, and $\boldsymbol{g}, \boldsymbol{h}$,$u$,$\boldsymbol{F}_k$, 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.

As aforementioned, 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.

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

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., $\boldsymbol{v}^{\boldsymbol{z}}$).[4] This type of extension is well formalized by Bünz, Maller, Mishra, and Vesely [21] in terms of two-tiered homomorphic commitment scheme [35]. The multi-exponentiation argument in [21] 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 [35] and the multi-exponentiation argument in [21] so that we can construct a protocol with shorter communication overhead, denoted by Protocol1. The full description of the scheme is provided in Fig. 7 in Supplementary Material B.2.

**Efficiency Analysis** Although this approach reduces communication overheads, compared to the generalized BP-IP, it is not quite beneficial for our

[4] 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., [52]. Our multi-exponentiation argument corresponds to this variant.

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.

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

**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 [39]. 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.
**Our solution: augmented aggregate multi-exponentiation argument.** 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.

$$\left\{ \begin{pmatrix} \boldsymbol{F}_k \in \mathbb{G}_2^{2n(2n-1)}, \boldsymbol{z}_k \in \mathbb{Z}_p^{2n(2n-1)}, H_k \in \mathbb{G}_2, P_k \in \mathbb{G}_t, q_k \in \mathbb{G}_1 \\ ; \boldsymbol{v}_{k,j} \in \mathbb{G}^{2n(2n-1)} \text{ for } k, j \in [m] \end{pmatrix} \right. \\ \left. : \bigwedge_{k \in [m]} \left( P_k = \prod_{j \in [m]} \boldsymbol{E}(\boldsymbol{v}_{k,j}, \boldsymbol{F}_j) \wedge q_k = \boldsymbol{v}_{k,k}^{\boldsymbol{z}_k} \wedge (\boldsymbol{v}_{k,j}^{\boldsymbol{z}_j} = 1_{\mathbb{G}_1} \text{ for } j \neq k) \right) \right\} \quad (4)$$

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 mentioned in the previous paragraph. 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 use $H_k$'s in the product argument, which will be explained in the next section, where we reduce from the augmented aggregation multi-exponentiation protocol.

The full description of Protocol2 using aAggMEA is given in Fig. 1.

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

The proof of Theorem 1 is relegated to Supplementary Material C.1.

**Efficiency Analysis** A main difference between Protocol1 and Protocol2 is the aggregating process for $\log_{2n} N$ multi-exponentiation arguments. Due to communication-efficient feature of aAggMEA, the communication overhead is improved from $O((\log n) \cdot \log_{2n} N)$ to $O(\log n + \log_{2n} N)$. If we can set $n$ to satisfy $O(\log_{2n} N) = O(\log n)$, 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 (see Supplementary Material C.2 and C.3 for the detail), 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.[5]

## 3.3 Aggregating Multi-Exponentiation Argument

In this section, we propose an augmented aggregation of multi-exponentiation arguments aAggMEA for the relation in Eq. (4). Vectors in Eq. (4) are of dimension $2n(2n-1)$. For the sake of simplicity, we set the dimension of vectors $N$

[5] 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.

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

**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 $i, j \in I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$,

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

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 send $v$ to $\mathcal{V}$.

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

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

In addition, $\mathcal{P}$ computes

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

**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. 1.** $\mathsf{Protocol2}$: Sublogarithmic IP Argument

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 $\mathsf{aAggMEA}$ is reduced to a proof system, denoted by $\mathsf{ProdMEA}$, 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. 2 and its security property is given in the following theorem.

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

**Step 1**: $\mathcal{V}$ chooses and sends $y \xleftarrow{\$} \mathbb{Z}_p$ 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. 2.** Reduction from aAggMEA to ProdMEA

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

The proof of Theorem 2 is relegated to Supplementary Material C.2.

The construction and the detailed explanation of the protocol ProdMEA is relegated to Supplementary Material C.3. The idea for the construction of ProdMEA is to use the resemblance to the inner-product between integers. Indeed, the relation $\mathcal{R}_{PMEA}$ can be considered as the IP argument between a vector of group element $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ and an integer vector $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m)$ since $P$ 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 inner-product between $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ and $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m)$.

## 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 discrete logarithm 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[6] 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 a 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 discrete logarithm relation assumption in terms of problem instance sampler to generalize it. Let $\mathsf{GDLRsp}$ be a sampler that takes the security parameter $\lambda$ as input and outputs $(p, g_1, \dots, g_n, \mathbb{G})$, where $\mathbb{G}$ is a group $\mathbb{G}$ of $\lambda$-bit prime-order $p$ and $g_1, \dots, g_n$ are generators of $\mathbb{G}$.

**Definition 7 (General Discrete Logarithm Relation Assumption).** *Let* $\mathsf{GDLRsp}$ *be a sampler. We say that* $\mathsf{GDLRsp}$ *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* $\lambda$.

**Definition 8.** *For a fixed integer* $N$*, the sampler* $\mathsf{GDLRsp}_{Rand}$ *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.** $\mathsf{GDLRsp}_{Rand}$ *satisfies the GDLR assumption if the discrete logarithm assumption holds for the same underlying group* $\mathbb{G}$.

In fact, the security 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 security theorem of BP-IP below.

[6] Note that this operation is also called "projecting bilinear map" in the context of converting composite-order bilinear groups to prime-order bilinear groups [29].

**Theorem 4 ([19]).** *The BP-IP (given in Fig. 4) has perfect completeness and computational witness-extended-emulation under the general discrete logarithm assumption.*

We propose another sampler that satisfies the GDLR assumption.

**Definition 9.** *For fixed integers $m$ and $n$, the sampler $\mathsf{GDLRsp}_{BM}$ is defined as follows.*

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

**Theorem 5.** $\mathsf{GDLRsp}_{BM}$ *satisfies the GDLR assumption if the discrete logarithm 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 $\mathbb{G}_t$.[7]

*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 $\boldsymbol{z} \overset{\$}{\leftarrow} \mathbb{Z}_p^n$ 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 general discrete logarithm relation assumption on $\mathbb{G}_2$ with a sampler $\mathsf{GDLRsp}_{Rand}$. Theorem 3 guarantees that there is no type-2 adversary breaking the GDLR assumption with $\mathsf{GDLRsp}_{BM}$ under the DL assumption on $\mathbb{G}_2$.

[7] Note that the DL assumption on $\mathbb{G}_1$ implies the DL assumption on $\mathbb{G}_t$ by the MOV attack [45].

*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 $\boldsymbol{x} = (x_1, \dots, x_m) \xleftarrow{\$} \mathbb{Z}_p^m$, $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_p^n$, and $w \xleftarrow{\$} \mathbb{Z}_p$, and set $\boldsymbol{g} = (g^{x_1}, \dots, g^{x_{k-1}}, \hat{g}, g^{x_{k+1}}, \dots, g^{x_m})$, $\boldsymbol{H} = H^{\boldsymbol{z}}$, and $u = e(g, H)^w$. Then, the distribution of $(\boldsymbol{g}, \boldsymbol{H}, u)$ is identical to the real GDLR instance. Let $\hat{\boldsymbol{x}} = (x_1, \dots, x_{k-1}, Dlog_g\hat{g}, x_{k+1}, \dots, x_m)$. Then, $\boldsymbol{g} = g^{\hat{\boldsymbol{x}}}$.

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 discrete logarithm relation assumption holds. The discrete logarithm assumption implies the general discrete logarithm relation 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 arbitrary sampler satisfying the general discrete logarithm relation assumption, instead of $\mathsf{GDLRsp}_{Rand}$ 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.

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m, \boldsymbol{H} \in \mathbb{G}_2^n, u, P \in \mathbb{G}_t;\ \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}) \\ :\ P = (\boldsymbol{g} \otimes \boldsymbol{H})^{\boldsymbol{a}} (\boldsymbol{h} \otimes \boldsymbol{H})^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G}_t \end{array} \right\} \tag{5}$$

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 general discrete logarithm relation assumption under the discrete logarithm 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.[8] 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

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

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 $\mathsf{Protocol3}$, 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 $e(g_1, \boldsymbol{H})$ and $e(h_1, \boldsymbol{H})$, 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 $\mathsf{Protocol3}$, in Fig. 3. The number of rounds and the communication cost in $\mathsf{Protocol3}$ 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 $e(g_1, \boldsymbol{H})$ and $e(h_1, \boldsymbol{H})$, but using a similar trick in the case $(m > 1)$, the verifier can update $\boldsymbol{H}$ only instead of $e(g_1, \boldsymbol{H})$ and $e(h_1, \boldsymbol{H})$ and then perform constant pairing operations only at the final stage.

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

**Theorem 6.** *The argument presented in Fig. 3 for the relation* (5) *has perfect completeness and computational witness-extended-emulation under the general discrete logarithm assumption with the sampler* $\mathsf{GDLRsp}_{BM}$.

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

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

**If** $m = 1$: $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP}_{\mathsf{IP}}(e(g_1, \boldsymbol{H}), e(h_1, \boldsymbol{H}), u, P; \boldsymbol{a}, \boldsymbol{b})$.
**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 $x \xleftarrow{\$} \mathbb{Z}_p^*$ 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. 3.** Protocol3: Another Generalization of BP-IP

*Proof.* Although the verification cost in Protocol3 is reduced compared with BP-IP, the prover's and the verifier's 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 in Theorem 7 except that the general discrete logarithm relation assumption is guaranteed by Theorem 5 instead of Theorem 3. □

## 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 [26]. 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 $2 \times 2^{20}$ group operations for verification. Protocol3 requires $2 \times 2^{10}$ $\mathbb{G}_1$ operations and $2 \times 2^{10}$ $\mathbb{G}_2$ operations for verification. According to the implementation results from [49], 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 the verification time of Protocol3 may be significantly faster (approximately 70×) than that of BP-IP.

## 5 Sublinear Verifier without Pairing

In this section, we propose another IP argument with sublinear verifier, particularly without pairings. The crucial ingredient for Protocol3 is pairing-based homomorphic commitments to group elements [2], 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 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 $\boldsymbol{g} \in \mathbb{G}_1^m$ 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 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 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** In this section, we use a pair of elliptic curve groups, denoted by $(\mathbb{G}_p, \mathbb{G}_q)$, of distinct order $p$ and $q$ such that $\mathbb{G}_p := E(\mathbb{Z}_q)$ and both $p$ and $q$ are primes. 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 sum of 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, Costello, and Batina [47] based on the work of Bosma and Lenstra [17].

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.,

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

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 [47, 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 it for group operations of $E(\mathbb{Z}_q)$ in our construction. For the convenience of readers, we provide the complete addtion formula and the algorithm given by Renes et al. in Supplementary Material D.1.

## 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 hold 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 $2\lambda$ for the security parameter $\lambda$ [48].

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 $\boldsymbol{a}$.[9] 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 Pederesen commitment with a *different* commitment key $\boldsymbol{G} = (G_1, \ldots, G_{3n}) \in \mathbb{G}_q^{3n}$. More precisely, the commitment to $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} = (X_i, Y_i, Z_i)_{i=1}^n$ is defined as

$$\prod_{i=1}^{n} G_{3i-2}^{X_i} G_{3i-1}^{Y_i} G_{3i}^{Z_i}, \text{ which is denoted by } \mathsf{Com}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}}; \mathbf{G}).$$

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) [19] allow 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 Supplementary Material D.6. If we prove $d$ group operations, then the circuit size is $O(d \cdot n)$, so that both the computational cost for the prover and the verifier is $O(d \cdot n)$ and the cost for round and communication is $O(\log n + \log d)$.

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 anormal 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.

[9] 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.

### 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.

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

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[10] 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. We present $\mathsf{Protocol4.Row}$ in Fig. 9 in Supplementary Material D.2.

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 sublinear 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 are 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 $\boldsymbol{p}_k = \boldsymbol{l}_k^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}^{x_k^{-2}}$, 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. (7) (for $\mathsf{Protocol4.row}$)

[10] 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 $P_k$.

and Eq. (8) (for 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 AggMEC.Row (AggMEC.Col, resp.), Protocol4.Row (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.).

$$\mathcal{R}_{\mathsf{AggMEC.Row}} = \left\{ \begin{array}{l} \left( \begin{bmatrix} (\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k) \\ (\ \cdot\ , \boldsymbol{F}_{\ell+1}, \cdot\ , P_{\ell+1}, \cdot\ ) \end{bmatrix} ; \begin{bmatrix} (\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k) \\ (\ \cdot, \ \cdot, \boldsymbol{p}_{\ell+1}) \end{bmatrix} \text{ for } k \in [\ell] \right) : \\ \wedge_{j=1}^{\ell+1} \left( P_j = \mathsf{Com}(\boldsymbol{p}_j; \boldsymbol{F}_j) \right) \\ \wedge_{k=1}^{\ell} \left( S_k = \mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k; \boldsymbol{S}_k) \wedge \boldsymbol{p}_k = \boldsymbol{l}_k^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k^{x_k^{-2}} \right) \\ \boxed{\wedge_{k=1}^{\ell} \boldsymbol{l}_k, \boldsymbol{r}_k \in \mathbb{G}_p^{2n} \wedge \boldsymbol{p}_{\ell+1} \in \mathbb{G}_p^{2n}} \end{array} \right\} \quad (7)$$

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

$$\mathcal{R}_{\mathsf{AggMEC.Col}} = \left\{ \begin{array}{l} \left( \begin{bmatrix} (\boldsymbol{D}_k, P_k, x_k) \\ (\boldsymbol{D}_{\ell+1}, P_{\ell+1}, \ \ \cdot\ \ ) \end{bmatrix} ; \begin{bmatrix} (\boldsymbol{p}_k) \text{ for } k \in [\ell] \\ (\boldsymbol{p}_{\ell+1}) \end{bmatrix} \right) : \\ \wedge_{j=1}^{\ell+1} \big( P_j = \mathsf{Com}(\boldsymbol{p}_j; \boldsymbol{D}_j) \big) \wedge \boxed{\boldsymbol{p}_{\ell+1} \in \mathbb{G}_p^{2^{\ell+1}}} \\ \wedge_{k=1}^{\ell} \big( \boldsymbol{p}_k = (\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} \big) \\ \text{where } \boldsymbol{p}_{k+1} = \boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1} \parallel \boldsymbol{p}_{4,k+1} \end{array} \right\} \quad (8)$$

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 Supplementary Material D.

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** The main protocol Protocol4.Row (Fig. 9) calls Protocol4.Col (Fig. 10) and AggMEC.Row (Fig. 11) as subprotocols that again calls AggMEC.Col (Fig. 12) as a subprotocol. We analyze the efficiency of each protocol step-by-step. The detailed analysis is given in Supplementary Material D.7. Below, we denote group operations and elements in a group $G$ by $G$-operations and elements, respectively. We also note that because the number of field operations (elements) in $\mathbb{Z}_p$ is negligible compared to those in $\mathbb{G}_q$ for all protocols, we neglect them in complexity analysis.

Roughly, the complexity of AggMEC.Row is dominated by proving the following relation by running $\mathsf{Comp.BP}_{AC}$.

$$\boldsymbol{l}_k{}^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k{}^{x_k^{-2}} - \boldsymbol{p}_k = \boldsymbol{0} \text{ for } k \in [\ell] \quad (9)$$

(As discussed in 5.2, proving a group membership is cheaper than proving a group operation.) Eq. (9) consists of $O(n\ell \log p)$ $\mathbb{G}_p$-operations with a small constant hidden in the big-O notation and a single $\mathbb{G}_p$-operation requires 12 multiplications in $\mathbb{Z}_q$. Thus the arithmetic circuit for computing Eq. (9) consists of $O(n\ell \log p)$ multiplication gates. Then, $\mathcal{P}$ and $\mathcal{V}$ invoke $\mathsf{Comp.BP}_{AC}$ for such arithmetic circuit with parameter setting $N \leftarrow O(n\ell \log p)$ and $M \leftarrow O(n\ell)$, which costs $O(n\ell \log p \log q)$ $\mathbb{G}_q$-operations for each $\mathcal{P}$ and $\mathcal{V}$ and transmissions of $O(\log n + \log \ell + \log\log p)$ $\mathbb{G}_q$-elements.

The efficiency of $\mathsf{AggMEC.Col}$ can be analyzed similarly with $\mathsf{AggMEC.Row}$. An arithmetic circuit computing $(\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} - \boldsymbol{p}_k = \mathbf{0} \in \mathbb{G}_p^{2k}$ for $k \in [\ell]$ consists of $O(2^\ell \log p)$ multiplication gates over $\mathbb{Z}_q$. Executing $\mathsf{Comp.BP}_{AC}$ for such the circuit with parameter setting $N \leftarrow O(2^\ell \log p)$ and $M \leftarrow O(2^\ell)$, the total computation complexity of both $\mathcal{P}$ and $\mathcal{V}$ becomes $O(2^\ell \log p \log q)$ $\mathbb{G}_q$-operations and the communication complexity equals to $O(\ell + \log\log p)$ $\mathbb{G}_q$-elements.

We proceed to $\mathsf{Protocol4.Col}$. First, we analyze the reduction part. The reduction part is basically equivalent to that of $\mathsf{Protocol3}$, except that it uses different commitment scheme and the most computational cost of $\mathcal{V}$ shifts to $n = 1$. Thus, the computational cost of $\mathcal{P}$ is linear in $n$, which is $O(n \log q)$ $\mathbb{G}_q$-operations, and $\mathcal{V}$ performs field operations only. The communication cost is logarithmic in $n$, that is, $O(\log n)$ $\mathbb{G}_q$-elements. When $n = 1$, computation and communication complexities are dominated by the execution of $\mathsf{AggMEC.Col}$ with parameter setting $\ell \leftarrow \log n$. Thus, the total computation complexity is $O(n \log p \log q)$ $\mathbb{G}_q$-operations and the communication complexity is $O(\log n + \log\log p)$ $\mathbb{G}_q$-elements.

Finally, we analyze the efficiency of $\mathsf{Protocol4.Row}$, which is also basically equivalent to that of $\mathsf{Protocol3}$, except for using different commitment scheme. In the reduction part ($m > 1$), 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. When $m = 1$, the complexities are determined by the subprotocol $\mathsf{AggMEC.Row}$ with $\ell \leftarrow \log m$ and $\mathsf{Protocol4.Col}$. Therefore, $\mathcal{P}$'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, 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 $f(x)$. In Supplementary Material E.1, the formal definition of commitment scheme and polynomial commitment scheme is provided. We also present how to use the proposed

IP arguments as polynomial commitments and provide a comparison table in Supplementary Material E.2.

### 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 [20]. 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 compelxity 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 [52, 20]. For example, we can extend commitment schemes used in the paper to have hiding factor 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 [15, 19]. We can apply this reduction to our IP arguments. We relegate the details in Supplementary Material E.2.

## Acknowledgement

We thank Taechan Kim for discussion on complete addition formulas for elliptic curves.

## References

1. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, 2010.
2. 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.
3. 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.
4. M. Backes, A. Datta, and A. Kate. Asynchronous computational vss with reduced communication complexity. In *CT-RSA 2013*, volume 7779 of *LNCS*, pages 259–276. Springer, 2013.

5. 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.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 1993*, pages 62–73. ACM, 1993.
7. 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.
8. 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.
9. 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.
10. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016*, volume 9986 of *LNCS*, pages 31–60. Springer, 2016.
11. 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.
12. 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.
13. 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.
14. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.
15. 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.
16. 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.
17. W. Bosma and H. W. Lenstra. Complete systems of two addition laws for elliptic curves. *Journal of Number Theory*, 53:229–240, 1995.
18. S. Bowe, J. Grigg, and D. Hopwood. Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021, 2019. https://eprint.iacr.org/2019/1021.
19. 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.
20. B. Bünz, B. Fisch, and A. Szepieniec. Transparent snarks from dark compilers. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 677–706. Springer, 2020.
21. B. Bünz, M. Maller, P. Mishra, and N. Vesely. Proofs for inner pairing products and applications. Cryptology ePrint Archive, Report 2019/1177, 2019. https://eprint.iacr.org/2019/1177.
22. J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 262–288. Springer, 2015.
23. J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, 2010.

24. 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.
25. 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.
26. dalek-cryptography: Bulletproofs. https://github.com/dalek-cryptography/bulletproofs, 2018.
27. 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.
28. 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.
29. 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.
30. G. Fuchsbauer, C. Hanser, and D. Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, 2019.
31. 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.
32. 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.
33. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 192–208. Springer, 2009.
34. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, 2010.
35. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 431–448. Springer, 2011.
36. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016.
37. 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.
38. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *SIAM Journal on Computing*, volume 41(5), page 1193–1232, 2012.
39. 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.
40. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, 2010.
41. R. W. F. Lai, G. Malavolta, and V. Ronge. Succinct arguments for bilinear group arithmetic: Practical structure-preserving cryptography. In *ACM CCS 2019*, pages 2057–2074, 2019.
42. libsnark. https://github.com/scipr-lab/libsnark, 2017.
43. 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.

44. 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.
45. 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.
46. 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.
47. J. Renes, C. Costello, and L. Batina. Complete addition formulas for prime order elliptic curves. In M. Fischlin and J. Coron, editors, *EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 403–428. Springer, 2016.
48. E. Savas, T. A. Schmidt, and Ç. K. Koç. Generating elliptic curves of prime order. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2001.
49. 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.
50. 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.
51. S. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *CRYPTO 2020*, volume 12172 of *LNCS*, pages 704–737. Springer, 2020.
52. 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.
53. 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.
54. J. Xu, A. Yang, J. Zhou, and D. S. Wong. Lightweight delegatable proofs of storage. In *ESORICS 2016*, volume 9878 of *LNCS*, pages 324–343. Springer, 2016.
55. 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.

# Supplementary Materials

# Supplementary Materials

## A Bulletproofs [19]

### A.1 Inner-Product Argument (without Zero-Knowledge)

We review Bulletproofs inner-product argument, which is denoted by $\mathsf{BP_{IP}}$ and given in Fig. 4, for the relation in Eq. (1).

$$\boxed{\mathsf{BP_{IP}}(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, u, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^N)}$$

```
If $N = 1$:
    Step 1: $\mathcal{P}$ sends $\mathcal{V}$ $a$ and $b$.
    Step 2: $\mathcal{V}$ outputs Accepts if and only if $P = g^a h^b u^{a \cdot b}$ holds.
Else ($N > 1$): Let $\widehat{N} = \frac{N}{2}$ and parse $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to
        $\boldsymbol{a} = \boldsymbol{a}_1 \| \boldsymbol{a}_{-1}, \quad \boldsymbol{b} = \boldsymbol{b}_1 \| \boldsymbol{b}_{-1}, \quad \boldsymbol{g} = \boldsymbol{g}_1 \| \boldsymbol{g}_{-1}, \quad \text{and } \boldsymbol{h} = \boldsymbol{h}_1 \| \boldsymbol{h}_{-1}.$
    Step 1: $\mathcal{P}$ calculates
        $L = \boldsymbol{g}_{-1}^{\boldsymbol{a}_1} \boldsymbol{h}_1^{\boldsymbol{b}_{-1}} u^{\langle \boldsymbol{a}_1, \boldsymbol{b}_{-1} \rangle} \in \mathbb{G}$ and $R = \boldsymbol{g}_1^{\boldsymbol{a}_{-1}} \boldsymbol{h}_{-1}^{\boldsymbol{b}_1} u^{\langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle} \in \mathbb{G}$,
      and then sends $L, R$ to $\mathcal{V}$.
    Step 2: $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.
    Step 3: Both $\mathcal{P}$ and $\mathcal{V}$ compute
        $\widehat{\boldsymbol{g}} = \boldsymbol{g}_1^{x^{-1}} \circ \boldsymbol{g}_{-1}^{x} \in \mathbb{G}^{\widehat{N}}, \ \widehat{\boldsymbol{h}} = \boldsymbol{h}_1^{x} \circ \boldsymbol{h}_{-1}^{x^{-1}} \in \mathbb{G}^{\widehat{N}}$, and $\widehat{P} = L^{x^2} P R^{x^{-2}} \in \mathbb{G}$.
      Additionally, $\mathcal{P}$ computes
        $\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1} \in \mathbb{Z}_p^{\widehat{N}}$ and $\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{N}}$.
    Step 4: Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP_{IP}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$.
```

**Fig. 4.** Bulletproofs (Inner-Product Argument)

The main idea of BP-IP is recursive reduction from an argument for $N$-length witness to an argument for ($\widehat{N} = \frac{N}{2}$)-length witness. For the sake of simplicity, we assume that $N$ is a power of 2 and one can pad the input if need be. First, the prover parses $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to two vectors of $\widehat{N}$-length, respectively, as follows.

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

$\mathcal{P}$ begins by computing and sending $\mathcal{V}$

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

$\mathcal{V}$ chooses a random challenge $x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$. Next, both $\mathcal{P}$ and $\mathcal{V}$ compute a common input for the next step argument

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

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

$$\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1} \in \mathbb{Z}_p^{\widehat{N}} \quad \text{and} \quad \widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{N}}.$$

One can easily check that the updated $\widehat{P}$ equals to the followings.

$$\begin{aligned}
&L^{x^2} P \, R^{x^{-2}} \\
=&(\boldsymbol{g}_{-1}^{\boldsymbol{a}_1} \boldsymbol{h}_1^{\boldsymbol{b}_{-1}} u^{\langle \boldsymbol{a}_1, \boldsymbol{b}_{-1} \rangle})^{x^2} (\boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}) (\boldsymbol{g}_1^{\boldsymbol{a}_{-1}} \boldsymbol{h}_{-1}^{\boldsymbol{b}_1} u^{\langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle})^{x^{-2}} \\
=&\boldsymbol{g}_1^{\boldsymbol{a}_1 + x^{-2} \boldsymbol{a}_{-1}} \boldsymbol{g}_{-1}^{x^2 \boldsymbol{a}_1 + \boldsymbol{a}_{-1}} \cdot \boldsymbol{h}_1^{x^2 \boldsymbol{b}_{-1} + \boldsymbol{b}_1} \boldsymbol{h}_{-1}^{\boldsymbol{b}_{-1} + x^{-2} \boldsymbol{b}_1} \cdot u^{x^2 \langle \boldsymbol{a}_1, \boldsymbol{b}_{-1} \rangle + \langle \boldsymbol{a}, \boldsymbol{b} \rangle + x^{-2} \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle} \\
=&(\boldsymbol{g}_1^{x^{-1}} \circ \boldsymbol{g}_{-1}^{x})^{x \boldsymbol{a}_1 + x^{-1} \boldsymbol{a}_{-1}} \cdot (\boldsymbol{h}_1^{x} \circ \boldsymbol{h}_{-1}^{x^{-1}})^{x^{-1} \boldsymbol{b}_1 + x \boldsymbol{b}_{-1}} \cdot u^{\langle x \boldsymbol{a}_1 + x^{-1} \boldsymbol{a}_{-1}, x^{-1} \boldsymbol{b}_1 + x \boldsymbol{b}_{-1} \rangle} \\
=&\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}} \cdot \widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}} \cdot u^{\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle}
\end{aligned}$$

Thus, $\widehat{P}$ satisfies again an inner-product relation $\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}} \cdot \widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}} \cdot u^{\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle}$ with a half-length witness $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$. Next, both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP}_{\mathsf{IP}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$ together. The full description of BP-IP is provided in Fig. 8.

**Theorem 7 ([19]).** *The inner-product argument of Bulletproofs (given in Fig. 4) has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption.*

In the above recursive step, $\mathcal{P}$ sends only two group elements in $\mathbb{G}$ and the length of witness vectors becomes a half in the next round. Thus, it requires $O(\log_2 N)$ group elements during $O(\log N)$ rounds. The verifier computes $\widehat{g}$ and $\widehat{h}$ that require $O(N)$ exponentiation for each round, but these computations can be optimized to be a single multi-exponentiation of $N$-length for all rounds.

## A.2 Zero-Knowledge Arguments for Arithmetic Circuits

Bulletproofs presents another zero-knowledge argument protocol that is for arbitrary arithmetic circuits. More precisely. this protocol proves the following relation and the detailed description of protocol is given in Fig. 5.

$$\left\{ \begin{array}{l} \left( \begin{array}{l} \boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, \boldsymbol{V} \in \mathbb{G}^M, g, h \in \mathbb{G}, \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O \in \mathbb{Z}_p^{Q \times N}, \boldsymbol{W}_V \in \mathbb{Z}_p^{Q \times M}, \\ \boldsymbol{c} \in \mathbb{Z}_p^Q; \quad \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O \in \mathbb{Z}_p^N, \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^M \end{array} \right) \\ : V_j = g^{v_j} h^{\gamma_j} \forall j \in [1, m] \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O \\ \quad \wedge \ \boldsymbol{W}_L \cdot \boldsymbol{a}_L + \boldsymbol{W}_R \cdot \boldsymbol{a}_R + \boldsymbol{W}_O \cdot \boldsymbol{a}_O = \boldsymbol{W}_V \cdot \boldsymbol{v} + \boldsymbol{c} \end{array} \right\}$$

$\boxed{\mathsf{BP}_{AC}(\boldsymbol{g},\boldsymbol{h},\boldsymbol{V},g,h,\boldsymbol{W}_L,\boldsymbol{W}_R,\boldsymbol{W}_O,\boldsymbol{W}_V,\boldsymbol{c};\boldsymbol{a}_L,\boldsymbol{a}_R,\boldsymbol{a}_O,\boldsymbol{v},\boldsymbol{\gamma})}$

**Step 1**: $\mathcal{P}$ computes
$$\alpha,\beta,\rho\xleftarrow{\$}\mathbb{Z}_p,\ A_I=h^{\alpha}\boldsymbol{g}^{\boldsymbol{a}_L}\boldsymbol{h}^{\boldsymbol{a}_R}\in\mathbb{G},\ A_O=h^{\beta}\boldsymbol{g}^{\boldsymbol{a}_O}\in\mathbb{G},$$
$$\boldsymbol{s}_L,\boldsymbol{s}_R\xleftarrow{\$}\mathbb{Z}_p^N,\ S=h^{\rho}\boldsymbol{g}^{\boldsymbol{s}_L}\boldsymbol{h}^{\boldsymbol{s}_R}\in\mathbb{G}$$

and send $A_I, A_O$, and $S$ to $\mathcal{V}$.

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

**Step 3**: Both $\mathcal{P}$ and $\mathcal{V}$ set

$$\boldsymbol{y}^N:=(1,y,y^2,\dots,y^{N-1})\in\mathbb{Z}_p^N,\quad \boldsymbol{z}^{Q+1}:=(z,z^2,\dots,z^Q)\in\mathbb{Z}_p^Q,$$
$$\delta(y,z):=\langle\boldsymbol{y}^{-N}\circ(\boldsymbol{z}^{Q+1}\cdot\boldsymbol{W}_R),\boldsymbol{z}^{Q+1}\cdot\boldsymbol{W}_L\rangle,$$
$$h_i':=h_i^{y^{-i+1}},\ \forall i\in[N]\text{ and }\boldsymbol{h}':=(h_1',\dots,h_N').$$

Additionally, $\mathcal{P}$ computes

$$l(X):=\boldsymbol{a}_L\cdot X+\boldsymbol{a}_O\cdot X^2+\boldsymbol{y}^{-N}\circ(\boldsymbol{z}^{Q+1}\cdot\boldsymbol{W}_R)\cdot X+\boldsymbol{s}_L\cdot X^3\in\mathbb{Z}_p^N[X],$$
$$r(X):=\boldsymbol{y}^N\circ\boldsymbol{a}_R\cdot X-\boldsymbol{y}^N+\boldsymbol{z}^{Q+1}\cdot(\boldsymbol{W}_L\cdot X+\boldsymbol{W}_O)+\boldsymbol{y}^N\circ\boldsymbol{s}_R\cdot X^3\in\mathbb{Z}_p^N[X],$$
$$t(X):=\langle l(X),r(X)\rangle=\sum_{i\in[6]}t_i\cdot X^i\in\mathbb{Z}_p[X],$$
$$\boldsymbol{w}:=\boldsymbol{W}_L\cdot\boldsymbol{a}_L+\boldsymbol{W}_R\cdot\boldsymbol{a}_R+\boldsymbol{W}_O\cdot\boldsymbol{a}_O,$$
$$t_2:=\langle\boldsymbol{a}_L,\boldsymbol{a}_R\circ\boldsymbol{y}^N\rangle-\langle\boldsymbol{a}_O,\boldsymbol{y}^N\rangle+\langle\boldsymbol{z}^{Q+1},\boldsymbol{w}\rangle+\delta(y,z)\in\mathbb{Z}_p,$$
$$\tau_i\xleftarrow{\$}\mathbb{Z}_p\ \forall i\in\{1,3,4,5,6\},\quad T_i:=g^{t_i}h^{\tau_i}\ \forall i\in\{1,3,4,5,6\},$$

and send $T_i$ for $i\in\{1,3,4,5,6\}$ to $\mathcal{V}$.

**Step 4**: $\mathcal{V}$ chooses $x\xleftarrow{\$}\mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.

**Step 5**: $\mathcal{P}$ computes

$$\boldsymbol{l}:=l(x)\in\mathbb{Z}_p^N,\quad \boldsymbol{r}:=r(x)\in\mathbb{Z}_p^N,\quad \hat{t}:=\langle\boldsymbol{l},\boldsymbol{r}\rangle\in\mathbb{Z}_p,$$
$$\tau_x:=\sum_{i=1,i\neq 2}^{6}\tau_i\cdot\ x^i+x^2\cdot\langle\boldsymbol{z}^{Q+1},\boldsymbol{W}_V\cdot\boldsymbol{\gamma}\rangle\in\mathbb{Z}_p,\quad \mu:=\alpha\cdot x+\beta\cdot x^2+\rho\cdot x^3\in\mathbb{Z}_p$$

and send $\tau_x,\mu,\hat{t}$ to $\mathcal{V}$.

**Step 6**: $\mathcal{V}$ send a random challenge $w\xleftarrow{\$}\mathbb{Z}_p^*$ to $\mathcal{P}$.

**Step 7**: $\mathcal{V}$ checks the following equation.

$$g^{\hat{t}}h^{\tau_x}\overset{?}{=}g^{x^2\cdot(\delta(y,z)+\langle\boldsymbol{z}^{Q+1},\boldsymbol{c}\rangle)}\cdot\boldsymbol{V}^{x^2\cdot(\boldsymbol{z}^{Q+1}\cdot\boldsymbol{W}_V)}\cdot T_1^x\cdot\prod_{i=3}^{6}T_i^{x^i}$$

If the equation holds, both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{g},\boldsymbol{h}',g^w,P\cdot h^{-\mu}\cdot g^{w\cdot\hat{t}};\boldsymbol{l},\boldsymbol{r})$ where

$$P=A_I^x\cdot A_O^{x^2}\cdot\boldsymbol{g}^{\boldsymbol{y}^{-N}\circ(\boldsymbol{z}^{Q+1}\cdot\boldsymbol{W}_R)}\cdot\boldsymbol{h}'^{-\boldsymbol{y}^N+\boldsymbol{z}^{Q+1}(\boldsymbol{W}_O+x\boldsymbol{W}_L)}\cdot S^{x^3}$$

**Fig. 5.** Bulletproofs (Arithmetic Circuit Argument)

```
$\boxed{\mathsf{GBP}_{\mathsf{IP}}(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, u, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^N)}$
If $N = 1$:
    Step 1: $\mathcal{P}$ sends $\mathcal{V}$ $a$ and $b$.
    Step 2: $\mathcal{V}$ outputs Accepts if and only if $P = g^a h^b u^{a \cdot b}$ holds.
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 $i, j \in I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$,
          $v_{i,j} = \boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j \boldsymbol{b}_i \rangle} \in \mathbb{G}$,
      and then sends $\{v_{i,j}\}_{\substack{i,j \in I_n \\ i \neq j}}$ to $\mathcal{V}$.
    Step 2: $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.
    Step 3: Both $\mathcal{P}$ and $\mathcal{V}$ compute
          $\widehat{\boldsymbol{g}} = \circ_{i \in I_n} \boldsymbol{g}_i^{x^{-i}} \in \mathbb{G}^{\widehat{N}},\ \widehat{\boldsymbol{h}} = \circ_{i \in I_n} \boldsymbol{h}_i^{x^i} \in \mathbb{G}^{\widehat{N}}$ and $\widehat{P} = P \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x^{j-i}} \in \mathbb{G}$.
      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 4: Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{GBP}_{\mathsf{IP}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$.
```

**Fig. 6.** Our First Generalization of BP-IP

## B Our First Generalization of BP-IP

The concrete description of the generalized BP-IP is given in Fig. 6.

We provide a sketch of the soundness proof. After receiving $v_{i,j}$'s, we rewind $\mathcal{P}^*$ $4n-1$ times and obtain $4n-1$ challenge-witness tuples $(x_k, \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k)$ for $k \in [4n-1]$. Then, we know that the extracted witness satisfies the following equality for all $k \in [4n-1]$.

$$P \prod_{s \in J_n} \Big( \prod_{\substack{i,j \in I_n \\ j-i=s}} v_{i,j} \Big)^{x_k^s} = P \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x_k^{j-i}} = \widehat{P} = \big( \circ_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i}} \big)^{\widehat{\boldsymbol{a}}_k} \big( \circ_{j \in I_n} \boldsymbol{h}_j^{x_k^j} \big)^{\widehat{\boldsymbol{b}}_k} u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle}$$
$$= \Big( \prod_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i} \widehat{\boldsymbol{a}}_k} \boldsymbol{h}_i^{x_k^i \widehat{\boldsymbol{b}}_k} \Big) u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \tag{10}$$

where $J_n := \{\pm 2, \pm 4, \pm 6, \ldots, \pm(4n-4), \pm(4n-2)\}$ of size $4n-2$. Assuming that all $x_i^2$'s are distinct, one can prove that the $(4n-1) \times (4n-1)$ matrix $M$ with $(k, j)$-entry $x_k^{-4n+2j}$ is invertible, where $M$'s $k$-th row is a vector of exponents used in the left-hand side of Eq. (10). Thus, we can use $M^{-1}$ to find exponents

$\{\boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r}\}_{r\in I_n}, c_P$ and $\{\boldsymbol{a}_{s,r}, \boldsymbol{b}_{s,r}\}_{r\in I_n}, c_s$ for $s \in J_N$ satisfying

$$P = \Big(\prod_{r\in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{P,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{P,r}}\Big) u^{c_P} \in \mathbb{G}, \qquad \Big(\prod_{\substack{i,j\in I_n\\ j-i=s}} v_{i,j}\Big) = \Big(\prod_{r\in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{s,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{s,r}}\Big) u^{c_s} \in \mathbb{G}.$$

Therefore, we successfully extract the exponents with bases $\boldsymbol{g}_r$ and $\boldsymbol{h}_r$ for $r \in I_n$, which are the witness vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. Of course, we must show that the exponent $c_P$ should be equal to $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$. To this end, we can use more rewinding to extract a tuple satisfying Eq. (10) and obtain the following theorem.

**Theorem 8.** *The inner-product argument in Fig. 6 has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption.*

The complete proof of Theorem 8 is given in the next subsection.

## B.1 Proof of Theorem 8

*Proof. (completeness)* For the recursive step, we observe that

$$P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a},\boldsymbol{b}\rangle} = \prod_{i\in I_n} \boldsymbol{g}_i^{\boldsymbol{a}_i} \boldsymbol{h}_i^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_i,\boldsymbol{b}_i\rangle}, \text{ and thus}$$

$$\begin{aligned}
\widehat{P} &= \Big(\prod_{i\in I_n} \boldsymbol{g}_i^{\boldsymbol{a}_i} \boldsymbol{h}_i^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_i,\boldsymbol{b}_i\rangle}\Big) \cdot \Big(\prod_{\substack{i,j\in I_n\\ i\neq j}} \big(\boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j,\boldsymbol{b}_i\rangle}\big)^{x^{j-i}}\Big) = \prod_{i,j\in I_n} \big(\boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j,\boldsymbol{b}_i\rangle}\big)^{x^{j-i}}\\
&= \big(\circ_{i\in I_n} \boldsymbol{g}_i^{x^{-i}}\big)^{\sum_{j\in I_n} \boldsymbol{a}_j x^j} \cdot \big(\circ_{j\in I_n} \boldsymbol{h}_j^{x^j}\big)^{\sum_{i\in I_n} \boldsymbol{b}_i x^{-i}} \cdot u^{\langle \sum_{j\in I_n} \boldsymbol{a}_j x^j, \sum_{i\in I_n} \boldsymbol{b}_i x^{-i}\rangle}\\
&= \widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}} \cdot \widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}} \cdot u^{\langle \widehat{\boldsymbol{a}},\widehat{\boldsymbol{b}}\rangle}.
\end{aligned} \tag{11}$$

Therefore, $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$ satisfies the IP relation in Eq. (1), and thus an argument of $N$-length is reduced to the same argument of $\widehat{N}$-length. Eq (11) guarantees the completeness of the protocol. For $N = 1$, it is straightforward since the prover provides a witness $(a, b)$ and the verifier checks the relation.

*(witness-extended emulation)* In order to show the computational witness-extended emulation, we construct an expected polynomial time extractor $\chi$ whose goal is to extract the witness by using a polynomially bounded tree of accepting transcripts. If then, similar to BP-IP, we can apply the following general forking lemma.

**Theorem 9 (General Forking Lemma [15]).** *Let $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ be a $(2k+1)$-move, public coin interactive protocol with $\mu$ challenges $x_1, \ldots, x_\mu$ in sequence. Let $n_i \geq 1$ for $1 \leq i \leq \mu$. Consider an $(n_1, \ldots, n_k)$-tree of accepting transcripts with challenges in the following tree format. The tree has depth $\mu$ and $\prod_{i=1}^{\mu} n_i$ leaves. The root of the tree is labeled with the statement. Each node of depth $i < \mu$ has exactly $n_i$ children, each labeled with a distinct value of the $i$-th challenge $x_i$.*

*Let $\chi$ be a witness extraction algorithm that succeeds with probability $1 - neg(\lambda)$ for some negligible function $neg(\lambda)$ in extracting a witness from an $(n_1, \ldots, n_k)$-tree of accepting transcripts in probabilistic polynomial time. Assume that $\prod_{i=1}^{k} n_i$ is bounded above by a polynomial in the security parameter $\lambda$. Then, $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation.*

The case $(N = 1)$ is straightforward since the prover sends the witness and then the verifier can directly check the correctness. Let us focus on the case $(N > 1)$. We prove that for each recursive step that on input $(\boldsymbol{g}, \boldsymbol{h}, u, P)$, we can efficiently extract from the prover a witness $(\boldsymbol{a}, \boldsymbol{b})$ under the discrete logarithm relation assumption whose instance is the CRS $(\boldsymbol{g}, \boldsymbol{h}, u)$. First, the extractor runs the prover to get $v_{i,j}$ for $i \neq j \in I_n$. At this point, the extractor rewinds the prover $12n - 5$ times and feeds $12n - 5$ non-zero challenges $x_k$ such that all $x_k^2$ are distinct. Then, the extractor obtains $12n - 5$ pairs $\widehat{\boldsymbol{a}}_k$ and $\widehat{\boldsymbol{b}}_k$ such that for each $k \in [12n - 5]$

$$\begin{aligned} P \prod_{s \in J_n} \Big( \prod_{\substack{i,j \in I_n \\ j-i=s}} v_{i,j} \Big)^{x_k^s} = P \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x_k^{j-i}} &= \widehat{P} \\ &= \Big( \circ_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i}} \Big)^{\widehat{\boldsymbol{a}}_k} \Big( \circ_{j \in I_n} \boldsymbol{h}_j^{x_k^{j}} \Big)^{\widehat{\boldsymbol{b}}_k} u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \\ &= \Big( \prod_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i} \widehat{\boldsymbol{a}}_k} \boldsymbol{h}_i^{x_k^{i} \widehat{\boldsymbol{b}}_k} \Big) u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \qquad (12) \end{aligned}$$

where $J_n := \{\pm 2, \pm 4, \pm 6, \ldots, \pm(4n-4), \pm(4n-2)\}$ of size $4n - 2$. We know that squares of the first $4n-1$ challenges, $x_1^2, \ldots, x_{4n-1}^2$, are distinct, so that the following matrix $M \in \mathbb{Z}_p^{(4n-1)\times(4n-1)}$ is invertible since it is a product of two invertible matrices, where one of which is a diagonal matrix and the other is a Vandermonde matrix with distinct rows.

$$\begin{aligned} M = & \begin{bmatrix} x_1^{-4n+2} & x_1^{-4n+4} & \cdots 1 \cdots & x_1^{4n-4} & x_1^{4n-2} \\ x_2^{-4n+2} & x_2^{-4n+4} & \cdots 1 \cdots & x_2^{4n-4} & x_2^{4n-2} \\ \vdots & \vdots & \cdots 1 \cdots & \vdots & \vdots \\ x_{4n-1}^{-4n+2} & x_{4n-1}^{-4n+4} & \cdots 1 \cdots & x_{4n-1}^{4n-4} & x_{4n-1}^{4n-2} \end{bmatrix} \\ = & \begin{bmatrix} x_1^{-4n+2} & 0 & \cdots & 0 \\ 0 & x_2^{-4n+2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & x_{4n-1}^{-4n+2} \end{bmatrix} \cdot \begin{bmatrix} 1 & x_1^2 & \cdots & x_1^{4n-2} & \cdots & x_1^{8n-6} & x_1^{8n-4} \\ 1 & x_2^2 & \cdots & x_2^{4n-2} & \cdots & x_2^{8n-6} & x_2^{8n-4} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 1 & x_{4n-1}^2 & \cdots & x_{4n-1}^{4n-2} & \cdots & x_{4n-1}^{8n-6} & x_{4n-1}^{8n-4} \end{bmatrix}. \end{aligned}$$

The extractor knows all the exponents $x_k^{j-i}$, $x_k^{-i}$, $x_k^{j}$, $\widehat{\boldsymbol{a}}_k$, and $\widehat{\boldsymbol{b}}_k$ in Eq. (12). There are $4n-1$ distinct powers of $x_k^2$ in the left-hand side in Eq. (12). Thus, by using the inverse matrix of $M$ and the elementary linear algebra in the public exponents of the first $4n - 1$ equalities in Eq. (12), we can find the exponents

$\{\boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r}\}_{r \in I_n}, c_P$ and $\{\boldsymbol{a}_{s,r}, \boldsymbol{b}_{s,r}\}_{r \in I_n}, c_s$ for $s \in J_N$ satisfying

$$P = \big(\prod_{r \in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{P,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{P,r}}\big) u^{c_P} \in \mathbb{G}, \tag{13}$$

$$\big(\prod_{\substack{i,j \in I_n \\ j-i=s}} v_{i,j}\big) = \big(\prod_{r \in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{s,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{s,r}}\big) u^{c_s} \in \mathbb{G}. \tag{14}$$

Next, we prove that the extracted exponents $\{\boldsymbol{a}_{P,r}\}_{r \in I_n}, \{\boldsymbol{b}_{P,r}\}_{r \in I_n}, c_P$ satisfy the desired relation $c_P = \sum_{r \in I_n} \langle \boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r} \rangle = \langle \boldsymbol{a}_P, \boldsymbol{b}_P \rangle$, where $\boldsymbol{a}_P$ and $\boldsymbol{b}_P$ are concatenations of $\boldsymbol{a}_{P,r}$'s and $\boldsymbol{b}_{P,r}$'s, respectively. Putting Eq. (13) and Eq. (14) into Eq. (12), we have for each $k \in [12n-5]$,

$$\begin{aligned} &\big(\prod_{r \in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{P,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{P,r}}\big) u^{c_P} \cdot \prod_{s \in J_n} \big(\prod_{r \in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{s,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{s,r}}\big)^{x_k^s} u^{c_s x_k^s} \\ &= \big(\prod_{r \in I_n} \boldsymbol{g}_r^{x_k^{-r} \widehat{\boldsymbol{a}}_k} \boldsymbol{h}_r^{x_k^r \widehat{\boldsymbol{b}}_k}\big) u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle}. \end{aligned}$$

This can be rewritten as

$$\begin{aligned} &\big(\prod_{r \in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{P,r} + \sum_{s \in J_n} x_k^s \boldsymbol{a}_{s,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{P,r} + \sum_{s \in J_n} x_k^s \boldsymbol{b}_{s,r}}\big) u^{c_P + \sum_{s \in J_n} c_s x_k^s} \\ &= \big(\prod_{r \in I_n} \boldsymbol{g}_r^{x_k^{-r} \widehat{\boldsymbol{a}}_k} \boldsymbol{h}_r^{x_k^r \widehat{\boldsymbol{b}}_k}\big) u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \text{ for } k \in [12n-5]. \end{aligned}$$

By the discrete logarithm relation assumption, the above equality implies that for $k \in [12n-5]$ and $r \in I_n$

$$\boxed{\boldsymbol{g}_r \text{ exponents}} : \boldsymbol{a}_{P,r} + \textstyle\sum_{s \in J_n} \boldsymbol{a}_{s,r} x_k^s = \widehat{\boldsymbol{a}}_k x_k^{-r} \tag{15}$$

$$\boxed{\boldsymbol{h}_r \text{ exponents}} : \boldsymbol{b}_{P,r} + \textstyle\sum_{s \in J_n} \boldsymbol{b}_{s,r} x_k^s = \widehat{\boldsymbol{b}}_k x_k^{r} \tag{16}$$

$$\boxed{u \text{ exponents}} : \quad c_P + \textstyle\sum_{s \in J_n} c_s x_k^s = \langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle. \tag{17}$$

If we find exponents satisfying Eq. (12), Eq. (13) and Eq. (14), but not one of the above Eq. (15), Eq. (16) and Eq. (17), it directly implies a non-trivial relation between the CRS and so we break the discrete logarithm relation assumption in $\mathbb{G}_1$.

As an intermediate step toward the relation $c_P = \sum_{r \in I_n} \langle \boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r} \rangle$, we find a relation between $\boldsymbol{a}_{P,r}$ and $\widehat{\boldsymbol{a}}_k$ and a relation between $\boldsymbol{b}_{P,r}$ and $\widehat{\boldsymbol{b}}_k$ since such relations can be combined with Eq. (17) to find the relation $c_P = \sum_{r \in I_n} \langle \boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r} \rangle$. From Eq. (15) and Eq. (16), we can remove $\widehat{\boldsymbol{a}}_k$ and $\widehat{\boldsymbol{b}}_k$ and deduce that for each $k \in [12n-5]$ and any $r, r' \in I_n$,

$$\boldsymbol{a}_{P,r} x_k^r + \sum_{s \in J_n} \boldsymbol{a}_{s,r} x_k^{s+r} - \boldsymbol{a}_{P,r'} x_k^{r'} - \sum_{s \in J_n} \boldsymbol{a}_{s,r'} x_k^{s+r'} = 0 \tag{18}$$

$$\boldsymbol{b}_{P,r} x_k^{-r} + \sum_{s \in J_n} \boldsymbol{b}_{s,r} x_k^{s-r} - \boldsymbol{b}_{P,r'} x_k^{-r'} - \sum_{s \in J_n} \boldsymbol{b}_{s,r'} x_k^{s-r'} = 0. \tag{19}$$

In both Eq. (18) and Eq. (19), degrees of $x_k$ range between $6n-3$ and $-6n+3$ according to $r \in I_n$ and $s \in J_n$. That is, the number of distinct integers in the range is $12n-5$ that is equal to the number of distinct challenges $x_k$ for $k \in [12n-5]$. That implies that the following polynomial equations in the variable $X$ hold for any $r, r' \in I_n$.

$$\boldsymbol{a}_{P,r}X^r + \sum_{s\in J_n} \boldsymbol{a}_{s,r}X^{s+r} - \boldsymbol{a}_{P,r'}X^{r'} - \sum_{s\in J_n} \boldsymbol{a}_{s,r'}X^{s+r'} = 0$$
$$\text{and } \boldsymbol{b}_{P,r}X^{-r} + \sum_{s\in J_n} \boldsymbol{b}_{s,r}X^{s-r} - \boldsymbol{b}_{P,r'}X^{-r'} - \sum_{s\in J_n} \boldsymbol{b}_{s,r'}X^{s-r'} = 0.$$

This can be rewritten as

$$\boldsymbol{a}_{P,r}X^r + \sum_{s\in J_n} \boldsymbol{a}_{s,r}X^{s+r} \text{ is the same polynomial for all } r \in I_n. \quad (20)$$
$$\boldsymbol{b}_{P,r}X^{-r} + \sum_{s\in J_n} \boldsymbol{b}_{s,r}X^{s-r} \text{ is the same polynomial for all } r \in I_n. \quad (21)$$

The only way to hold the above two equations is that

1. The polynomial in Eq. (20) is $\sum_{s\in I_n} \boldsymbol{a}_{P,s}X^s$, regardless of $r$.
2. The polynomial in Eq. (21) is $\sum_{s\in I_n} \boldsymbol{b}_{P,s}X^{-s}$, regardless of $r$.

Fix an $r \in I_n$, say $r = 2n-1$. Then, putting the above result into Eq. (20) and Eq. (21), we have

$$\boldsymbol{a}_{P,2n-1}X^{2n-1} + \sum_{s\in J_n} \boldsymbol{a}_{s,2n-1}X^{s+2n-1} = \sum_{s\in I_n} \boldsymbol{a}_{P,s}X^s$$
$$\boldsymbol{b}_{P,2n-1}X^{-2n+1} + \sum_{s\in J_n} \boldsymbol{b}_{s,2n-1}X^{s-2n+1} = \sum_{s\in I_n} \boldsymbol{b}_{P,s}X^{-s}$$

Combining the above result with Eq. (15) and Eq. (16), we have for $k \in [12n-5]$

$$\boldsymbol{a}_{P,2n-1}x_k^{2n-1} + \sum_{s\in J_n} \boldsymbol{a}_{s,2n-1}x_k^{s+2n-1} = \widehat{\boldsymbol{a}}_k = \sum_{s\in I_n} \boldsymbol{a}_{P,s}x_k^s$$
$$\boldsymbol{b}_{P,2n-1}x_k^{-2n+1} + \sum_{s\in J_n} \boldsymbol{b}_{s,2n-1}x_k^{s-2n+1} = \widehat{\boldsymbol{b}}_k = \sum_{s\in I_n} \boldsymbol{b}_{P,s}x_k^{-s}.$$

Thus, we obtain the relations $\widehat{\boldsymbol{a}}_k = \sum_{s\in I_n} \boldsymbol{a}_{P,s}x_k^s$ and $\widehat{\boldsymbol{b}}_k = \sum_{s\in I_n} \boldsymbol{b}_{P,s}x_k^{-s}$ for $k \in [12n-5]$, which is the intermediate step toward the desired relation $c_P = \sum_{r\in I_n}\langle \boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r}\rangle$.

As aforementioned, we combine these relations with Eq. (17) and obtain that for $k \in [12n-5]$

$$c_P + \sum_{s\in J_n} c_s x_k^s = \langle \sum_{s\in I_n} \boldsymbol{a}_{P,s}x_k^s, \sum_{s\in I_n} \boldsymbol{b}_{P,s}x_k^{-s}\rangle = \sum_{s,s'\in I_n} \langle \boldsymbol{a}_{P,s}, \boldsymbol{b}_{P,s'}\rangle x_k^{s-s'}$$
$$= \sum_{\substack{s,s'\in I_n \\ s'\neq s}} \langle \boldsymbol{a}_{P,s}, \boldsymbol{b}_{P,s'}\rangle x_k^{s-s'} + \sum_{s\in I_n} \langle \boldsymbol{a}_{P,s}, \boldsymbol{b}_{P,s}\rangle$$

Since this relation holds for all $x_1, \ldots, x_{12n-5}$, it must be that

$$c_P = \sum_{s \in I_n} \langle \boldsymbol{a}_{P,s}, \boldsymbol{b}_{P,s} \rangle = \langle \boldsymbol{a}_P, \boldsymbol{b}_P \rangle.$$

Therefore, we construct the extractor that outputs $\boldsymbol{a}_P, \boldsymbol{b}_P$ and $c_P$ satisfying the above inner-product relation. The extractor rewinds $12n - 5$ times for each recursive step. Thus, it uses $(12n-5)^{\log_{2n} N}$ transcripts in total and thus runs in expected polynomial time in $N$ and $\lambda$. Then, by the general forking lemma, we conclude that the argument has computational witness-extended emulation. □

## B.2 Protocol1 : Generalization with Multi-Exponentiation Argument

We provide a full description of our generalized BP-IP with Multi-Exponentiation Argument, denoted by Protocol1 in Fig. 7.

In the protocol description , 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.

```
$\boxed{\mathsf{Protocol1}(\boldsymbol{g}, \boldsymbol{h}, u, \boldsymbol{F}_k \text{ for } k \in [m], P \in \mathbb{G}_1, st_V; \boldsymbol{a}, \boldsymbol{b}, st_P), \text{ where } m = \log_{2n} N}$
**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. 7.** Protocol1

## C Sublogarithmic Protocol from Aggregated Multi-Exponentiation Arguments

### C.1 Protocol2: Proof of Theorem 1

*Proof. (Completeness)* For $N > 1$, the protocol resembles the generalization of the BP-IP in Fig. 6. It is sufficient to show that for the case $(N = 1)$, the correctly generated state information $st_P$ successfully passes the augmented aggregation of multi-exponentiation argument protocols. For $k \in [m]$, $u_k = \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)$ and $v_k = \boldsymbol{v}_k^{\boldsymbol{x}_k}$, where $\boldsymbol{x}_k$ is the challenge vector used in the $k$-th recursive stage. Since the prover sets $\boldsymbol{v}_{k,j} = \boldsymbol{1}_{\mathbb{G}_1}$ for $j \neq k$ and $\boldsymbol{v}_{k,k} = \boldsymbol{v}_k$ for $k \in [m]$, we have

$$u_k = \prod_{j\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,j}, \boldsymbol{F}_j) \wedge v_k = \boldsymbol{v}_{k,k}^{\boldsymbol{x}_k} \wedge (\boldsymbol{v}_{k,j}^{\boldsymbol{x}_j} = 1_{\mathbb{G}_1} \text{ for } j \neq k),$$

which is the relation in Eq. (4). Therefore, it will correctly pass values to the aAggMEA protocol.

*(Witness-Extended-Emulation)* Due to the general forking lemma, it is sufficient to construct an extractor $\chi$ that extracts a witness from a suitable tree of accepting transcripts in probabilistic polynomial time.

We begin with $(\underbrace{12n-5,\ldots,12n-5}_{\log_{2n} N}, 2m, \underbrace{7,\ldots,7}_{\log(2n(2n-1)\log_{2n} N)})$-tree of accepting transcriptions. Note that the number of all challenges in the tree is $(12n-5)^{\log_{2n} N} \cdot 2m \cdot 7^{\log(2n(2n-1)\log_{2n} N)}$ that is bounded above by a polynomial in $N$ and $\lambda$, so that we can apply the general forking lemma.

If $N = 1$ and $st_P$ is empty, then it is straightforward since the verifier receives a witness $a$ and $b$ satisfying $P = g^a h^b u^{ab}$ and check the validity for any accepting transcript in the tree. If $N = 1$ but $st_P$ is non-empty, then we use the fact that the lower position subtree of $(1+\log(2n(2n-1)\log_{2n} N)$-depth is $(2m, 7, \ldots, 7)$-tree of accepting transcripts for aAggMEA. aAggMEA has the witness-extended emulation under the DPair assumption by Theorem 2 and Theorem 11. Using the $(2m, 7, \ldots, 7)$-tree of accepting transcripts, the extractor can extract $\boldsymbol{v}_{k,j}$ satisfying

$$\bigwedge_{k\in[m]} \Big(u_k = \prod_{j\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,j}, \boldsymbol{F}_j) \;\wedge\; v_k = \boldsymbol{v}_{k,k}^{\boldsymbol{x}_k} \;\wedge\; (\boldsymbol{v}_{k,j}^{\boldsymbol{x}_j} = 1_{\mathbb{G}_1} \text{ for } j \neq k)\Big), \quad (22)$$

for each accepting transcript in the upper position $(\underbrace{12n-5,\ldots,12n-5}_{\log_{2n} N})$-subtree of the main tree. Note that for $j \neq k$, $\boldsymbol{v}_{k,j}$ may not equal to $\boldsymbol{1}_{\mathbb{G}_1}$ unlike the original protocol. However, what we need from Eq. (22) is only the relation $v_k = \boldsymbol{v}_{k,k}^{\boldsymbol{x}_k}$ such that $\boldsymbol{v}_{k,k}$ is committed by the prover before receiving $\boldsymbol{x}_k$.

All the remaining process is equivalent to the proof of Theorem 8; the format of the upper position subtree of accepting transcripts is exactly the same as that required in the proof of generalized BP-IP in Theorem 8. Furthermore, for each upper position accepting transcript, we have shown that the extractor can extract the $\boldsymbol{v}_{k,k}$, which is used in the computation of $\widehat{P} = P \cdot v_k$ in the $k$-th recursive round. Thus, what the extractor has for each accepting transcript for this protocol is exactly the same as that of the extractor for the generalized BP-IP protocol, so that we can employ the extractor in the proof of Theorem 8 under the discrete logarithm relation assumption in $\mathbb{G}_1$.

As we aforementioned, we use $(\underbrace{12n-5, \dots, 12n-5}_{\log_{2n} N}, 2m, \underbrace{7, \dots, 7}_{\log(2n(2n-1)\log_{2n} N)})$-tree of accepting transcriptions, which has leaves of polynomially bounded size in $N$ and $\lambda$. Therefore, we can apply the general forking lemma. □

## C.2 aAggMEA: Proof of Theorem 2

We begin with providing the definition of the $q$-double pairing assumption, which will be used in the proof for the witness-extended emulation of the scheme.

**Definition 10 ($q$-Double Pairing Assumption).** *Let $\mathcal{G}$ be an asymmetric bilinear group generator. We say that $\mathcal{G}$ satisfies the q-Double Pairing assumption in $\mathbb{G}_2$ if all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

$$\Pr\left[\begin{array}{c} \boldsymbol{g} \neq \mathbf{1}_{\mathbb{G}_1} \\ \wedge \\ \boldsymbol{E}(\boldsymbol{g}, \boldsymbol{F}) = 1_{\mathbb{G}_t} \end{array}\middle| \begin{array}{c} (p, g, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e) \leftarrow \mathcal{G}(1^\lambda); \\ \boldsymbol{F} \overset{\$}{\leftarrow} \mathbb{G}_2^q; \\ \boldsymbol{g} \leftarrow \mathcal{A}(p, \boldsymbol{F}, g, H, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e) \end{array}\right] < negl(\lambda),$$

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

**Theorem 10.** *The q-double pairing assumption holds in $\mathbb{G}_2$ for polynomially bounded q if the double pairing is assumed hard. In particular, the reduction is tight.*

*Proof.* Assume that there exists a PPT adversary $\mathcal{A}$ who breaks the $q$-double paring assumption for polynomial-size $q$. Given a double pairing instance $(G, G^a) \in \mathbb{G}_2^2$, we construct a reduction using $\mathcal{A}$ as a subroutine. The reduction chooses a random vector $(r_1, \dots, r_q) \overset{\$}{\leftarrow} \mathbb{Z}_p^q$, sets $\boldsymbol{F} = ((G^a)^{r_1}, G^{r_2}, \dots, G^{r_q})$, and sends to $\mathcal{A}$. Due to the random exponents $r_i$'s, $\boldsymbol{F}$ looks uniformly distributed from $\mathcal{A}$'s viewpoint. Thus, $\mathcal{A}$ successfully outputs $\boldsymbol{g} = (g_1, \dots, g_q)$ such that $\boldsymbol{E}(\boldsymbol{g}, \boldsymbol{F}) = 1_{\mathbb{G}_t}$ with non-negligible probability. Then, the reduction outputs a pair $(g_1^{r_1}, g_2^{r_2} \cdots g_q^{r_q})$ as an answer to the double pairing problem.

Since $\boldsymbol{E}(\boldsymbol{g}, \boldsymbol{F}) = 1_{\mathbb{G}_t}$ holds, we know that the following should holds as well.

$$\boldsymbol{E}((g_1^{r_1}, \dots, g_q^{r_q}), (G^a, G, \dots, G)) = e(g_1^{r_1}, G^a) \cdot e(g_2^{r_2} \cdots g_q^{r_q}, G) = 1_{\mathbb{G}_t}$$

Thus, the above equality guarantees the correctness of the reduction's output.

It is clear that the reduction is tight in the sense that the reduction's advantage for breaking the double pairing assumption is exactly identical to the advantage of $\mathcal{A}$ for breaking the $q$-double pairing assumption. □

Now we are ready to prove Theorem 2 about the protocol for the following relation.

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

*(Completeness)* When setting $\tilde{\boldsymbol{F}}_k$, $\tilde{\boldsymbol{z}}_k$, $\tilde{H}_k$, $\tilde{P}$, and $\tilde{\boldsymbol{v}}_k$ as in the protocol description, we have

$$\begin{aligned}
&\tilde{P} \\
&= \prod_{k\in[m]} \Big(P_k^{y^{k-1}} \cdot e(q_k^{y^{k-1}}, H_k^{y^m})\Big) \\
&= \prod_{k\in[m]} \Big(\big(\prod_{j\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,j}\boldsymbol{F}_j)\big)^{y^{k-1}} \cdot e((\boldsymbol{v}_{k,k}^{\boldsymbol{z}_k})^{y^{k-1}}, H_k^{y^m})\Big) \\
&= \Big(\prod_{j\in[m]} \boldsymbol{E}(\circ_{k\in[m]}\boldsymbol{v}_{k,j}^{y^{k-1}}, \boldsymbol{F}_j)\Big) \cdot \Big(\prod_{k\in[m]} e(\boldsymbol{v}_{k,k}^{y^{k-1}\boldsymbol{z}_k}, H_k^{y^m})\Big) \\
&= \prod_{k\in[m]} \boldsymbol{E}(\circ_{j\in[m]}\boldsymbol{v}_{j,k}^{y^{j-1}}, \boldsymbol{F}_k) \cdot e(\boldsymbol{v}_{k,k}^{y^{k-1}\boldsymbol{z}_k}, H_k^{y^m}) \text{ // Change index } j \text{ with index } k \\
&= \prod_{k\in[m]} \boldsymbol{E}(\circ_{j\in[m]}\boldsymbol{v}_{j,k}^{y^{j-k}}, \boldsymbol{F}_k^{y^{k-1}}) \cdot e(\boldsymbol{v}_{k,k}^{y^{k-1}\boldsymbol{z}_k}, H_k^{y^m}) \\
&= \prod_{k\in[m]} \boldsymbol{E}(\tilde{\boldsymbol{v}}_k, \tilde{\boldsymbol{F}}_k) \cdot e\big((\circ_{j\in[m]}\boldsymbol{v}_{j,k}^{y^{j-k}})^{y^{k-1}\boldsymbol{z}_k}, \tilde{H}_k) \text{ // Use the relation } \boldsymbol{v}_{j,k}^{\boldsymbol{z}_k} = 1_{\mathbb{G}_1} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } j \neq k \\
&= \prod_{k\in[m]} \boldsymbol{E}(\tilde{\boldsymbol{v}}_k, \tilde{\boldsymbol{F}}_k) \cdot e(\tilde{\boldsymbol{v}}_k^{\tilde{\boldsymbol{z}}_k}, \tilde{H}_k)
\end{aligned}$$

Thus, we know that $\tilde{\boldsymbol{F}}_k$, $\tilde{\boldsymbol{z}}_k$, $\tilde{H}_k$, $\tilde{P}$, and $\tilde{\boldsymbol{v}}_k$ satisfy the relation in Eq. (5). Thus, if ProdMEA has the completeness, aAggMEA has also the completeness.

*(Witness-Extended-Emulation)* Due to the general forking lemma, it is sufficient to construct an extractor $\chi$ that extracts a witness from a suitable tree of accepting transcripts in probabilistic polynomial time.

Suppose that ProdMEA has witness-extended emulation and the double pairing assumption holds. Thus, we can employ the extractor for ProdMEA and the $q$-double pairing assumption holds in $\mathbb{G}_2$. The extractor rewinds the prover in the reduction by feeding $2m$ distinct challenges $y_i$ for $i \in [2m]$ and obtains the witness $\tilde{\boldsymbol{v}}_k^{(i)}$ for $k \in [m]$ and $i \in [2m]$ satisfying ProdMEA argument. That is, for $i \in [2m]$ we have

$$\begin{aligned}
\prod_{k\in[m]} \Big(P_k^{y_i^{k-1}} e(q_k, H_k)^{y_i^{m+k-1}}\Big) &= \tilde{P}_i \\
&= \prod_{j\in[m]} \boldsymbol{E}((\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{j-1}}, \boldsymbol{F}_j) e((\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{m+j-1}\boldsymbol{z}_j}, H_j). \quad (23)
\end{aligned}$$

All exponents in the above equation are powers of $y_i$. Let $M$ be an $2m \times 2m$ Vandermonde matrix with $i$-th row $(1, y_i, \ldots, y_i^{2m-1})$. Let $\boldsymbol{a}_k = (a_{k,1}, \ldots, a_{k,2m})$ be the $k$-th row of $M^{-1}$ for $k \in [2m]$. Then, using elementary linear algebra in the exponent of Eq. (23), the extractor can compute $\boldsymbol{w}_{k,j} \in \mathbb{G}_1^N$ and $\nu_{k,j} \in \mathbb{G}_1$ for $k, j \in [m]$ such that

$$P_k = \prod_{i=1}^{m} \tilde{P}_i^{a_{k,i}} = \prod_{j \in [m]} \boldsymbol{E}(\boldsymbol{w}_{k,j}, \boldsymbol{F}_j) e(\nu_{k,j}, H_j). \tag{24}$$

From now, we show that the extracted $\boldsymbol{w}_{k,j}$ and $\nu_{k,j}$ satisfy the desired relations $\nu_{k,j} = 1_{\mathbb{G}_1}$ for all $k, j \in [m]$, $q_k = \boldsymbol{w}_{k,k}^{\boldsymbol{z}_k}$ for all $k \in [m]$, and $\boldsymbol{w}_{j,k}^{z_k} = 1_{\mathbb{G}_1}$ for $j \neq k$. When we put the above equality into Eq. (23), we have

$$\prod_{k \in [m]} \Big( \big( \prod_{j \in [m]} \boldsymbol{E}(\boldsymbol{w}_{k,j}, \boldsymbol{F}_j)^{y_i^{k-1}} e(\nu_{k,j}, H_j)^{y_i^{k-1}} \big) \cdot e(q_k, H_k)^{y_i^{m+k-1}} \Big)$$
$$= \prod_{j \in [m]} \Big( \boldsymbol{E}(\tilde{\boldsymbol{v}}_j^{(i)^{y_i^{j-1}}}, \boldsymbol{F}_j) e((\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{m+j-1} \boldsymbol{z}_j}, H_j) \Big).$$

By the $q$-double pairing assumption, we can separate the above equation as follows. For each $i \in [2m]$ and $j \in [m]$, we have

$$\boxed{\boldsymbol{F}_j \text{ correspondence}} : \quad \circ_{k \in [m]} \boldsymbol{w}_{k,j}^{y_i^{k-1}} = (\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{j-1}} \tag{25}$$

$$\boxed{H_j \text{ correspondence}} : \big( \textstyle\prod_{k \in [m]} \nu_{k,j}^{y_i^{k-1}} \big) \cdot q_j^{y_i^{m+j-1}} = (\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{m+j-1} \boldsymbol{z}_j} \tag{26}$$

Combining Eq. (25) and Eq. (26), we can remove $\tilde{\boldsymbol{v}}_{i,j}$'s and obtain the followings.

$$\big( \prod_{k \in [m]} \nu_{k,j}^{y_i^{k-1}} \big) \cdot q_j^{y_i^{m+j-1}} = \prod_{k \in [m]} \boldsymbol{w}_{k,j}^{y_i^{m+k-1} \boldsymbol{z}_j} \tag{27}$$

We observe that all the values except $y_i$ in the above equation is fixed before choosing the challenge $y_i$, the degree of $y_i$'s vary between 0 and $2m-1$, and the above equation holds for each challenge $y_i$. Since the number of distinct challenges is larger than the maximum degree of $y_i$, the only way to hold Eq. (27) is that

| | Left-Hand Side | | Right-Hand Side |
|---|---|---|---|
| $\boxed{\text{For } k \in [m],\ y_i^{k-1} \text{ correspondence:}}$ | $\nu_{k,j}$ | $=$ | $1_{\mathbb{G}_1}$ |
| $\boxed{y_i^{m+j-1} \text{ correspondence:}}$ | $q_j$ | $=$ | $\boldsymbol{w}_{j,j}^{\boldsymbol{z}_j}$ |
| $\boxed{\text{For } k \neq j,\ y_i^{m+k-1} \text{ correspondence:}}$ | $1_{\mathbb{G}_1}$ | $=$ | $\boldsymbol{w}_{k,j}^{\boldsymbol{z}_j}$. |

Thus, putting the above result into Eq. (24), the extractor eventually has $\boldsymbol{w}_{k,j}$ for $k \in [m]$ satisfying

$$q_k = \boldsymbol{w}_{k,k}^{\boldsymbol{z}_k} \wedge P_k = \prod_{j \in [m]} \boldsymbol{E}(\boldsymbol{w}_{k,j}, \boldsymbol{F}_j) \wedge (\boldsymbol{w}_{k,j}^{\boldsymbol{z}_j} = 1_{\mathbb{G}_1} \text{ for all } j \neq k).$$

$\square$

## C.3 ProdMEA Protocol

We propose a ProdMEA protocol for the relation $\mathcal{R}_{PMEA}$. The ProdMEA protocol recursively reduces from an argument for $N$-length witness to an argument for $\widehat{N} = \frac{N}{2}$-length witness. First, parse $\boldsymbol{F}_k, \boldsymbol{z}_k$, and $\boldsymbol{v}_k$ to two vectors of $\widehat{N}$-length, respectively, as follows.

$$\boldsymbol{F}_k = \boldsymbol{F}_{k,1} \| \boldsymbol{F}_{k,-1}, \quad \boldsymbol{z}_k = \boldsymbol{z}_{k,1} \| \boldsymbol{z}_{k,-1}, \quad \text{and } \boldsymbol{v}_k = \boldsymbol{v}_{k,1} \| \boldsymbol{v}_{k,-1}$$

$\mathcal{P}$ begins with computing and sending $\mathcal{V}$ for $k \in [m]$

$$\begin{aligned} L &= \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}, \boldsymbol{F}_{k,-1}) e(\boldsymbol{v}_{k,1}^{\boldsymbol{z}_{k,-1}}, H_k) \in \mathbb{G}_t \\ \text{and } R &= \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,-1}, \boldsymbol{F}_{k,1}) e(\boldsymbol{v}_{k,-1}^{\boldsymbol{z}_{k,1}}, H_k) \in \mathbb{G}_t. \end{aligned}$$

$\mathcal{V}$ chooses a random challenge $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$. Next, both $\mathcal{P}$ and $\mathcal{V}$ compute a common input for the next step argument as follows. For $k \in [m]$,

$$\widehat{\boldsymbol{F}}_k = \boldsymbol{F}_{k,1}^{x^{-1}} \circ \boldsymbol{F}_{k,-1}^{x} \in \mathbb{G}_2^{\widehat{N}}, \ \widehat{\boldsymbol{z}}_k = \boldsymbol{z}_{k,1} x^{-1} + \boldsymbol{z}_{k,-1} x \in \mathbb{Z}_p^{\widehat{N}}, \ \widehat{P} = L^{x^2} P \, R^{x^{-2}} \in \mathbb{G}_t$$

and $\mathcal{P}$ computes a half-dimension witness for the next step argument for $k \in [m]$,

$$\widehat{\boldsymbol{v}}_k = \boldsymbol{v}_{k,1}^{x} \circ \boldsymbol{v}_{k,-1}^{x^{-1}} \in \mathbb{G}_1^{\widehat{N}}.$$

One can easily check that $\widehat{P}$ equals the followings.

$$\begin{aligned} \widehat{P} = &\Big( \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}, \boldsymbol{F}_{k,-1}) e(\boldsymbol{v}_{k,1}^{\boldsymbol{z}_{k,-1}}, H_k) \Big)^{x^2} \cdot \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k) e(\boldsymbol{v}_k^{\boldsymbol{z}_k}, H_k) \\ &\cdot \Big( \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,-1}, \boldsymbol{F}_{k,1}) e(\boldsymbol{v}_{k,-1}^{\boldsymbol{z}_{k,1}}, H_k) \Big)^{x^{-2}} \\ = &\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}^{x}, \boldsymbol{F}_{k,-1}^{x}) \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k) \boldsymbol{E}(\boldsymbol{v}_{k,-1}^{x^{-1}}, \boldsymbol{F}_{k,1}^{x^{-1}}) e(\boldsymbol{v}_{k,1}^{\boldsymbol{z}_{k,-1} x^2} \cdot \boldsymbol{v}_k^{\boldsymbol{z}_k} \cdot \boldsymbol{v}_{k,-1}^{\boldsymbol{z}_{k,1} x^{-2}}, H_k) \\ = &\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}^{x} \circ \boldsymbol{v}_{k,-1}^{x^{-1}}, \boldsymbol{F}_{k,-1}^{x} \circ \boldsymbol{F}_{k,1}^{x^{-1}}) e((\boldsymbol{v}_{k,1}^{x} \circ \boldsymbol{v}_{k,-1}^{x^{-1}})^{\boldsymbol{z}_{k,1} x^{-1} + \boldsymbol{z}_{k,-1} x}, H_k) \\ = &\prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k, \widehat{\boldsymbol{F}}_k) e(\widehat{\boldsymbol{v}}_k^{\widehat{\boldsymbol{z}}_k}, H_k). \end{aligned}$$

Thus, $\widehat{P}$ satisfies again a ProdMEA relation with half-length witness $\widehat{\boldsymbol{v}}_k$'s, so that both $\mathcal{P}$ and $\mathcal{V}$ run ProdMEA$(\widehat{\boldsymbol{F}}_k, \widehat{z}_k, H_k, \widehat{P}; \widehat{\boldsymbol{v}}_k)$ together. The full description of ProdMEA is provided in Fig. 8.

We briefly provide a sketch of the soundness proof. That is, given a successful prover $\mathcal{P}^*$, we extract a witness $\boldsymbol{v}_k$. The extractor runs $\mathcal{P}^*$ and receives $L$ and

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

**If** $N = 1$:

**Step 1**: $\mathcal{P}$ sends $v_1, \ldots, v_m$ to $\mathcal{V}$.

**Step 2**: $\mathcal{V}$ outputs *Accepts* if and only if $P = \prod_{k\in[m]} e(v_k, F_k)e(v_k^{z_k}, H_k)$ holds.

**Else** ($N > 1$):

Let $\widehat{N} = \frac{N}{2}$ and for $k \in [m]$ parse $\boldsymbol{F}_k$, $\boldsymbol{z}_k$ and $\boldsymbol{v}_k$ to $\boldsymbol{F}_k = \boldsymbol{F}_{k,1} \| \boldsymbol{F}_{k,-1}$, $\boldsymbol{z}_k = \boldsymbol{z}_{k,1} \| \boldsymbol{z}_{k,-1}$, and $\boldsymbol{v}_k = \boldsymbol{v}_{k,1} \| \boldsymbol{v}_{k,-1}$, respectively.

**Step 1**:$\mathcal{P}$ computes for $k \in [m]$

$$L = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}, \boldsymbol{F}_{k,-1})e(\boldsymbol{v}_{k,1}^{\boldsymbol{z}_{k,-1}}, H_k) \in \mathbb{G}_t$$

$$\text{and } R = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,-1}, \boldsymbol{F}_{k,1})e(\boldsymbol{v}_{k,-1}^{\boldsymbol{z}_{k,1}}, H_k) \in \mathbb{G}_t.$$

Then, $\mathcal{P}$ sends $L$ and $R$ 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

$$\widehat{\boldsymbol{F}}_k = \boldsymbol{F}_{k,1}^{x^{-1}} \circ \ \boldsymbol{F}_{k,-1}^{x} \in \mathbb{G}_2^{\widehat{N}}, \ \widehat{\boldsymbol{z}}_k = \boldsymbol{z}_{k,1}x^{-1} + \boldsymbol{z}_{k,-1}x \in \mathbb{Z}_p^{\widehat{N}},$$

$$\text{and } \widehat{P} = L^{x^2} P \ R^{x^{-2}} \in \mathbb{G}_t.$$

Additionally, $\mathcal{P}$ computes for $k \in [m]$, $\widehat{\boldsymbol{v}}_k = \boldsymbol{v}_{k,1}^{x} \circ \boldsymbol{v}_{k,-1}^{x^{-1}} \in \mathbb{G}_1^{\widehat{N}}$.

**Step 4**: Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{ProdMEA}(\widehat{\boldsymbol{F}}_k, \widehat{\boldsymbol{z}}_k, H_k, \widehat{P}; \widehat{\boldsymbol{v}}_k)$.

**Fig. 8.** ProdMEA protocol

$R$. By rewinding $\mathcal{P}^*$ three times and feeding $\mathcal{P}^*$ three challenges $x_i$ for $i \in [3]$ such that $x_i^2 \neq x_j^2$ for $i \neq j$, the extractor obtains $\widehat{\boldsymbol{v}}_k^{(i)}$'s for $i \in [3]$ and $k \in [m]$ satisfying the following.

$$\begin{aligned} L^{x_i^2} P R^{x_i^{-2}} &= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \widehat{\boldsymbol{F}}_k)e(\widehat{\boldsymbol{v}}_k^{(i)\widehat{\boldsymbol{z}}_k}, H_k) \\ &= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \boldsymbol{F}_{k,1}^{x_i^{-1}} \circ \boldsymbol{F}_{k,-1}^{x_i})e(\widehat{\boldsymbol{v}}_k^{(i)\boldsymbol{z}_{k,1}x_i^{-1} + \boldsymbol{z}_{k,-1}x_i}, H_k) \\ &= \prod_{i\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i^{-1}}, \boldsymbol{F}_{k,1})\boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i}, \boldsymbol{F}_{k,-1})e(\widehat{\boldsymbol{v}}_k^{(i)\boldsymbol{z}_{k,1}x_i^{-1} + \boldsymbol{z}_{k,-1}x_i}, H_k) \end{aligned} \quad (28)$$

Since $x_i^2$'s are distinct, the matrix $\begin{bmatrix} x_1^{-2} & 1 & x_1^2 \\ x_2^{-2} & 1 & x_2^2 \\ x_3^{-2} & 1 & x_3^2 \end{bmatrix}$ is invertible. Therefore, by using the elementary linear algebra in the exponent, we can obtain $\boldsymbol{g}_{P,k,1}, \boldsymbol{g}_{P,k,-1}$ and

$g_{P,k,H}$ for $k \in [m]$ satisfying

$$P = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{P,k,1}, \boldsymbol{F}_{k,1})\boldsymbol{E}(\boldsymbol{g}_{P,k,-1}, \boldsymbol{F}_{k,-1})e(g_{P,k,H}, H_k).$$

Therefore, we extract a witness $\boldsymbol{v}_k = \boldsymbol{g}_{P,k,1} \| \boldsymbol{g}_{P,k,-1}$ for $k \in [m]$. Of course, we have to show that $g_{P,k,H} = \boldsymbol{v}_k^{\boldsymbol{z}_k}$ for $k \in [m]$. To this end, we can use more rewinding to extract a tuple satisfying Eq. (28) and obtain the following theorem.

**Theorem 11.** *The* ProdMEA *protocol in Fig. 8 has perfect completeness and computational witness-extended-emulation under the double pairing assumption.*

*Proof. (Completeness)* If $N = 1$, it is straightforward to check the completeness of the scheme in Figure 8. For $N > 1$, we have

$$P = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)e(\boldsymbol{v}_k^{\boldsymbol{z}_k}, H_k)$$

and so

$$\begin{aligned}
\widehat{P} = &\Big(\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}, \boldsymbol{F}_{k,-1})e(\boldsymbol{v}_{k,1}^{\boldsymbol{z}_{k,-1}}, H_k)\Big)^{x^2} \cdot \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)e(\boldsymbol{v}_k^{\boldsymbol{z}_k}, H_k) \\
&\cdot \Big(\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,-1}, \boldsymbol{F}_{k,1})e(\boldsymbol{v}_{k,-1}^{\boldsymbol{z}_{k,1}}, H_k)\Big)^{x^{-2}}. \\
= &\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}^{x}, \boldsymbol{F}_{k,-1}^{x})\boldsymbol{E}(\boldsymbol{v}_k, \boldsymbol{F}_k)\boldsymbol{E}(\boldsymbol{v}_{k,-1}^{x^{-1}}, \boldsymbol{F}_{k,1}^{x^{-1}})e(\boldsymbol{v}_{k,1}^{\boldsymbol{z}_{k,-1}x^2} \cdot \boldsymbol{v}_k^{\boldsymbol{z}_k} \cdot \boldsymbol{v}_{k,-1}^{\boldsymbol{z}_{k,1}x^{-2}}, H_k) \\
= &\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{v}_{k,1}^{x} \circ \boldsymbol{v}_{k,-1}^{x^{-1}}, \boldsymbol{F}_{k,-1}^{x} \circ \boldsymbol{F}_{k,1}^{x^{-1}})e((\boldsymbol{v}_{k,1}^{x} \circ \boldsymbol{v}_{k,-1}^{x^{-1}})^{\boldsymbol{z}_{k,1}x^{-1}+\boldsymbol{z}_{k,-1}x}, H_k) \\
= &\prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k, \widehat{\boldsymbol{F}}_k)e(\widehat{\boldsymbol{v}}_k^{\widehat{\boldsymbol{z}}_k}, H_k).
\end{aligned}$$

Therefore, the completeness is satisfied.

*(Witness-Extended Emulation)* The double pairing assumption implies the $q$-double pairing assumption in $\mathbb{G}_2$ by Theorem 10, so that from now we assume that the $q$-double pairing assumption holds in $\mathbb{G}_2$. In order to show the computational witness-extended emulation, we construct an expected polynomial time extractor $\chi$ whose goal is to extract the witness by using a $poly(\lambda)$-bounded tree of accepting transcripts. If then, we can apply the general forking lemma.

The case $(N = 1)$ is straightforward since the prover sends the witness and the verifier can directly check the correctness. Let us focus on the case $(N > 1)$. We prove that for each recursive step that on input $(\boldsymbol{F}_k, \boldsymbol{z}_k, H_k, P)$ for $k \in [m]$, we can efficiently extract from the prover a witness $\boldsymbol{v}_k$ for $k \in [m]$ under the $q$-double pairing assumption for the CRS $\boldsymbol{F}_k$'s and $H_k$'s. First, the extractor runs the prover to get $L$ and $R$. At this point, the extractor rewinds the prover 7 times and feeds 7 non-zero challenges $x_i$ such that $x_i^2 \neq x_j^2$ for all $j \neq i$. Then, the extractor obtains 7 tuples $(x_i, \widehat{\boldsymbol{v}}_1^{(i)}, \ldots, \widehat{\boldsymbol{v}}_m^{(i)})$ such that for $i \in [7]$

$$\begin{aligned}
&L^{x_i^2} P R^{x_i^{-2}} \\
&= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \widehat{\boldsymbol{F}}_k) e(\widehat{\boldsymbol{v}}_k^{(i)\widehat{\boldsymbol{z}}_k}, H_k) \\
&= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \boldsymbol{F}_{k,1}^{x_i^{-1}} \circ \boldsymbol{F}_{k,-1}^{x_i}) e(\widehat{\boldsymbol{v}}_k^{(i)(\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i)}, H_k) \\
&= \prod_{i\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i^{-1}}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i}, \boldsymbol{F}_{k,-1}) e(\widehat{\boldsymbol{v}}_k^{(i)(\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i)}, H_k) \qquad (29)
\end{aligned}$$

We know that squares of the first 3 challenges $x_1^2, x_2^2, x_3^2$ are distinct, so that the following matrix $M \in \mathbb{Z}_p^{3\times 3}$ is invertible since it is a product of two invertible matrices, where one of which is a diagonal matrix and the other is a Vandermonde matrix.

$$M = \begin{bmatrix} x_1^{-2} & 1 & x_1^2 \\ x_2^{-2} & 1 & x_2^2 \\ x_3^{-2} & 1 & x_3^2 \end{bmatrix} = \begin{bmatrix} x_1^{-2} & 0 & 0 \\ 0 & x_2^{-2} & 0 \\ 0 & 0 & x_3^{-2} \end{bmatrix} \cdot \begin{bmatrix} 1 & x_1^2 & x_1^4 \\ 1 & x_1^2 & x_2^4 \\ 1 & x_1^2 & x_3^4 \end{bmatrix}.$$

By using the inverse matrix of $M$ and the elementary linear algebra in the public exponents of the first 3 equalities for $i = 1, 2, 3$ in Eq. (29), we can find tuples group element vectors $(\boldsymbol{g}_{P,k,1}, \boldsymbol{g}_{P,k,-1}, g_{P,k,H})$, $(\boldsymbol{g}_{L,k,1}, \boldsymbol{g}_{L,k,-1}, g_{L,k,H})$, and $(\boldsymbol{g}_{R,k,1}, \boldsymbol{g}_{R,k,-1}, g_{R,k,H})$ satisfying

$$P = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{P,k,1}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\boldsymbol{g}_{P,k,-1}, \boldsymbol{F}_{k,-1}) e(g_{P,k,H}, H_K) \qquad (30)$$

$$L = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{L,k,1}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\boldsymbol{g}_{L,k,-1}, \boldsymbol{F}_{k,-1}) e(g_{L,k,H}, H_K) \qquad (31)$$

$$R = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{R,k,1}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\boldsymbol{g}_{R,k,-1}, \boldsymbol{F}_{k,-1}) e(g_{R,k,H}, H_K). \qquad (32)$$

Next, we prove that the extracted group elements $\boldsymbol{g}_{P,k,1}, \boldsymbol{g}_{P,k,-1}, g_{P,k,H}$ satisfy the desired relation $g_{P,k,H} = \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k}$, where $\boldsymbol{g}_{P,k}$ is the concatenation of $\boldsymbol{g}_{P,k,1}$ and $\boldsymbol{g}_{P,k,-1}$ for $k \in [m]$. Putting Eq. (30), Eq. (31) and Eq. (32) into Eq. (29), we have for each $i \in [7]$,

$$\begin{aligned}
&\prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{L,k,1}^{x_i^2} \circ \boldsymbol{g}_{P,k,1} \circ \boldsymbol{g}_{R,k,1}^{x_i^{-2}}, \boldsymbol{F}_{k,1}) \cdot \boldsymbol{E}(\boldsymbol{g}_{L,k,-1}^{x_i^2} \circ \boldsymbol{g}_{P,k,-1} \circ \boldsymbol{g}_{R,k,-1}^{x_i^{-2}}, \boldsymbol{F}_{k,-1}) \\
&\quad \cdot e(g_{L,k,H}^{x_i^2} \cdot g_{P,k,H} \cdot g_{R,k,H}^{x_i^{-2}}, H_k) \\
&= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i^{-1}}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i}, \boldsymbol{F}_{k,-1}) e(\widehat{\boldsymbol{v}}_k^{(i)(\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i)}, H_k)
\end{aligned}$$

By the $q$-double pairing assumption, the above equality implies that for $i \in [7]$ and $k \in [m]$

$$\boxed{\boldsymbol{F}_{k,1} \text{ correspondence}} : \quad \boldsymbol{g}_{L,k,1}^{x_i^2} \circ \boldsymbol{g}_{P,k,1} \circ \boldsymbol{g}_{R,k,1}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)\,x_i^{-1}} \tag{33}$$

$$\boxed{\boldsymbol{F}_{k,-1} \text{ correspondence}} : \boldsymbol{g}_{L,k,-1}^{x_i^2} \circ \boldsymbol{g}_{P,k,-1} \circ \boldsymbol{g}_{R,k,-1}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)\,x_i} \tag{34}$$

$$\boxed{H_k \text{ correspondence}} : \quad g_{L,k,H}^{x_i^2} \cdot g_{P,k,H} \cdot g_{R,k,H}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)\,\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i} \tag{35}$$

If we find exponents satisfying Eq. (29), Eq. (30), Eq. (31) and Eq. (32), but not one of the above Eq. (33), Eq. (34) and Eq. (35), it directly implies a non-trivial relation between the generators $\boldsymbol{F}_k$'s and $H_k$'s and so we break the $q$-double pairing assumption.

As an intermediate step toward the relation $g_{P,k,H} = \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k}$, we find a relation between $\boldsymbol{g}_{P,k}$ and $\widehat{\boldsymbol{v}}_k^{(i)}$ for $k \in [m]$ since such the relation can be combined with Eq. (35) to find the desired relation $g_{P,k,H} = \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k}$. From Eq. (33) and Eq. (34), we can deduce that for each $i \in [7]$ and $k \in [m]$,

$$\boldsymbol{g}_{L,k,1}^{x_i^3} \circ \boldsymbol{g}_{P,k,1}^{x_i} \circ \boldsymbol{g}_{R,k,1}^{x_i^{-1}} = \boldsymbol{g}_{L,k,-1}^{x_i} \circ \boldsymbol{g}_{P,k,-1}^{x_i^{-1}} \circ \boldsymbol{g}_{R,k,-1}^{x_i^{-3}} \in \mathbb{G}_1^N. \tag{36}$$

Eq. (36) can be interpreted that seven $x_i$'s are solutions of $m \times N$ equations. That is, for each equation, we have 7 solutions. Since the degree of $x_i$ in each equation varies between -3 and 3, each equation should hold for all $x \in \mathbb{Z}_p$. The only way to hold the above bunch of equations is that for $k \in [m]$

$$\boldsymbol{g}_{L,k,1} = \boldsymbol{1}_{\mathbb{G}_1}, \quad \boldsymbol{g}_{P,k,1} = \boldsymbol{g}_{L,k,-1}, \quad \boldsymbol{g}_{R,k,1} = \boldsymbol{g}_{P,k,-1}, \quad \boldsymbol{g}_{R,k,-1} = \boldsymbol{1}_{\mathbb{G}_1}.$$

Putting the above result into Eq. (33), for $i \in [7]$ and $k \in [m]$ we have

$$\boldsymbol{g}_{P,k,1} \circ \boldsymbol{g}_{P,k,-1}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)\,x_i^{-1}}.$$

Thus, we obtain the relation $\widehat{\boldsymbol{v}}_k^{(i)} = \boldsymbol{g}_{P,k,1}^{x_i} \circ \boldsymbol{g}_{P,k,-1}^{x_i^{-1}}$ for $i \in [7]$, which is the intermediate step toward the desired relation $g_{P,k,H} = \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k}$.

As aforementioned, we combine these relations with Eq. (35) and obtain that for $i \in [7]$ and $k \in [m]$

$$\begin{aligned}
(g_{L,k,H})^{x_i^2} \cdot g_{P,k,H} \cdot (g_{R,k,H})^{x_i^{-2}} &= \widehat{\boldsymbol{v}}_k^{(i)(\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i)} \\
&= \left(\boldsymbol{g}_{P,k,1}^{x_i} \circ \boldsymbol{g}_{P,k,-1}^{x_i^{-1}}\right)^{\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i} \\
&= \boldsymbol{g}_{P,k,1}^{\boldsymbol{z}_{k,1}} \cdot \boldsymbol{g}_{P,k,1}^{\boldsymbol{z}_{k,-1}x_i^2} \cdot \boldsymbol{g}_{P,k,-1}^{\boldsymbol{z}_{k,1}x_i^{-2}} \cdot \boldsymbol{g}_{P,k,-1}^{\boldsymbol{z}_{k,-1}} \\
&= (\boldsymbol{g}_{P,k,1})^{\boldsymbol{z}_{k,-1}x_i^2} \cdot \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k} \cdot (\boldsymbol{g}_{P,k,-1})^{\boldsymbol{z}_{k,1}x_i^{-2}}.
\end{aligned}$$

Since this relation holds for all $i \in [7]$, it must be that

$$g_{P,k,H} = \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k} \text{ for } k \in [m].$$

Therefore, we construct the extractor that outputs $\boldsymbol{g}_{P,k}$ and $g_{P,k,H}$ satisfying the above desired relation. The extractor rewinds 7 times for each recursive step. Thus, it uses $7^{\log_2 N}$ transcripts in total and thus runs in expected polynomial time in $N$ and $\lambda$. Then, by the general forking lemma, we conclude that the argument has computational witness-extended emulation. $\square$

**Tradeoff between Rounds and Communications** If $N = 1$, the prover sends $v_1, \ldots, v_m$ to the verifier and the verifier checks if $P = \prod_{k\in[m]} e(v_k, F_k)e(v_k^{z_k}, H_k)$ holds. This procedure requires to transmit $m$ group elements in $\mathbb{G}_1$. When running additional rounds, we can reduce this transmission cost to be logarithmic in $m$.

If $m = 1$, the prover sends $v_1 \in \mathbb{G}_1$ to the verifier. If $m > 1$, for the sake of simplicity, we assume that $m$ is a power of 2. The prover and the verifier compute and set $B_k := F_k \cdot H_k^{z_k} \in \mathbb{G}_2$. Let $\tilde{\boldsymbol{v}} = (v_1, \ldots, v_m)$ and $\boldsymbol{B} = (B_1, \ldots, B_m)$. Next, both the prover and the verifier run the protocol for the relation $P = \boldsymbol{E}(\tilde{\boldsymbol{v}}, \boldsymbol{B})$ where $P$ and $\boldsymbol{B}$ are common inputs and $\tilde{\boldsymbol{v}}$ is a witness. For example, both can run the single multi-exponentiation argument protocol (Fig. 8 with no product) with zero coefficient vector $\boldsymbol{z} = \boldsymbol{0}$, which requires the prover to send $2\log m$ group elements in $\mathbb{G}_t$ and one group element in $\mathbb{G}_1$.

**Efficiency** *Basic Protocol.* The prover sends two group elements in $\mathbb{G}_t$ for each recursive stage and $m$ group elements in $\mathbb{G}_1$ at the final stage. Overall, the protocol runs $O(\log N)$ rounds and the prover sends $2\log N$ group elements in $\mathbb{G}_t$ and $m$ group elements in $\mathbb{G}_1$. For the $i$-th recursive stage, prover's computational cost is dominated by $O(\frac{mN}{2^{i-1}})$ bilinear map computation. Overall, the prover computes $O(mN)$ bilinear maps. The verifier's computational cost is $O(N + m)$ since the final stage requires $O(m)$ operations for verification and the verifier's work in each recursive stage is exactly the same as that of BP-IP so that it can be batched and reduced to a single multi-exponentiation of length $N$.

*Protocol with Low Communication Cost.* At the final stage of this version, the protocol runs a single multi-exponentiation argument protocol instead of directly sending the witness. It will pose additional $\log m$ rounds but $O(\log m)$ communication cost instead of $m$. Overall, the protocol uses $O(\log N + \log m)$ rounds and the prover sends $O(\log N + \log m)$ group elements in $\mathbb{G}_t$ and one group element in $\mathbb{G}_1$. Both the prover and the verifier's asymptotic computational costs are unchanged; that is, $O(mN)$ bilinear maps and $O(N + m)$ group/field operations, respectively.

## D Sublinear Verifier without Pairing

### D.1 Complete Addition Formula for Prime Order Elliptic Curve

We provide the complete addition formula for a prime order elliptic curve group given by the short Weierstrass equation in two-dimensional projective space [47].

**Algorithm 1** Complete, projective point addition for arbitrary prime order short Weierstrass curves

```
Require: $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, Z_2)$, $E(\mathbb{Z}_q) : Y^2Z = X^3 + aXZ^2 + bZ^3$,
    and $b_3 = 3 \cdot b$.
Ensure: $(X_3, Y_3, Z_3) = P_1 + P_2$.
 1: $t_0 \leftarrow X_1 \cdot X_2$
 2: $t_1 \leftarrow Y_1 \cdot Y_2$
 3: $t_2 \leftarrow Z_1 \cdot Z_2$
 4: $t_3 \leftarrow X_1 + Y_1$
 5: $t_4 \leftarrow X_2 + Y_2$
 6: $t_3 \leftarrow t_3 \cdot t_4$
 7: $t_4 \leftarrow t_0 + t_1$
 8: $t_3 \leftarrow t_3 - t_4$
 9: $t_4 \leftarrow X_1 + Z_1$
10: $t_5 \leftarrow X_2 + Z_2$
11: $t_4 \leftarrow t_4 \cdot t_5$
12: $t_5 \leftarrow t_0 + t_2$
13: $t_4 \leftarrow t_4 - t_5$
14: $t_5 \leftarrow Y_1 + Z_1$
15: $X_3 \leftarrow Y_2 + Z_2$
16: $t_5 \leftarrow t_5 \cdot X_3$
17: $X_3 \leftarrow t_1 + t_2$
18: $t_5 \leftarrow t_5 - X_3$
19: $Z_3 \leftarrow a \cdot t_4$
20: $X_3 \leftarrow b_3 \cdot t_2$
21: $Z_3 \leftarrow X_3 + Z_3$
22: $X_3 \leftarrow t_1 - Z_3$
23: $Z_3 \leftarrow t_1 + Z_3$
24: $Y_3 \leftarrow X_3 \cdot Z_3$
25: $t_1 \leftarrow t_0 + t_0$
26: $t_1 \leftarrow t_1 + t_0$
27: $t_2 \leftarrow a \cdot t_2$
28: $t_4 \leftarrow b_3 \cdot t_4$
29: $t_1 \leftarrow t_1 + t_2$
30: $t_2 \leftarrow t_0 - t_2$
31: $t_2 \leftarrow a \cdot t_2$
32: $t_4 \leftarrow t_4 + t_2$
33: $t_0 \leftarrow t_1 \cdot t_4$
34: $Y_3 \leftarrow Y_3 + t_0$
35: $t_0 \leftarrow t_5 \cdot t_4$
36: $X_3 \leftarrow t_3 \cdot X_3$
37: $X_3 \leftarrow X_3 - t_0$
38: $t_0 \leftarrow t_3 \cdot t_1$
39: $Z_3 \leftarrow t_5 \cdot Z_3$
40: $Z_3 \leftarrow Z_3 + t_0$
```

Given two points $(X_1, Y_1, Z_1), (X_2, Y_2, Z_2) \in \{(X, Y, Z) \in \mathbb{Z}_q^3 | Y^2Z = X^3 + aXZ^2 + bZ^3\}$, $(X_3, Y_3, Z_3) = (X_1, Y_1, Z_1) + (X_2, Y_2, Z_2)$ can be computed as follows:

$$\begin{aligned}
X_3 &= (X_1Y_2 + X_2Y_1)(Y_1Y_2 - a(X_1Z_2 + X_2Z_1) - 3bZ_1Z_2) \\
&\quad - (Y_1Z_2 + Y_2Z_1)(aX_1X_2 + 3b(X_1Z_2 + X_2Z_1) - a2Z_1Z_2), \\
Y_3 &= (3X_1X_2 + aZ_1Z_2)(aX_1X_2 + 3b(X_1Z_2 + X_2Z_1) - a2Z_1Z_2) \\
&\quad + (Y_1Y_2 + a(X_1Z_2 + X_2Z_1) + 3bZ_1Z_2)(Y_1Y_2 - a(X_1Z_2 + X_2Z_1) - 3bZ_1Z_2), \\
Z_3 &= (Y_1Z_2 + Y_2Z_1)(Y_1Y_2 + a(X_1Z_2 + X_2Z_1) + 3bZ_1Z_2) \\
&\quad + (X_1Y_2 + X_2Y_1)(3X_1X_2 + aZ_1Z_2).
\end{aligned}$$

Algorithm 1 computes the above formulas using 12 multiplications, 5 multiplications by constant, and 23 additions over $\mathbb{Z}_q$ [47, Algorithm 1].

## D.2 Protocol4.Row

We provide a full description of Protocol4.Row in Fig. 9. In a nutshell, the protocol Protocol4.Row consist of the following process. First, the prover and the verifier run row-reduction Protocol4.Row recursively until $m = 1$. After the row-reduction, the prover and the verifier run two subprotocols Protocol4.Col and AggMEC.Row which are described in Fig. 10 and Fig. 11 respectively. We denote $\mathsf{pp}_{col}$ and $\mathsf{pp}_1$ to the public parameters to run the subprotocols Protocol4.Col and AggMEC.Row respectively. Note that $\mathsf{pp}_{col}$ contain the public parameters $\mathsf{pp}_2$ for AggMEC.Col.

$\boxed{\begin{array}{l}\mathsf{Protocol4.Row}(\boldsymbol{g},\boldsymbol{h},\boldsymbol{F}_{\ell+1},(\boldsymbol{S}_k,\boldsymbol{F}_k)_{k=1}^{\ell},\mathsf{pp}_1,\mathsf{pp}_{col},P,c,st_V;\boldsymbol{a},\boldsymbol{b},st_p)\\ \quad \text{where } \ell=\log m \text{ and } \mathsf{pp}_1=(\boldsymbol{G},\boldsymbol{H},\boldsymbol{K},U)\in\mathbb{G}_q^{O(n)}\times\mathbb{G}_q^{O(n)}\times\mathbb{G}_q^{18\ell n+6n}\times\mathbb{G}_q,\\ \quad \mathsf{pp}_{col}=((\boldsymbol{D}_k)_{k=1}^{\log n},\mathsf{pp}_2) \text{ where } \boldsymbol{D}_k\in\mathbb{G}_q^{3\cdot 2^k}\end{array}}$

**If** $(m = 1)$:

**Step 1** : If $st_P$ is empty, then follow the **Step 2**.
Otherwise, both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{AggMEC.Row}(\mathsf{pp}_1, st_V; st_P)$

**Step 2** : Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{Protocol4.Col}(g, h, \boldsymbol{F}_1, \mathsf{pp}_{col}, P, c; \boldsymbol{a}, \boldsymbol{b})$

**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** : $\mathcal{P}$ computes $\boldsymbol{p}_{\ell+1} = \overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \parallel \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}} \in \mathbb{G}_p^{2n}$. If the lists $st_P$ and $st_V$ are empty, then $\mathcal{P}$ and $\mathcal{V}$ add a tuple $(\,\cdot\,,\,\cdot\,,\boldsymbol{p}_{\ell+1})$ on $st_P$ and $(\,\cdot\,,\boldsymbol{F}_{\ell+1},\,\cdot\,,P,\,\cdot\,)$ on $st_V$ respectively.

**Step 2**: $\mathcal{P}$ calculates

$$\boldsymbol{l} = \overrightarrow{\boldsymbol{g}_{-1}{}^{\boldsymbol{a}_1}} \parallel \overrightarrow{\boldsymbol{h}_1{}^{\boldsymbol{b}_{-1}}} \in \mathbb{G}_p^{2n},\ \boldsymbol{r} = \overrightarrow{\boldsymbol{g}_1{}^{\boldsymbol{a}_{-1}}} \parallel \overrightarrow{\boldsymbol{h}_{-1}{}^{\boldsymbol{b}_1}} \in \mathbb{G}_p^{2n}$$
$$c_L = \langle \boldsymbol{a}_1, \boldsymbol{b}_{-1}\rangle \in \mathbb{Z}_p,\ c_R = \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1\rangle \in \mathbb{Z}_p,\ S_\ell = \mathsf{Com}(\boldsymbol{l} \parallel \boldsymbol{r}; \boldsymbol{S}_\ell) \in \mathbb{G}_q$$

and sends $c_L$, $c_R$ and $S_\ell$ to $\mathcal{V}$.

**Step 3**: $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.

**Step 4**: $\mathcal{P}$ computes

$$\boldsymbol{p}_\ell = \boldsymbol{l}^{x^2} \circ \boldsymbol{p}_{\ell+1} \circ \boldsymbol{r}^{x^{-2}} \in \mathbb{G}_p^{2n}$$
$$P_\ell = \mathsf{Com}(\boldsymbol{p}_\ell; \boldsymbol{F}_\ell) \in \mathbb{G}_q$$

and sends $P_\ell$ to $\mathcal{V}$

**Step 5**: Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\widehat{\boldsymbol{g}} = \boldsymbol{g}_1^{x^{-1}} \circ \boldsymbol{g}_{-1}^{x} \in \mathbb{G}_p^{\widehat{m}},\ \widehat{\boldsymbol{h}} = \boldsymbol{h}_1^{x} \circ \boldsymbol{h}_{-1}^{x^{-1}} \in \mathbb{G}_p^{\widehat{m}},\ \widehat{c} = x^2 c_L + c + x^{-2} c_R \in \mathbb{Z}_p$$

In addition, $\mathcal{P}$ computes

$$\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1} \in \mathbb{Z}_p^{\widehat{m}\times n} \text{ and } \widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{m}\times n}.$$

**Step 6**: $\mathcal{V}$ updates $st_V$ by adding a tuple $(\boldsymbol{S}_\ell, \boldsymbol{F}_\ell, S_\ell, P_\ell, x)$ into the top row. $\mathcal{P}$ updates $st_P$ by adding a tuple $(\boldsymbol{l}, \boldsymbol{r}, \boldsymbol{p}_\ell)$ into the top row. Both $\mathcal{P}$ and $\mathcal{V}$ run the protocol

$$\mathsf{Protocol4.Row}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \boldsymbol{F}_\ell, (\boldsymbol{S}_k, \boldsymbol{F}_k)_{k=1}^{\ell-1}, \mathsf{pp}_1, \mathsf{pp}_{col}, P_\ell, \widehat{c}, st_V; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, st_P)$$

**Fig. 9.** Protocol4.Row

**Theorem 12.** *Assume that both proof systems* AggMEC.Row *and* Protocol4.Col *have perfect completeness and computational witness-extended-emulation respectively. Then,* Protocol4.Row *has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption.*

*Proof.* The protocol $\mathsf{Protocol4.Row}$ separates into two cases **Case 1**($m = 1$) and **Case 2**($m > 1$) according to $m$. For the sake of simplicity, we assume that $m$ is power of 2.

(*Completeness*) In **Case 1**, the prover and the verifier perform two subprotocols $\mathsf{Protocol4.Col}$ and $\mathsf{AggMEC.Row}$. $\mathsf{Protocol4.Col}$ is a proof system for the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$, and thus its perfect completeness guarantees that the verifier accepts any correctly generated proof. In addition, we have to check the correctness of the input of $\mathsf{AggMEC.Row}$. We consider it after proving the perfect completeness of **Case 2** since such inputs are related to the steps in the **Case 2**.

**Case 2** is the reduction step of $\mathsf{Protocol4.Row}$. For the completeness of this case, we show that $\mathsf{Protocol4.Row}$ correctly reduce a valid instance for the relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$ to a valid instance of the relation $\mathcal{R}_{\mathsf{IP}}^{m/2,n}$. Assume that a statement $(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{F}_{\ell+1}, P, c)$ is in a language $\mathcal{L}^{m,n}$ associated with the relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$ such that there exists a witness pair $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_p^{m\times n} \times \mathbb{Z}_p^{m\times n}$ satisfying the following equations.

$$P = \mathsf{Com}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \parallel \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}; \boldsymbol{F}_{\ell+1}) \tag{37}$$
$$c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle \tag{38}$$

The prover's computation for the reduction step guarantees that $P_\ell$ and $\hat{c}$ satisfy the following equations.

$$\begin{aligned}
P_\ell =& \mathsf{Com}(\boldsymbol{p}_\ell; \boldsymbol{F}_\ell)\\
=& \mathsf{Com}(\boldsymbol{l}^{x^2} \circ \boldsymbol{p}_{\ell+1} \circ \boldsymbol{r}^{x^{-2}}; \boldsymbol{F}_\ell) \text{ \textbf{Step 4} in Fig.9}\\
=& \mathsf{Com}((\overrightarrow{\boldsymbol{g}_{-1}{}^{\boldsymbol{a}_1}} \parallel \overrightarrow{\boldsymbol{h}_1{}^{\boldsymbol{b}_{-1}}})^{x^2} \circ (\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \parallel \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}) \circ (\overrightarrow{\boldsymbol{g}_1{}^{\boldsymbol{a}_{-1}}} \parallel \overrightarrow{\boldsymbol{h}_{-1}{}^{\boldsymbol{b}_1}})^{x^{-2}}; \boldsymbol{F}_\ell) \text{ \textbf{Step 1}, \textbf{Step 2} in Fig.9}\\
=& \mathsf{Com}(\overrightarrow{(\boldsymbol{g}_1^{x^{-1}} \circ \boldsymbol{g}_{-1}^{x})^{x\boldsymbol{a}_1 + x^{-1}\boldsymbol{a}_{-1}}} \parallel \overrightarrow{(\boldsymbol{h}_1^{x} \circ \boldsymbol{h}_{-1}^{x^{-1}})^{x^{-1}\boldsymbol{b}_1 + x\boldsymbol{b}_{-1}}}; \boldsymbol{F}_\ell) \text{ By homomorphic property}\\
=& \mathsf{Com}(\overrightarrow{\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}}} \parallel \overrightarrow{\widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}}}; \boldsymbol{F}_\ell)
\end{aligned} \tag{39}$$

$$\begin{aligned}
\widehat{c} =& x^2 c_L + c + x^{-2} c_R\\
=& x^2 \langle \boldsymbol{a}_1, \boldsymbol{a}_{-1} \rangle + \langle \boldsymbol{a}_1, \boldsymbol{b}_1 \rangle + \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_{-1} \rangle + x^{-2} \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle \text{ By Eq.(38) \& \textbf{Step 2} in Fig.9}\\
=& \langle x\boldsymbol{a}_1 + x^{-1}\boldsymbol{a}_{-1}, x^{-1}\boldsymbol{b}_1 + x\boldsymbol{b}_{-1} \rangle\\
=& \langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle
\end{aligned} \tag{40}$$

From the resemblance between pairs of equations (Eq. (37), Eq. (38)) and (Eq. (39), Eq. (40)), we know that the tuple $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \boldsymbol{F}_\ell, P_\ell, \widehat{c})$ belongs to the language $\mathcal{L}^{m/2,n}$ associated with the relation $\mathcal{R}_{\mathsf{IP}}^{m/2,n}$ with a witness $(\widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}) \in \mathbb{Z}_p^{m/2\times n} \times \mathbb{Z}_p^{m/2\times n}$. In other words, each recursion step satisfies perfect completeness. By $\ell$ times recursion, we converts an instance for the relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$ to a instance for the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ perfectly.

Finally, we check the acceptance of $\mathsf{AggMEC.Row}$ verifier. From $\ell$ times recursion, the lists $st_P$ and $st_V$ must be formed $\ell + 1$ rows. By $\mathcal{P}$ computation in **Step 2** and **Step 4** in Fig.9, the $k$-th tuples of $st_P$ satisfies the following equation.

$$P_k = \mathsf{Com}(\boldsymbol{p}_k; \boldsymbol{F}_k),\ S_k = \mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k; \boldsymbol{S}_k),\ \boldsymbol{p}_k = \boldsymbol{l}_k^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k^{x_k^{-2}} \text{ for } k \in [\ell]$$

The $\ell + 1$-th row $(\boldsymbol{F}_{\ell+1}, P_{\ell+1}; \boldsymbol{p}_{\ell+1})$ satisfies $P_{\ell+1} = \mathsf{Com}(\boldsymbol{p}_{\ell+1}; \boldsymbol{F}_{\ell+1})$ from **Step 1** in Fig.9. Then, the instance $\begin{bmatrix} (\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k) \\ (\ \cdot\ , \boldsymbol{F}_{\ell+1},\ \cdot\ , P_{\ell+1},\ \cdot\ ) \end{bmatrix}$ satisfies the conditions in the relation $\mathcal{R}_{\mathsf{AggMEC.Row}}$. By the perfect completeness of AggMEC.Row, the acceptance of AggMEC.Row verifier is guaranteed.

(*Witness-Extended Emulation*) For the computational witness-extended emulation, we construct an expected polynomial time extractor $\chi$ whose goal is to extract a witness by using a polynomially bounded tree of accepting transcripts. To construct the extractor $\chi$, we use two extractors $\mathcal{E}_1$ and $\mathcal{E}_{col}$, which are given by the hypothesis, that extract a witness of $\mathcal{R}_{\mathsf{AggMEC.Row}}$ and $\mathcal{R}_{\mathsf{IP}}^{1,n}$, respectively. By the general forking lemma, it is sufficient to construct an extractor $\chi$ that extracts a witness from a suitable tree of accepting transcripts in probabilistic polynomial time. We begin with $(\underbrace{7,\ldots,7}_{\log_2 m})$-tree of accepting transcripts. Since the number of challenges in the tree $(7^{\log_2 m})$ is bounded above by a polynomial in $N = mn$ and security parameter $\lambda$, we can apply the general forking lemma.

In **Case 1**, if $st_P$ is empty, it means the protocol has never run **Case 2**, so that the proof system proves the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ that is exactly the goal of Protocol4.Col. Therefore, $\mathcal{E}_{col}$ is sufficient for $\chi$. If $st_P$ is non-empty, then $\chi$ runs two extractor $\mathcal{E}_{col}$ and $\mathcal{E}_1$ for all leaf nodes of the tree of accepting transcripts. Note that the tree of accepting transcripts is polynomially bounded and both $\mathcal{E}_{col}$ and $\mathcal{E}_1$ are expected polynomial time, so that $\chi$ spends polynomially bounded operations in this step. By Theorem 13 and Theorem 14, $\mathcal{E}_{col}$ and $\mathcal{E}_1$ extract a witness $(\boldsymbol{a}, \boldsymbol{b})$ of $\mathcal{R}_{\mathsf{IP}}^{1,n}$ and a witness $(\boldsymbol{l}, \boldsymbol{r}, \boldsymbol{p})$ of $\mathcal{R}_{\mathsf{AggMEC.Row}}$, respectively. We would like to emphasize that $\mathcal{E}_1$ has run for all accepting transcripts, so that from now we can assume that for each accepting transcript, $\chi$ knows a tuple $(\boldsymbol{l}, \boldsymbol{r}, \widehat{\boldsymbol{p}}, \boldsymbol{p})$ satisfying the following equations, regardless of the challenges.

$$P = \mathsf{Com}(\boldsymbol{p}; \boldsymbol{F}_{\ell+1}) \tag{41}$$

$$\widehat{P} = \mathsf{Com}(\widehat{\boldsymbol{p}}; \boldsymbol{F}_{\ell}) \tag{42}$$

$$\widehat{\boldsymbol{p}} = \boldsymbol{l}^{x^2} \circ \boldsymbol{p} \circ \boldsymbol{r}^{x^{-2}} \tag{43}$$

Next, we show that for **Case 2**, $\chi$ can extract a witness $(\boldsymbol{a}, \boldsymbol{b})$ of each recursive round by using the previous round witness $(\widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$ that can be extracted by either the previous round extractor of **Case 2** or the extractor of **Case 1**.

The extractor $\chi$ runs the prover until **Step 2** to get $c_L, c_R$, and $S_\ell$. At this point of time, $\chi$ rewinds the prover with 7 distinct challenges $x_1, \ldots, x_7$ and obtains tuples $(\widehat{\boldsymbol{a}}_i, \widehat{\boldsymbol{b}}_i)$ for $i \in [7]$ that satisfy the following equations.

$$\widehat{P}_i = \mathsf{Com}(\overrightarrow{\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}_i}} \parallel \overrightarrow{\widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}_i}}; \boldsymbol{F}_\ell) \tag{44}$$

$$\widehat{c}_i = \langle \widehat{\boldsymbol{a}}_i, \widehat{\boldsymbol{b}}_i \rangle \tag{45}$$

$\widehat{P}_i$ is contained in $st_V$, so that, by discrete logarithm relation on $\mathbb{G}_q$, the openings of Eq. (42) and Eq. (44) must be identical for all $i \in [7]$. That is, $\widehat{\boldsymbol{p}} = \overrightarrow{\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}_i}} \parallel \overrightarrow{\widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}_i}}$. Combining this equation with Eq. (43), we have

$$\begin{aligned} \boldsymbol{l}^{x_i^2} \circ \boldsymbol{p} \circ \boldsymbol{r}^{x_i^{-2}} = \widehat{\boldsymbol{p}}_i &= \overrightarrow{\widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}_i}} \parallel \overrightarrow{\widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}_i}} \\ &= \overrightarrow{(\boldsymbol{g}_1^{x_i^{-1}} \circ \boldsymbol{g}_{-1}^{x_i})^{\widehat{\boldsymbol{a}}_i}} \parallel \overrightarrow{(\boldsymbol{h}_1^{x_i} \circ \boldsymbol{h}_{-1}^{x_i^{-1}})^{\widehat{\boldsymbol{b}}_i}} \end{aligned} \tag{46}$$

where all the exponents are known to the extractor.

The right-hand side of Eq.(46) consists of a vector of products of the CRS $\boldsymbol{g}$ and $\boldsymbol{h}$ whose discrete logarithm relation is unknown. The left-hand side of Eq.(46) has exponents $(x_i^2, 1, x_i^{-2})$. Hence, similarly to the proof of the original BP-IP, for three challenges $x_1, x_2$ and $x_3$ the extractor computes the inverse matrix of a $3\times 3$ matrix with rows $(x_i^2, 1, x_i^{-2})_{i=1}^3$. Then, by using it, $\chi$ finds exponent matrices $\boldsymbol{a}_{t,s}, \boldsymbol{b}_{t,s} \in \mathbb{Z}_p^{m/2\times n}$ for $t \in \{l, p, r\}$ and $s \in \{1, -1\}$ that satisfies the following equations .

$$\begin{aligned} \boldsymbol{l} &= \overrightarrow{\boldsymbol{g}_1^{\boldsymbol{a}_{l,1}}} \circ \overrightarrow{\boldsymbol{g}_{-1}^{\boldsymbol{a}_{l,-1}}} \parallel \overrightarrow{\boldsymbol{h}_1^{\boldsymbol{b}_{l,1}}} \circ \overrightarrow{\boldsymbol{h}_{-1}^{\boldsymbol{b}_{l,-1}}} \in \mathbb{G}_p^{2n} \\ \boldsymbol{p} &= \overrightarrow{\boldsymbol{g}_1^{\boldsymbol{a}_{p,1}}} \circ \overrightarrow{\boldsymbol{g}_{-1}^{\boldsymbol{a}_{p,-1}}} \parallel \overrightarrow{\boldsymbol{h}_1^{\boldsymbol{b}_{p,1}}} \circ \overrightarrow{\boldsymbol{h}_{-1}^{\boldsymbol{b}_{p,-1}}} \in \mathbb{G}_p^{2n} \\ \boldsymbol{r} &= \overrightarrow{\boldsymbol{g}_1^{\boldsymbol{a}_{r,1}}} \circ \overrightarrow{\boldsymbol{g}_{-1}^{\boldsymbol{a}_{r,-1}}} \parallel \overrightarrow{\boldsymbol{h}_1^{\boldsymbol{b}_{r,1}}} \circ \overrightarrow{\boldsymbol{h}_{-1}^{\boldsymbol{b}_{r,-1}}} \in \mathbb{G}_p^{2n} \end{aligned} \tag{47}$$

Putting Eq.(47) into Eq.(46), by the discrete logarithm relation assumption on $\mathbb{G}_p$, the exponents of the bases $\boldsymbol{g}_1, \boldsymbol{g}_{-1}, \boldsymbol{h}_1$, and $\boldsymbol{h}_{-1}$ satisfy the following equations for each $i \in [7]$.

$$\boxed{\boldsymbol{g}_1 \ \text{exponents}} : x_i^2 \boldsymbol{a}_{l,1} + \boldsymbol{a}_{p,1} + x_i^{-2} \boldsymbol{a}_{r,1} = x_i^{-1} \widehat{\boldsymbol{a}}_i \tag{48}$$

$$\boxed{\boldsymbol{g}_{-1} \ \text{exponents}} : x_i^2 \boldsymbol{a}_{l,-1} + \boldsymbol{a}_{p,-1} + x_i^{-2} \boldsymbol{a}_{r,-1} = x_i \widehat{\boldsymbol{a}}_i \tag{49}$$

$$\boxed{\boldsymbol{h}_1 \ \text{exponents}} : x_i^2 \boldsymbol{b}_{l,1} + \boldsymbol{b}_{p,1} + x_i^{-2} \boldsymbol{b}_{r,1} = x_i \widehat{\boldsymbol{b}}_i \tag{50}$$

$$\boxed{\boldsymbol{h}_{-1} \ \text{exponents}} : x_i^2 \boldsymbol{b}_{l,-1} + \boldsymbol{b}_{p,-1} + x_i^{-2} \boldsymbol{b}_{r,-1} = x_i^{-1} \widehat{\boldsymbol{b}}_i \tag{51}$$

Combining Eq.(48) and (49), we get the following Eq.(52). Similarly, combining Eq.(50) and (51), we get the following Eq.(53).

$$x_i^3 \boldsymbol{a}_{l,1} + x_i \boldsymbol{a}_{p,1} + x_i^{-1} \boldsymbol{a}_{r,1} = x_i \boldsymbol{a}_{l,-1} + x_i^{-1} \boldsymbol{a}_{p,-1} + x_i^{-3} \boldsymbol{a}_{r,-1} \tag{52}$$

$$x_i \boldsymbol{b}_{l,1} + x_i^{-1} \boldsymbol{b}_{p,1} + x_i^{-3} \boldsymbol{b}_{r,1} = x_i^3 \boldsymbol{b}_{l,-1} + x_i \boldsymbol{b}_{p,-1} + x_i^{-1} \boldsymbol{b}_{r,-1} \tag{53}$$

for $i \in [7]$. In both Eq.(52) and Eq.(53), degrees of $x_i$ range between $-3$ and 3. Then, we can construct two polynomials $A(X)$ and $B(X)$ with 7 roots $x_1, \cdots, x_7$.

$$A(X) = \boldsymbol{a}_{l,1} X^6 + (\boldsymbol{a}_{p,1} - \boldsymbol{a}_{l,-1}) X^4 + (\boldsymbol{a}_{r,1} - \boldsymbol{a}_{p,-1}) X^2 - \boldsymbol{a}_{r,-1} = 0 \tag{54}$$

$$B(X) = \boldsymbol{b}_{l,-1} X^6 + (\boldsymbol{b}_{p,-1} - \boldsymbol{b}_{l,1}) X^4 + (\boldsymbol{b}_{r,-1} - \boldsymbol{b}_{p,1}) X^2 - \boldsymbol{b}_{r,1} = 0 \tag{55}$$

From the facts that the degree of $A(X)$ and $B(X)$ is at most 6 and $A(X)$ and $B(X)$ have 7 distinct roots, the polynomials $A(X)$ and $B(X)$ must be zero polynomial, so that we have

$$\boldsymbol{a}_{p,1} = \boldsymbol{a}_{l,-1}, \quad \boldsymbol{a}_{p,-1} = \boldsymbol{a}_{r,1}, \quad \boldsymbol{a}_{l,1} = \boldsymbol{a}_{r,-1} = \boldsymbol{0} \tag{56}$$

$$\boldsymbol{b}_{p,1} = \boldsymbol{b}_{r,-1}, \quad \boldsymbol{b}_{p,-1} = \boldsymbol{b}_{l,1}, \quad \boldsymbol{b}_{l,-1} = \boldsymbol{b}_{r,1} = \boldsymbol{0} \tag{57}$$

Now we claim that the desired matrices $\boldsymbol{a}_p = \boldsymbol{a}_{p,1} \parallel \boldsymbol{a}_{p,-1}$ and $\boldsymbol{b}_p = \boldsymbol{b}_{p,1} \parallel \boldsymbol{b}_{p,-1}$ are a witness of the instance $(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{F}_{\ell+1}, P, c)$ of the relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$. That is, we show that $P = \mathsf{Com}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}_p}} \parallel \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}_p}}; \boldsymbol{F}_{\ell+1}) \wedge c = \langle \boldsymbol{a}_p, \boldsymbol{b}_p \rangle$ holds. The first constraint is rather straightforward; from Eq.(41), we get

$$P = \mathsf{Com}(\overrightarrow{\boldsymbol{g}_1^{\boldsymbol{a}_{p,1}}} \circ \overrightarrow{\boldsymbol{g}_{-1}^{\boldsymbol{a}_{p,-1}}} \parallel \overrightarrow{\boldsymbol{h}_1^{\boldsymbol{b}_{p,1}}} \circ \overrightarrow{\boldsymbol{h}_{-1}^{\boldsymbol{b}_{p,-1}}}; \boldsymbol{F}_{\ell+1}) = \mathsf{Com}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}_p}} \parallel \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}_p}}; \boldsymbol{F}_{\ell+1}). \tag{58}$$

To check the inner product relation $c = \langle \boldsymbol{a}_p, \boldsymbol{b}_p \rangle$, we consider the equation

$$\begin{aligned} & c + x_i^2 c_L + x_i^{-2} c_R \\ =& \langle \widehat{\boldsymbol{a}}_i, \widehat{\boldsymbol{b}}_i \rangle \\ =& \langle x_i \boldsymbol{a}_{p,1} + x_i^{-1} \boldsymbol{a}_{p,-1}, x_i^{-1} \boldsymbol{b}_{p,1} + x_i \boldsymbol{b}_{p,-1} \rangle \\ =& \langle \boldsymbol{a}_{p,1}, \boldsymbol{b}_{p,1} \rangle + \langle \boldsymbol{a}_{p,-1}, \boldsymbol{b}_{p,-1} \rangle + x_i^2 \langle \boldsymbol{a}_{p,1}, \boldsymbol{b}_{p,-1} \rangle + x_i^{-2} \langle \boldsymbol{a}_{p,-1}, \boldsymbol{b}_{p,1} \rangle \end{aligned} \tag{59}$$

for all challenges $x_i$. Since $x_1^2$, $x_2^2$ and $x_3^2$ are distinct, a Vandermonde matrix $M \in \mathbb{Z}_p^{3\times 3}$ is invertible.

$$M = \begin{bmatrix} x_1^{-2} & 1 & x_1^2 \\ x_2^{-2} & 1 & x_2^2 \\ x_3^{-2} & 1 & x_3^2 \end{bmatrix} \tag{60}$$

Using the matrix $M$, we can rewrite Eq.(59) to the following equation.

$$M \begin{bmatrix} c_R \\ c \\ c_L \end{bmatrix} = M \begin{bmatrix} \langle \boldsymbol{a}_{p,-1}, \boldsymbol{b}_{p,1} \rangle \\ \langle \boldsymbol{a}_{p,1}, \boldsymbol{b}_{p,1} \rangle + \langle \boldsymbol{a}_{p,-1}, \boldsymbol{b}_{p,-1} \rangle \\ \langle \boldsymbol{a}_{p,1}, \boldsymbol{b}_{p,-1} \rangle \end{bmatrix} \tag{61}$$

Since $M$ is invertible, the equation $c = \langle \boldsymbol{a}_{p,1}, \boldsymbol{b}_{p,1} \rangle + \langle \boldsymbol{a}_{p,-1}, \boldsymbol{b}_{p,-1} \rangle = \langle \boldsymbol{a}_p, \boldsymbol{b}_p \rangle$. holds. Therefore, we conclude that the desired witness $(\boldsymbol{a}_p, \boldsymbol{b}_p)$ is a witness of the instance $(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{F}_{\ell+1}, P, c)$ for relation $\mathcal{R}_{\mathsf{IP}}^{m,n}$.

For each recursion, the extractor $\chi$ extracts a witness in at most polynomial time. Therefore, we conclude that the protocol Protocol4.Row provides computational witness-extended-emulation by the general forking lemma. □

## D.3 Protocol4.Col

In this section, we describe Protocol4.Col, which is an argument of knowledge for the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$. In the similar way in Protocol4.Row, Protocol4.Col consist of recursive reduction and run a subprotocol AggMEC.Col. We denote $\mathsf{pp}_2$ to the public parameters for AggMEC.Col.

**Theorem 13.** *Assume that a proof system* AggMEC.Col *has perfect completeness and computational witness-extended-emulation. Then,* Protocol4.Col *has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption.*

*Proof.* The Protocol4.Col separates into two cases **Case 1**$(n = 1)$ and **Case 2**$(n > 1)$ according to $n$. For the sake of simplicity, we assume that $n$ is power of 2.
(*Completeness*) In **Case 1**, the prover and the verifier perform a subprotocol AggMEC.Col. After running the subprotocol AggMEC.Col, the prover sends witness $a$ and $b$ to the verifier and then the verifier checks the relation $\mathcal{R}_{\mathsf{IP}}^{1,1}$ directly. From this fact, the correctness of a witness $(a, b)$ is guaranteed for relation $\mathcal{R}_{\mathsf{IP}}^{1,1}$. In addition, we have to check the correctness of the input of AggMEC.Col. We consider it after proving the perfect completeness of **Case 2** since such inputs are related to the steps in the **Case 2**.

**Case 2** is the reduction step of Protocol4.Col. For the completeness of this case, we show that Protocol4.Col correctly reduce a valid instance for the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ to a valid instance of the relation $\mathcal{R}_{\mathsf{IP}}^{1,n/2}$. Assume that a statement $(g, h, \boldsymbol{D}_{\ell+1}, P, c)$ is in a

$\mathsf{Protocol4.Col}(g, h, \boldsymbol{D}_{\ell+1}, (\boldsymbol{D}_k)_{k=1}^{\ell}, \mathsf{pp}_2, P, c, st_V; \boldsymbol{a}, \boldsymbol{b}, st_P)$
where $\ell = \log n$ and $\mathsf{pp}_2 = (\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{K}, U) \in \mathbb{G}_q^{O(n)} \times \mathbb{G}_q^{O(n)} \times \mathbb{G}_q^{3\cdot(2^{\ell+2}-2)} \times \mathbb{G}_q$

**If** $n = 1$:
**Step 1**: If $st_P$ is empty, then follow the **Step 2**.
Otherwise, both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{AggMEC.Col}(\mathsf{pp}_2, st_V; st_P)$
**Step 2**: $\mathcal{P}$ sends $a$ and $b$ and then $\mathcal{V}$ checks the following equations

$$c \overset{?}{=} a \cdot b, \quad P \overset{?}{=} \mathsf{com}(g^a \parallel h^b; \boldsymbol{D}_1)$$

**Else** $(n > 1)$: Let $\widehat{n} = \frac{n}{2}$. Parse $\boldsymbol{a}, \boldsymbol{b}$ to

$$\boldsymbol{a} = \boldsymbol{a}_1 \| \boldsymbol{a}_{-1}\ , \quad \boldsymbol{b} = \boldsymbol{b}_1 \| \boldsymbol{b}_{-1}$$

**Step 1**: $\mathcal{P}$ computes $\boldsymbol{p}_{\ell+1} = g^{\boldsymbol{a}} \parallel h^{\boldsymbol{b}} \in \mathbb{G}_p^{2n}$. If the lists $st_P$ and $st_V$ are empty, then $\mathcal{P}$ and $\mathcal{V}$ add a tuple $(\boldsymbol{p}_{\ell+1})$ on $st_P$ and $(\boldsymbol{D}_{\ell+1}, P, \cdot)$ on $st_V$ respectively. Additionally, $\mathcal{P}$ parses $\boldsymbol{p}_{\ell+1}$ to 4 vectors $\boldsymbol{p}_{1,\ell+1}, \boldsymbol{p}_{2,\ell+1}, \boldsymbol{p}_{3,\ell+1}, \boldsymbol{p}_{4,\ell+1} \in \mathbb{G}^{\widehat{n}}$.

$$\boldsymbol{p}_{\ell+1} = \boldsymbol{p}_{1,\ell+1} \parallel \boldsymbol{p}_{2,\ell+1} \parallel \boldsymbol{p}_{3,\ell+1} \parallel \boldsymbol{p}_{4,\ell+1}$$

**Step 2**: $\mathcal{P}$ calculates

$$c_L = \langle \boldsymbol{a}_1, \boldsymbol{b}_{-1} \rangle \in \mathbb{Z}_p, \quad c_R = \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle \in \mathbb{Z}_p$$

and sends $c_L$, $c_R$ to $\mathcal{V}$.
**Step 3**: $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$.
**Step 4**: $\mathcal{P}$ computes

$$\boldsymbol{p}_\ell = (\boldsymbol{p}_{1,\ell+1} \parallel \boldsymbol{p}_{4,\ell+1})^x \circ (\boldsymbol{p}_{2,\ell+1} \parallel \boldsymbol{p}_{3,\ell+1})^{x^{-1}} \in \mathbb{G}_p^{2\widehat{n}}$$
$$P_\ell = \mathsf{Com}(\boldsymbol{p}_\ell; \boldsymbol{D}_\ell) \in \mathbb{G}_q$$

and sends $P_\ell$ to $\mathcal{V}$
**Step 5**: Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\widehat{c} = x^2 c_L + c + x^{-2} c_R \in \mathbb{Z}_p$$

In addition, $\mathcal{P}$ computes $\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1} \in \mathbb{Z}_p^{\widehat{n}}$ and $\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{n}}$
**Step 6**: $\mathcal{V}$ updates $st_V$ by adding a tuple $(\boldsymbol{D}_\ell, P_\ell, x)$ into the top row. $\mathcal{P}$ updates $st_P$ by adding a tuple $(\boldsymbol{p}_\ell)$ into the top row. Both $\mathcal{P}$ and $\mathcal{V}$ run the protocol

$$\mathsf{Protocol4.Col}(g, h, \boldsymbol{D}_\ell, (\boldsymbol{D}_k)_{k=1}^{\ell-1}, \mathsf{pp}_2, P_\ell, \widehat{c}, st_V; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, st_P)$$

**Fig. 10.** Protocol4.Col

language $\mathcal{L}^{1,n}$ associated with the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ such that there exists a witness pair

$(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{Z}_p^{1\times n} \times \mathbb{Z}_p^{1\times n}$ satisfying the following equations.

$$P = \mathsf{Com}(g^{\boldsymbol{a}} \parallel h^{\boldsymbol{b}}; \boldsymbol{D}_{\ell+1}) \tag{62}$$
$$c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle \tag{63}$$

The prover's computation for the reduction step guarantees that $P_\ell$ and $\widehat{c}$ satisfy the following equations.

$$\begin{aligned}
P_\ell =& \mathsf{Com}(\boldsymbol{p}_\ell; \boldsymbol{D}_\ell) \\
=& \mathsf{Com}((\boldsymbol{p}_{1,\ell+1} \parallel \boldsymbol{p}_{4,\ell+1})^x \circ (\boldsymbol{p}_{2,\ell+1} \parallel \boldsymbol{p}_{3,\ell+1})^{x^{-1}}; \boldsymbol{D}_\ell) \\
=& \mathsf{Com}((g^{\boldsymbol{a}_1} \parallel h^{\boldsymbol{b}_{-1}})^x \circ (g^{\boldsymbol{a}_{-1}} \parallel h^{\boldsymbol{b}_1})^{x^{-1}}; \boldsymbol{D}_\ell) \\
=& \mathsf{Com}(g^{x\boldsymbol{a}_1 + x^{-1}\boldsymbol{a}_{-1}} \parallel h^{x^{-1}\boldsymbol{b}_1 + x\boldsymbol{b}_{-1}}; \boldsymbol{D}_\ell) \text{ By homomorphic property} \\
=& \mathsf{Com}(g^{\widehat{\boldsymbol{a}}} \parallel h^{\widehat{\boldsymbol{b}}}; \boldsymbol{D}_\ell)
\end{aligned} \tag{64}$$

$$\begin{aligned}
\widehat{c} =& x^2 c_L + c + x^{-2} c_R \\
=& x^2 \langle \boldsymbol{a}_1, \boldsymbol{a}_{-1} \rangle + \langle \boldsymbol{a}_1, \boldsymbol{b}_1 \rangle + \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_{-1} \rangle + x^{-2} \langle \boldsymbol{a}_{-1}, \boldsymbol{b}_1 \rangle \text{ By Eq.(63) \& \textbf{Step 1} in Fig.10} \\
=& \langle x\boldsymbol{a}_1 + x^{-1}\boldsymbol{a}_{-1}, x^{-1}\boldsymbol{b}_1 + x\boldsymbol{b}_{-1} \rangle \\
=& \langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle
\end{aligned} \tag{65}$$

From the resemblance between pairs of equations (Eq.(62), Eq.(63)) and (Eq.(64), Eq.(65)), we know that the tuple $(g, h, \boldsymbol{D}_\ell, P_\ell, \widehat{c})$ belongs to the language $\mathcal{L}^{1,n/2}$ associated with the relation $\mathcal{R}_{\mathsf{IP}}^{1,n/2}$ with a witness $(\widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}) \in \mathbb{Z}_p^{1\times n/2} \times \mathbb{Z}_p^{1\times n/2}$. In other wods, each recursion step satisfies perfect completeness. By $\ell$ times recursion, we converts an instance for the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$ to an instance for the relation $\mathcal{R}_{\mathsf{IP}}^{1,1}$ perfectly.

Finally, we check the acceptance of AggMEC.Col verifier. From $\ell$ times recursion, the lists $st_P$ and $st_V$ must be formed $\ell + 1$ rows. By $\mathcal{P}$ computation in **Step 2** and **Step 4** in Fig.10, the $k$-th tuples of $st_P$ satisfies the following equation.

$$P_k = \mathsf{Com}(\boldsymbol{p}_k, \boldsymbol{D}_k),\ \boldsymbol{p}_k = (\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} \text{ for } k \in [\ell] \tag{66}$$

The $\ell + 1$-th row $(\boldsymbol{F}, P_\ell, \boldsymbol{p}_\ell)$ satisfies $P_{\ell+1} = \mathsf{Com}(\boldsymbol{p}_{\ell+1}; \boldsymbol{D}_{\ell+1})$ from **Step 1** in Fig.10. Then, the instance $\begin{bmatrix} (\boldsymbol{D}_k, P_k, x_k) \text{ for } k \in [\ell] \\ (\boldsymbol{D}_{\ell+1}, P_{\ell+1}, \ \cdot\ ) \end{bmatrix}$ satisfies the conditions in the relation $\mathcal{R}_{\mathsf{AggMEC.Col}}$. By the perfect completeness of AggMEC.Col, acceptance of AggMEC.Col verifier is guaranteed.

(*Witness-Extended-Emulation*) For the computational witness-extended emulation, we construct an expected polynomial time extractor $\chi$ whose goal is to extract the witness by using a polynomially bounded tree of accepting transcripts. To construct the extractor $\chi$, we use a extractor $\mathcal{E}_2$, which extract a witness by using transcript of AggMEC.Col. From hypothesis, an polynomial time extractor $\mathcal{E}_2$ are given.

Due to the general forking lemma, it is sufficient to construct an extractor $\chi$ that extracts a witness from a suitable tree of accepting transcripts in probabilistic polynomial time.

We begin with $\underbrace{(3,\ldots,3)}_{\log_2 n}$-tree of accepting transcripts. Since the number of challenges in the tree ($3^{\log n}$) is bounded above by a polynomial in $n$ and security parameter $\lambda$, we can apply the general forking lemma.

In **Case 1**, if $st_P$ is empty, it means the protocol has never run **Case 2**, so that the proof system proves the relation $\mathcal{R}_{\mathsf{IP}}^{1,1}$. Since the prover sends the witness $(a,b)$ to the verifier, the extractor $\chi$ get a witness from the prover directly. If $st_P$ is non-empty, then $\chi$ run a extractor $\mathcal{E}_2$ for all leaf nodes of the tree of accepting transcripts. Note that the tree of accepting transcripts is polynomially bounded and $\mathcal{E}_2$ is expected polynomial time, so that $\chi$ spends polynomially bounded operations in this step. By Theorem 15, $\mathcal{E}_2$ extract a witness $(\widehat{\boldsymbol{p}},\boldsymbol{p})$ of $\mathcal{R}_{\mathsf{AggMEC.Col}}$. We would like to emphasize that $\mathcal{E}_2$ has run for all accepting transcripts, so that from now we can assume that for each accepting transcript, $\chi$ knows a tuple $(\widehat{\boldsymbol{p}},\boldsymbol{p})$ satisfying the following equations, regardless of the challenges.

$$P = \mathsf{Com}(\boldsymbol{p};\boldsymbol{D}_{\ell+1}) \tag{67}$$
$$\widehat{P} = \mathsf{Com}(\widehat{\boldsymbol{p}};\boldsymbol{D}_{\ell}) \tag{68}$$
$$\widehat{\boldsymbol{p}} = (\boldsymbol{p}_1 \parallel \boldsymbol{p}_4)^{x} \circ (\boldsymbol{p}_2 \parallel \boldsymbol{p}_3)^{x^{-1}} \tag{69}$$

Next, we show that for **Case 2**, $\chi$ can extract a witness $(\boldsymbol{a},\boldsymbol{b})$ of each recursive round by using the previous round witness $(\widehat{\boldsymbol{a}},\widehat{\boldsymbol{b}})$ that can be extracted by either the previous round extractor of **Case 2** or the extractor of **Case 1**.

The extractor $\chi$ runs the prover until **Step 2** to get $c_L$ and $c_R$. At this point of time, $\chi$ rewinds the prover with 3 distinct challenges $x_1, x_2, x_3$ and obtains tuples $(\widehat{\boldsymbol{a}}_i,\widehat{\boldsymbol{b}}_i)$ for $i \in [3]$ that satisfy the following equations.

$$\widehat{P}_i = \mathsf{Com}(g^{\widehat{\boldsymbol{a}}_i} \parallel h^{\widehat{\boldsymbol{b}}_i};\boldsymbol{D}_{\ell}) \tag{70}$$
$$\widehat{c}_i = \langle \widehat{\boldsymbol{a}}_i, \widehat{\boldsymbol{b}}_i \rangle \tag{71}$$

$\widehat{P}_i$ is contained in $st_V$, so that, by the discrete logarithm relation on $\mathbb{G}_q$, the openings of Eq.(68) and Eq.(70) must be identical for all $i \in [3]$. That is, $\widehat{\boldsymbol{p}} = g^{\widehat{\boldsymbol{a}}_i} \parallel h^{\widehat{\boldsymbol{b}}_i}$. Combining this equation with Eq.(69), we have

$$(\boldsymbol{p}_1 \parallel \boldsymbol{p}_4)^{x_i} \circ (\boldsymbol{p}_2 \parallel \boldsymbol{p}_3)^{x_i^{-1}} = \widehat{\boldsymbol{p}}_i = g^{\widehat{\boldsymbol{a}}_i} \parallel h^{\widehat{\boldsymbol{b}}_i} \tag{72}$$

where all the exponents are known to the extractor.

The right-hand side of Eq.(72) consists of a vector of products of the CRS $g$ and $h$ whose discrete log is unknown. The left-hand side of Eq.(72) has exponents $(x_i, x_i^{-1})$. Hence, similar to the proof of the original BP-IP, for two challenges $x_1$ and $x_2$, the extractor computes the inverse matrix of a $2 \times 2$ matrix with row $(x_i, x_i^{-1})_{i=1}^{2}$. Then, by using it, $\chi$ finds exponent vectors $\boldsymbol{a}_1^*, \boldsymbol{a}_{-1}^*, \boldsymbol{b}_1^*, \boldsymbol{b}_{-1}^* \in \mathbb{Z}_p^{n/2}$ satisfying the following equations.

$$\begin{aligned} g^{\boldsymbol{a}_1^*} &= \boldsymbol{p}_1, \quad g^{\boldsymbol{a}_{-1}^*} = \boldsymbol{p}_2 \\ h^{\boldsymbol{b}_1^*} &= \boldsymbol{p}_3, \quad h^{\boldsymbol{b}_{-1}^*} = \boldsymbol{p}_4 \end{aligned} \tag{73}$$

Putting Eq.(73) into Eq.(72), by the discrete logarithm relation assumption on $\mathbb{G}_p$, the exponents of the bases $g$ and $h$ satisfy the following relation.

$$\boxed{g \text{ exponents}} : \widehat{\boldsymbol{a}}_i = \ x_i \boldsymbol{a}_1^* + x_i^{-1} \boldsymbol{a}_{-1}^* \tag{74}$$

$$\boxed{h \text{ exponents}} : \widehat{\boldsymbol{b}}_i = \ x_i^{-1} \boldsymbol{b}_1^* + x_i \boldsymbol{b}_{-1}^* \tag{75}$$

From given two challenge-witness pair $(x_1, (\widehat{\boldsymbol{a}}_1, \widehat{\boldsymbol{b}}_1))$ and $(x_2, (\widehat{\boldsymbol{a}}_2, \widehat{\boldsymbol{b}}_2))$, the extractor $\chi$ computes $\boldsymbol{a}_1^*$, $\boldsymbol{a}_{-1}^*$, $\boldsymbol{b}_1^*$ and $\boldsymbol{b}_{-1}^*$.

Now we claim that the desired vectors $\boldsymbol{a}^* = \boldsymbol{a}_1^* \parallel \boldsymbol{a}_{-1}^*$ and $\boldsymbol{b}^* = \boldsymbol{b}_1^* \parallel \boldsymbol{b}_{-1}^*$ are a witness of the instance $(g, h, \boldsymbol{D}_{\ell+1}, P)$ of the relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$. That is, we show that $P = \mathsf{Com}(g^{\boldsymbol{a}^*} \parallel h^{\boldsymbol{b}^*}; \boldsymbol{D}_{\ell+1}) \wedge c = \langle \boldsymbol{a}^*, \boldsymbol{b}^* \rangle$ holds. The first constraint is rather straightforward; from Eq.(67), we get

$$P = \mathsf{Com}(\boldsymbol{p}_1 \parallel \boldsymbol{p}_2 \parallel \boldsymbol{p}_3 \parallel \boldsymbol{p}_4; \boldsymbol{D}_{\ell+1}) = \mathsf{Com}(g^{\boldsymbol{a}^*} \parallel h^{\boldsymbol{b}^*}; \boldsymbol{D}_{\ell+1}) \tag{76}$$

To check the inner product relation $c = \langle \boldsymbol{a}^*, \boldsymbol{b}^* \rangle$, we consider the equation.

$$\begin{aligned} c + x_i^2 c_L + x_i^{-2} c_R = \widehat{c} =& \langle \widehat{\boldsymbol{a}}_i, \widehat{\boldsymbol{b}}_i \rangle \\ =& \langle x_i \boldsymbol{a}_1^* + x_i^{-1} \boldsymbol{a}_{-1}^*, x_i^{-1} \boldsymbol{b}_1^* + x_i \boldsymbol{b}_{-1}^* \rangle \\ =& \langle \boldsymbol{a}_1^*, \boldsymbol{b}_1^* \rangle + \langle \boldsymbol{a}_{-1}^*, \boldsymbol{b}_{-1}^* \rangle + x_i^2 \langle \boldsymbol{a}_1^*, \boldsymbol{b}_{-1}^* \rangle + x_i^{-2} \langle \boldsymbol{a}_{-1}^*, \boldsymbol{b}_1^* \rangle \end{aligned} \tag{77}$$

In the similar way in Eq.(59), the equation $c = \langle \boldsymbol{a}_1^*, \boldsymbol{b}_1^* \rangle + \langle \boldsymbol{a}_{-1}^*, \boldsymbol{b}_{-1}^* \rangle = \langle \boldsymbol{a}^*, \boldsymbol{b}^* \rangle$ holds. Therefore, we conclude that the desired witness $(\boldsymbol{a}^*, \boldsymbol{b}^*)$ is a witness of the instance $(g, h, \boldsymbol{D}_{\ell+1}, P)$ for relation $\mathcal{R}_{\mathsf{IP}}^{1,n}$.

For each recursion, the extractor $\chi$ extracts a witness in at most polynomial time. Therefore, we conclude that the protocol Protocol4.Col provides computational witness-extended-emulation by the general forking lemma. □

## D.4 AggMEC.Row

In this section, we focus on AggMEC.Row, the argument of knowledge for the relation $\mathcal{R}_{\mathsf{AggMEC.Row}}$. We provide a full description of AggMEC.Row in Fig.11.

**Uniform Random String** Public inputs of the protocol AggMEC.Row consist of a uniform random string $(\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{K}, U) \in \mathbb{G}_q^{O(n)} \times \mathbb{G}_q^{O(n)} \times \mathbb{G}_q^{18n \log m + 6n} \times \mathbb{G}_q$ and lists of instance $(\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k)$. The $\boldsymbol{G}$ and $\boldsymbol{H}$ are used in a subprotocol Comp.BP$_{AC}$. In a nutshell, $\boldsymbol{G}$ and $\boldsymbol{H}$ are commitment keys for left/right wires and output wires of arithmetic circuit $\mathcal{C}$ respectively. $\boldsymbol{K}$ and $U$ are used in inner product arguments, which guarantee knowledge of the input wires $\boldsymbol{a}_V$ and its inner product. Note that the URS $(\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{K}, U)$ is corresponding to $\mathsf{pp}_1$ in Protocol4.Row.

**Two Inner Product Arguments** To aggregate proofs, both the prover and the verifier aggregate each instance $\boldsymbol{F}_k, S_k$ and $P_k$ with challenge $w$. Additionally, the prover aggregates $\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k$ to $\boldsymbol{a}_V$. For guarantee the knowledge of $\boldsymbol{a}_V$, the prover sends a commitment $C$ before receiving the challenge $w$ (**Step 1** in Fig.11). And the prover and the verifier run two inner product arguments (**Step 5**). The two arguments guarantee that $\langle \boldsymbol{a}_1, \boldsymbol{u} \circ \boldsymbol{w} \rangle = \langle \boldsymbol{a}_2 \circ \boldsymbol{w}, \boldsymbol{u} \rangle$ for $C = \boldsymbol{F}_{base}^{\boldsymbol{a}_1}$, $V = \boldsymbol{F}_{base}^{\boldsymbol{a}_2 \circ \boldsymbol{w}}$ with random $u$ and $v$. In other words, the message $\boldsymbol{a}_V$ of a commitment $C$ is identical to the witness $\boldsymbol{a}_V$ of Comp.BP$_{AC}$

$$\boxed{\mathsf{AggMEC.Row}(\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{K}, U, \begin{bmatrix} (\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k) \\ (\ \cdot\ , \boldsymbol{F}_{\ell+1},\ \cdot\ , P_{\ell+1},\ \cdot) \end{bmatrix}; \begin{bmatrix} (\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k) \\ (\ \cdot\ ,\ \cdot\ , \boldsymbol{p}_{\ell+1}) \end{bmatrix} \text{ for } k \in [\ell])}$$

Let $\mathcal{C}$ be a fan-in 2 arithmetic circuit over $\mathbb{Z}_q$ that takes $(\boldsymbol{l}_k \parallel \boldsymbol{p}_{k+1} \parallel \boldsymbol{r}_k) \in \mathbb{Z}_q^{18n}$ for $k \in [\ell]$ as input and outputs $\boldsymbol{l}_k{}^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k{}^{x_k^{-2}} - \boldsymbol{p}_k \in \mathbb{Z}_q^{6n}$ and $Y^2Z - X^3 - aX - b \in \mathbb{Z}_q$ for all components $(X, Y, Z) \in \mathbb{Z}_q^3$ of the vectors $(\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_{\ell+1})$ for all $k$.

Let $\boldsymbol{a}_L, \boldsymbol{a}_R$, and $\boldsymbol{a}_O$ be vectors indicating values of left input wires, right input wires, and output wires of all multiplicative gates in $\mathcal{C}$ respectively, except that the final output wires, which are all set as zeros.

Let $\boldsymbol{a}_V$ be $\boldsymbol{l}_1 \parallel \boldsymbol{r}_1 \parallel \boldsymbol{l}_2 \parallel \boldsymbol{r}_2 \parallel \cdots \parallel \boldsymbol{l}_\ell \parallel \boldsymbol{r}_\ell \parallel \boldsymbol{p}_1 \parallel \cdots \parallel \boldsymbol{p}_{\ell+1} \in \mathbb{Z}_q^{18\ell n + 6n}$.

Let $(\boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O, \boldsymbol{W}_V, \boldsymbol{c}; \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O, \boldsymbol{a}_V)$ be an Hadamard-product relation such that $\boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O$ and $\boldsymbol{W}_L \cdot \boldsymbol{a}_L + \boldsymbol{W}_R \cdot \boldsymbol{a}_R + \boldsymbol{W}_O \cdot \boldsymbol{a}_O = \boldsymbol{W}_V \cdot \boldsymbol{a}_V + \boldsymbol{c}$ are equations expressing the relations among wires in $\mathcal{C}$.

**Step 1**: $\mathcal{P}$ sends $C = \boldsymbol{F}_{base}^{\boldsymbol{a}_V}$, a commitment to $\boldsymbol{a}_V$, where

$$\boldsymbol{F}_{base} := \boldsymbol{S}_1 \parallel \cdots \parallel \boldsymbol{S}_\ell \parallel \boldsymbol{F}_1 \parallel \cdots \parallel \boldsymbol{F}_{\ell+1} \in \mathbb{G}_q^{18\ell n + 6n}.$$

**Step 2**: $\mathcal{V}$ chooses $u, w \xleftarrow{\$} \mathbb{Z}_q^*$ and sends it to $\mathcal{P}$

**Step 3**: Let $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{Z}_q^{18\ell n + 6n}$ be vectors consisting of powers of $w$ satisfying

$$\boldsymbol{F}_{base}^{\boldsymbol{w}} = \boldsymbol{S}_1 \parallel \cdots \parallel \boldsymbol{S}_\ell^{w^{\ell-1}} \parallel \boldsymbol{F}_1^{w^\ell} \parallel \cdots \parallel \boldsymbol{F}_{\ell+1}^{w^{2\ell}} \in \mathbb{G}_q^{18\ell n + 6n}.$$
$$\boldsymbol{u} = (1, u, u^2, \cdots, u^{18\ell n + 6n - 1})$$

$\mathcal{P}$ sends $t := \langle \boldsymbol{a}_V, \boldsymbol{u} \circ \boldsymbol{w} \rangle$ to $\mathcal{V}$.

**Step 4**: $\mathcal{V}$ chooses $s \xleftarrow{\$} \mathbb{Z}_q^*$ and sends it to $\mathcal{P}$

**Step 5** : $\mathcal{P}$ and $\mathcal{V}$ set

$$V = \left( \prod_{k=1}^{\ell} S_k^{w^{k-1}} \right) \cdot \left( \prod_{j=1}^{\ell+1} P_j^{w^{\ell+j-1}} \right) \in \mathbb{G}_q$$

and run

$\mathsf{Comp.BP}_{AC}(\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{F}_{base}^{\boldsymbol{w}}, \boldsymbol{K}, U, V, \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O, \boldsymbol{W}_V, \boldsymbol{c}; \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O, \boldsymbol{a}_V)$

$\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{F}_{base}, \boldsymbol{K}, U^s, \boldsymbol{K}^{\boldsymbol{u} \circ \boldsymbol{w}} \cdot U^{s \cdot t} \cdot C^{s^2}; \boldsymbol{a}_V, \boldsymbol{u} \circ \boldsymbol{w})$

$\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{F}_{base}^{\boldsymbol{w}}, \boldsymbol{K}, U^s, \boldsymbol{K}^{\boldsymbol{u}} \cdot U^{s \cdot t} \cdot V^{s^2}; \boldsymbol{a}_V \circ \boldsymbol{w}, \boldsymbol{u})$

**Fig. 11.** AggMEC.Row

**Theorem 14.** *Let* Comp.BP$_{\mathsf{AC}}$ *have perfect completeness and computational witness-extended-emulation. Then, the protocol* AggMEC.Row *has perfect completeness and computational witness-extended-emulation under the general discrete logarithm assumption.*

*Proof.* (*Completeness*) In order to show the completeness, we employ the perfect completeness of $\mathsf{Comp.BP}_{AC}$, so that it is sufficient to show the input of $\mathsf{Comp.BP}_{AC}$ is well constructed. That is, we show that the three constraints in $\mathsf{Comp.BP}_{AC}$ holds: (1) $V = \mathsf{Com}(\boldsymbol{a}_V; \boldsymbol{F}^{\boldsymbol{w}}_{base})$, (2) $\boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O$, (3) $\boldsymbol{W}_L \cdot \boldsymbol{a}_L + \boldsymbol{W}_R \cdot \boldsymbol{a}_R + \boldsymbol{W}_O \cdot \boldsymbol{a}_O = \boldsymbol{W}_V \cdot \boldsymbol{a}_V + \boldsymbol{c}$

Assume that an instance $\begin{bmatrix} (\boldsymbol{S}_k, \boldsymbol{F}_k, S_k, P_k, x_k) \text{ for } k \in [\ell] \\ (\ \cdot\ , \boldsymbol{F}_{\ell+1},\ \cdot\ , P_{\ell+1},\ \cdot) \end{bmatrix}$ has a witness $\begin{bmatrix} (\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k) \text{ for } k \in [\ell] \\ (\ \cdot\ ,\ \cdot\ , \boldsymbol{p}_{\ell+1}) \end{bmatrix}$ for the relation $\mathcal{R}_{\mathsf{AggMEC.Row}}$.

In **Step 3**, $V$ is defined as

$$\begin{aligned} V &= \left(\prod_{k=1}^{\ell} S_k^{w^{k-1}}\right) \cdot \left(\prod_{j=1}^{\ell+1} P_j^{w^{\ell+j-1}}\right) \\ &= \prod_{k=1}^{\ell} \mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k; \boldsymbol{S}_k^{w^{k-1}}) \prod_{j=1}^{\ell+1} \mathsf{Com}(\boldsymbol{p}_j; \boldsymbol{F}_j^{w^{\ell+j-1}}). \end{aligned}$$

Since $\boldsymbol{a}_V$ is defined as $\boldsymbol{l}_1 \parallel \boldsymbol{r}_1 \parallel \boldsymbol{l}_2 \parallel \boldsymbol{r}_2 \parallel \cdots \parallel \boldsymbol{l}_\ell \parallel \boldsymbol{r}_\ell \parallel \boldsymbol{p}_1 \parallel \cdots \parallel \boldsymbol{p}_{\ell+1} \in \mathbb{Z}_q^{18\ell n + 6n}$, the above equation is equal to $\mathsf{Com}(\boldsymbol{a}_V; \boldsymbol{F}^{\boldsymbol{w}}_{base})$. Then, the first constraint (1) holds.

By the definition of the Hadamard-product relation set in the protocol, the constraints (2) and (3) hold if the circuit $\mathcal{C}$, which takes $\boldsymbol{a}_V$ as input, outputs zeros. Since $(\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_k, \boldsymbol{p}_{k+1})$'s are a witness of $\mathcal{R}_{\mathsf{AggMEC.Row}}$, the equality ${\boldsymbol{l}_k}^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ {\boldsymbol{r}_k}^{x_k^{-2}} = \boldsymbol{p}_k$ holds. Therefore, the circuit $\mathcal{C}$'s output ${\boldsymbol{l}_k}^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ {\boldsymbol{r}_k}^{x_k^{-2}} - \boldsymbol{p}_k$ should be equal to a zero vector. This completes the perfect completeness proof.

(*Witness-Extended-Emulation*) For the computational witness-extended emulation for $\mathsf{AggMEC.Row}$, we construct a polynomial time extractor $\chi$ whose goal is to extract the witness by using polynomially many accepting transcripts. By the hypothesis, the extractor $\chi$ can employ a witness-extended-emulation $\mathcal{E}$ for $\mathsf{Comp.BP}_{AC}$.

$\chi$ begins with running the prover to get $C$. Then, $\chi$ rewinds the prover $2\ell+1$ times and feeds $2\ell+1$ distinct challenges $w_i$ for $i \in [2\ell+1]$. Then $\chi$ runs $\mathcal{E}$ to obtain $2\ell+1$ tuples $(\boldsymbol{a}_L^{(i)}, \boldsymbol{a}_R^{(i)}, \boldsymbol{a}_O^{(i)}, \boldsymbol{a}_V^{(i)})$. We claim that $\boldsymbol{a}_V^{(i)}$ is identical regardless of $w_i$. Indeed, $C$ is committed before the time of rewinding and thus its opening is unique under the discrete logarithm relation assumption. Then the final argument system in **Step 3** proving the equality between the opening of $C$ and the opening of $V$ guarantees that $\boldsymbol{a}_V^{(i)} = \boldsymbol{a}_V^{(j)}$ holds for $i \neq j$. Thus, for the sake of simplicity, we simply use $\boldsymbol{a}_V$ to denote $\boldsymbol{a}_V^{(i)}$ regardless of $w$ challenges.

We know that $\boldsymbol{a}_V$ passes the protocol $\mathsf{Comp.BP}_{AC}$, so that the following equations hold.

$$\boldsymbol{a}_V = \boldsymbol{l}_1 \parallel \boldsymbol{r}_1 \parallel \cdots \parallel \boldsymbol{l}_\ell \parallel \boldsymbol{r}_\ell \parallel \boldsymbol{p}_1 \parallel \cdots \parallel \boldsymbol{p}_{\ell+1} \tag{78}$$

$$\boldsymbol{p}_k = {\boldsymbol{l}_k}^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ {\boldsymbol{r}_k}^{x_k^{-2}} \text{ for } k \in [\ell] \tag{79}$$

To complete the proof, we show that the extracted values $(\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_j)$ for $k \in [\ell], j \in [\ell+1]$) are a witness of $\mathcal{R}_{\mathsf{AggMEC.Row}}$. One constraint is already satisfied by Eq.(79). We consider two remained constraints of $\mathcal{R}_{\mathsf{AggMEC.Row}}$ such that $P_j$ and $S_k$ are commitment to $\boldsymbol{p}_j$ for $j \in [\ell+1]$ and $\boldsymbol{l}_k \| \boldsymbol{r}_k$ for $k \in [\ell]$, respectively.

$V$ is defined as not only a product of $S_k$'s and $P_j$'s in **Step 3**, but also a commitment to $\boldsymbol{a}_V$ with the commitment key $\boldsymbol{F}^{\boldsymbol{w}}_{base}$. Thus, we have the following equality.

$$\left(\prod_{k=1}^{\ell} S_k^{w_i^{k-1}}\right)\cdot\left(\prod_{j=1}^{\ell+1} P_j^{w_i^{\ell+j-1}}\right) = V = \mathsf{Com}(\boldsymbol{a}_V;\boldsymbol{F}^{\boldsymbol{w}}_{base})$$
$$= \prod_{k=1}^{\ell}\mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k;\boldsymbol{S}_k^{w_i^{k-1}})\prod_{j=1}^{\ell+1}\mathsf{Com}(\boldsymbol{p}_j;\boldsymbol{F}_j^{w_i^{\ell+j-1}})$$
$$= \prod_{k=1}^{\ell}\mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k;\boldsymbol{S}_k)^{w_i^{k-1}}\prod_{j=1}^{\ell+1}\mathsf{Com}(\boldsymbol{p}_j;\boldsymbol{F}_j)^{w_i^{\ell+j-1}} \tag{80}$$

From $2\ell+1$ equations with distinct $w_i$'s, we can separate Eq.(80) into the following desired $2\ell+1$ equations.

$$S_k = \mathsf{Com}(\boldsymbol{l}_k \parallel \boldsymbol{r}_k;\boldsymbol{S}_k) \text{ for } k\in[\ell] \tag{81}$$
$$P_j = \mathsf{Com}(\boldsymbol{p}_j;\boldsymbol{F}_j) \text{ for } j\in[\ell+1] \tag{82}$$

Then $(\boldsymbol{l}_k, \boldsymbol{r}_k, \boldsymbol{p}_j$ for $k\in[\ell], j\in[\ell+1])$ is a witness for the relation $\mathcal{R}_{\mathsf{AggMEC.Row}}$. The extractor $\chi$ extracts a witness from $2\ell+1$ transcripts with $2\ell+1$ times running of polynomial time extractor $\mathcal{E}$. Therefore the protocol AggMEC.Row provides computational witness-extended emulation. □

## D.5 AggMEC.Col

In this section, we focus on AggMEC.Col, the argument of knowledge for the relation $\mathcal{R}_{\mathsf{AggMEC.Col}}$. We provide a full description of AggMEC.Col in Fig.12. In the similar way in AggMEC.Row, AggMEC.Col requires uniform random string $(\boldsymbol{G},\boldsymbol{H},\boldsymbol{K},U)\in\mathbb{G}_q^{O(n)}\times\mathbb{G}_q^{O(n)}\times\mathbb{G}_q^{6\cdot(2^{\ell+1}-1}\times\mathbb{G}_q$. Note that the URS $(\boldsymbol{G},\boldsymbol{H},\boldsymbol{K},U)$ is corresponding to $\mathsf{pp}_2$ in Protocol4.Col.

**Theorem 15.** *Let* Comp.BP$_{\mathsf{AC}}$ *have perfect completeness and computational witness-extended-emulation. Then, the protocol* AggMEC.Col *has perfect completeness and computational witness-extended-emulation under the general discrete logarithm assumption.*

*Proof.* (*Completeness*) In order to show the completeness, we employ the perfect completeness of Comp.BP$_{\mathsf{AC}}$, so that it is sufficient to show the input of Comp.BP$_{\mathsf{AC}}$ is well constructed. That is, we show that the three constraints in Comp.BP$_{\mathsf{AC}}$ holds: (1) $V=\mathsf{Com}(\boldsymbol{a}_V;\boldsymbol{D}^{\boldsymbol{w}}_{base})$, (2) $\boldsymbol{a}_L\circ\boldsymbol{a}_R=\boldsymbol{a}_O$, (3) $\boldsymbol{W}_L\cdot\boldsymbol{a}_L+\boldsymbol{W}_R\cdot\boldsymbol{a}_R+\boldsymbol{W}_O\cdot\boldsymbol{a}_O=\boldsymbol{W}_V\cdot\boldsymbol{a}_V+\boldsymbol{c}$

Assume that an instance $\begin{bmatrix}(\boldsymbol{D}_k,P_k,x_k)\text{ for }k\in[\ell]\\(\boldsymbol{D}_{\ell+1},P_{\ell+1},\ \cdot\ )\end{bmatrix}$ has a witness $\begin{bmatrix}(\boldsymbol{p}_k)\text{ for }k\in[\ell]\\(\boldsymbol{p}_{\ell+1})\end{bmatrix}$ for the relation $\mathcal{R}_{\mathsf{AggMEC.Col}}$.
In **Step 3**, $V$ is defined as

$$V=\left(\prod_{k=1}^{\ell+1}P_k^{w^{k-1}}\right)=\prod_{k=1}^{\ell+1}\mathsf{Com}(\boldsymbol{p}_k;\boldsymbol{D}_k^{w^{k-1}})$$

Since $\boldsymbol{a}_V$ is defined as $\boldsymbol{p}_1\parallel\cdots\parallel\boldsymbol{p}_{\ell+1}\in\mathbb{Z}_q^{6\cdot(2^{\ell+1}-1)}$, the above equation is equal to $\mathsf{Com}(\boldsymbol{a}_V;\boldsymbol{D}^{\boldsymbol{w}}_{base})$. Then, the first constraint (1) holds.

$$\boxed{\mathsf{AggMEC.Col}(\boldsymbol{G},\boldsymbol{H},\boldsymbol{K},U,\begin{bmatrix}(\boldsymbol{D}_k,P_k,x_k)\\(\boldsymbol{D}_{\ell+1},P_{\ell+1},\ \cdot\ )\end{bmatrix};\begin{bmatrix}(\boldsymbol{p}_k)\\(\boldsymbol{p}_{\ell+1})\end{bmatrix}\ \text{for } k\in[\ell])}$$

Let $\mathcal{C}$ be a fan-in 2 arithmetic circuit over $\mathbb{Z}_q$ that takes $(\boldsymbol{p}_{k+1})\in\mathbb{Z}_q^{3\cdot 2^{k+1}}$ for $k\in[\ell]$ as input and outputs $(\boldsymbol{p}_{1,k+1}\parallel\boldsymbol{p}_{4,k+1})^{x_k}\circ(\boldsymbol{p}_{2,k+1}\parallel\boldsymbol{p}_{3,k+1})^{x_k^{-1}}-\boldsymbol{p}_k\in\mathbb{Z}_q^{3\cdot 2^k}$ and $Y^2Z-X^3-aX-b\in\mathbb{Z}_q$ for all components $(X,Y,Z)\in\mathbb{Z}_q^3$ of the vectors $(\boldsymbol{l}_k,\boldsymbol{r}_k,\boldsymbol{p}_{\ell+1})$ for all $k$.

Let $\boldsymbol{a}_L,\boldsymbol{a}_R$, and $\boldsymbol{a}_O$ be vectors indicating values of left input wires, right input wires, and output wires of all multiplicative gates in $\mathcal{C}$ respectively, except that the final output wires, which are all set as zeros.

Let $\boldsymbol{a}_V$ be $\boldsymbol{p}_1\parallel\cdots\parallel\boldsymbol{p}_{\ell+1}\in\mathbb{Z}_q^{6\cdot(2^{\ell+1}-1)}$.

Let $(\boldsymbol{W}_L,\boldsymbol{W}_R,\boldsymbol{W}_O,\boldsymbol{W}_V,\boldsymbol{c};\boldsymbol{a}_L,\boldsymbol{a}_R,\boldsymbol{a}_O,\boldsymbol{a}_V)$ be an Hadamard-product relation such that $\boldsymbol{a}_L\circ\boldsymbol{a}_R=\boldsymbol{a}_O$ and $\boldsymbol{W}_L\cdot\boldsymbol{a}_L+\boldsymbol{W}_R\cdot\boldsymbol{a}_R+\boldsymbol{W}_O\cdot\boldsymbol{a}_O=\boldsymbol{W}_V\cdot\boldsymbol{a}_V+\boldsymbol{c}$ are equations expressing the relations among wires in $\mathcal{C}$.

**Step 1**: $\mathcal{P}$ sends $C=\boldsymbol{D}_{base}^{\boldsymbol{a}_V}$, a commitment to $\boldsymbol{a}_V$, where

$$\boldsymbol{D}_{base}:=\boldsymbol{D}_1\parallel\cdots\parallel\boldsymbol{D}_{\ell+1}\in\mathbb{G}_q^{6\cdot(2^{\ell+1}-1)}.$$

**Step 2**: $\mathcal{V}$ chooses $u,w\xleftarrow{\$}\mathbb{Z}_q^*$ and sends it to $\mathcal{P}$

**Step 3**: Let $\boldsymbol{u},\boldsymbol{w}\in\mathbb{Z}_q^{6\cdot(2^{\ell+1}-1)}$ be vectors consisting of powers of $w$ satisfying

$$\boldsymbol{D}_{base}^{\boldsymbol{w}}=\boldsymbol{D}_1\parallel\cdots\parallel\boldsymbol{D}_{\ell+1}^{w^\ell}\in\mathbb{G}_q^{6\cdot(2^{\ell+1}-1)}.$$
$$\boldsymbol{u}=(1,u,u^2,\cdots,u^{6\cdot(2^{\ell+1}-1)})$$

$\mathcal{P}$ sends $t:=\langle\boldsymbol{a}_V,\boldsymbol{u}\circ\boldsymbol{w}\rangle$ to $\mathcal{V}$.

**Step 4**: $\mathcal{V}$ chooses $s\xleftarrow{\$}\mathbb{Z}_q^*$ and sends it to $\mathcal{P}$

**Step 5** : $\mathcal{P}$ and $\mathcal{V}$ set

$$V=\left(\prod_{k=1}^{\ell+1}P_k^{w^{k-1}}\right)\in\mathbb{G}_q$$

and run

$\mathsf{Comp.BP}_{AC}(\boldsymbol{G},\boldsymbol{H},\boldsymbol{D}_{base}^{\boldsymbol{w}},\boldsymbol{K},U,V,\boldsymbol{W}_L,\boldsymbol{W}_R,\boldsymbol{W}_O,\boldsymbol{W}_V,\boldsymbol{c};\boldsymbol{a}_L,\boldsymbol{a}_R,\boldsymbol{a}_O,\boldsymbol{a}_V)$

$\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{D}_{base},\boldsymbol{K},U^s,\boldsymbol{K}^{\boldsymbol{u}\circ\boldsymbol{w}}\cdot U^{s\cdot t}\cdot C^{s^2};\boldsymbol{a}_V,\boldsymbol{u}\circ\boldsymbol{w})$

$\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{D}_{base}^{\boldsymbol{w}},\boldsymbol{K},U^s,\boldsymbol{K}^{\boldsymbol{u}}\cdot U^{s\cdot t}\cdot V^{s^2};\boldsymbol{a}_V\circ\boldsymbol{w},\boldsymbol{u})$

**Fig. 12.** AggMEC.Col

By the definition of the Hadamard-product relation set in the protocol, the constraints (2) and (3) hold if the circuit $\mathcal{C}$, which takes $\boldsymbol{a}_V$ as input, outputs zeros. Since $(\boldsymbol{p}_k,\boldsymbol{p}_{k+1})$'s are a witness of $\mathcal{R}_{\mathsf{AggMEC.Col}}$, the equality $(\boldsymbol{p}_{1,k+1}\parallel\boldsymbol{p}_{4,k+1})^{x_k}\circ(\boldsymbol{p}_{2,k+1}\parallel$

$\boldsymbol{p}_{3,k+1})^{x_k^{-1}} = \boldsymbol{p}_k$ holds. Therefore, the circuit $\mathcal{C}$'s output $(\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} - \boldsymbol{p}_k$ should be equal to a zero-vector. This completes the perfect completeness proof.

(*Witness-Extended-Emulation*) For the computational witness-extended emulation for $\mathsf{AggMEC.Col}$, we construct a polynomial time extractor $\chi$ whose goal is to extract the witness by using polynomially many accepting transcripts. By the hypothesis, the extractor $\chi$ can employ a witness-extended-emulation $\mathcal{E}$ for $\mathsf{Comp.BP}_{AC}$.

$\chi$ begins with running the prover to get $C$. Then, $\chi$ rewinds the prover $\ell+1$ times and feeds $\ell+1$ distinct challenges $w_i$ for $i \in [\ell+1]$. Then $\chi$ runs $\mathcal{E}$ to obtain $\ell+1$ tuples $(\boldsymbol{a}_L^{(i)}, \boldsymbol{a}_R^{(i)}, \boldsymbol{a}_O^{(i)}, \boldsymbol{a}_V^{(i)})$. In the same way in proof of Theorem D.4, $\boldsymbol{a}_V^{(i)}$ is identical regardless of $w_i$. Thus, we simply use $\boldsymbol{a}_V$ to denote $\boldsymbol{a}_V^{(i)}$ regardless of $w$ challenges.

We know that $\boldsymbol{a}_V$ passes the protocol $\mathsf{Comp.BP}_{AC}$, so that the following equations hold.

$$\boldsymbol{a}_V = \boldsymbol{p}_1 \parallel \cdots \parallel \boldsymbol{p}_{\ell+1} \tag{83}$$

$$\boldsymbol{p}_k = (\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} \text{ for } k \in [\ell] \tag{84}$$

To complete the proof, we show that the extracted values $(\boldsymbol{p}_k)$ for $k \in [\ell+1]$ are a witness of $\mathcal{R}_{\mathsf{AggMEC.Row}}$. One constraint is already satisfied by Eq.(84). We consider two remained constraints of $\mathcal{R}_{\mathsf{AggMEC.Col}}$ such that $P_k$ are commitment to $\boldsymbol{p}_k$ for $k \in [\ell+1]$.

$V$ is defined as not only a product of $S_k$'s and $P_j$'s in **Step 3**, but also a commitment to $\boldsymbol{a}_V$ with the commitment key $\boldsymbol{D}_{base}^{\boldsymbol{w}}$. Thus, we have the following equality.

$$\begin{aligned}\left(\prod_{k=1}^{\ell+1} P_k^{w_i^{k-1}}\right) = V &= \mathsf{Com}(\boldsymbol{a}_V; \boldsymbol{D}_{base}^{\boldsymbol{w}}) \\ &= \prod_{k=1}^{\ell+1} \mathsf{Com}(\boldsymbol{p}_k; \boldsymbol{D}_k^{w_i^{k-1}}) \\ &= \prod_{k=1}^{\ell+1} \mathsf{Com}(\boldsymbol{p}_k; \boldsymbol{D}_k)^{w_i^{k-1}} \end{aligned} \tag{85}$$

From $\ell+1$ equations with distinct $w_i$'s, we can separate Eq.(85) into the following desired $\ell+1$ equations.

$$P_k = \mathsf{Com}(\boldsymbol{p}_k; \boldsymbol{D}_k) \text{ for } k \in [\ell+1] \tag{86}$$

Then ($\boldsymbol{p}_k$ for $k \in [\ell+1]$) is a witness for the relation $\mathcal{R}_{\mathsf{AggMEC.Col}}$. The extractor $\chi$ extracts a witness from $\ell+1$ transcripts with $\ell+1$ times running of polynomial time extractor $\mathcal{E}$. Therefore, the protocol $\mathsf{AggMEC.Col}$ provides computational witness-extended emulation. □

## D.6 $\mathsf{Comp.BP}_{AC}$

The commitment scheme is often used in proof systems for committing to elements of the witness. For example, Bünz et al.'s proof system for arithmetic circuits [19] allows inputs to be Pedersen commitments to elements of the witness. In particular, it uses the original Pedersen commitment scheme (without hiding property) to an integer. Using the homomorphic property of Pedersen commitment, the verifier can perform some verification process using the separated committed integers. However, the separated committed integers derive the linear size of commitments in the number of committed integers.

In this section, we propose an argument for arithmetic circuits, which allows inputs to be a generalized Pedersen commitment, denoted by $V := \boldsymbol{F}^{\boldsymbol{a}_V} \in \mathbb{G}_q$, to a vector of elements of the witness $\boldsymbol{a}_V \in \mathbb{Z}_q^M$, where $\mathbb{G}_q$ is a group of order $q$ and $\boldsymbol{F} \in \mathbb{G}_q^M$ is the commitment key. It enables to design a much shorter proof since it compresses a vector of commitments to an integer in [19] into a commitment to a vector of integers. Instead, the verifier in our protocol cannot publicly perform homomorphic operation over the committed values, so that this verification should be proven by an additional subprotocol, which is not significant compared to the original task.

Using the reduction from the arithmetic circuit to Hadamard-product relation in [19], the following relation is sufficient for the proof systems for the arithmetic circuit with additional input $V := \boldsymbol{F}^{\boldsymbol{a}_V} \in \mathbb{G}_q$.

$$\mathcal{R}_{AC} = \left\{ \begin{array}{l} \big(\boldsymbol{G}, \boldsymbol{H} \in \mathbb{G}_q^N, \boldsymbol{F}, \boldsymbol{K} \in \mathbb{G}_q^M, U, V \in \mathbb{G}_q, \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O \in \mathbb{Z}_q^{Q\times N}, \\ \boldsymbol{W}_V \in \mathbb{Z}_q^{Q\times M}, \boldsymbol{c} \in \mathbb{Z}_q^Q; \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O \in \mathbb{Z}_q^N, \boldsymbol{a}_V \in \mathbb{Z}_q^M\big) : \\ \boxed{V = \boldsymbol{F}^{\boldsymbol{a}_V}} \wedge \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_o \\ \wedge \boldsymbol{W}_L \cdot \boldsymbol{a}_L + \boldsymbol{W}_R \cdot \boldsymbol{a}_R + \boldsymbol{W}_O \cdot \boldsymbol{a}_O = \boldsymbol{W}_V \cdot \boldsymbol{a}_V + \boldsymbol{c} \end{array} \right\} \quad (87)$$

We build a proof system for the relation $\mathcal{R}_{AC}$ on the basis of the inner-product argument $\mathsf{BP}_{\mathsf{IP}}$ of the Bulletproofs [19]. The proposed protocol, called $\mathsf{Comp.BP}_{AC}$, is a reduction to $\mathsf{BP}_{\mathsf{IP}}$ and described in Fig.13.

**Efficiency Analysis** We discuss the efficiency of $\mathsf{Comp.BP}_{AC}$. In this section, we neglect the field operations in $\mathbb{Z}_q$ in complexity analysis. And we assume that the circuit size $N$ is larger than the circuit input length $M$.

First, we consider the computational complexity. In **Step 1**, $\mathcal{P}$ performs $O(N \log q)$ $\mathbb{G}_q$-operations for $A_I$ and $A_O$. And both $\mathcal{P}$ and $\mathcal{V}$ perform $O(N \log q)$ $\mathbb{G}_q$-operations for $\boldsymbol{H}'$ and $P$ in **Step 3** and **Step 7** respectively. Finally, both $\mathcal{P}$ and $\mathcal{V}$ update $\mathsf{BP}_{\mathsf{IP}}$ instances $U^w, P \cdot U^{w\cdot\hat{t}}, \boldsymbol{F}^{w^2}, V^{w^2} \cdot \boldsymbol{K}^{\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_V} \cdot U^{w\cdot v}$ which are required $O(N \log q)$ $\mathbb{G}_q$-operations. After reduction, $\mathcal{P}$ and $\mathcal{V}$ run two $\mathsf{BP}_{\mathsf{IP}}$ protocols, which require $O((N \log q + M \log q)$ $\mathbb{G}_q$-operations in total. Since $N$ is larger than $M$, we conclude total $\mathcal{P}/\mathcal{V}$ computation complexity is $O(N \log q)$ $\mathbb{G}_q$-operations.

Let consider the proof size. In the reduction process, $\mathcal{P}$ sends four $\mathbb{Z}_q$-elements $(v, t_1, t_3, \hat{t})$ to the verifier. After reduction, $\mathcal{P}$ and $\mathcal{V}$ run two $\mathsf{BP}_{\mathsf{IP}}$. Each $\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{G}, \boldsymbol{H}', U^w, P \cdot U^{w\cdot\hat{t}}; \boldsymbol{l}, \boldsymbol{r})$ and $\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{F}^{w^2}, \boldsymbol{K}, U^w, V^{w^2} \cdot \boldsymbol{K}^{\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_V} \cdot U^{w\cdot v}; \boldsymbol{a}_V, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V)$ require $\log N$ $\mathbb{G}_q$-elements and $\log M$ $\mathbb{G}_q$-elements, respectively. Then, total proof size is bound $O(\log N)$ $\mathbb{G}_q$-elements.

**Theorem 16.** *The* $\mathsf{Comp.BP}_{AC}$ *has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption.*

*Proof.* Since $\mathsf{BP}_{\mathsf{IP}}$ has perfect completeness and computational witness-extended emulation under the discrete logarithm relation assumption (Thereom 4), we prove only the reduction step in $\mathsf{Comp.BP}_{AC}$ for perfect completeness and computational witness-extended emulation.

(*Completeness*) In the last step, both the prover and the verifier run two inner-product arguments $\mathsf{BP}_{\mathsf{IP}}$. Hence, it is sufficient to show that the inputs of two inner-product arguments are well calculated. That is, $(\boldsymbol{G}, \boldsymbol{H}', U, P \cdot U^{w\cdot\hat{t}}; \boldsymbol{l}, \boldsymbol{r}) \in \mathcal{R}_{\mathsf{IP}}$ and

$(\boldsymbol{F}^{w^2}, \boldsymbol{K}, U^w, V^{w^2} \cdot \boldsymbol{K}^{\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_V} \cdot U^{w\cdot v}; \boldsymbol{a}_V, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V) \in \mathcal{R}_{\mathsf{IP}}$. The first inclusion relation is rather straightforward from the calculation in **Step 3**. For the second inclusion relation, one can check $P \cdot U^{w\hat{t}}$ is a valid Pedersen commitment to a vector $(\boldsymbol{l}, \boldsymbol{r}, \langle \boldsymbol{l}, \boldsymbol{r}\rangle)$ with the commitment key $(\boldsymbol{G}, \boldsymbol{H}', U^w)$ from the following equalities.

$$\begin{aligned}
\boldsymbol{G}^{\boldsymbol{l}} \cdot \boldsymbol{H}'^{\boldsymbol{r}} =& \boldsymbol{G}^{\boldsymbol{a}_L \cdot x + \boldsymbol{a}_O \cdot x^2 + \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R)\cdot x} \cdot \boldsymbol{H}'^{\boldsymbol{y}^N \circ \boldsymbol{a}_R \cdot x - \boldsymbol{y}^N + \boldsymbol{z}^{Q+1}\cdot (\boldsymbol{W}_L \cdot x + \boldsymbol{W}_O)} \\
=& (\boldsymbol{G}^{\boldsymbol{a}_L} \boldsymbol{H}'^{\boldsymbol{y}^N \circ \boldsymbol{a}_R})^x \cdot (\boldsymbol{G}^{\boldsymbol{a}_O})^{x^2} \cdot \boldsymbol{H}'^{-\boldsymbol{y}^N + \boldsymbol{z}^{Q+1}\cdot(\boldsymbol{W}_L \cdot x + \boldsymbol{W}_O)} \cdot \boldsymbol{G}^{\boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_R)\cdot x} \\
=& A_I^x \cdot A_O^{x^2} \cdot \boldsymbol{H}'^{-\boldsymbol{y}^N + \boldsymbol{z}^{Q+1}\cdot (x\boldsymbol{W}_L + \boldsymbol{W}_O)} \cdot \boldsymbol{G}^{\boldsymbol{y}^{-N}\circ(\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_R) x} \\
=& P \\
\langle \boldsymbol{l}, \boldsymbol{r}\rangle =& t_1 x + t_2 x^2 + t_3 x^3 \\
=& t_1 x + (\langle \boldsymbol{a}_L, \boldsymbol{a}_R \circ \boldsymbol{y}^N\rangle - \langle \boldsymbol{a}_O, \boldsymbol{y}^N\rangle + \langle \boldsymbol{z}^{Q+1}, \boldsymbol{w}\rangle + \delta(y,z))x^2 + t_3 x^3 \\
=& t_1 x + (\langle \boldsymbol{z}^{Q+1}, \boldsymbol{W}_V \cdot \boldsymbol{a}_V + \boldsymbol{c}\rangle + \delta(y,z) +)x^2 + t_3 x^3 \\
=& t_1 x + (v + \langle \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V, \boldsymbol{c}\rangle + \delta(y,z))x^2 + t_3 x^3 \\
=& \widehat{t}
\end{aligned}$$

(*Witness-Extended-Emulation*) By the general forking lemma, it is sufficient to construct an expected polynomial time extractor $\chi$ that extracts the witness by using a polynomially bounded tree of accepting transcripts. By Theorem 4, two subprotocols for the inner-product have the computational witness-extended emulation under the discrete logarithm relation assumption. Thus, the extractor $\chi$ can employ $\mathsf{BP}_{\mathsf{IP}}$ extractor $\mathcal{E}$ at most polynomial time.

For each challenge, the extractor begins with running two subprotocol's extractors to obtain $\boldsymbol{l}$, $\boldsymbol{r}$, $\boldsymbol{a}_V$, and $\boldsymbol{a}_W$ satisfying the condition

$$P \cdot U^{w\cdot \hat{t}} = \boldsymbol{G}^{\boldsymbol{l}} \boldsymbol{H}'^{\boldsymbol{r}} U^{w\cdot \langle \boldsymbol{l}, \boldsymbol{r}\rangle} \tag{88}$$

of the first inner-product relation and the condition

$$V^{w^2} \cdot \boldsymbol{K}^{\boldsymbol{z}^{Q+1}\cdot W_V} \cdot U^{w\cdot v} = (\boldsymbol{F}^{w^2})^{\boldsymbol{a}_V} \boldsymbol{K}^{\boldsymbol{a}_W} U^{w\cdot \langle \boldsymbol{a}_V, \boldsymbol{a}_W\rangle} \tag{89}$$

of the second inner-product relation. From the above two equations, we claim that $\hat{t} = \langle \boldsymbol{l}, \boldsymbol{r}\rangle$ and $v = \langle \boldsymbol{a}_V, \boldsymbol{a}_W\rangle$. If for a different challenge $w'$, we can compute a different pair $(\boldsymbol{l}', \boldsymbol{r}')$, then we have

$$P = \boldsymbol{G}^{\boldsymbol{l}} \boldsymbol{H}^{\boldsymbol{y}^{-N}\circ \boldsymbol{r}} U^{w\cdot(\langle \boldsymbol{l}, \boldsymbol{r}\rangle - \hat{t})} = \boldsymbol{G}^{\boldsymbol{l}'} \boldsymbol{H}^{\boldsymbol{y}^{-N}\circ \boldsymbol{r}'} U^{w'\cdot(\langle \boldsymbol{l}', \boldsymbol{r}'\rangle - \hat{t})}$$

that yields a non-trivial solution of the discrete logarithm relation problem. Thus, under the discrete logarithm relation assumption, we can set $\boldsymbol{l}$ and $\boldsymbol{r}$ obtained by the extractor is independent from the choice of the challenge $w$. Since $P$ is publicly computable by the equation in **Step 7**, we have

$$\boldsymbol{G}^{\boldsymbol{l}} \boldsymbol{H}'^{\boldsymbol{r}} U^{w\langle \boldsymbol{l}, \boldsymbol{r}\rangle} = A_I^x \cdot A_O^{x^2} \cdot \boldsymbol{H}'^{-\boldsymbol{y}^N + \boldsymbol{z}^{Q+1}\cdot (x\boldsymbol{W}_L + \boldsymbol{W}_O)} \cdot \boldsymbol{G}^{\boldsymbol{y}^{-N}\circ(\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_R)x} \cdot U^{w\hat{t}}$$

Using two challenges $w$, the above equation separates into

$$\boldsymbol{G}^{\boldsymbol{l}} \boldsymbol{H}'^{\boldsymbol{r}} = A_I^x \cdot A_O^{x^2} \cdot \boldsymbol{H}'^{-\boldsymbol{y}^N + \boldsymbol{z}^{Q+1}\cdot (x\boldsymbol{W}_L + \boldsymbol{W}_O)} \cdot \boldsymbol{G}^{\boldsymbol{y}^{-N}\circ(\boldsymbol{z}^{Q+1}\cdot \boldsymbol{W}_R)x} \tag{90}$$

$$U^{\langle \boldsymbol{l}, \boldsymbol{r}\rangle} = U^{\hat{t}} \tag{91}$$

The equation Eq. (91), we obtain a desired result $\hat{t} = \langle \boldsymbol{l}, \boldsymbol{r} \rangle$. Applying a similar argument with three challenges to Eq. (89), we can obtain another desired result $v = \langle \boldsymbol{a}_V, \boldsymbol{a}_W \rangle$, $V = \boldsymbol{F}^{\boldsymbol{a}_V}$, and $\boldsymbol{a}_W = \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V$, which can be re-written as

$$V = \boldsymbol{F}^{\boldsymbol{a}_V} \tag{92}$$

$$v = \langle \boldsymbol{a}_V, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V \rangle. \tag{93}$$

Until now we have extracted only a witness $\boldsymbol{a}_V$ using $w$ challenges. Next, we use $x, y, z$ challenges to extract the other witness $\boldsymbol{a}_L$, $\boldsymbol{a}_R$, and $\boldsymbol{a}_O$ and to show all the witness satisfy the remained desired constraints. The upper equation Eq. (90) consists of a product of $\boldsymbol{G}, \boldsymbol{H}', A_I$ and $A_O$. Using two challenges $x_1$ and $x_2$ such that $(x_i, x_i^2)$ are linearly independent, we can find $\boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_{O,L}$, and $\boldsymbol{a}_{O,R}$ that satisfy the following equations.

$$A_I = \boldsymbol{G}^{\boldsymbol{a}_L} \cdot \boldsymbol{H}^{\boldsymbol{a}_R}$$
$$A_O = \boldsymbol{G}^{\boldsymbol{a}_{O,L}} \cdot \boldsymbol{H}^{\boldsymbol{a}_{O,R}}$$

The discrete logarithm relation assumption guarantees the uniqueness of the above representations, particularly regardless of the challenges. Combining the above equations with (90), we obtain the following equalities under the discrete logarithm relation assumption.

$$\boldsymbol{l} = \boldsymbol{a}_L \cdot x + \boldsymbol{a}_{O,L} \cdot x^2 + \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R) x \tag{94}$$

$$\boldsymbol{r} = \boldsymbol{y}^N \circ \boldsymbol{a}_R \cdot x + \boldsymbol{y}^N \circ \boldsymbol{a}_{O,R} \cdot x^2 - \boldsymbol{y}^N + \boldsymbol{z}^{Q+1} \cdot (x \boldsymbol{W}_L + \boldsymbol{W}_O) \tag{95}$$

In particular, the above equalities hold regardless of the challenges. From Eq. (91), Eq. (93) and the equation in **Step 6**, we know that

$$\begin{aligned} \langle \boldsymbol{l}, \boldsymbol{r} \rangle = \hat{t} &= t_1 x + \big(\delta(y,z) + v + \langle \boldsymbol{z}^{Q+1}, \boldsymbol{c} \rangle\big) x^2 + t_3 x^3 \\ &= t_1 x + \big(\delta(y,z) + \langle \boldsymbol{a}_V, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V \rangle + \langle \boldsymbol{z}^{Q+1}, \boldsymbol{c} \rangle\big) x^2 + t_3 x^3, \end{aligned} \tag{96}$$

where all coefficients in $x$, $x^2$, and $x^3$ are chosen or defined before the selection of the challenge $x$. Thus, for each $(y, z)$ challenge pair, we consider three polynomials $l(X), r(X), t(X)$ of degree at most 4 whose evaluation at $x$ is the value in Eq. (94), Eq. (95), Eq. (96), respectively. In particular, we consider evaluations for 4 distinct $x$ challenges. Then, all coefficients are uniquely defined since we have 4 points of polynomials of degree at most 4. (In fact, all the other values except $x$ is predefined regardless of $x$, and thus we know all coefficients already.)

Computing the inner product between $l(X)$ and $r(X)$, the coefficient in $X^2$ is

$$\begin{aligned} &\langle \boldsymbol{a}_L + \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R), \boldsymbol{y}^N \circ \boldsymbol{a}_R + \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_L \rangle + \langle \boldsymbol{a}_{O,L}, -\boldsymbol{y}^N + \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_O \rangle \\ =&\langle \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R), \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_L \rangle + \langle \boldsymbol{a}_L, \boldsymbol{y}^N \circ \boldsymbol{a}_R \rangle - \langle \boldsymbol{a}_{O,L}, \boldsymbol{y}^N \rangle \\ &+ \langle \boldsymbol{a}_L, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_L \rangle + \langle \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R), \boldsymbol{y}^N \circ \boldsymbol{a}_R \rangle + \langle \boldsymbol{a}_{O,L}, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_O \rangle \\ =&\delta(y,z) + \langle \boldsymbol{a}_L, \boldsymbol{y}^N \circ \boldsymbol{a}_R \rangle - \langle \boldsymbol{a}_{O,L}, \boldsymbol{y}^N \rangle + \langle \boldsymbol{z}^{Q+1}, \boldsymbol{W}_L \boldsymbol{a}_L + \boldsymbol{W}_R \boldsymbol{a}_R + \boldsymbol{W}_O \boldsymbol{a}_{O,L} \rangle. \end{aligned}$$

Considering the above equation with the coefficient in $X^2$ of $t(X)$, we have

$$\langle \boldsymbol{a}_{O,L} - \boldsymbol{a}_L \circ \boldsymbol{a}_R, \boldsymbol{y}^N \rangle = \langle \boldsymbol{z}^{Q+1}, \boldsymbol{W}_L \boldsymbol{a}_L + \boldsymbol{W}_R \boldsymbol{a}_R + \boldsymbol{W}_O \boldsymbol{a}_{O,L} - \boldsymbol{W}_V \boldsymbol{a}_V - \boldsymbol{c} \rangle.$$

The above equality holds for any challenges $y$ and $z$. Fixing a $z$ (and so $\boldsymbol{z}^{Q+1}$), $\langle \boldsymbol{a}_{O,L} - \boldsymbol{a}_L \circ \boldsymbol{a}_R, \boldsymbol{y}^N \rangle$ is a constant regardless of $\boldsymbol{y}^N$. Thus, considering $n+1$ different $y$ challenges (and so $\boldsymbol{y}_i^n$ for $i \in [n+1]$), we know that $\boldsymbol{a}_{O,L} - \boldsymbol{a}_L \circ \boldsymbol{a}_R$ is orthogonal to $n$ linearly independent vectors $\boldsymbol{y}_i^n - \boldsymbol{y}_{n+1}^n$ for $i \in [n]$, so that we have

$$\boldsymbol{a}_{O,L} - \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{0}. \tag{97}$$

Similarly, using $Q+1$ different $z$ challenges, we obtain the following equation.

$$\boldsymbol{W}_L \cdot \boldsymbol{a}_L + \boldsymbol{W}_R \cdot \boldsymbol{a}_R + \boldsymbol{W}_O \cdot \boldsymbol{a}_{O,L} - \boldsymbol{W}_V \cdot \boldsymbol{a}_V - \boldsymbol{c} = \boldsymbol{0} \tag{98}$$

The three equations Eq. (92), Eq. (97), and Eq. (98) are exactly the constraints in the relation $\mathcal{R}_{AC}$. Therefore, the extracted tuple $(\boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_{O,L}, \boldsymbol{a}_V)$ is a witness of the relation $\mathcal{R}_{AC}$. $\square$

$\boxed{\mathsf{Comp.BP}_{AC}(\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{F}, \boldsymbol{K}, U, V, \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O, \boldsymbol{W}_V, \boldsymbol{c}; \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O, \boldsymbol{a}_V)}$

**Step 1**: $\mathcal{P}$ computes

$$A_I = \boldsymbol{G}^{\boldsymbol{a}_L}\boldsymbol{H}^{\boldsymbol{a}_R} \in \mathbb{G}_q,\ A_O = \boldsymbol{G}^{\boldsymbol{a}_O} \in \mathbb{G}_q$$

and sends $A_I, A_O$ to $\mathcal{V}$.
**Step 2**: $\mathcal{V}$ chooses $y, z \xleftarrow{\$} \mathbb{Z}_q^*$ and returns it to $\mathcal{P}$.
**Step 3**: Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$\begin{aligned}
&\boldsymbol{y}^N := (1, y, y^2, \dots, y^{N-1}) \in \mathbb{Z}_q^N,\quad \boldsymbol{z}^{Q+1} := (z, z^2, \dots, z^Q) \in \mathbb{Z}_q^Q,\\
&\delta(y,z) := \langle \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R), \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_L \rangle,\\
&H_i' := H_i^{y^{-i+1}},\ \forall i \in [N] \text{ and } \boldsymbol{H}' := (H_1', \dots, H_N').
\end{aligned}$$

Additionally, $\mathcal{P}$ computes

$$\begin{aligned}
&l(X) := \boldsymbol{a}_L \cdot X + \boldsymbol{a}_O \cdot X^2 + \boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R) \cdot X \in \mathbb{Z}_q^N[X],\\
&r(X) := \boldsymbol{y}^N \circ \boldsymbol{a}_R \cdot X - \boldsymbol{y}^N + \boldsymbol{z}^{Q+1} \cdot (\boldsymbol{W}_L \cdot X + \boldsymbol{W}_O) \in \mathbb{Z}_q^N[X],\\
&t(X) := \langle l(X), r(X) \rangle = \sum_{i \in [3]} t_i \cdot X^i \in \mathbb{Z}_q[X],\\
&\boldsymbol{w} := \boldsymbol{W}_L \cdot \boldsymbol{a}_L + \boldsymbol{W}_R \cdot \boldsymbol{a}_R + \boldsymbol{W}_O \cdot \boldsymbol{a}_O \in \mathbb{Z}_q^Q,\\
&t_2 := \langle \boldsymbol{a}_L, \boldsymbol{a}_R \circ \boldsymbol{y}^N \rangle - \langle \boldsymbol{a}_O, \boldsymbol{y}^N \rangle + \langle \boldsymbol{z}^{Q+1}, \boldsymbol{w} \rangle + \delta(y,z) \in \mathbb{Z}_q,\\
&v := \langle \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V, \boldsymbol{a}_V \rangle \in \mathbb{Z}_q
\end{aligned}$$

and sends $v$ and $t_i$ for $i \in \{1, 3\}$ to $\mathcal{V}$.
**Step 4**: $\mathcal{V}$ chooses $x \xleftarrow{\$} \mathbb{Z}_q^*$ and returns it to $\mathcal{P}$.
**Step 5**: $\mathcal{P}$ computes

$$\boldsymbol{l} := l(x) \in \mathbb{Z}_q^N,\quad \boldsymbol{r} := r(x) \in \mathbb{Z}_q^N,\quad \hat{t} := \langle \boldsymbol{l}, \boldsymbol{r} \rangle \in \mathbb{Z}_q$$

and sends $\hat{t}$ to $\mathcal{V}$.
**Step 6**: $\mathcal{V}$ checks the following equation.

$$\hat{t} = t_1 x + \big(\delta(y,z) + v + \langle \boldsymbol{z}^{Q+1}, \boldsymbol{c} \rangle\big)x^2 + t_3 x^3$$

If the equation holds, $\mathcal{V}$ chooses $w \xleftarrow{\$} \mathbb{Z}_q^*$ and returns it to $\mathcal{P}$.
**Step 7**: Both $\mathcal{P}$ and $\mathcal{V}$ compute

$$P = A_I^x \cdot A_O^{x^2} \cdot \boldsymbol{H}'^{-\boldsymbol{y}^N + \boldsymbol{z}^{Q+1} \cdot (x\boldsymbol{W}_L + \boldsymbol{W}_O)} \cdot \boldsymbol{G}^{\boldsymbol{y}^{-N} \circ (\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_R)x} \in \mathbb{G}_q$$

**Step 8**: Both $\mathcal{P}$ and $\mathcal{V}$ run two protocols $\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{G}, \boldsymbol{H}', U^w, P \cdot U^{w \cdot \hat{t}}; \boldsymbol{l}, \boldsymbol{r})$ and $\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{F}^{w^2}, \boldsymbol{K}, U^w, V^{w^2} \cdot \boldsymbol{K}^{\boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V} \cdot U^{w \cdot v}; \boldsymbol{a}_V, \boldsymbol{z}^{Q+1} \cdot \boldsymbol{W}_V)$

**Fig. 13.** $\mathsf{Comp.BP}_{AC}$: A Generalization of Bulletproofs-AC For Compact Input

## D.7 Efficiency Analysis

We are interested in the efficiency of Protocol4.Row (Fig. 9). To do this, we present complexities for AggMEC.Row (Fig. 11), AggMEC.Col (Fig. 12), and Protocol4.Col (Fig. 10) in advance. Below, we denote group operations and elements in a group $G$ by $G$-operations and elements, respectively. We also note that because the number of field operations (elements) in $\mathbb{Z}_p$ is negligible compared to those in $\mathbb{G}_q$ for all protocols, we neglect them in complexity analysis.

First, we discuss the efficiency of AggMEC.Row. In Step 1, $\mathcal{P}$ performs $O(n\ell \log q)$ $\mathbb{G}_q$-operations. Both $\mathcal{P}$ and $\mathcal{V}$ perform $O(n\ell \log q)$ $\mathbb{G}_q$-operations in **Step 3** and $O(\ell \log q)$ $\mathbb{G}_q$-operations in **Step 5**. An arithmetic circuit to compute $\boldsymbol{l}_k{}^{x_k^2} \circ \boldsymbol{p}_{k+1} \circ \boldsymbol{r}_k{}^{x_k^{-2}} - \boldsymbol{p}_k \in \mathbb{G}_p^{2n}$ for $k \in [\ell]$ consists of $O(n\ell \log p)$ $\mathbb{G}_p$-operations. Because a single $\mathbb{G}_p$-operation requires 12 multiplication gates over $\mathbb{Z}_q$, the circuit consists of $O(n\ell \log p)$ multiplication gates. In **Step 5**, $\mathcal{P}$ and $\mathcal{V}$ invoke Comp.BP$_{AC}$ for $N \leftarrow O(n\ell \log p)$ and $M \leftarrow O(n\ell)$, which requires $O(n\ell \log p \log q)$ $\mathbb{G}_q$-operations for each $\mathcal{P}$ and $\mathcal{V}$. In addition, $\mathcal{P}$ and $\mathcal{V}$ invoke BP$_{\mathsf{IP}}$ twice for $N \leftarrow O(n\ell)$, which requires $O(n\ell \log q)$ $\mathbb{G}_q$-operations for each $\mathcal{P}$ and $\mathcal{V}$. Thus, the total computation complexity of both $\mathcal{P}$ and $\mathcal{V}$ equals to $O(n\ell \log p \log q)$ $\mathbb{G}_q$-operations. The communication complexity is dominated by that of Comp.BP$_{AC}$, i.e., $O(\log n + \log \ell + \log\log p)$ $\mathbb{G}_q$-elements.

The efficiency of AggMEC.Col can be analyzed similarly with AggMEC.Row. An arithmetic circuit to compute $(\boldsymbol{p}_{1,k+1} \parallel \boldsymbol{p}_{4,k+1})^{x_k} \circ (\boldsymbol{p}_{2,k+1} \parallel \boldsymbol{p}_{3,k+1})^{x_k^{-1}} - \boldsymbol{p}_k \in \mathbb{G}_p^{2k}$ for $k \in [\ell]$ consists of $O(2^\ell \log p)$ multiplication gates over $\mathbb{Z}_q$. **Step 5** executes Comp.BP$_{AC}$ for $N \leftarrow O(2^\ell \log p)$ and $M \leftarrow O(2^\ell)$ and BP$_{\mathsf{IP}}$ for $N \leftarrow O(2^\ell)$ twice. The total computation complexity of both $\mathcal{P}$ and $\mathcal{V}$ becomes $O(2^\ell \log p \log q)$ $\mathbb{G}_q$-operations. The communication complexity equals to $O(\ell + \log\log p)$ $\mathbb{G}_q$-elements.

We proceed to Protocol4.Col. First, we analyze the reduction part. For each $n > 1$, $\mathcal{P}$ performs $O(n \log p)$ $\mathbb{G}_p$-operations in **Step 1** and $O(n \log p)$ $\mathbb{G}_p$-operations, $O(n \log q)$ $\mathbb{G}_q$-operations in **Step 4**. Because $n$ is cut in half at each iteration, $\mathcal{P}$ still performs $O(n \log p)$ $\mathbb{G}_p$-operations and $O(n \log q)$ $\mathbb{G}_q$-operations across all iterations. The communication complexity is dominated by $O(\log n)$ $\mathbb{G}_q$-elements from **Step 4**. When $n = 1$, computation and communication complexities are dominated by the execution of AggMEC.Col with $\ell \leftarrow \log n$. Thus, the total computation complexity of $\mathcal{P}$ consists of $O(n \log p)$ $\mathbb{G}_p$-operations and $O(n \log p \log q)$ $\mathbb{G}_q$-operations. The total computation complexity of $\mathcal{V}$ is $O(n \log p \log q)$ $\mathbb{G}_q$-operations. The communication complexity is $O(\log n + \log\log p)$ $\mathbb{G}_q$-elements.

Finally, we analyze the efficiency of Protocol4.Row. In the reduction part ($m > 1$), the computation cost for $\mathcal{P}$ is dominated by $O(mn \log p)$ $\mathbb{G}_p$-operations (**Step 1** and **Step 2**) and $O(n \log m \log q)$ $\mathbb{G}_q$-operations (**Step 2** and **Step 4**). The computation cost for $\mathcal{V}$ is $O(m \log p)$ $\mathbb{G}_p$-operations from **Step 5**. $\mathcal{P}$ and $\mathcal{V}$ communicate with $O(\log m)$ $\mathbb{G}_q$-elements from **Step 4**. When $m = 1$, the complexities are determined by AggMEC.Row with $\ell \leftarrow \log m$ and Protocol4.Col. Therefore, the total computation complexity of $\mathcal{P}$ consists of $O(mn \log p)$ $\mathbb{G}_p$-operations and $O(n \log m \log p \log q)$ $\mathbb{G}_q$-operations. The total computation complexity of $\mathcal{V}$ consists of $O(m \log p)$ $\mathbb{G}_p$-operations and $O(n \log m \log p \log q)$ $\mathbb{G}_q$-operations. The total communication complexity is $O(\log n + \log m + \log\log p)$ $\mathbb{G}_q$-elements.

## E Extensions

### E.1 Transparent Polynomial Commitment Scheme

**Definition 11 (Commitment Scheme).** *A commitment scheme for a message space $\mathcal{M}$ is a pair of algorithms* $(\mathsf{Setup}, \mathsf{Com})$ *such that*

- $\mathsf{Setup}(1^\lambda) \to ck$*: takes a security parameter $\lambda$ and outputs a commitment key $ck$.*
- $\mathsf{Com}(ck, M) \to c$*: takes a commitment key and a message $M \in \mathcal{M}$ and outputs a commitment $c$.*

*We say that a commitment scheme $\mathcal{C} = (\mathsf{Setup}, \mathsf{Com})$ for a message space $\mathcal{M}$ is* binding *if for all polynomial time adversaries $\mathcal{A}$ the following probability is negligible in $\lambda$*

$$\Pr\left[\begin{array}{c} M_0, M_1 \in \mathcal{M} \ \wedge M_0 \neq M_1 \\ \wedge\ \mathsf{Com}(ck, M_0) = \mathsf{Com}(ck, M_1) \end{array} \middle| \begin{array}{c} \mathsf{Setup}(1^\lambda) \to ck; \\ \mathcal{A}(ck) \to (M_0, M_1) \end{array}\right]$$

*We say that a commitment scheme $\mathcal{C} = (\mathsf{Setup}, \mathsf{Com})$ for a message space $\mathcal{M}$ is* hiding *if for all polynomial time adversaries $\mathcal{A}$ the following probability is negligible in $\lambda$*

$$\left| \Pr\left[ b = \widehat{b} \ \middle| \begin{array}{c} \mathsf{Setup}(1^\lambda) \to ck; \mathcal{A}(ck) \to (M_0, M_1, st); \\ \{0,1\} \xrightarrow{\$} b; \mathsf{Com}(ck, M_b) \to c; \mathcal{A}(ck, st) \to \widehat{b} \end{array}\right] - \frac{1}{2} \right|$$

**Definition 12 (Extractable Polynomial Commitment Scheme).** *A polynomial commitment scheme consists of a 5-tuple of algorithms* $(\mathsf{Setup}, \mathsf{Com}, \mathsf{Eval.Setup}, \mathsf{Eval.Prove}, \mathsf{Eval.Verify})$ *such that*

- $\mathsf{Setup}(\mathbb{F}, d) \to ck$*: takes a polynomial-coefficient field and a maximum degree as input and outputs a commitment key $ck$.*
- $\mathsf{Com}(ck, f(X)) \to c$*: takes a commitment key $ck$ and a polynomial $f(X) \in \mathbb{F}[X]$ of maximum degree $d$ as input and outputs a commitment $c$.*
- $(\mathsf{Eval.Setup}, \mathsf{Eval.Prove}, \mathsf{Eval.Verify})$*: is a (CRS generator, prover, verifier)-tuple of an interactive argument of knowledge with respect to the relation*

  $$\big\{(ck, c, x, z; f(X) \in \mathbb{F}) : \mathsf{Com}(ck, f(X)) \to c \ \wedge \ deg(f(X)) \leq d \ \wedge \ f(x) = z\big\},$$

  *where* $\mathsf{Setup}(\mathbb{F}, d) \to ck$*.*

*We say that a polynomial commitment $\mathcal{PC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Eval.Setup}, \mathsf{Eval.Prove}, \mathsf{Eval.Verify})$ is* extractable *if* $(\mathsf{Setup}, \mathsf{Com})$ *is a binding commitment scheme and if* $(\mathsf{Eval.Setup}, \mathsf{Eval.Prove}, \mathsf{Eval.Verify})$ *has witness-extended emulation.*

The inner-product argument with the Pedersen commitment scheme such as BP-IP can be naturally considered a transparent polynomial commitment scheme and is already used in many prior works (e.g., [52, 18, 21]). For example, a polynomial $f(X) = \sum_{i \in [N]} a_i X^{i-1} \in \mathbb{Z}_p[X]$ can be represented by a vector $\boldsymbol{a} = (a_1, \ldots, a_N)$ of its coefficients. Then, the prover can commit to $\boldsymbol{a}$ using Pedersen commitment (that is, $\boldsymbol{g}^{\boldsymbol{a}}$, where $\boldsymbol{g}$ is the commitment key.) and prove an evaluation at any point $x$ by proving an inner-product relation between $\boldsymbol{a}$ and the vector $(1, x, x^2, \ldots, x^{N-1})$.

In Table 2, we present a comparison for asymptotic complexities of polynomial commitment schemes.

| Scheme | CRS size | Commit size | Opening proof size | $\mathcal{P}$'s computation | | $\mathcal{V}$'s computation | Assump. |
|---|---|---|---|---|---|---|---|
| | | | | Commit | Open | | |
| Groth [35] | $O(\sqrt[3]{N})\mathbb{G}_2$ | $O(\sqrt[3]{N})\mathbb{G}_t$ | $O(\sqrt[3]{N})\mathbb{G}_1$ | $O(N)\tau_1$ | $O(\sqrt[\frac{2}{3}]{N})\tau_1$ | $O(\sqrt[3]{N})\tau_1$ | DPair |
| BP [19] | $O(N)\mathbb{G}_1$ | $O(1)\mathbb{G}_1$ | $O(\log N)\mathbb{G}_1$ | $O(N)\tau_1$ | | | DL |
| Hyrax [52] | $O(\sqrt{N})\mathbb{G}_1$ | | $O(\log N)\mathbb{G}_1$ | $O(N)\tau_1$ | $O(\sqrt{N})\tau_1$ | | DL |
| BFS [20] | $O(N)\mathbb{G}_U$ | $O(1)\mathbb{G}_U$ | $O(\log N)\mathbb{G}_U$ | $O(N)\mathsf{u}$ | $O(N\log_2 N)\mathsf{u}$ | $O(\log_2 N)\mathsf{u}$ | UO |
| Virgo [55] | $O(1)$ | $O(1)\mathbb{H}$ | $O((\log N)^2)\mathbb{H}$ | $O(N\log_2 N)\mathsf{h}$ | | $O((\log_2 N)^2)\mathsf{h}$ | CR hash |
| BMMV [21] | $O(\sqrt{N})\mathbb{G}_2$ | $O(1)\mathbb{G}_t$ | $O(\log N)\mathbb{G}_t$ | $O(N)\tau_1$ | $O(\sqrt{N})\tau_1$ | $O(\sqrt{N})\tau_2$ | DPair |
| Protocol2 | $O(N)\mathbb{G}_1$ | $O(1)\mathbb{G}_1$ | $O(\sqrt{\log N})\mathbb{G}_t$ | $O(N)\tau_1$ | $O(N2^{\sqrt{\log N}})\tau_1$ | $O(N)\tau_1$ | DL&DPair |
| Protocol3 | $O(\sqrt{N})\mathbb{G}_2$ | $O(1)\mathbb{G}_t$ | $O(\log N)\mathbb{G}_t$ | $O(N)\tau_1$ | $O(N)\tau_1$ | $O(\sqrt{N})\tau_2$ | DL |
| Protocol4 | $O(\sqrt{N})\mathbb{G}_q$ | $O(1)\mathbb{G}_q$ | $O(\log N)\mathbb{G}_q$ | $O(N)\tau_p$ | $O(N)\tau_p$ | $O(\sqrt{N}\log N)\tau_q$ | DL |

$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t)$: 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,
$\tau_i$: group operations in $\mathbb{G}_i$, $\mathsf{u}$: group operation in $\mathbb{G}_U$, $\mathsf{p}$: pairing operation, $\mathsf{h}$: hash operation,
$N$: degree of polynomial, DL: discrete logarithm assumption, DPair: double pairing assumption, UO: strong RSA assumption and adaptive root assumption in unknown order groups, CR hash: collision-resistant hashes

**Table 2.** Transparent polynomial commitment schemes

## E.2 Zero-Knowledge Argument for Arithmetic Circuits

**Definition 13 (Perfect Special Honest Verifier Zero-Knowledge).** *A public coin argument $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is perfect special honest verifier zero-knowledge (SHVZK) for $\mathcal{R}$ if there exists probabilistic polynomial time simulator $\mathsf{S}$ such that for all pairs of interactive adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr\left[\begin{array}{c}\mathcal{A}_2(tr) = 1\\ \wedge(\sigma, x, w) \in \mathcal{R}\end{array}\middle|\ \sigma \leftarrow \mathcal{K}(1^\lambda); (x, w, \rho) \leftarrow \mathcal{A}_1(\sigma); tr \leftarrow \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x; \rho)\rangle\right]$$

$$= \Pr\left[\begin{array}{c}\mathcal{A}_2(tr) = 1\\ \wedge(\sigma, x, w) \in \mathcal{R}\end{array}\middle|\ \sigma \leftarrow \mathcal{K}(1^\lambda); (x, w, \rho) \leftarrow \mathcal{A}_1(\sigma); tr \leftarrow \mathsf{S}(x, \rho)\right],$$

*where $\rho$ is the public coin randomness used by $\mathcal{V}$.*

In the above definition, the adversary $\mathcal{A}_1$ chooses a tuple of the statement, witness, and the source of randomness used by $\mathcal{V}$, but $\mathcal{A}_2$ cannot distinguish between the honestly generated transcript by $\mathcal{P}$ and the simulated transcript by $\mathcal{S}$.

Bootle et al. [15] presents a conversion from an arbitrary arithmetic circuit with $N$ multiplication fan-in two gates into a certain relation containing a Hadamard product with linear constraints. Bünz et al. [19] slightly generalizes the relation to include committed values as inputs to the arithmetic circuit, so that the converted relation contains the committed values as well. The formal description of these relations in [15, 19] is given as follows.

$$\left\{\begin{array}{l}\left(\begin{array}{l}\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, \boldsymbol{V} \in \mathbb{G}^M, g, h \in \mathbb{G}, \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O \in \mathbb{Z}_p^{Q\times N}, \boldsymbol{W}_V \in \mathbb{Z}_p^{Q\times M},\\ \boldsymbol{c} \in \mathbb{Z}_p^Q;\ \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O \in \mathbb{Z}_p^N, \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^M\end{array}\right)\\ : V_j = g^{v_j} h^{\gamma_j} \forall j \in [1, m]\ \wedge\ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O\\ \quad \wedge\ \boldsymbol{W}_L \boldsymbol{a}_L^\top + \boldsymbol{W}_R \boldsymbol{a}_R^\top + \boldsymbol{W}_O \boldsymbol{a}_O^\top = \boldsymbol{W}_V \boldsymbol{v}^\top + \boldsymbol{c}^\top\end{array}\right\} \tag{99}$$

where $\boldsymbol{W}_V \in \mathbb{Z}_p^{Q \times M}$ is of rank $M$.

BP-AC is a zero-knowledge argument for Eq. (99), and in particular, Bünz et al. [19] proved the following theorem.

**Theorem 17 (Theorem 5 in [19]).** *There exists an efficient arithmetic circuit protocol for the relation in Eq.* (99) *using the argument for the inner-product relation in Eq.* (1). *In particular, the arithmetic circuit protocol has perfect completeness, SHVZK and computational witness-extended emulation if the discrete logarithm relation assumption holds and the underlying inner-product argument has perfect completeness and computational witness-extended emulation.*

We provide BP-AC in Fig. 5 for the self-containedness. In the reduction phase of BP-AC, the prover sends only 8 group elements and 3 field elements to the verifier for constant rounds, so that the overall communication overhead is asymptotically the same as that of the underlying inner-product argument.

Our sublogarithmic proof system Protocol2 can be combined with BP-AC by replacing BP-IP. In fact, the language in Eq. (2) differs from the relation (1) on the CRS only and the relation is equivalent. Therefore, we can still employ Theorem 17 taking Protocol2 as the underlying inner-product argument. Similarly, we can employ Theorem 17 with taking Protocol3 and Protocol4 as the underlying inner-product argument.