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

Hyoenbum Lee and Jae Hong Seo**

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

Abstract. We propose two zero-knowledge arguments for arithmetic circuits with fan-in 2 gates in the uniform random string model. Our first protocol features $O(\sqrt{\log_2 N})$ communication and round complexities and $O(N)$ computational complexity for the verifier, where N is the size of the circuit. Our second protocol features $O(\log_2 N)$ communication and $O(\sqrt{N})$ computational complexity for the verifier. We prove the soundness of our arguments under the discrete logarithm assumption or the double pairing assumption, which is at least as reliable as the decisional Diffie-Hellman assumption.

The main ingredient of our arguments is two different generalizations of Bünz et al.'s Bulletproofs inner-product argument (IEEE S&P 2018) that convinces a verifier of knowledge of two vectors satisfying an inner-product relation. For a protocol with sublogarithmic communication, we devise a novel method to aggregate multiple arguments for bilinear operations such as multi-exponentiations, which is essential for reducing the communication overhead. For a protocol with a sublinear verifier, we develop a generalization of the discrete logarithm relation assumption, which is essential for reducing verification overhead while keeping the soundness proof solely relying on the discrete logarithm assumption. These techniques are of independent interest.

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 circuits 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 [41, 28], 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 [21].

** Corresponding Author

The first sublinear ZK argument for arithmetic circuits solely based on the hardness of the DL problem is due to Groth [30] and improved by Seo [45]. These works feature constant round complexity as well.

Groth [32] 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 [17], which is called Bulletproofs. Hoffmann, Klooß, and Rupp [36] 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, 17, 36] have linear verifiers and other DL-based ZK argument systems with different asymptotic performance, in particular sublinear verifier, have proposed. e.g., Hyrax [47] and Spartan [46]. Recently, Bünz, Maller, Mishra, and Vesely [19] 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, 34] prior to Bulletproofs. Logarithmic communication complexity in these works is attained with relatively large round complexity, compared to [30, 45].

Relying on the non-standard but reliable assumptions, there exists a ZK argument system with better asymptotic performance due to Bünz, Fisch, and Szepieniec [18] that achieves logarithmic communications and logarithmic verifier simultaneously, but it relies on 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) [31, 40, 12, 29, 13, 43, 8, 11, 34, 33, 41, 28, 48, 22] 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, 23, 50] 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 $\log_2^2 N$ for the circuit size N, which is larger than that of the DL based approach [15, 17].

1.1 Our Results

We propose two ZK arguments for arithmetic circuits with fan-in 2 gates. In particular, for a given arithmetic circuit of size N,

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

2. we present the first ZK argument with *logarithmic communication and sublinear verifier computation* such that its soundness is solely based on the *discrete logarithm assumption*.

Note that all our proposals do not require the structured reference string, so that the proposals are *transparent* proof systems; that is, it does not require a trusted setup. We first propose two different generalizations of Bulletproofs inner-product argument that convinces a verifier of knowledge of two vectors satisfying an inner-product relation, and then we use a method to convert from inner-product argument (without ZK) to ZK argument for arithmetic circuits in [15, 17].

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, 39, 41, 28, 22, 24] 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 inner-product argument with improved complexity, sublogarithmic communications or sublinear verifier. Having inner-product arguments, we can follow Bulletproofs approach for the reduction from inner-product arguments to ZK arguments for arithmetic circuits.

Sublogarithmic Communications. We consider the inner-product 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 $\boldsymbol{a}, \boldsymbol{b}$ are witness vectors and $\boldsymbol{g}, \boldsymbol{h}, u, P$ consist of a statement.

First, we observe that the length of witness is halved for each recursive step in Bulletproofs 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 Bulletproofs. Then, in our generalization, the recursive steps are quickly finished in $\log_{2n} N$ times compared with $\log_2 N$ of Bulletproofs, but the prover sends $2n(2n-1)$ group elements in the generalization, which is larger than Bulletproofs. One can easily verify that the communication cost $O(n^2 \log_{2n} N)$ becomes minimal when $n = 1$, so that the original Bulletproofs has a minimal case of this naïve generalization in terms of proof size. Nevertheless, this naïve generalization is a good starting point toward our target, sublogarithmic inner-product argument without a trusted setup.

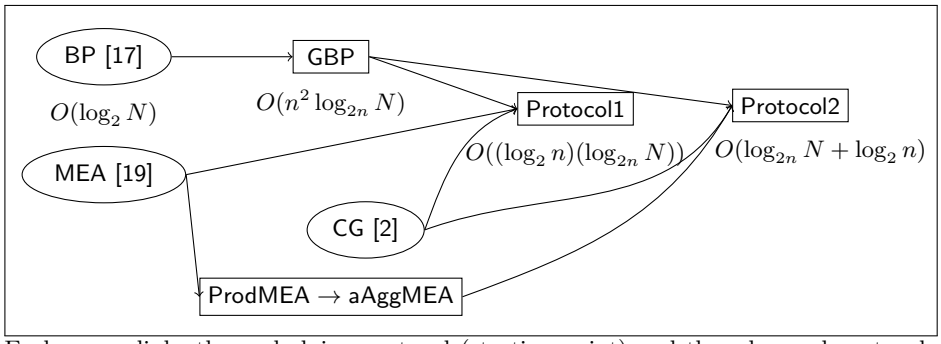
Second, we apply the commit-and-prove approach inside the generalized Bulletproofs to reduce transmission overhead. More precisely, instead of sending $2n(2n-1)$ group elements, the prover sends only a commitment to a vector of $2n(2n-1)$ group elements using homomorphic commitment scheme to group elements (e.g., [1, 2]). In fact, those group elements are necessary for the verifier to update from P to $\widehat{P}$ for the next recursive step and this process cannot be performed by the verifier without knowing those $2n(2n-1)$ group elements. Thus,

Scheme	Communication	$\mathcal{P}$'s comp.	$\mathcal{V}$'s comp.	Assump.
Groth [30] & Seo [45]	$O(\sqrt{N})$	$O(N)$	$O(N)$	DL
Groth [32]	$O(\sqrt[3]{N})$	$O(N)$	$O(\sqrt[3]{N})$	DPair
BulletP. [15, 17] & HKR [36]	$O(\log N)$	$O(N)$	$O(N)$	DL
Hyrax [47]	$O(\sqrt{w} + d \log N)$	$O(N \log N)$	$O(\sqrt{w} + d \log N)$	DL
Spartan 1 [46]	$O(\sqrt{N})$	$O(N)$	$O(\sqrt{N})$	DL
Spartan 2 [46]	$O(\log^2 N)$	$O(N \log N)$	$O(\log^2 N)$	DL
BMMV [19]	$O(\log N)$	$O(N)$	$O(\sqrt{N})$	DPair
Supersonic [18]	$O(\log N)$	$O(N \log N)$	$O(\log N)$	UOGroup
Ligero [3]	$O(\sqrt{N})$	$O(N \log N)$	$O(N)$	CR hash
STARK [7]	$O(\log^2 N)$	$O(N \log^2 N)$	$O(\log^2 N)$	CR hash
Aurora [9]	$O(\log^2 N)$	$O(N \log N)$	$O(N)$	CR hash
Fractal [23]	$O(\log^2 N)$	$O(N \log N)$	$O(\log^2 N)$	CR hash
Virgo [50]	$O(d \log N)$	$O(N \log N)$	$O(d \log N)$	CR hash
Our IP arguments + Transformation to AC arguments (Theorem 12):				
Protocol2 (Section 4)	$O(\sqrt{\log N})$	$O(N2^{\sqrt{\log N}})$	$O(N)$	DL & DPair
Protocol3 (Section 6)	$O(\log N)$	$O(N)$	$O(\sqrt{N})$	DL

Table 1. Comparison for transparent ZK arguments
IP: inner-product, AC: arithmetic circuit, N: circuit size, d: circuit depth, w: input size, DL: discrete logarithm assumption, DPair: double pairing assumption, UOGroup: unknown-order group (strong RSA assumption & adaptive root assumption), CR hash: collision-resistant hashes
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 [25].

the prover sends $\widehat{P}$ along with its proof to the verifier. This process of committing and proving can be achieved using a multi-exponentiation argument (e.g., [19]). Nevertheless, if we naively apply this commit-and-prove approach here, it ends up with asymptotically the same proof size as Bulletproofs since we must prove *multiple* multi-exponentiation arguments. More precisely, the communication cost $O(n^2 \log_{2n} N)$ of the recursive step in the generalized Bulletproofs 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 $O\big((\log_{2n} N) \cdot (\log_2 n^2)\big)$. Overall, its communication cost is $O\big((\log_{2n} N) \cdot (\log_2 n^2)\big) = O(\log_2 N)$, which is asymptotically the same as that of Bulletproofs.

Lastly, in order 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-



Each arrow links the underlying protocol (starting point) and the advanced protocol (ending point). The big-O notation under each protocol indicates communication complexity of the protocol.
The oval nodes indicate known results; BP: Bulletproofs [15, 17], MEA: multi-exponentiation argument [19], CG: Commitment to group elements [2]. The rectangle nodes indicate the proposed protocols; GBP: Generalized Bulletproofs in Fig. 2, ProdMEA: Product of multi-exponentiation argument in Fig. 6, aAggMEA: Augmented aggregation of multi-exponentiation argument in Fig. 5. Protocol1: Inner-product argument in Fig. 3. Protocol2: Sublogarithmic inner-product argument in Fig. 4.
N is the dimension of witness vector. n is a positive integer parameter for GBP, where $n = 1$ implies the original Bulletproofs.

Fig. 1. Our Approach for Logarithmic Communications

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 [17], linear combinations of protocols [36]). 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 Bulletproofs. However, we find that this strategy does not work well. The detailed explanation about difficulty we faced is given in Section 4.2. Instead, we devise an *augmented aggregated multi-exponentiation argument* called aAggMEA; we add redundant values in the target relation for aAggMEA such that the final relation can cover the desired aggregated multi-exponentiation relation, and then we reduce from aAggMEA to a product argument called 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 Bulletproofs and achieve $O\big((\log_{2n} N) + (\log_2 n^2)\big)$ communication overhead. Using aAggMEA, we finally achieve $O(\sqrt{\log_2 N})$ communication cost when setting n to satisfy $O(\log_{2n} N) = O(\log_2 n^2)$. For example, if $N = (2n)^m$ and $n = 2^m$, then the

communication cost is $O(m) = O(\sqrt{\log_2 N})$. We provide a high-level pictorial explanation for our approach in Fig. 1.

Sublinear Verifier From Discrete Logarithms. Now, we consider another protocol whose goal is to achieve sublinear verifier while keeping benefits of Bulletproofs such as logarithmic communications and soundness proof from the discrete logarithm assumption.

First, we start from an observation such that indeed the soundness of Bulletproofs inner-product 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 7). 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.

1.3 Related Work

Argument for Algebraic Relations. Groth [30] 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 inner-product argument to be logarithmic in the dimension of witness [15, 17]. There are proposals for other algebraic relations. For example, Groth and Sahai [35] 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 [38] and Bünz, Maller, Mishra, Vesely [19] 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 [32, 47, 41, 28, 48, 16, 22, 46, 50] but also for constructing many other crypto primitives such as verifiable secret sharing [37, 4], anonymous credentials [20, 27], and proofs of storage and replication [49]. 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 inner-product argument [15, 17, 47, 16].

1.4 Organization

After providing necessary definitions in the next section, we present a naïve generalization of Bulletproofs inner-product argument in Section 3 and reduce its communication overhead in Section 4 by using a newly proposed building block in Section 5. We present another generalization that achieves sublinear CRS size and verifier computation in Section 6. In Section 7, we extend our inner-product arguments to ZK arguments for arithmetic circuits. In Section 8, we discuss a way to combine ideas for two our generalized Bulletproofs protocols.

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 σ. In this paper, the CRS is a randomly generated group element, so that indeed we are in the common random string model, where an argument consists of two interactive PPT algorithms $(\mathcal{P}, \mathcal{V})$ such that $\mathcal{P}$ and $\mathcal{V}$ are called the prover and the verifier, respectively. The transcript produced by an interaction between $\mathcal{P}$ and $\mathcal{V}$ on inputs x and y is denoted by $tr \leftarrow \langle \mathcal{P}(x), \mathcal{V}(y) \rangle$. Since we are in the common random string model, for the sake of simplicity we omit the process of running $\mathcal{K}$ and assume the CRS is given as common input to both $\mathcal{P}$ and $\mathcal{V}$. At the end of transcript, the verifier $\mathcal{V}$ outputs b, which is denoted by $\langle \mathcal{P}(x), \mathcal{V}(y) \rangle = b$, where $b = 1$ if $\mathcal{V}$ accepts and $b = 0$ if $\mathcal{V}$ rejects.

Let $\mathcal{R}$ be a polynomial time verifiable ternary relation consisting of tuples of the CRS σ, a statement x, and a witness w. We define a CRS-dependent language L_σ as the set of statements x that have a witness w such that $(\sigma, x, w) \in \mathcal{R}$. That is, $L_\sigma = \{\ x \mid \exists\ w \text{ satisfying } (\sigma, x, w) \in \mathcal{R}\ \}$. 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 λ.

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 $\mathcal{A}$,*

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

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.*

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

$\mathcal{E}$ 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 [25] in the random oracle model; all $\mathcal{V}$'s challenges can be replaced with hash values of the transcript up to that point.

Definition 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^λ 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, $\{1, \ldots, m\}$. Let $\mathcal{G}$ be an asymmetric bilinear group generator that takes the security parameter λ 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 λ, 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.[1]

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 $\boldsymbol{g}_1 \circ \boldsymbol{g}_2$.

[1] 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 $\sum_{i=\pm 1} \boldsymbol{a}_i x^i$, a witness for the next recursive round.

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 $\boldsymbol{g} = (g_1, \ldots, g_n) \in \mathbb{G}_1^n$ and $\boldsymbol{H} = (H_1, \ldots, H_n) \in \mathbb{G}_2^n$, the inner-pairing product[2] 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 A Generalization of Bulletproofs

Bulletproofs inner-product argument is a proof system, denoted by $\mathsf{BP}_{\mathsf{IP}}$, convincing of the following relation.

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

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. We provide a review of Bulletproofs in the supplementary material.

In this section, we present a generalization of Bulletproofs inner-product argument for the relation in Eq. (1). In each round of $\mathsf{BP}_{\mathsf{IP}}$, each vector in the CRS and witness is split into two equal-length subvectors. We generalize

[2] We call this operation "inner-pairing product" that is named by [19]. Note that this operation is also called "canceling bilinear map" in the context of converting composite-order bilinear groups to prime-order bilinear groups [26].

Bulletproofs by splitting a vector of length N into $2n$ subvectors of length $N/2n$ in each round, where $n = 1$ implies the original Bulletproofs protocol. Similar to Bulletproofs, we assume N is a power of $2n$ for the sake of simplicity. Then, for proving the relation in Eq. (1), let $\widehat{N} = \frac{N}{2n}$ and the prover begins with parsing $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to We provide a review of Bulletproofs in the supplementary material.

In this section, we present a generalization of Bulletproofs inner-product argument for the relation in Eq. (1). In each round of $\mathsf{BP}_{\mathsf{IP}}$, each vector in the CRS and the witness are split into two equal-length subvectors. We generalize Bulletproofs by splitting a vector of length N into $2n$ subvectors of length $N/2n$ in each round, where $n = 1$ implies the original Bulletproofs protocol. Similar to Bulletproofs, we assume N is a power of $2n$ for the sake of simplicity. Then, for proving the relation in Eq. (1), let $\widehat{N} = \frac{N}{2n}$ and the prover begins with parsing $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to

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

Let $I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$ be a $2n$-size index set. In each recursive round of Bulletproofs, 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 Bulletproofs, 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 show that this process is a reduction to a one $2n$-th length inner-product argument. That is, $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$ satisfies Eq. (1). We observe that

$$\begin{aligned} P &= \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} = \prod_{i \in I_n} \boldsymbol{g}_i^{\boldsymbol{a}_i} \boldsymbol{h}_i^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_i, \boldsymbol{b}_i \rangle}, \text{ and thus} \\ \widehat{P} &= \Big(\prod_{i \in I_n} \boldsymbol{g}_i^{\boldsymbol{a}_i} \boldsymbol{h}_i^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_i, \boldsymbol{b}_i \rangle} \Big) \cdot \Big(\prod_{\substack{i,j \in I_n \\ i \neq j}} \big(\boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \big)^{x^{j-i}} \Big) = \prod_{i,j \in I_n} \big(\boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \big)^{x^{j-i}} \\ &= \big(\circ_{i \in I_n} \boldsymbol{g}_i^{x^{-i}} \big)^{\sum_{j \in I_n} \boldsymbol{a}_j x^j} \cdot \big(\circ_{j \in I_n} \boldsymbol{h}_j^{x^j} \big)^{\sum_{i \in I_n} \boldsymbol{b}_i x^{-i}} \cdot u^{\langle \sum_{j \in I_n} \boldsymbol{a}_j x^j, \sum_{i \in I_n} \boldsymbol{b}_i x^{-i} \rangle} \\ &= \widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}} \cdot \widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}} \cdot u^{\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle}. \end{aligned} \tag{2}$$

Therefore, $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$ satisfies the inner-product relation in Eq. (1), and thus an argument of N-length is reduced to the same argument of $\widehat{N}$-length.

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

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

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

Step 1: $\mathcal{P}$ calculates for all distinct $i, j \in I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$,

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

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

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

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. 2. A Generalization of Bulletproofs

We briefly 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 $k \in [4n-1]$.

$$P \prod_{s \in J_n} \Big(\prod_{\substack{i,j \in I_n \\ j-i=s}} v_{i,j} \Big)^{x_k^s} = P \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x_k^{j-i}} = \widehat{P} = \Big(\circ_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i}} \Big)^{\widehat{\boldsymbol{a}}_k} \Big(\circ_{j \in I_n} \boldsymbol{h}_j^{x_k^j} \Big)^{\widehat{\boldsymbol{b}}_k} u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle}$$
$$= \Big(\prod_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i} \widehat{\boldsymbol{a}}_k} \boldsymbol{h}_i^{x_k^i \widehat{\boldsymbol{b}}_k} \Big) u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \tag{3}$$

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. (3). 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

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

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. (3) and obtain the following theorem.

Theorem 1. *The inner-product argument in Fig. 2 has perfect completeness and computational witness-extended-emulation under the discrete logarithm relation assumption.*

The proof of Theorem 1 is relegated to the supplementary material.

3.1 Efficiency

The prover repeats the $(N > 1)$ case $\log_{2n} N$ times and 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 original Bulletproofs inner-product argument protocol. Therefore, the generalization in this section does not have any benefit, at least compared with with the original Bulletproofs.

4 Sublogarithmic Inner-Product Argument from Diffie-Hellman

We improve our generalization of Bulletproofs inner-product argument by using the pairing-based homomorphic commitment scheme to group elements [1, 2]. Furthermore, this additional commitment scheme requires to put the commitment key into the common random string, so that we slightly extend our target relation by adding some uniformly distributed group elements, which become a part of the common random string in our argument. That is, the desired relation that our proof system proves is as follows.

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^N, u \in \mathbb{G}_1, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_m \in \mathbb{G}_2^{2n(2n-1)}, \boldsymbol{H} \in \mathbb{G}_2^m, P \in \mathbb{G}_1; \boldsymbol{a}, \boldsymbol{b}) \\ : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G}_1 \end{array} \right\} \quad (4)$$

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 inner-product protocols will use them to run a subprotocol for multi-exponentiation arguments given in the following subsections.

4.1 Proof Size Reduction using Multi-Exponentiation Argument

As aforementioned, the generalized Bulletproofs in Fig. 2 with $n > 1$ carries larger communication overhead than that of Bulletproofs. 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 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.

$$\left\{ (\boldsymbol{F} \in \mathbb{G}_2^N, \boldsymbol{z} \in \mathbb{Z}_p^N, P \in \mathbb{G}_t, q \in \mathbb{G}_1; \boldsymbol{v} \in \mathbb{G}_1^N) : P = \boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}) \wedge q = \boldsymbol{v}^{\boldsymbol{z}} \right\}, \quad (5)$$

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 (5), but provide an intuitive idea for it. In fact, Bulletproofs inner-product argument can be naturally extended to this proof system due to the resemblance between the standard inner-product 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 inner-product between two integer vectors (e.g., $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$) can be substituted with multi-exponentiation (e.g., $\boldsymbol{v}^{\boldsymbol{z}}$).[3] This type of extension is well formulated by Bünz, Maller, Mishra, and Vesely [19] in terms of two-tiered homomorphic commitment scheme [32]. The multi-exponentiation argument in [19] costs the same complexities as those of Bulletproofs inner-product argument; $O(\log N)$ 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 [32] and the multi-exponentiation argument in [19] so that we can construct a protocol with shorter communication overhead. The full description of the scheme is provided in Fig. 3. 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.

Although this approach reduces communication overheads, compared to the generalized Bulletproofs, it is not quite beneficial for our purpose. More precisely, the communication overhead $O(n^2 \log_{2n} N)$ of the generalized Bulletproofs is reduced to $O((\log_2 n) \cdot (\log_{2n} N))$ since the communication overhead per round $O(n^2)$ is reduced to its logarithm $O(\log_2 n)$ by the multi-exponentiation argument. Although the communication overhead is reduced to $O((\log_2 n) \cdot (\log_{2n} N))$

[3] The Bulletproofs 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., [47]. Our multi-exponentiation argument corresponds to this variant.

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

If $N = 1$:

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

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

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

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

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

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

Step 1: $\mathcal{P}$ calculates for all distinct $i \neq j \in I_n = \{\pm 1, \pm 3, \ldots, \pm(2n-1)\}$,

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

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

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

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. 3. Protocol1

compared with the generalized Bulletproofs ($n > 1$), the resulting complexity is equal to $O(\log_2 N)$, which is asymptotically the same as the communication overhead of Bulletproofs inner-product argument. Therefore, this protocol is no better than Bulletproofs, at least in terms of communication complexity. Nevertheless, Protocol1 in Fig. 3 is a good basis for our sublogarithmic protocol presented in the next subsection.

4.2 Sublogarithmic Protocol from Aggregated Multi-Exponentiation Arguments

We build our main protocol for sublogarithmic transparent inner-product arguments on the basis of Protocol1 in Fig. 3. To this end, we develop an aggregation technique to prove multiple multi-exponentiation arguments at once, which

proves the following aggregated relation.

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

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 [36]. 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 random linear combination technique. Unfortunately, however, the relation $\mathcal{R}_{AggMEA}$ consists of two distinct type 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 3). 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 all 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.

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

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 our inner-product argument protocol using aAggMEA is given in Fig. 4.

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

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

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

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

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. 4. Protocol2: Sublogarithmic Inner-Product Argument

Theorem 2. *The inner-product argument in Fig. 4 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 2 is relegated to the supplementary material.

4.3 Efficiency

A main difference between Protocol1 and Protocol2 is the aggregating process for $\log_{2n} N$ multi-exponentiation arguments. Due to communication-efficient feature

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

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

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

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. 5. Reduction from aAggMEA to ProdMEA

of aAggMEA, the communication overhead is improved from $O((\log_2 n)\cdot\log_{2n} N)$ to $O(\log_2 n + \log_{2n} N)$. If we can set n to satisfy $O(\log_{2n} N) = O(\log_2 n)$, then the communication complexity becomes $O(\log_2 n + \log_{2n} N) = O(\sqrt{\log_2 N})$.

As for the computational overhead, compared to generalized Bulletproofs, only a run of aAggMEA protocol is imposed. Our proposal for the aAggMEA protocol is an extended variant of Bulletproofs (see the next section for the detail), so that its computational complexity is still linear in the length of witness vector that is $O(n^2 \log_{2n} N)$ though it requires expensive pairing operations. 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 Bulletproofs.[4]

5 Aggregating Multi-Exponentiation Argument

In this section, we propose an augmented aggregation of multi-exponentiation arguments aAggMEA for the relation in Eq. (6). Vectors in Eq. (6) are of dimension $2n(2n-1)$. For the sake of simplicity, we set the dimension of vectors N in this section and, by introducing dummy components, we can without loss of generality assume that N is a power of 2. The proposed protocol consists of two parts. First, the aAggMEA is reduced to a proof system, denoted by ProdMEA, for the following relation $\mathcal{R}_{PMEA}$ for a product of multi-exponentiation.

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

[4] Note that when the communication complexity is evaluated, we set $n = 2^{\sqrt{\log_2 N}}$ that is much smaller than $\sqrt{N} = 2^{\frac{1}{2}\log_2 N}$, and thus our estimation for computational cost makes sense.

The reduction is provided in Fig. 5 and its security property is given in the following theorem.

Theorem 3. *The* aAggMEA *protocol in Fig. 5 has perfect completeness and computational witness-extended-emulation if the* ProdMEA *protocol used in Fig. 5 has perfect completeness and computational witness-extended-emulation and the double pairing assumption holds.*

The proof of Theorem 3 is relegated to the supplementary material.

Next, 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 $k \in [m]$

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

$\mathcal{V}$ 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.

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

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

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 $x \xleftarrow{\$} \mathbb{Z}_p^*$ 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. 6. ProdMEA protocol

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 $\mathsf{ProdMEA}(\widehat{\boldsymbol{F}}_k, \widehat{z}_k, H_k, \widehat{P}; \widehat{\boldsymbol{v}}_k)$ together. The full description of ProdMEA is provided in Fig. 6. 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 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.

$$\begin{aligned} L^{x_i^2} P R^{x_i^{-2}} &= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \widehat{\boldsymbol{F}}_k)e(\widehat{\boldsymbol{v}}_k^{(i)\widehat{\boldsymbol{z}}_k}, H_k) \\ &= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \boldsymbol{F}_{k,1}^{x_i^{-1}} \circ \boldsymbol{F}_{k,-1}^{x_i})e(\widehat{\boldsymbol{v}}_k^{(i)\boldsymbol{z}_{k,1}x_i^{-1} + \boldsymbol{z}_{k,-1}x_i}, H_k) \\ &= \prod_{i\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i^{-1}}, \boldsymbol{F}_{k,1})\boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i}, \boldsymbol{F}_{k,-1})e(\widehat{\boldsymbol{v}}_k^{(i)\boldsymbol{z}_{k,1}x_i^{-1} + \boldsymbol{z}_{k,-1}x_i}, H_k) \end{aligned} \quad (6)$$

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. (6) and obtain the following theorem.

Theorem 4. *The* ProdMEA *protocol in Fig. 6 has perfect completeness and computational witness-extended-emulation under the double pairing assumption.*

The proof of Theorem 4 is relegated to the supplementary material.

5.1 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. 6 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 $\mathbb{G}_1$.

5.2 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 Bulletproofs 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.

6 Inner-Product Argument with Sublinear Verifier from Discrete Logarithms

In this section, we propose an inner-product argument with logarithmic communication and sublinear verifier computation, solely based on the discrete logarithm assumption.

6.1 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[5] 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.

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

6.2 General Discrete Logarithm Relation Assumption

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

Definition 7 (General Discrete Logarithm Relation Assumption). *Let* GDLRsp *be a sampler. We say that* GDLRsp *satisfies the general discrete logarithm relation (GDLR) assumption if all non-uniform polynomial-time adversaries* $\mathcal{A}$*, the following inequality holds.*

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

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

Definition 8. *For a fixed integer* N*, the sampler* GDLRsp_{Rand} *is defined as follows.*

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

Theorem 5. 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 Bulletproofs inner-product argument holds under the GDLR assumption; it uses only the fact that no adversary can find non-trivial relation, regardless of the distribution of generators $\boldsymbol{g}$. We restate the security theorem of Bulletproofs below.

Theorem 6 ([17]). *The inner-product argument of Bulletproofs (given in Fig. 8) 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* GDLRsp_{BM} *is defined as follows.*

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

Theorem 7. 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 $\mathbb{G}_t$.[6]

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

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

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

Type-2 adversary. This type of adversary can be used as an attacker breaking the general discrete logarithm relation assumption on $\mathbb{G}_2$ with a sampler GDLRsp_{Rand}. Theorem 5 guarantees that there is no type-2 adversary breaking the GDLR assumption with GDLRsp_{BM} under the DL assumption on $\mathbb{G}_2$.

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

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 $Dlog_g \hat{g}$. □

6.3 Another Generalization of Bulletproofs with Sublinear Verifier

In Bulletproofs, 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

[6] Note that the DL assumption on $\mathbb{G}_1$ implies the DL assumption on $\mathbb{G}_t$ by the MOV attack [42].

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 Bulletproofs. We can generalize Bulletproofs while keeping the soundness proof by using arbitrary sampler satisfying the general discrete logarithm relation assumption, instead of GDLRsp_{Rand} to create the CRS.

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

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m, \boldsymbol{H} \in \mathbb{G}_2^n, u, P \in \mathbb{G}_t;\ \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}) \\ :\ P = (\boldsymbol{g} \otimes \boldsymbol{H})^{\boldsymbol{a}} (\boldsymbol{h} \otimes \boldsymbol{H})^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G}_t \end{array} \right\} \tag{7}$$

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 Bulletproofs, $\boldsymbol{g} \otimes \boldsymbol{H}$ 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 7.

Sublinear Verification. If we set $m = n = \sqrt{N}$, the above modification can reduce the CRS size to be a square root of Bulletproofs. 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.[7] 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 Bulletproofs 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

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

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

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

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. 7. Protocol3: Another Generalization of Bulletproofs

and $\mathcal{P}$ additionally computes $\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 x + \boldsymbol{a}_{-1} x^{-1}$ and $\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 x^{-1} + \boldsymbol{b}_{-1} x \in \mathbb{Z}_p^{\widehat{m}}$. Then, $\widehat{P}$ is well computed since L and R are equivalent to those in Bulletproofs. In Bulletproofs, however, $\tilde{\boldsymbol{g}}_1^{x^{-1}} \circ \tilde{\boldsymbol{g}}_{-1}^{x}$ and $\tilde{\boldsymbol{h}}_1^{x} \circ \tilde{\boldsymbol{h}}_{-1}^{x^{-1}}$ should be computed as the new bases for the next round argument with witness $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$. Instead, in Protocol3, we use the equality $\widehat{\boldsymbol{g}} \otimes \boldsymbol{H} = \tilde{\boldsymbol{g}}_1^{x^{-1}} \circ \tilde{\boldsymbol{g}}_{-1}^{x}$ and $\widehat{\boldsymbol{h}} \otimes \boldsymbol{H} = \tilde{\boldsymbol{h}}_1^{x} \circ \tilde{\boldsymbol{h}}_{-1}^{x^{-1}}$ such that $\widehat{\boldsymbol{g}}$ and $\widehat{\boldsymbol{h}}$ are the bases for the next argument with $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$. Therefore, both $\mathcal{P}$ and $\mathcal{V}$ can run the protocol with $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, \boldsymbol{H}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$. If $m = 1$, the CRS is of the form $e(g_1, \boldsymbol{H})$ and $e(h_1, \boldsymbol{H})$, which is uniform in $\mathbb{G}_t$, so that we can directly run Bulletproofs over $\mathbb{G}_t$. We present the full description of our protocol, denoted by Protocol3, in Fig. 7. The number of rounds and the communication cost in Protocol3 are the same as those of Bulletproofs 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, Protocol3 is asymptotically the same as Bulletproofs inner-product argument since we can consider Protocol3 as Bulletproofs 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 8. *The argument presented in Fig. 7 for the relation* (7) *has perfect completeness and computational witness-extended-emulation under the general discrete logarithm assumption with the sampler* GDLRsp_{BM}.

Proof. Although the verification cost in Protocol3 is reduced compared with Bulletproofs, the prover's and the verifier's computation in Protocol3 is equivalent to that of Bulletproofs 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 9 except that the general discrete logarithm relation assumption is guaranteed by Theorem 7 instead of Theorem 5. □

6.4 Practical verification of Protocol 3

When it comes to asymptotic complexity, Protocol3 is definitely better than Bulletproofs. However, for practical performance, we consider the computation time of group operations which depends on choice of elliptic curve. Actually, both scheme use different elliptic curves. More specifically, Bulletproofs uses ed25519 curve for efficiency. However Protocol3 cannot apply ed25519 curve because this curve does not support pairing operation. For Protocol3, we consider pairing friendly elliptic curve like BLS12-381.

We consider a typical parameter setting $N = 2^{20}$ in 128-bits security and ed25519 curve for Bulletproofs, BLS12-381 curve for Protocol3. Bulletproofs requires 2×2^{20} group operations for verification. Protocol3 requires 2×2^{10} $\mathbb{G}_1$ operations and 2×2^{10} $\mathbb{G}_2$ operations for verification. The computation time of group operations $\mathbb{G}_1$ and $\mathbb{G}_2$ in BLS12-381 curve are roughly $5\times$ and $10\times$ slower than that of group operation in ed25519 curve respectively [44]. Then we can conclude that the verification time of Protocol3 is $67\times$ faster than that of Bulletproofs.

7 Extensions

7.1 Zero-Knowledge Argument for Arithmetic Circuits

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. The inner-product argument is an important ingredient for the SHVZK argument for arithmetic circuits [15, 17]. We can apply this well-known approach with our inner-product arguments. In the supplementary material, we provide the formal definition of SHVZK and present how to use our inner-product arguments for the SHVZK argument for arithmetic circuits.

7.2 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

Scheme	CRS size	Commit size	Opening proof size	$\mathcal{P}$'s computation		$\mathcal{V}$'s computation	Assump.
				Commit	Open		
Groth [32]	$O(\sqrt[3]{N})\mathbb{G}_2$	$O(\sqrt[3]{N})\mathbb{G}_t$	$O(\sqrt[3]{N})\mathbb{G}_1$	$O(N)\tau_1$	$O(\sqrt[2]{\sqrt[3]{N}})\tau_1$	$O(\sqrt[3]{N})\mathsf{p}$	DPair
BulletP. [17]	$O(N)\mathbb{G}_1$	$O(1)\mathbb{G}_1$	$O(\log_2 N)\mathbb{G}_1$	$O(N)\tau_1$			DL
Hyrax [47]	$O(\sqrt{N})\mathbb{G}_1$		$O(\log_2 N)\mathbb{G}_1$	$O(N)\tau_1$	$O(\sqrt{N})\tau_1$		DL
BFS [18]	$O(N)\mathbb{G}_U$	$O(1)\mathbb{G}_U$	$O(\log_2 N)\mathbb{G}_U$	$O(N)\mathsf{u}$	$O(N\log_2 N)\mathsf{u}$	$O(\log_2 N)\mathsf{u}$	UO
Virgo [50]	$O(1)$	$O(1)\mathbb{H}$	$O((\log_2 N)^2)\mathbb{H}$	$O(N\log_2 N)\mathsf{h}$		$O((\log_2 N)^2)\mathsf{h}$	CR hash
BMMV [19]	$O(\sqrt{N})\mathbb{G}_2$	$O(1)\mathbb{G}_t$	$O(\log_2 N)\mathbb{G}_t$	$O(N)\tau_1$	$O(\sqrt{N})\mathsf{p}$	$O(\sqrt{N})\tau_2$	DPair
Protocol2	$O(N)\mathbb{G}_1$	$O(1)\mathbb{G}_1$	$O(\sqrt{\log_2 N})\mathbb{G}_t$	$O(N)\tau_1$	$O(N2^{\sqrt{\log N}})\tau_1$	$O(N)\tau_1$	DL&DPair
Protocol3	$O(\sqrt{N})\mathbb{G}_2$	$O(1)\mathbb{G}_t$	$O(\log_2 N)\mathbb{G}_t$	$O(N)\tau_t$	$O(N)\tau_t$	$O(\sqrt{N})\tau_2$	DL

$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t)$: bilinear groups, $\mathbb{G}_U$: group of unknown order, $\mathbb{H}$: hash function,
τ_i: group operations in $\mathbb{G}_i$, u: group operation in $\mathbb{G}_U$, p: pairing operation, 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

by a verifier) and convinces a verifier of correctness of $f(x)$. We provide the formal definitions of commitment scheme and polynomial commitment scheme in the supplementary material.

The inner-product argument with the Pedersen commitment scheme such as Bulletproofs can be naturally considered a transparent polynomial commitment scheme and is already used in many prior works (e.g., [47, 16, 19]). 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 $(1, x, x^2, \ldots, x^{N-1})$.

Bulletproofs inner-product argument convinces about the relation in Eq. (1). Our main protocol Protocol2 is a proof system for the relation in Eq. (4), which is equivalent to the relation in Eq. (1) except for the CRS. Therefore, Protocol2 can be used as a transparent polynomial commitment scheme in exactly the same way as Bulletproofs. In Table 2, we present a comparison for asymptotic complexities of polynomial commitment schemes. Note that although both $\mathcal{P}$ and $\mathcal{V}$ perform bilinear operations in Protocol2, those are sublinear in the dimension N of the witness vector, so that group operations with $O(N)$ complexity dominate computational complexities.

8 Discussion on Best of Protocol2 and Protocol3

There are several directions to improve our results. One of improvements is to combine ideas of two protocols Protocol2 and Protocol3. It seems difficult to get

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

We share an idea to circumvent the above difficulty. In Protocol2, $\mathcal{P}$ commits to a vector $\boldsymbol{v}$ whose components form $v_{i,j} = \boldsymbol{g}_i^{\boldsymbol{a}_j} \boldsymbol{h}_j^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \in \mathbb{G}_1$. Similarly, we can generalize Protocol3 so that $\mathcal{P}$ commits to $v_{i,j}$ whose components consist of $(\boldsymbol{g}_i \otimes \boldsymbol{H})^{\boldsymbol{a}_j} (\boldsymbol{h}_j \otimes \boldsymbol{H})^{\boldsymbol{b}_i} u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \in \mathbb{G}_t$. Since $v_{i,j}$ is in the target group, we cannot again commit to $v_{i,j}$ by using homomorphic commitment schemes. Nevertheless, we observe that $v_{i,j}$ can be represented as $\boldsymbol{E}(\boldsymbol{\alpha}_{i,j}, \boldsymbol{H})$, where $\boldsymbol{\alpha}_{i,j} \in \mathbb{G}^n$ depends on $\boldsymbol{a}_j$ and $\boldsymbol{b}_i$. Since $\boldsymbol{H}$ is a public parameter, the prover may commit to $\boldsymbol{\alpha}_{i,j}$'s instead of $v_{i,j}$'s and then prove suitable relations. We hope this observation is helpful for tackling the difficulty for best of both protocols.

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. Association for Computing Machinery, 2017.
4. M. Backes, A. Datta, and A. Kate. Asynchronous computational vss with reduced communication complexity. In *CT-RSA 2013*, pages 259–276, 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. In *https://eprint.iacr.org/2018/046*, page 46, 2018.
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*, 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*, pages 103–128. Springer, 2019.
10. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, 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 *ITCS12*, LNCS, 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 *TCC 2013*, LNCS, pages 111–120. Springer, 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*, LNCS, pages 327–357. Springer, 2016.
16. S. Bowe, J. Grigg, and D. Hopwood. Recursive proof composition without a trusted setup. In *https://eprint.iacr.org/2019/1021*, page 1021, 2019.
17. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy 2018*, pages 315–334. IEEE, 2018.
18. B. Bünz, B. Fisch, and A. Szepieniec. Transparent snarks from dark compilers. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 677–706. Springer, 2020.
19. B. Bünz, M. Maller, P. Mishra, and N. Vesely. Proofs for inner pairing products and applications. *IACR Cryptology ePrint Archive*, 2019:1177, 2019.
20. J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *ASIACRYPT '15*, pages 262–288, 2015.
21. J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, 2010.
22. 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.
23. A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT 2020 Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, 2020.
24. V. Daza, C. Ràfols, and A. Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In *PKC 2020*, volume 12110 of *LNCS*, pages 527–557. Springer, 2020.
25. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.
26. D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.
27. 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.
28. 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.
29. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT 2013*, pages 626–645. Springer, 2013.
30. J. Groth. Linear algebra with sub-linear zero-knowledge arguments. In *CRYPTO 2009*, LNCS, pages 192–208. Springer, 2009.

31. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, 2010.
32. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 431–448. Springer, 2011.
33. J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, pages 305–326. Springer, 2016.
34. 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. Springer, 2015.
35. 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.
36. 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.
37. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT '10*, LNCS, pages 177–194. Springer, 2010.
38. R. W. F. Lai, G. Malavolta, and V. Ronge. Succinct arguments for bilinear group arithmetic: Practical structure-preserving cryptography. In *ACM CCS '19*, pages 2057–2074, 2019.
39. libsnark. https://github.com/scipr-lab/libsnark, 2017.
40. 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.
41. 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.
42. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inf. Theory*, 39(5):1639–1646, 1993.
43. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.
44. M. Scott. On the deployment of curve based cryptography for the internet of things. *IACR Cryptology ePrint Archive*, page 514, 2020.
45. J. H. Seo. Round-efficient sub-linear zero-knowledge arguments for linear algebra. In *PKC 2011*, LNCS, pages 387–402. Springer, 2011.
46. S. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *CRYPTO 2020*, volume 12172 of *LNCS*, pages 704–737. Springer, 2020.
47. 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.
48. T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO (3) 2019*, volume 11694 of *LNCS*, pages 733–764. Springer, 2019.
49. J. Xu, A. Yang, J. Zhou, and D. S. Wong. Lightweight delegatable proofs of storage. In *ESORICS '16*, pages 324–343, 2016.
50. J. Zhang, T. Xie, Y. Zhang, and D. Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, pages 859–876. IEEE, 2019.

Supplementary Materials

A Bulletproofs [17]

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

We review Bulletproofs inner-product argument, which is denoted by $\mathsf{BP}_{\mathsf{IP}}$ and given in Fig. 8, for the relation in Eq. (1).

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

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 $\widehat{N} = \frac{N}{2}$ 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

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

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

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

Additionally, $\mathcal{P}$ computes

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

Step 4: Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP}_{\mathsf{IP}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, u, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}})$.

Fig. 8. Bulletproofs (Inner-Product Argument)

The main idea of Bulletproofs 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 $x \xleftarrow{\$} \mathbb{Z}_p^*$ and returns it to $\mathcal{P}$. Next, both $\mathcal{P}$ and $\mathcal{V}$ compute a common input for the next step argument

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

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 Bulletproofs is provided in Fig. 8.

Theorem 9 ([17]). *The inner-product argument of Bulletproofs (given in Fig. 8 of the supplementary material) 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 $O(\log_2 N)$ rounds. The verifier computes $\widehat{g}$ and $\widehat{h}$ that require $O(N)$ exponentiations 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. 9.

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

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

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

$$\boldsymbol{s}_L, \boldsymbol{s}_R \xleftarrow{\$} \mathbb{Z}_p^N,\ S = h^{\rho} \boldsymbol{g}^{\boldsymbol{s}_L} \boldsymbol{h}^{\boldsymbol{s}_R} \in \mathbb{G}$$

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 $\mathcal{V}$ compute

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

Additionally, $\mathcal{P}$ computes

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

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

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

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: Both $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{BP}_{\mathsf{IP}}(\boldsymbol{g}, \boldsymbol{h}', g^w, P \cdot h^{-\mu} \cdot g^{w \cdot \hat{t}}; \boldsymbol{l}, \boldsymbol{r})$.

Fig. 9. Bulletproofs (Arithmetic Circuit Argument)

B Proof of Theorem 1

B.1 General Forking Lemma

Theorem 10 (General Forking Lemma [15]). *Let $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ be a $(2k+1)$-move, public coin interactive protocol with μ 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 μ 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 χ 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 λ. Then, $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation.

B.2 Proof of Theorem 1

Proof. (completeness) For the recursive step, Eq (2) guarantees the completeness of the protocol. For $N = 1$, it is straightforward since the prover provides a witness a and 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 χ whose goal is to extract the witness by using a polynomially bounded tree of accepting transcripts. If then, similar to Bulletproofs, we can apply the general forking lemma.

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 $\boldsymbol{a}$ and $\boldsymbol{b}$ under the discrete logarithm relation assumption whose instance is the CRS $\boldsymbol{g}$, $\boldsymbol{h}$, and 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 $k \in [12n-5]$

$$
\begin{aligned}
P \prod_{s \in J_n} \Big(\prod_{\substack{i,j \in I_n \\ j-i=s}} v_{i,j} \Big)^{x_k^s} = P \prod_{\substack{i,j \in I_n \\ i \neq j}} v_{i,j}^{x_k^{j-i}} &= \widehat{P} \\
&= \big(\circ_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i}} \big)^{\widehat{\boldsymbol{a}}_k} \big(\circ_{j \in I_n} \boldsymbol{h}_j^{x_k^j} \big)^{\widehat{\boldsymbol{b}}_k} u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \\
&= \Big(\prod_{i \in I_n} \boldsymbol{g}_i^{x_k^{-i} \widehat{\boldsymbol{a}}_k} \boldsymbol{h}_i^{x_k^i \widehat{\boldsymbol{b}}_k} \Big) u^{\langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle} \qquad (8)
\end{aligned}
$$

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.

$$M = \begin{bmatrix} x_1^{-4n+2} & x_1^{-4n+4} & \cdots & 1 & \cdots & x_1^{4n-4} & x_1^{4n-2} \\ x_2^{-4n+2} & x_2^{-4n+4} & \cdots & 1 & \cdots & x_2^{4n-4} & x_2^{4n-2} \\ \vdots & \vdots & & \cdots 1 \cdots & & \vdots & \vdots \\ x_{4n-1}^{-4n+2} & x_{4n-1}^{-4n+4} & \cdots & 1 & \cdots & x_{4n-1}^{4n-4} & x_{4n-1}^{4n-2} \end{bmatrix}$$

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

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. (8). There are $4n-1$ distinct powers of x_k^2 in the left-hand side in Eq. (8). 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. (8), 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

$$P = \big(\prod_{r\in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{P,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{P,r}}\big) u^{c_P} \in \mathbb{G}, \tag{9}$$

$$\big(\prod_{\substack{i,j\in I_n \\ j-i=s}} v_{i,j}\big) = \big(\prod_{r\in I_n} \boldsymbol{g}_r^{\boldsymbol{a}_{s,r}} \boldsymbol{h}_r^{\boldsymbol{b}_{s,r}}\big) u^{c_s} \in \mathbb{G}. \tag{10}$$

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. (9) and Eq. (10) into Eq. (8), 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 $r \in I_n$

$$\boxed{\boldsymbol{g}_r \text{ exponents}} : \boldsymbol{a}_{P,r} + \textstyle\sum_{s\in J_n} \boldsymbol{a}_{s,r} x_k^s = \widehat{\boldsymbol{a}}_k x_k^{-r} \tag{11}$$

$$\boxed{\boldsymbol{h}_r \text{ exponents}} : \boldsymbol{b}_{P,r} + \textstyle\sum_{s\in J_n} \boldsymbol{b}_{s,r} x_k^s = \widehat{\boldsymbol{b}}_k x_k^{r} \tag{12}$$

$$\boxed{u \text{ exponents}} : \quad c_P + \textstyle\sum_{s\in J_n} c_s x_k^s \quad = \langle \widehat{\boldsymbol{a}}_k, \widehat{\boldsymbol{b}}_k \rangle. \tag{13}$$

If we find exponents satisfying Eq. (8), Eq. (9) and Eq. (10), but not one of the above Eq. (11), Eq. (12) and Eq. (13), 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. (13) to find the relation $c_P = \sum_{r\in I_n} \langle \boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r} \rangle$. From Eq. (11) and Eq. (12), we can remove $\widehat{\boldsymbol{a}}_k$ and $\widehat{\boldsymbol{b}}_k$ and deduce that for each $k \in [12n-5]$ and any $r, r' \in I_n$,

$$\boldsymbol{a}_{P,r} x_k^r + \sum_{s\in J_n} \boldsymbol{a}_{s,r} x_k^{s+r} - \boldsymbol{a}_{P,r'} x_k^{r'} - \sum_{s\in J_n} \boldsymbol{a}_{s,r'} x_k^{s+r'} = 0 \tag{14}$$

$$\boldsymbol{b}_{P,r} x_k^{-r} + \sum_{s\in J_n} \boldsymbol{b}_{s,r} x_k^{s-r} - \boldsymbol{b}_{P,r'} x_k^{-r'} - \sum_{s\in J_n} \boldsymbol{b}_{s,r'} x_k^{s-r'} = 0. \tag{15}$$

In both Eq. (14) and Eq. (15), 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

$$\boldsymbol{a}_{P,r} X^r + \sum_{s\in J_n} \boldsymbol{a}_{s,r} X^{s+r} \text{ is the same polynomial for all } r \in I_n. \tag{16}$$

$$\boldsymbol{b}_{P,r} X^{-r} + \sum_{s\in J_n} \boldsymbol{b}_{s,r} X^{s-r} \text{ is the same polynomial for all } r \in I_n. \tag{17}$$

The only way to hold the above two equations is that

1. The polynomial in Eq. (16) is $\sum_{s\in I_n} \boldsymbol{a}_{P,s} X^s$, regardless of r.
2. The polynomial in Eq. (17) 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. (16) and Eq. (17), 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. (11) and Eq. (12), 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. (13) and obtain that for $k \in [12n-5]$

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

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 λ. Then, by the general forking lemma, we conclude that the argument has computational witness-extended emulation. □

C Proof of Theorem 2

Proof. (Completeness) For $N > 1$, the protocol resembles the generalization of the bulletproofs in Fig. 2. 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 arguments protocol. 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 $\boldsymbol{v}_{k,j} = \mathbf{1}_{\mathbb{G}_1}$ 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. (6). 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 χ that extracts a witness from a suitable tree of accepting transcripts in probabilistic polynomial time.

We begin with $(\underbrace{12n-5, \ldots, 12n-5}_{\log_{2n} N}, 2m, \underbrace{7, \ldots, 7}_{\log_2(2n(2n-1)\log_{2n} N)})$-tree of accepting transcriptions. Note that the number of all challenges in the tree is $(12n-5)^{\log_{2n} N} \cdot 2m \cdot 7^{\log_2(2n(2n-1)\log_{2n} N)}$ that is bounded above by a polynomial in N and λ, 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 $(1+\log_2(2n(2n-1)\log_{2n} N)$-depth is $(2m, 7, \ldots, 7)$-tree of accepting transcripts for aAggMEA. aAggMEA has the witness-extended emulation under the co-sCDH assumption by Theorem 3 and Theorem 4. Using the $(2m, 7, \ldots, 7)$-tree of accepting transcripts, the extractor can extract $\boldsymbol{v}_{k,j}$ satisfying

$$\bigwedge_{k \in [m]} \Big(u_k = \prod_{j \in [m]} \boldsymbol{E}(\boldsymbol{v}_{k,j}, \boldsymbol{F}_j) \ \wedge \ v_k = \boldsymbol{v}_{k,k}^{\boldsymbol{x}_k} \ \wedge \ (\boldsymbol{v}_{k,j}^{\boldsymbol{x}_j} = 1_{\mathbb{G}_1} \text{ for } j \neq k) \Big), \quad (18)$$

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 $\mathbf{1}_{\mathbb{G}_1}$ unlike the original protocol. However, what we need from Eq. (18) 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 1; the format of the upper position subtree of accepting transcripts is exactly the same as that required in the proof of generalized Bulletproofs in Theorem 1. 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 Bulletproofs protocol, so that we can employ the extractor in the proof of Theorem 1 under the discrete logarithm relation assumption in $\mathbb{G}_1$.

As we aforementioned, we use $(\underbrace{12n-5, \ldots, 12n-5}_{\log_{2n} N}, 2m, \underbrace{7, \ldots, 7}_{\log_2(2n(2n-1)\log_{2n} N)})$-tree of accepting transcriptions, which has leaves of polynomially bounded size in N and λ. Therefore, we can apply the general forking lemma. □

D Proof of Theorem 3

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.*

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

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

Theorem 11. *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 $(r_1, \ldots, r_q) \xleftarrow{\$} \mathbb{Z}_p^q$, sets $\boldsymbol{F} = ((G^a)^{r_1}, G^{r_2}, \ldots, G^{r_q})$, 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 $\boldsymbol{g} = (g_1, \ldots, g_q)$ 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.

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

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 3 about the protocol for the following relation.

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

(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

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

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. (6). 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 χ 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

$$
\begin{aligned}
\prod_{k\in[m]} \Big(P_k^{y_i^{k-1}} e(q_k, H_k)^{y_i^{m+k-1}}\Big) &= \tilde{P}_i \\
&= \prod_{j\in[m]} \boldsymbol{E}((\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{j-1}}, \boldsymbol{F}_j) e((\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{m+j-1}\boldsymbol{z}_j}, H_j). \quad (19)
\end{aligned}
$$

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. (19), 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

$$P_k = \prod_{i=1}^{m} \tilde{P}_i^{a_{k,i}} = \prod_{j \in [m]} \boldsymbol{E}(\boldsymbol{w}_{k,j}, \boldsymbol{F}_j) e(\nu_{k,j}, H_j). \tag{20}$$

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. (19), we have

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

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

$$\boxed{\boldsymbol{F}_j \text{ correspondence}}: \quad \circ_{k \in [m]} \boldsymbol{w}_{k,j}^{y_i^{k-1}} = (\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{j-1}} \tag{21}$$

$$\boxed{H_j \text{ correspondence}}: \big(\textstyle\prod_{k \in [m]} \nu_{k,j}^{y_i^{k-1}} \big) \cdot q_j^{y_i^{m+j-1}} = (\tilde{\boldsymbol{v}}_j^{(i)})^{y_i^{m+j-1} \boldsymbol{z}_j} \tag{22}$$

Combining Eq. (21) and Eq. (22), we can remove $\tilde{\boldsymbol{v}}_{i,j}$'s and obtain the followings.

$$\big(\prod_{k \in [m]} \nu_{k,j}^{y_i^{k-1}} \big) \cdot q_j^{y_i^{m+j-1}} = \prod_{k \in [m]} \boldsymbol{w}_{k,j}^{y_i^{m+k-1} \boldsymbol{z}_j} \tag{23}$$

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. (23) is that

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

Thus, putting the above result into Eq. (20), the extractor eventually has $\boldsymbol{w}_{k,j}$ for $k \in [m]$ satisfying

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

□

E Proof of Theorem 4

Proof. (Completeness) If $N = 1$, it is straightforward to check the completeness of the scheme in Figure 6. 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

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

Therefore, the completeness is satisfied.

(Witness-Extended Emulation) The double pairing assumption implies the q-double pairing assumption in $\mathbb{G}_2$ by Theorem 11, 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 χ 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 $i \in [7]$

$$\begin{aligned}
& L^{x_i^2} P R^{x_i^{-2}} \\
&= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \widehat{\boldsymbol{F}}_k) e(\widehat{\boldsymbol{v}}_k^{(i)\widehat{\boldsymbol{z}}_k}, H_k) \\
&= \prod_{k\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)}, \boldsymbol{F}_{k,1}^{x_i^{-1}} \circ \boldsymbol{F}_{k,-1}^{x_i}) e(\widehat{\boldsymbol{v}}_k^{(i)(\boldsymbol{z}_{k,1} x_i^{-1} + \boldsymbol{z}_{k,-1} x_i)}, H_k) \\
&= \prod_{i\in[m]} \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i^{-1}}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\widehat{\boldsymbol{v}}_k^{(i)x_i}, \boldsymbol{F}_{k,-1}) e(\widehat{\boldsymbol{v}}_k^{(i)(\boldsymbol{z}_{k,1} x_i^{-1} + \boldsymbol{z}_{k,-1} x_i)}, H_k) \quad (24)
\end{aligned}$$

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. (24), 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

$$P = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{P,k,1}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\boldsymbol{g}_{P,k,-1}, \boldsymbol{F}_{k,-1}) e(g_{P,k,H}, H_K) \quad (25)$$

$$L = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{L,k,1}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\boldsymbol{g}_{L,k,-1}, \boldsymbol{F}_{k,-1}) e(g_{L,k,H}, H_K) \quad (26)$$

$$R = \prod_{k\in[m]} \boldsymbol{E}(\boldsymbol{g}_{R,k,1}, \boldsymbol{F}_{k,1}) \boldsymbol{E}(\boldsymbol{g}_{R,k,-1}, \boldsymbol{F}_{k,-1}) e(g_{R,k,H}, H_K). \quad (27)$$

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. (25), Eq. (26) and Eq. (27) into Eq. (24), 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 $k \in [m]$

$$\boxed{\boldsymbol{F}_{k,1} \text{ correspondence}}: \quad \boldsymbol{g}_{L,k,1}^{x_i^2} \circ \boldsymbol{g}_{P,k,1} \circ \boldsymbol{g}_{R,k,1}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)x_i^{-1}} \tag{28}$$

$$\boxed{\boldsymbol{F}_{k,-1} \text{ correspondence}}: \boldsymbol{g}_{L,k,-1}^{x_i^2} \circ \boldsymbol{g}_{P,k,-1} \circ \boldsymbol{g}_{R,k,-1}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)x_i} \tag{29}$$

$$\boxed{H_k \text{ correspondence}}: \quad g_{L,k,H}^{x_i^2} \cdot g_{P,k,H} \cdot g_{R,k,H}^{x_i^{-2}} = \widehat{\boldsymbol{v}}_k^{(i)\boldsymbol{z}_{k,1}x_i^{-1}+\boldsymbol{z}_{k,-1}x_i} \tag{30}$$

If we find exponents satisfying Eq. (24), Eq. (25), Eq. (26) and Eq. (27), but not one of the above Eq. (28), Eq. (29) and Eq. (30), 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. (30) to find the desired relation $g_{P,k,H} = \boldsymbol{g}_{P,k}^{\boldsymbol{z}_k}$. From Eq. (28) and Eq. (29), we can deduce that for each $i \in [7]$ and $k \in [m]$,

$$\boldsymbol{g}_{L,k,1}^{x_i^3} \circ \boldsymbol{g}_{P,k,1}^{x_i} \circ \boldsymbol{g}_{R,k,1}^{x_i^{-1}} = \boldsymbol{g}_{L,k,-1}^{x_i} \circ \boldsymbol{g}_{P,k,-1}^{x_i^{-1}} \circ \boldsymbol{g}_{R,k,-1}^{x_i^{-3}} \in \mathbb{G}_1^N. \tag{31}$$

Eq. (31) 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 $k \in [m]$

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

Putting the above result into Eq. (28), 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. (30) 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 λ. Then, by the general forking lemma, we conclude that the argument has computational witness-extended emulation. □

F Extensions

F.1 Special Honest Verifier Zero-Knowledge

Definition 11. *(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 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 ρ 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 $\mathcal{S}$.

F.2 Zero-Knowledge Argument for Arithmetic Circuits

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. [17] 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, 17] is given as follows.

$$\left\{\begin{array}{l}\Big(\boldsymbol{g},\boldsymbol{h}\in\mathbb{G}^N,\boldsymbol{V}\in\mathbb{G}^M,g,h\in\mathbb{G},\boldsymbol{W}_L,\boldsymbol{W}_R,\boldsymbol{W}_O\in\mathbb{Z}_p^{Q\times N},\boldsymbol{W}_V\in\mathbb{Z}_p^{Q\times M},\\ \quad\boldsymbol{c}\in\mathbb{Z}_p^Q;\ \boldsymbol{a}_L,\boldsymbol{a}_R,\boldsymbol{a}_O\in\mathbb{Z}_p^N,\boldsymbol{v},\boldsymbol{\gamma}\in\mathbb{Z}_p^M\\ :V_j=g^{v_j}h^{\gamma_j}\forall j\in[1,m]\ \wedge\ \boldsymbol{a}_L\circ\boldsymbol{a}_R=\boldsymbol{a}_O\\ \quad\wedge\ \boldsymbol{W}_L\boldsymbol{a}_L^\top+\boldsymbol{W}_R\boldsymbol{a}_R^\top+\boldsymbol{W}_O\boldsymbol{a}_O^\top=\boldsymbol{W}_V\boldsymbol{v}^\top+\boldsymbol{c}^\top\end{array}\Big)\right\}\tag{32}$$

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

Bulletproofs for arithmetic circuits is a zero-knowledge argument for Eq. (32), and in particular, Bünz et al. [17] proved the following theorem.

Theorem 12 (Theorem 5 in [17]). *There exists an efficient arithmetic circuit protocol for the relation in Eq.* (32) *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 Bulletproofs arithmetic circuit protocol in the supplementary material (Fig. 9) for the self-containedness. Bulletproofs arithmetic circuit argument is indeed a reduction to an inner-product argument. In the reduction phase, 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 Bulletproofs arithmetic circuit argument by replacing Bulletproofs inner-product argument. In fact, the language in Eq. (4) differs from the relation (1) on the CRS only and the relation is equivalent. Therefore, we can still employ Theorem 12 taking Protocol2 as the underlying inner-product argument. Similarly, we can employ Theorem 12 with taking Protocol3 as the underlying inner-product argument.

F.3 Commitment Schemes

Definition 12 (Binding 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 λ 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 λ*

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

Definition 13 (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.*