TENET : Sublogarithmic Proof, Sublinear Verifier Inner Product Argument without a Trusted Setup

Hyeonbum 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 a new inner product argument (IPA), called TENET, which features sublogarithmic proof size and sublinear verifier without a trusted setup. IPA is a core primitive for various advanced proof systems including range proofs, circuit satisfiability, and polynomial commitment, particularly where a trusted setup is hard to apply. At ASIACRYPT 2022, Kim, Lee, and Seo showed that pairings can be utilized to exceed the complexity barrier of the previous discrete logarithm-based IPA without a trusted setup. More precisely, they proposed two pairing-based IPAs, one with sublogarithmic proof size and the other one with sublinear verifier cost, but they left achieving both complexities simultaneously as an open problem. We investigate the obstacles for this open problem and then provide our solution TENET, which achieves both sublogarithmic proof size and sublinear verifier. We prove the soundness of TENET under the discrete logarithm assumption and double pairing assumption.

Keywords: Inner product argument, Transparent setup, Zero knowledge proof

1 Introduction

An argument system is a protocol between two parties, the prover and the verifier, such that the prover can convince the verifier that a statement is true [15]. One of the most useful argument systems is an inner product argument (IPA), an argument for the inner product relation of two committed vectors. Bootle, Cerulli, Chaidos, Groth, and Petit [5] proposed the first IPA with logarithmic proof size under the discrete logarithm assumption and then used it to construct a zero knowledge (ZK) argument for circuit satisfiability. Bünz, Bootle, Boneh, Poelstra, Wuille, and Maxwell [7] proposed an improved IPA, called Bulletproofs, and showed its efficacy by applying to prove range and arithmetic circuit relations.

Kim, Lee, and Seo [18] proposed two pairing-based IPAs without a trusted setup, Protocol2 and Protocol3. Protocol2 and Protocol3 provide sublogarithmic

** Corresponding Author

proof size and sublinear verifier costs, respectively. However, they do not achieve both complexity simultaneously and leave it as an open problem.

We focus on generalization of pairing-based IPA without a trusted setup. More concretely, we aim to combine two arguments, Protocol2 and Protocol3, to achieve sublogarithmic proof size and sublinear verifier simultaneously.

1.1 Our Results

Generalization of the Inner Product Argument Without a Trusted Setup. We propose generalization of pairing-based IPA without a trusted setup. Specifically, we focus on a combination of two ideas of pairing-based IPAs, Protocol2 and Protocol3. One of the core ideas of Protocol2 is *commit-and-prove* for relation of group elements, which are messages of the prover. In this phase, pairing-based group commitment scheme [1] is used to commit prover's messages. Meanwhile, the prover's message in Protocol3 belongs to the target group. To combine two schemes, the prover should make commitments of his messages that are not put to pairing-based group commitment schemes.
Structure of Prover's Message : The prover's message consists of multiple target groups of the form $v = \prod_{i,j} e(g[i], H[j])^{a[i,j]}$, where $\boldsymbol{H}$ is public. From the bilinear structure, the message construction can be viewed as $v = \prod_i e(\boldsymbol{g}^{\boldsymbol{a}[j]}, H[j])$. Owing to this structure, we substitute the prover's message v with the source group elements $\boldsymbol{g}^{\boldsymbol{a}[j]}$. After substitution, we apply pairing-based group commitments to $\boldsymbol{g}^{\boldsymbol{a}[j]}$ for a *commit-and-prove* approach.

Optimization Technique for Sublogarithmic Size IPA. We introduce optimization techniques for sublogarithmic size IPA, Protocol2 and TENET. The optimization affects CRS size, proof size and verifier cost. More concretely, the prover generates several group vectors of the form $\boldsymbol{v} \in \mathbb{G}_1^{2d(2d-1)}$. Then, the prover sends commitments to each group vector with knowledge proof of them. Both the proof size and verifier cost for the proof rely on the size of group vectors, $O(d^2)$. However, we show that only knowledge proof for $O(d)$ length vectors is sufficient for soundness.

TENET : Sublogarithmic Proof Size and Sublinear Verifier Under DL and DPair. After the generalization and optimization, we analyze the arguments and then find appropriate parameters to achieve both sublogarithm proof size and sublinear verifier cost. Certainly, we prove security of TENET with perfect completeness and computational witness extended emulation under discrete logarithm (DL) and double pairing (DPair) assumption. From our IPA TENET, one can construct sublogarithm proof size and sublinear verifiable polynomial commitment schemes, which can be used on Sonic [21], Plonk [14], and Marlin [10] to get efficiency without a trusted setup.

	Comm.	$\mathcal{P}$'s cost	$\mathcal{V}$'s cost	Assumption	Trusted Setup
Bootle et al.[5]	$O(\log N)$	$O(N)$	$O(N)$	DL	No
Bulletproofs[7]	$O(\log N)$	$O(N)$	$O(N)$	DL	No
Chung et al.[11]	$O(\log N)$	$O(N)$	$O(N)$	DL	No
Daza et al.[12]	$O(\log N)$	$O(N)$	$O(\log N)$	DL, DPair	Yes
Zhou et al.[23]	$O(\log N)$	$O(N)$	$O(\log N)$	DL, DPair	Yes
Protocol2[18]	$O(\sqrt{\log N})$	$O(N2^{\sqrt{\log N}})$	$O(N)$	DL, DPair	No
Protocol3[18]	$O(\log N)$	$O(N)$	$O(\sqrt{N})$	DL	No
Protocol4[18]	$O(\log N)$	$O(N)$	$O(\sqrt{N}\log N)$	DL	No
TENET(Ours)	$O(\sqrt{\log N})$	$O(N2^{\sqrt{\log N}})$	$O(N/2^{\sqrt{\log N}})$	DL, DPair	No

Table 1. Comparison Table of Inner Product Arguments from Discrete Logarithms

1.2 Related Work

Inner Product Argument. Inner product arguments are used as building blocks for range proof and zero knowledge proof, which can be used in numerous applications such as verifiable computation, confidential transactions, and decentralized identification.

There are many variants of Bulletproofs [3, 8, 9, 11, 12, 18, 23], which are based on inner product reduction. In [11], the zero knowledge weighted IPA was proposed and used to construct a variant of Bulletproofs, called Bulletproofs+, with a shorter proof size. In [12, 23], the structured common reference string and bilinear maps are used to achieve both logarithmic communication and verification. In [18], three IPAs without a trusted setup are proposed: Protocol2 with sublogarithmic proof size, Protocol3 with sublinear verifier, and Protocol4 with sublinear verifier. The difference between Protocol3 and Protocol4 is reliance on pairing-based elliptic curves. We provide a comparison among various IPAs of Bulletproofs in Table 1.

Zero Knowledge Argument and Polynomial Commitment Schemes Bootle et al. [5] first proposed the logarithmic size ZK argument for circuit satisfiability without a trusted setup. To construct the ZK argument, they applied their IPA, which provides a logarithm proof size. The core idea to achieve logarithm size is to construct an efficient reduction protocol that can run recursively. This idea is widely used to construct ZK arguments without a trusted setup [7, 22, 6, 8, 20].

Kate, Zaverucha, and Goldberg [17] first introduced the polynomial commitment scheme (PCS), which allows the prover to claim the polynomial evaluation at a point without opening the polynomial itself. In addition, they constructed a constant size PCS, called KZG PCS. KZG PCS is the core building block of

ZK arguments with a constant proof size [16, 21, 10, 14]. However, the arguments require a trusted setup.

Bünz, Fisch, and Szepieniec [8] proposed PCS without a trusted setup, called DARK, and they introduced the paradigm of construction ZK argument, combining a polynomial interactive oracle proof system [4] and PCS. From their paradigm, they constructed logarithmic proof size and verifiable ZK argument without a trusted setup by replacing KZG PCS with DARK. In their paradigm, the complexity and cryptographic properties of ZK arguments are inherited from those of PCS.

IPA can be converted to a PCS scheme because polynomial evaluations are a kind of inner product relation; thus, some recent works [9, 20, 2] have focused on efficient IPA to construct efficient PCS and ZK arguments.

2 Preliminary

2.1 Definitions

We first define notations used in the paper. Some notations are inspired by [18]. $[m]$ denotes a set of integers from 1 to m, $\{1, \cdots, m\}$. Specifically, we define two index sets I_d and J_d. I_d is the set of continuous odd integers from $-2d+1$ to $2d-1$, $I_d = \{\pm 1, \pm 3, \cdots, \pm(2d-1)\}$. And J_d is the set of continuous even integers excluding 0 from $-4d+2$ to $4d-2$, $J_d = \{\pm 2, \pm 4, \cdots, \pm(4d-2)\}$. Note that J_d consists of all possible differences between two distinct elements of I_d. We define $\mathcal{G}$ as an asymmetric bilinear group generator. $\mathcal{G}$ takes the security parameter λ and outputs $(p, g, G, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$, where $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_t$ are distinct groups of prime order p of length λ, g and G are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively; and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ is a non-degenerate bilinear map. We use bold font to represent vectors in $\mathbb{Z}_p^m$ or $\mathbb{G}^m$. For a vector $\boldsymbol{a} \in \mathbb{Z}_p^m$, we use subscript index $i \in I_d$ to denote $2d$-separation of $\boldsymbol{a}$. Starting from 1 for the first upper subvector subscript, following the order : $\{1, -1, 3, -3, \ldots, 2d-1, -2d+1\}$, small absolute value is in front of a large one, and positive is in front of negative, for lower subvector subscript. We denote $\boldsymbol{a}_1 \parallel \boldsymbol{a}_{-1}$ for sticking two vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_{-1}$, and the notation $\parallel$ can be used when sticking several vectors sequentially. To represent the i-th element of the vector $\boldsymbol{a}$, we use a_i(non-bold style letter with subscript i); that is $\boldsymbol{a} = (a_1, a_2, \ldots, a_m)$. Now, we define notation for some vector operations.

Component-Wise Multiplication : For $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^m$, we denote $\boldsymbol{g} \circ \boldsymbol{h} = (g_1 h_1, \ldots, g_m h_m)$. In general, we denote $\bigcirc_{i \in [I]} \boldsymbol{g}_i = (\prod_{i \in [I]} g_{i,m}, \cdots, \prod_{i \in [I]} g_{i,m})$ for several vectors $\boldsymbol{g}_i = (g_{i,1}, \cdots, g_{i,m}) \in \mathbb{G}^m$ for $i \in I$.

Inner Product : For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^m$, we denote $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i \in [m]} a_i b_i$.

Multi-Exponentiation : For $\boldsymbol{x} \in \mathbb{Z}_p^m$ and $\boldsymbol{g} \in \mathbb{G}$, we denote $\boldsymbol{g}^{\boldsymbol{x}} = \prod_{i \in [m]} g_i^{x_i}$.

Inner Pairing Product : For $\boldsymbol{g} \in \mathbb{G}_1^m$ and $\boldsymbol{H} \in \mathbb{G}_2^m$, we denote $\boldsymbol{E}(\boldsymbol{g}, \boldsymbol{H}) = \prod_{i \in [m]} e(g_i, H_i)$.

Parallel Multi-Exponentiations. We denote two types of parallel multi-exponentiation. One is parallel multi-exponentiation of common base elements, and the other is parallel multi-exponentiation to common vectors.

1. *Parallel multi-exponentiation of common base elements* : Let $\mathbf{a} \in \mathbb{Z}_p^{m \times n}$ be a matrix and $\boldsymbol{g} \in \mathbb{G}^m$ be group elements. We denote $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} := (\boldsymbol{g}^{\boldsymbol{a}_1}, \ldots, \boldsymbol{g}^{\boldsymbol{a}_n})$, where $\boldsymbol{a}_i$ is the i-th column vector of matrix $\boldsymbol{a}$.

2. *Parallel multi-exponentiation to common vectors* : Let $\mathbf{a} \in \mathbb{Z}_p^n$ be a vector and $\boldsymbol{g} \in \mathbb{G}^{m \times n}$ be a group matrix. We denote $\widehat{\boldsymbol{g}^{\boldsymbol{a}}} := (\boldsymbol{g}_1^{\boldsymbol{a}}, \ldots, \boldsymbol{g}_m^{\boldsymbol{a}})$, where $\boldsymbol{g}_i$ is the i-th row group vector of group matrix $\boldsymbol{g}$.

Outer-Pairing Product. We define an outer pairing product, which is a way of generating a target group matrix from source group vectors. For $\boldsymbol{g} \in \mathbb{G}_1^m$ and $\boldsymbol{H} \in \mathbb{G}_2^n$, we denote

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

Argument System for Relation $\boldsymbol{\mathcal{R}}$. A set $\mathcal{R}$ is a polynomial-time verifiable relation consisting of common reference string (CRS), statement, and witness, denoted by σ, x, and w respectively. From the relation, we define language $\mathcal{L}_\sigma = \{\ x \mid \exists\ w \text{ such that } (\sigma, x, w) \in \mathcal{R}\ \}$. We call the statement x true if the statement belongs to the language $\mathcal{L}_\sigma$, and we call w a witness of the statement x under the relation $\mathcal{R}$ if (σ, x, w) belongs to $\mathcal{R}$. For simplicity, we sometimes omit CRS σ and simply write $(x, w) \in \mathcal{R}$.

An interactive argument system for relation $\mathcal{R}$ consists of three probabilistic polynomial-time algorithms (PPTs), key generation algorithms, prover algorithms, verifier algorithms $(\mathcal{K}, \mathcal{P}, \mathcal{V})$. The $\mathcal{K}$ algorithm takes the security parameter λ and outputs CRS, which is the input of $\mathcal{P}$ and $\mathcal{V}$. $\mathcal{P}$ and $\mathcal{V}$ generate transcript interactively, denoted by $tr \leftarrow \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x) \rangle$. At the end of the transcript, $\mathcal{V}$ outputs a bit, 0 or 1, which means reject or accept, respectively. The purpose of $\mathcal{P}$ is to obtain acceptance from $\mathcal{V}$, and the purpose of $\mathcal{V}$ is to check the statement x belongs to $\mathcal{L}_\sigma$.

Argument of Knowledge. An argument of knowledge is a special case of an argument system. Informally, the purpose of $\mathcal{V}$ is to check the knowledge of the witness w of statement x, $(x, w) \in \mathcal{R}$, which guarantees $x \in \mathcal{L}_\sigma$. Arguments of knowledge should satisfy the properties of completeness and witness extractability.

Definition 1 (Perfect Completeness). *Let $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ be an argument system and $\mathcal{R}$ be a polynomial-time verifiable relation. We say that the argument system $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}$ has* ***perfect completeness*** *if, the following* *probability equation holds for all* $\sigma \leftarrow \mathcal{K}(1^\lambda)$*:*

$$\Pr_{(\sigma,x,w)\in\mathcal{R}}\left[\langle \mathcal{P}(\sigma,x,w), \mathcal{V}(\sigma,x)\rangle = 1\right] = 1.$$

Definition 2 (Computational Witness Extended Emulation). *Let* $(\mathcal{K},\mathcal{P},\mathcal{V})$ *be an argument system and* $\mathcal{R}$ *be a polynomial-time verifiable relation. We say that the argument* $(\mathcal{P},\mathcal{V})$ *has* ***witness-extended emulation*** *if, for every deterministic polynomial prover* $\mathcal{P}^*$*, which may not follow* $\mathcal{P}$*, there exists a polynomial time emulator* $\mathcal{E}$ *for which the following inequality holds:*

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

where $negl(\lambda)$ *is a negligible function in* λ*. Emulator* $\mathcal{E}$ *can access the oracle* $\langle\mathcal{P}^*(\sigma,x,s),\mathcal{V}(\sigma,x)\rangle$*, which outputs the transcript between* $\mathcal{P}^*$ *and* $\mathcal{V}$*.* $\mathcal{E}$ *permits to rewind* $\mathcal{P}^*$ *at a specific round and rerun* $\mathcal{V}$ *using fresh randomness.*

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

Trusted Setup. In some arguments, the CRS generator algorithm takes a trapdoor that should not be revealed to anyone, including the prover and verifier. In this case, CRS generation should be run by a trusted third party. The setting requiring trusted party is called a trusted setup.

Non-interactive Argument in the Random Oracle Model. We call an interactive argument a public coin if $\mathcal{V}$ outputs without decision bits constituting a uniformly random message without dependency of $\mathcal{P}$'s messages. Fiat and Shamir [13] proposed that any public coin interactive argument can be converted to a non-interactive one in the random oracle model. The idea is to substitute $\mathcal{V}$'s random messages to random oracle outputs with inputs of previous messages at this point.

Assumptions Let $\mathcal{G}$ be a group generator. $\mathcal{G}$ takes security parameters λ and then outputs $\mathbb{G}$, describing a group of order p.

Definition 4 (Discrete Logarithm Relation Assumption). *We say that* $\mathbb{G}$ *satisfies the discrete logarithm relation assumption if, for all non-uniform polynomial-time adversaries* $\mathcal{A}$*, the following inequality holds:*

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

where $negl(\lambda)$ *is a negligible function in* λ*.*

Definition 5 (q-Pairing Assumption). *We say that the asymmetric bilinear group generator $\mathcal{G}_b$ satisfies the q-pairing assumption if, for all non-uniform polynomial-time adversaries $\mathcal{A}$, the following inequality holds.*

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

The discrete logarithm relation (DLR) assumption is equivalent to thet DL assumption. Similarly, the q-pairing assumption is equivalent to the 2-pairing assumption, DPair assumption.

2.2 Inner Product Argument

An inner product argument is an argument of knowledge for the inner product relation between two vectors [5]. More concretely, the inner product argument is an argument of knowledge for the following relation:

$$\mathcal{R}_{\mathsf{IPA}} = \{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, A, B \in \mathbb{G}, c \in \mathbb{Z}_p; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^N) : A = \boldsymbol{g}^{\boldsymbol{a}} \wedge B = \boldsymbol{h}^{\boldsymbol{b}} \wedge c = \langle \boldsymbol{a}, \boldsymbol{b} \rangle\}$$

Bünz et al. [7] proposed an improved inner product argument by relation reduction. To achieve low communication cost, they provided a reduction technique from relation $\mathcal{R}_{\mathsf{IPA}}$ to the following relation $\mathcal{R}_{\mathsf{BPIP}}$ using Pedersen commitment of the inner product value c:

$$\mathcal{R}_{\mathsf{BPIP}} = \{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^N, u, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^N) : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}\}$$

After the reduction, they constructed an argument of knowledge for $\mathcal{R}_{\mathsf{BPIP}}$ using recursive reduction. The improved inner product argument, denoted by $\mathsf{BP}_{\mathsf{IP}}$, provides the $O(\log N)$ proof size and $O(N)$ prover and verifier cost.

Lai, Malavolta, and Ronge [19] proposed an inner pairing product argument, and Bünz, Maller, Mishra, and Vesely [9] optimized it. We denote the inner pairing product argument as IPP, which is an argument for the below relation $\mathcal{R}_{\mathsf{IPP}}$. The core structure of IPP is similar to that of $\mathsf{BP}_{\mathsf{IP}}$, and its complexity is $O(\log N)$ size with the $O(N)$ prover and verifier cost.

$$\mathcal{R}_{\mathsf{IPP}} = \{(\boldsymbol{h} \in \mathbb{G}_2^N, P \in \mathbb{G}_t;\ \mathbf{g} \in \mathbb{G}_1^N) : P = \boldsymbol{E}(\boldsymbol{g}, \boldsymbol{h})\}$$

Kim, Lee, and Seo [18] proposed two pairing-based inner product arguments: sublogarithmic proof size Protocol2 and sublinear verifier Protocol3. Before describing our protocols, we briefly explain two schemes: Protocol2 and Protocol3.

Protocol2: Sublogarithm Communication Inner Product Argument. Protocol2 is an argument of knowledge for the relation $\mathcal{R}_{\mathsf{BPIP}}$. The construction of Protocol2 consists of three steps: *round reducing, commit-and-prove, aggregating technique*. First, they construct refined reduction, which induces decreasing total rounds. However, there is no benefit in terms of communication costs because refined reduction results in high communication costs per round. To reduce communication cost, they apply the *commit-and-prove* approach, which commits the prover's message per round and then proves the knowledge of the prover's message. This approach reduces total communication cost, but logarithmic complexity remains. To further reduce communication cost, they apply the *aggregating technique*, which delays the proof for each round until the last time the prover generates proof for the previous claims. To achieve sublogarithm communication, they proposed augmented aggregating multi-exponentiation argument, aAggMEA.

Protocol3: Sublinear Verifier Inner Product Argument. Protocol3 is an inner product argument with a sublinear verifier for the below relation $\mathcal{R}_{\mathsf{PT3}}$. The reduction process is equivalent to $\mathsf{BP}_{\mathsf{IP}}$, but one difference is the common reference string. The CRS of Protocol3 is $\boldsymbol{g}, \boldsymbol{h}$, and $\boldsymbol{H}$, whose length is the square root of the witness length. The CRS structure makes the verifier avoid linear computation.

$$\mathcal{R}_{\mathsf{PT3}} = \left\{ \begin{array}{c} \left(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m, \boldsymbol{H} \in \mathbb{G}_2^n, u, P \in \mathbb{G}_t; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}\right) : \\ P = (\boldsymbol{g} \otimes \boldsymbol{H})^{\boldsymbol{a}} \cdot (\boldsymbol{h} \otimes \boldsymbol{H})^{\boldsymbol{b}} \cdot u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \end{array} \right\}$$

3 Main Results

3.1 Motivation and Reducing Round

This paper mainly aims to construct a pairing-based inner product argument that provides a sublogarithmic proof size and sublinear verifier cost simultaneously. To construct it, we focus on combining two protocols: Protocol2 and Protocol3. Our approach is to apply the idea of Protocol2 on Protocol3 for the relation $\mathcal{R}_{\mathsf{PT3}}$. Rather than half reduction, it is $2d$ times smaller per round. The following protocol RRPT3 in Fig. 1 is a round reduced version of Protocol3, applying a *round-reducing* technique.

The next step is *commit-and-prove*. To reduce communication cost per round, we substitute sending whole commitment $\boldsymbol{v}$ with sending commitment to $\boldsymbol{v}$ with proof for $\boldsymbol{v}$. In the case of Protocol2, $2d(2d-1)$ group elements are committed by the pairing-based commitment scheme by Abe et. al. [1] because they belong to the source group $\mathbb{G}_1$. Meanwhile, it is difficult to apply pairing-based group commitments directly on RRPT3 because group elements $\boldsymbol{v}$ belong to the target group $\mathbb{G}_t$. To the best of our knowledge, there are no homomorphic commitment schemes for target group elements of bilinear groups.

Key Idea : Decompose Commitment of $\boldsymbol{a}$ and $\boldsymbol{b}$. To detour the obstacle, we observe the bilinear structure of the following product:

$$(\boldsymbol{g} \otimes \boldsymbol{H})^{\boldsymbol{a}} \cdot (\boldsymbol{h} \otimes \boldsymbol{H})^{\boldsymbol{b}} = \boldsymbol{E}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}, \boldsymbol{H})$$

From the bilinear property, we can change operation order, outer product, and multi-exponentiation to parallel multi-exponentiation and inner pairing

$\boxed{\mathsf{RRPT3}(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{H}, u, P; \boldsymbol{a}, \boldsymbol{b})}$

If $m = 1$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Run $\mathsf{BP}_{\mathsf{IP}}(g \otimes \boldsymbol{H}, h \otimes \boldsymbol{H}, u, P; \boldsymbol{a}, \boldsymbol{b})$

Else $(m > 1)$: Let $m' = \frac{m}{2d}$. Parse $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to

$$\boldsymbol{a} = [\boldsymbol{a}_1 \parallel \boldsymbol{a}_{-1} \parallel \cdots \parallel \boldsymbol{a}_{2d-1} \parallel \boldsymbol{a}_{-2d+1}], \quad \boldsymbol{g} = \boldsymbol{g}_1 \parallel \boldsymbol{g}_{-1} \parallel \cdots \parallel \boldsymbol{g}_{2d-1} \parallel \boldsymbol{g}_{-2d+1},$$
$$\boldsymbol{b} = [\boldsymbol{b}_1 \parallel \boldsymbol{b}_{-1} \parallel \cdots \parallel \boldsymbol{b}_{2d-1} \parallel \boldsymbol{b}_{-2d+1}], \quad \boldsymbol{h} = \boldsymbol{h}_1 \parallel \boldsymbol{h}_{-1} \parallel \cdots \parallel \boldsymbol{h}_{2d-1} \parallel \boldsymbol{h}_{-2d+1}$$

$\boxed{\mathcal{P}}$: Calculate $v[i,j]$ for all distinct $i, j \in I_n$, such that

$$v[i,j] = (\boldsymbol{g}_i \otimes \boldsymbol{H})^{\boldsymbol{a}_j} \cdot (\boldsymbol{h}_j \otimes \boldsymbol{H})^{\boldsymbol{b}_i} \cdot u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \in \mathbb{G}_t$$

and concatenate $v[i,j]$ to $\boldsymbol{v} \in \mathbb{G}_t^{2d(2d-1)}$ in lexicographic order

$\boxed{\mathcal{P} \rightarrow \mathcal{V}}$: $\boldsymbol{v}$

$\boxed{\mathcal{V} \rightarrow \mathcal{P}}$: $x \xleftarrow{\$} \mathbb{Z}_p^*$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Set $\boldsymbol{x} = (x^{j-i}) \in \mathbb{Z}_p^{2d(2d-1)}$ in lexicographic order. Then, computes

$$\boldsymbol{g}' = \circ_{i \in I_d} \boldsymbol{g}_i^{x^{-i}}, \quad \boldsymbol{h}' = \circ_{i \in I_d} \boldsymbol{h}_i^{x^i}, \quad P' = P \cdot \boldsymbol{v}^{\boldsymbol{x}}$$

$\boxed{\mathcal{P}}$: Compute

$$\boldsymbol{a}' = \sum_{i \in I_n} \boldsymbol{a}_i x^i \in \mathbb{Z}_p^{m' \times n}, \quad \boldsymbol{b}' = \sum_{i \in I_n} \boldsymbol{b}_i x^{-i} \in \mathbb{Z}_p^{m' \times n}$$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Run $\mathsf{RRPT3}(\boldsymbol{g}', \boldsymbol{h}', \boldsymbol{H}, u, P'; \boldsymbol{a}', \boldsymbol{b}')$.

Fig. 1. RRPT3: Round Reduced Protocol3

product. Let us focus on the term $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}$. By DL assumption on $\mathbb{G}_1$, $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}} \in \mathbb{G}_1^n$ can be a valid binding commitment of $\boldsymbol{a}$ and $\boldsymbol{b}$. In addition, we can apply pairing-based group commitment for $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}$.

Inner Product Term. In the above change, we only substitute commitment of witness vectors $\boldsymbol{a}, \boldsymbol{b}$, not their inner product $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$. In Protocol3, the inner product part $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ is committed using single exponentiation on the base $u \in \mathbb{G}_t$. To apply pairing-based group commitments to the exponentiation $u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}$, we use $\mathbb{G}_1$ base for commitment, not $\mathbb{G}_t$. Then, we add additional CRS $U \in \mathbb{G}_2$ to combine vector commitment and inner product terms to one target group element $P \in \mathbb{G}_t$

Now, we describe VRPT3, a variant of RRPT3, as shown in Fig. 2. VRPT3 is an argument of knowledge for the following relation:

$\mathsf{VRPT3}(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{H}, u, U, P; \boldsymbol{a}, \boldsymbol{b})$

If $m = 1$

$\boxed{\mathcal{P}\ \&\ \mathcal{V}}$: Run $\mathsf{BP}_{\mathsf{IP}}(g \otimes \boldsymbol{H}, h \otimes \boldsymbol{H}, u, P; \boldsymbol{a}, \boldsymbol{b})$

Else $(m > 1)$: Let $m' = \frac{m}{2d}$. Parse $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to

$$\boldsymbol{a} = [\boldsymbol{a}_1 \parallel \boldsymbol{a}_{-1} \parallel \cdots \parallel \boldsymbol{a}_{2d-1} \parallel \boldsymbol{a}_{-2d+1}], \quad \boldsymbol{g} = \boldsymbol{g}_1 \parallel \boldsymbol{g}_{-1} \parallel \cdots \parallel \boldsymbol{g}_{2d-1} \parallel \boldsymbol{g}_{-2d+1},$$
$$\boldsymbol{b} = [\boldsymbol{b}_1 \parallel \boldsymbol{b}_{-1} \parallel \cdots \parallel \boldsymbol{b}_{2d-1} \parallel \boldsymbol{b}_{-2d+1}], \quad \boldsymbol{h} = \boldsymbol{h}_1 \parallel \boldsymbol{h}_{-1} \parallel \cdots \parallel \boldsymbol{h}_{2d-1} \parallel \boldsymbol{h}_{-2d+1}$$

$\boxed{\mathcal{P}}$: Compute $\boldsymbol{v}[i,j]$ and $w[i,j]$ for all distinct $i, j \in I_d$ such that

$$\boldsymbol{v}[i,j] = \overrightarrow{\boldsymbol{g}_i^{\boldsymbol{a}_j}} \circ \overrightarrow{\boldsymbol{h}_j^{\boldsymbol{b}_i}} \in \mathbb{G}_1^n, w[i,j] = u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \in \mathbb{G}_1$$

and concatenate $\boldsymbol{v}[i,j]$ and $w[i,j]$ to $\boldsymbol{v} \in \mathbb{G}_1^{n \times 2d(2d-1)}$ and $\boldsymbol{w} \in \mathbb{G}_1^{2d(2d-1)}$ in lexicographic order, respectively.

$\boxed{\mathcal{P} \to \mathcal{V}}$: $\boldsymbol{v}$, $\boldsymbol{w}$

$\boxed{\mathcal{V} \to \mathcal{P}}$: $x \xleftarrow{\$} \mathbb{Z}_p^*$

$\boxed{\mathcal{P}\ \&\ \mathcal{V}}$: Set $\boldsymbol{x} = (x^{j-i}) \in \mathbb{Z}_p^{2d(2d-1)}$ in lexicographic order. Then, compute

$$\nu = \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H}) \in \mathbb{G}_t, \quad \mu = \boldsymbol{w}^{\boldsymbol{x}} \in \mathbb{G}_1, \quad P' = P \cdot \nu \cdot e(\mu, U) \in \mathbb{G}_t$$
$$\boldsymbol{g}' = \bigcirc_{i \in I_d} \boldsymbol{g}_i^{x^{-i}} \in \mathbb{G}_1^{m'}, \quad \boldsymbol{h}' = \bigcirc_{i \in I_d} \boldsymbol{h}_i^{x^i} \in \mathbb{G}_1^{m'}$$

$\boxed{\mathcal{P}}$: Compute

$$\boldsymbol{a}' = \sum_{i \in I_n} \boldsymbol{a}_i x^i \in \mathbb{Z}_p^{m' \times n}, \quad \boldsymbol{b}' = \sum_{i \in I_n} \boldsymbol{b}_i x^{-i} \in \mathbb{Z}_p^{m' \times n}$$

$\boxed{\mathcal{P}\ \&\ \mathcal{V}}$: Run $\mathsf{VRPT3}(\boldsymbol{g}', \boldsymbol{h}', \boldsymbol{H}, u, P'; \boldsymbol{a}', \boldsymbol{b}')$.

Fig. 2. VRPT3: Variant Round Reduced Protocol3

$$\mathcal{R}_{\mathsf{VRPT3}} = \left\{ \begin{array}{c} \left(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m, \boldsymbol{H} \in \mathbb{G}_2^n, u \in \mathbb{G}_1, U \in \mathbb{G}_2, P \in \mathbb{G}_t; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{m \times n}\right) : \\ P = \boldsymbol{E}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}, \boldsymbol{H}) \cdot e(u, U)^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \end{array} \right\}$$

Theorem 1. VRPT3 *provides perfect completeness and witness extended emulator under the discrete logarithm assumption.*

The main idea of the proof is similar to that of generalized-BP [18]. For more details, please refer to Appendix A.

3.2 Commit-and-Prove approach

In this section, we apply the *commit-and-prove* approach to $\mathsf{VRPT3}$. Instead of sending $\boldsymbol{v}, \boldsymbol{w}$, the prover sends commitments $V = \boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F})$ and $W = \boldsymbol{E}(\boldsymbol{w}, \boldsymbol{K})$, where $\boldsymbol{F} \in \mathbb{G}_2^{n\times 2n(2n-1)}$ and $\boldsymbol{K} \in \mathbb{G}_2^{2n(2n-1)}$ are additional CRS for commitments. After receiving commitments V and W, the verifier sends a random challenge to the Prover. Unlike $\mathsf{VRPT3}$, the verifier cannot update instance P' because the verifier does not know $\boldsymbol{v}$ and $\boldsymbol{w}$. Thus, the prover sends $\nu = \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H})$ and $\mu = \boldsymbol{w}^{\boldsymbol{x}}$ to the verifier to update P', and then they run additional argument for knowledge $\boldsymbol{v}$ and $\boldsymbol{w}$. The argument should guarantee knowledge for $\boldsymbol{v}$ and $\boldsymbol{w}$ such that $\nu = \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H})$ and $\mu = \boldsymbol{w}^{\boldsymbol{x}}$.

Parallel Multi-Exponentiation Argument. Let us focus on the argument of knowledge for $\boldsymbol{v}$. In the argument system, the prover's claim is the knowledge of $\boldsymbol{v}$, which satisfies $V = \boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F})$ and $\nu = \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H})$. We can construct an argument system using the half-reduction idea of $\mathsf{BP_{IP}}$. We denote the argument for $\boldsymbol{v}$ as parallel multi-exponentiation argument (PMEA). PMEA is an argument system for the following relation:

$$\mathcal{R}_{\mathsf{PMEA}} = \left\{ \begin{matrix} (\boldsymbol{F} \in \mathbb{G}_2^{n\times c}, \boldsymbol{x} \in \mathbb{Z}_p^c, \boldsymbol{H} \in \mathbb{G}_2^n, V, \nu \in \mathbb{G}_t; \boldsymbol{v} \in \mathbb{G}_1^{n\times c}) : \\ V = \boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}) \wedge \nu = \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H}) \end{matrix} \right\}$$

Argument of Knowledge for $\boldsymbol{w}$. Certainly, the prover claims knowledge of $\boldsymbol{w}$, which satisfies multi-exponentiation relation $\mu = \boldsymbol{w}^{\boldsymbol{x}}$. We can apply the MEA protocol [18] for knowledge of $\boldsymbol{w}$. Therefore, we do not explain the details of the argument for $\boldsymbol{w}$.

Using two protocols PMEA and MEA, we can construct a reduced communication protocol. However, for a similar reason as constructing $\mathsf{Protocol2}$, we should apply an *Aggregation* technique to achieve sublogarithmic communication cost. The aggregation of multi MEA, called $\mathsf{aAggMEA}$, was already proposed by Kim, Lee, and Seo [18]. Inspired by the idea of $\mathsf{aAggMEA}$, we construct aggregated arguments for PMEA.

Aggregation PMEA. In this section, we focus on aggregating PMEA protocols to apply our main protocol. One of the aggregating techniques is random linear combination. However, naïve random combination does not guarantee unuseness of $\boldsymbol{F}_s$ to construct V_ℓ for all $s \neq \ell$. To detour it, we use idea of $\mathsf{aAggMEA}$, which are used in $\mathsf{Protocol2}$. Similarly, we add redundant witness $\boldsymbol{v}_{\ell,r}$ and construct an argument for the following relation.

$$\mathcal{R}_{\mathsf{APMEA}} = \left\{ \begin{matrix} (\boldsymbol{F}_\ell \in \mathbb{G}_2^{n\times c}, \boldsymbol{x}_\ell \in \mathbb{Z}_p^c, \boldsymbol{H} \in \mathbb{G}_2^n, V_\ell, \nu_\ell \in \mathbb{G}_t; \boldsymbol{v}_{\ell,r} \in \mathbb{G}_1^{n\times c}, \ell, r \in [R]) : \\ \wedge_{\ell\in[R]} \left(V_\ell = \prod_{s\in[R]} \boldsymbol{E}(\boldsymbol{v}_{\ell,s}, \boldsymbol{F}_s) \wedge \nu = \boldsymbol{E}(\widehat{\boldsymbol{v}_{\ell,\ell}{}^{\boldsymbol{x}_\ell}}, \boldsymbol{H}) \right) \\ \wedge \left(\wedge_{\ell,r\in[R]\wedge \ell\neq r} \widehat{\boldsymbol{v}_{\ell,r}{}^{\boldsymbol{x}_r}} = \boldsymbol{1} \right) \end{matrix} \right\}$$

We construct a protocol APEMA for the relation $\mathcal{R}_{\mathsf{APMEA}}$. We describe details in Fig. 3. APEMA consists of two steps, the aggregation and the recursive reduction.

$\boxed{\mathsf{APMEA}(\boldsymbol{F}_\ell, \boldsymbol{x}_\ell, \boldsymbol{H}, V_\ell, \nu_\ell; \boldsymbol{v}_{\ell,r}))}$

$\boxed{\mathcal{V} \to \mathcal{P}}$: $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Compute $\boldsymbol{F}'_\ell \in \mathbb{G}_2^{n\times c}, \boldsymbol{x}'_\ell \in \mathbb{Z}_p^c, \boldsymbol{H}' \in \mathbb{G}_2^n, P \in \mathbb{G}_t$ such that

$$\boldsymbol{F}'_\ell = \boldsymbol{F}_\ell^{\alpha^{\ell-1}}, \quad \boldsymbol{x}'_\ell = \alpha^{\ell-1}\boldsymbol{x}_\ell, \quad \boldsymbol{H}' = \boldsymbol{H}^{\alpha^R}, \quad P = \prod_{\ell\in[R]} V_\ell^{\alpha^{\ell-1}} \nu_\ell^{\alpha^{R+\ell-1}}$$

$\boxed{\mathcal{P}}$: Compute $\boldsymbol{v}'_\ell = \bigcirc_{s\in[R]} \boldsymbol{v}_{s,\ell}^{\alpha^{s-\ell}}$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Run $\mathsf{ProdPMEA}(\boldsymbol{F'}_\ell, \boldsymbol{x}'_\ell, \boldsymbol{H}', P; \boldsymbol{v}'_\ell)$

$\boxed{\mathsf{ProdPMEA}(\boldsymbol{F}_\ell, \boldsymbol{x}_\ell, \boldsymbol{H}, P; \boldsymbol{v}_\ell)}$

If $c = 1$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Set $\boldsymbol{F}'_\ell = \boldsymbol{F}_\ell \circ \boldsymbol{H}_\ell^{x_\ell} \in \mathbb{G}_2^n, \ \forall \ell \in [R]$ and then concatenate ℓ vectors $\boldsymbol{F}'_\ell$ into $\boldsymbol{F}' \in \mathbb{G}_2^{nR}$.

$\boxed{\mathcal{P}}$: Concatenate all of $\boldsymbol{v}_\ell \in \mathbb{Z}_p^n$ into $\boldsymbol{v} \in \mathbb{G}_1^{nR}$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Run $\mathsf{IPP}(\boldsymbol{F}', P; \boldsymbol{v})$

Else $(c > 1)$: Let $c' = \frac{c}{2}$ and parse $\boldsymbol{F}_\ell$, $\boldsymbol{x}_\ell$, $\boldsymbol{v}_\ell$

$$\boldsymbol{F}_\ell = [\boldsymbol{F}_{\ell,1} \parallel \boldsymbol{F}_{\ell,-1}], \quad \boldsymbol{x}_\ell = \boldsymbol{x}_{\ell,1} \parallel \boldsymbol{x}_{\ell,-1}, \quad \boldsymbol{v}_\ell = [\boldsymbol{v}_{\ell,1} \parallel \boldsymbol{v}_{\ell,-1}]$$

$\boxed{\mathcal{P}}$: Calculate $L, R \in \mathbb{G}_t$ such that

$$L = \prod_{\ell\in[R]} \boldsymbol{E}(\boldsymbol{v}_{\ell,1}, \boldsymbol{F}_{\ell,-1})\boldsymbol{E}(\widehat{\boldsymbol{v}_{\ell,1}{}^{\boldsymbol{x}_{\ell,-1}}}, \boldsymbol{H}), \ R = \prod_{\ell\in[R]} \boldsymbol{E}(\boldsymbol{v}_{\ell,-1}, \boldsymbol{F}_{\ell,1})\boldsymbol{E}(\widehat{\boldsymbol{v}_{\ell,-1}{}^{\boldsymbol{x}_{\ell,1}}}, \boldsymbol{H})$$

$\boxed{\mathcal{P} \to \mathcal{V}}$: L, R

$\boxed{\mathcal{V} \to \mathcal{P}}$: $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: Compute $\boldsymbol{F}'_\ell \in \mathbb{G}_2^{n\times c'}, \boldsymbol{x}'_\ell \in \mathbb{Z}_p^{c'}, P' \in \mathbb{G}_t$ such that

$$\boldsymbol{F}'_\ell = \boldsymbol{F}_{\ell,1}^{\alpha^{-1}} \circ \boldsymbol{F}_{\ell,-1}^{\alpha}, \ \boldsymbol{x}'_\ell = \alpha^{-1}\boldsymbol{x}_{\ell,1} + \alpha\boldsymbol{x}_{\ell,-1}, \ P' = L^{\alpha^2} P R^{\alpha^{-2}}$$

$\boxed{\mathcal{P}}$: Compute $\boldsymbol{v}'_\ell = \boldsymbol{v}_{\ell,1}^{\alpha} \circ \boldsymbol{v}_{\ell,-1}^{\alpha^{-1}} \in \mathbb{G}_1^{n\times c'}$

$\boxed{\mathcal{P} \ \& \ \mathcal{V}}$: $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{ProdPMEA}(\boldsymbol{F}'_\ell, \boldsymbol{x}'_\ell, \boldsymbol{H}, P'; \boldsymbol{v}'_\ell)$

Fig. 3. APMEA: Augmented Aggregating Parallel Multi-Exponentiation Argument

Using the verifier challenges, the protocol lets R distinct commitments V_ℓ and evaluation ν_ℓ aggregate to one group element P. After the aggregating step, the prover and verifier run ProdPMEA for the following relation:

$$\mathcal{R}_{\mathsf{ProdPMEA}} = \left\{ \begin{matrix} (\boldsymbol{F}_\ell \in \mathbb{G}_2^{n\times c}, \boldsymbol{x}_\ell \in \mathbb{Z}_p^c, \boldsymbol{H} \in \mathbb{G}_2^n, P \in \mathbb{G}_t; \boldsymbol{v}_\ell \in \mathbb{G}_1^{n\times c}, \ell \in [R]) : \\ P = \prod_{s\in[R]} \boldsymbol{E}(\boldsymbol{v}_s, \boldsymbol{F}_s) \cdot \boldsymbol{E}(\widehat{\boldsymbol{v}_s{}^{\boldsymbol{x}_s}}, \boldsymbol{H}) \end{matrix} \right\}$$

Theorem 2. *Let* ProdPMEA *provide perfect completeness and witness extended emulator. Then, the* APMEA *protocol provides perfect completeness and witness extended emulator under the double pairing assumption.*

Theorem 3. *The* ProdPMEA *protocol provides perfect completeness and witness extended emulator under the double pairing assumption.*

Proof Sketch. We sketch the proof for witness extended emulation (WEE) of APMEA. In a similar way to ProdMEA [18], we can construct WEE of ProdPMEA. One difference is that the WEE of ProdPMEA runs the WEE of IPP as a subroutine. Once getting WEE of ProdPMEA, one can construct a WEE of APMEA, which uses the WEE of ProdPMEA as a subroutine. From $2R$ distinct extracted witnesses from the WEE of ProdPMEA, one can extract witness $\boldsymbol{v}_{\ell,r}$.

3.3 Main Protocols

In this section, we explain our main protocol TENET, sublogarithm communication, and sublinear verifier without a trusted setup. TENET consists of four phases: row reduction, column reduction, APMEA, and aAggMEA.

The purpose of row reduction is to reduce witness size from $m \times n$ to $\frac{m}{2d} \times n$ per round. For each round, the prover sends commitments V and W to the verifier. After receiving the verifier's challenges, the prover sends evaluation ν and μ to the verifier. Then, the prover and verifier run the protocol recursively without checking the proof for knowledge $\boldsymbol{v}$ and $\boldsymbol{w}$. Rather than checking per round, the prover and verifier store statements $\boldsymbol{v}, \boldsymbol{w}, \nu, \mu$ per round. Then, the prover and verifier run column reduction, which is identical to $\mathsf{BP}_{\mathsf{IP}}$ on base $\mathbb{G}_t$. If the $\mathsf{BP}_{\mathsf{IP}}$ verifier outputs 1, the prover and verifier run APMEA and aAggMEA using stored statements from row reduction. We describe TENET in Fig.4, which is applied using the following optimization technique.

Optimization : Compress Columns of $\boldsymbol{v}$ and $\boldsymbol{w}$. In this section, we present an optimization technique for APMEA and aAggMEA. Certainly, this idea can be applied to Protocol2, which contains aAggMEA as a subprotocol. Let us focus on the prover's message of VRPT3 in Fig. 2. For each round, the prover sends $\boldsymbol{v}$ and $\boldsymbol{w}$, which are commitments to parsed matrices and inner products with $2d(2d-1)$ columns. When computing multi-exponentiation to $\boldsymbol{x}$, each of the columns $\boldsymbol{v}[i,j]$ and $w[i,j]$ meet an exponent x^{j-i}. In this case, some columns meet the same exponent x^{j-i}. More concretely, we can rewrite multi-exponentiation of $4d-2$ group elements by the following equations:

$$\bigcirc_{i,j \in I_d \wedge i \neq j} \boldsymbol{v}[i,j]^{x^{i-j}} = \bigcirc_{s \in J_d} \big(\bigcirc_{s=i-j} \boldsymbol{v}[i,j] \big)^{x^s}, \quad \prod_{i,j \in I_d \wedge i \neq j} w[i,j]^{x^{i-j}} = \prod_{s \in J_d} \big(\prod_{s=i-j} w[i,j] \big)^{x^s}$$

This implies that only $4d-2$ different terms are sufficient to update P'. Then, is $4d-2$ terms sufficient to guarantee knowledge of witness $\boldsymbol{a}$ and $\boldsymbol{b}$? Let D_s be a set of tuples such that $D_s = \{i, j \in I_d | s = i - j\}$. Then, any tuples in D_s cannot have a common entry with each other. Since the tuple is related to base

$\boxed{\mathsf{TENET}(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{H}, \boldsymbol{F}_\ell, \boldsymbol{K}_\ell, u, U, P, st_I; \boldsymbol{a}, \boldsymbol{b}, st_W \text{ for } \ell \in [R])}$

If $m = 1$:

$\boxed{\mathcal{P} \,\&\, \mathcal{V}}$: Run $\mathsf{BP_{IP}}(g \otimes \boldsymbol{H}, h \otimes \boldsymbol{H}, e(u, U), P; \boldsymbol{a}, \boldsymbol{b})$

If $st_P = \perp$:

$\boxed{\mathcal{V}}$ *Accept* the protocol.

Else : Let $(V_\ell, \nu_\ell, W_\ell, \mu_\ell, \boldsymbol{x}_\ell; \boldsymbol{v}_{\ell,r}, \boldsymbol{w}_{\ell,r})$ be the ℓ-th row in (st_I, st_W)

$\boxed{\mathcal{P}}$: Set $\boldsymbol{v}_{\ell,\ell} = \boldsymbol{v}_\ell$, $\boldsymbol{w}_{\ell,\ell} = \boldsymbol{w}_\ell$ for all $\ell \in [R]$ and $\boldsymbol{v}_{\ell,r} = \boldsymbol{1}_{\mathbb{G}_1}$, $\boldsymbol{w}_{\ell,r} = \boldsymbol{1}_{\mathbb{G}_1}$ for all distinct $\ell, r \in [R]$

$\boxed{\mathcal{P} \,\&\, \mathcal{V}}$: Run $\mathsf{APMEA}(\boldsymbol{F}_\ell, \boldsymbol{x}_\ell, \boldsymbol{H}, \boldsymbol{K}, V_\ell, \nu_\ell; \boldsymbol{v}_{\ell,r})$, $\mathsf{aAggMEA}(\boldsymbol{K}_\ell, \boldsymbol{x}_\ell, W_\ell, \mu_\ell; \boldsymbol{w}_{\ell,r})$

Else $(m > 1)$: Let $m' = \frac{m}{2d}$. Parse $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}$, and $\boldsymbol{h}$ to

$$\boldsymbol{a} = [\boldsymbol{a}_1 \parallel \boldsymbol{a}_{-1} \parallel \cdots \parallel \boldsymbol{a}_{2d-1} \parallel \boldsymbol{a}_{-2d+1}], \quad \boldsymbol{g} = \boldsymbol{g}_1 \parallel \boldsymbol{g}_{-1} \parallel \cdots \parallel \boldsymbol{g}_{2d-1} \parallel \boldsymbol{g}_{-2d+1},$$
$$\boldsymbol{b} = [\boldsymbol{b}_1 \parallel \boldsymbol{b}_{-1} \parallel \cdots \parallel \boldsymbol{b}_{2d-1} \parallel \boldsymbol{b}_{-2d+1}], \quad \boldsymbol{h} = \boldsymbol{h}_1 \parallel \boldsymbol{h}_{-1} \parallel \cdots \parallel \boldsymbol{h}_{2d-1} \parallel \boldsymbol{h}_{-2d+1}$$

$\boxed{\mathcal{P}}$: Compute $\boldsymbol{v}[s] = \bigcirc_{s=i-j} (\overrightarrow{\boldsymbol{g}_i^{\boldsymbol{a}_j}} \circ \overrightarrow{\boldsymbol{h}_j^{\boldsymbol{b}_i}}) \in \mathbb{G}_1^n$, $w[s] = \prod_{s=i-j} u^{\langle \boldsymbol{a}_j, \boldsymbol{b}_i \rangle} \in \mathbb{G}_1 \ \forall s \in J_d$, and then concatenate $\boldsymbol{v}[s]$ and $w[s]$ to $\boldsymbol{v} \in \mathbb{G}_1^{n \times 4d-2}$ and $\boldsymbol{w} \in \mathbb{G}_1^{4d-2}$ in ascending order. And then compute $V = \boldsymbol{E}(\boldsymbol{v}, \boldsymbol{F}_\ell)$ and $W = \boldsymbol{E}(\boldsymbol{w}, \boldsymbol{K}_\ell)$

$\boxed{\mathcal{P} \to \mathcal{V}}$: V, W

$\boxed{\mathcal{V} \to \mathcal{P}}$: $x \xleftarrow{\$} \mathbb{Z}_p^*$

$\boxed{\mathcal{P}}$: Set $\boldsymbol{x} = (x^s)_{s \in J_d} \in \mathbb{Z}_p^{4d-2}$ in ascending order. Then, compute ν, μ

$$\nu = \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H}) \in \mathbb{G}_t, \quad \mu = \boldsymbol{w}^{\boldsymbol{x}} \in \mathbb{G}_1$$

$\boxed{\mathcal{P} \to \mathcal{V}}$: ν, μ

$\boxed{\mathcal{P} \,\&\, \mathcal{V}}$: Compute $\boldsymbol{g}', \boldsymbol{h}', P'$

$$\boldsymbol{g}' = \circ_{i \in I_d} \boldsymbol{g}_i^{x^{-i}} \in \mathbb{G}_1^{m'}, \ \boldsymbol{h}' = \circ_{i \in I_d} \boldsymbol{h}_i^{x^i} \in \mathbb{G}_1^{m'}, P' = P \cdot \nu \cdot e(\mu, U) \in \mathbb{G}_t$$

Then, update st_I by adding a tuple $(V, \nu, W, \mu, \boldsymbol{x})$ to the bottom.

$\boxed{\mathcal{P}}$: Compute $\boldsymbol{a}' = \sum_{i \in I_n} \boldsymbol{a}_i x^i \in \mathbb{Z}_p^{m' \times n}$, $\quad \boldsymbol{b}' = \sum_{i \in I_n} \boldsymbol{b}_i x^{-i} \in \mathbb{Z}_p^{m' \times n}$

Then, update st_W by adding a tuple $(\boldsymbol{v}, \boldsymbol{w})$ to the bottom.

$\boxed{\mathcal{P} \,\&\, \mathcal{V}}$: Run $\mathsf{TENET}(\boldsymbol{g}', \boldsymbol{h}', \boldsymbol{H}, \boldsymbol{F}_\ell, \boldsymbol{K}_\ell, u, U, P, st_V; \boldsymbol{a}', \boldsymbol{b}', st_P \text{ for } \ell \in [R-1])$

Fig. 4. TENET

group elements, $\boldsymbol{v}[i,j] = \overrightarrow{\boldsymbol{g}_i^{\boldsymbol{a}_j}} \circ \overrightarrow{\boldsymbol{h}_j^{\boldsymbol{b}_i}}$ have distinct bases from each other on tuple set D_s. Under the DLR assumption, witness vectors $\boldsymbol{a}_j$ and $\boldsymbol{b}_i$ are extractable from products of $\boldsymbol{v}[i,j]$. For more details on witness extraction, please refer to Appendix A.

Let us define the column-reduced vector $\bar{\boldsymbol{v}} \in \mathbb{G}_1^{n \times 4d-2}$ and $\bar{\boldsymbol{w}} \in \mathbb{G}_1^{4d-2}$ as $\bar{\boldsymbol{v}} = \big(\bigcirc_{(i,j) \in D_s} \boldsymbol{v}[i,j] \big)_{s \in J_d}$ and $\bar{\boldsymbol{w}} = \big(\prod_{(i,j) \in D_s} w[i,j] \big)_{s \in J_d}$. Then, we can adjust the prover's action in VRPT3 by generating $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{w}}$ and sending them to the verifier. After applying the *commit-and-prove* approach, the prover's action is changed to sending commitment to $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{w}}$ and their proofs, rather than to $\boldsymbol{v}$ and $\boldsymbol{w}$. Since the witness size is reduced from $O(d^2)$(resp. $\boldsymbol{v}, \boldsymbol{w}$) to $O(d)$(resp. $\bar{\boldsymbol{v}}, \bar{\boldsymbol{w}}$), the required CRS size $\boldsymbol{F}$ and $\boldsymbol{K}$ decrease to $O(d)$, and proof size and verifier cost for APMEA and aAggMEA can be decreased.

Uniform Reference Strings. The required common reference strings for TENET are $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}_1^m$, $\boldsymbol{H} \in \mathbb{G}_2^n$, $\boldsymbol{F}_\ell \in \mathbb{G}_2^{n \times 4d-2}$, $\boldsymbol{K}_\ell \in \mathbb{G}_2^{4d-2}$, and $(u, U) \in \mathbb{G}_1 \times \mathbb{G}_2$, which are all chosen randomly from a uniform distribution, not depending on a trusted party. The total size of common reference strings is $(2m+1)|\mathbb{G}_1| + (R(4d-2)(n+1)+1)|\mathbb{G}_2|$.

Theorem 4. TENET *provides perfect completeness and witness extended emulator under the discrete logarithm assumption and double pairing assumption if* APMEA *and* aAggMEA *provide perfect completeness and witness extended emulator.*

Proof Sketch. The proof idea of TENET is similar to that of Protocol2. The witness extended emulator of APMEA and that of aAggMEA extract prover's messages $\boldsymbol{v}$ and $\boldsymbol{w}$ for all rounds. Using them, we can construct a witness extended emulator for TENET following the witness extended emulator of VRPT3.

Efficiency. We explain the cost of TENET, communication cost, verifier computational cost and prover computational cost. We analyze TENET in four parts: row reduction, column reduction, APMEA, and aAggMEA. We describe the efficiency of them in Table 2.

Row Reduction.

Communication: For each row reduction, the prover sends 3 $\mathbb{G}_t$ elements and 1 $\mathbb{G}_1$ element. Since the total round of row reduction is $R = \log_{2d} m$, total communication is $O(R)$.

Prover's Complexity: To compute $\boldsymbol{v}$ and $\boldsymbol{w}$ per round, the prover computes $\frac{m}{2d} \cdot n \cdot 2d(2d-1)$ $\mathbb{G}_1$-exp with $n \cdot (4d-2)$ pairing and $2d(2d-1)$ $\mathbb{G}_1$-exp with $4d-2$ pairing. After receiving a challenge, the prover constructs ν and μ, whose costs are $n \cdot (4d-2)$ $\mathbb{G}_1$-exp with n pairing and $4d-2$ $\mathbb{G}_1$-exp, respectively. Then, the prover computes $2m$ $\mathbb{G}_1$-exp and mn field operations with constant pairing for updating instance and witness steps. Since the size of m is shrinking by $1/2d$ times per round, the overwhelming term of prover complexity is $O(mnd + nd^2R)$.

Verifier's Complexity: The verifier updates instances $\boldsymbol{g}, \boldsymbol{h}$, and P, whose computation costs are $2m$ $\mathbb{G}_1$-exp in total. Similarly prover complexity, the overwhelming term of verifier complexity is $O(m)$.

Column Reduction.

The column-reduction phase is only running $\mathsf{BP}_{\mathsf{IP}}$ on $\mathbb{G}_t$. However, the CRS update step can be changed to updating $\boldsymbol{H} \in \mathbb{G}_2^n$, rather than $g \otimes \boldsymbol{H} \in$

$\mathbb{G}_t^n$. Therefore, total communication is $O(\log n)$ $\mathbb{G}_t$-exp, and the prover and verifier computation is $O(n)$ $\mathbb{G}_2$-exp.

APMEA.

Communication: In the aggregating phase, the verifier sends one challenge to the prover, but sending a challenge can be substituted by using random oracle by the Fiat-Shamir transform [13]. In the recursive reduction phase, the prover sends two $\mathbb{G}_t$ elements per round, so that the total communication cost is $O(\log(ndR)$.

Prover's Complexity: In the aggregating phase, the prover computes $n(4d-2)R$ $\mathbb{G}_1$-exp for updating witness $\boldsymbol{v}_\ell$ [1] , $n(4d-2)R$ $\mathbb{G}_2$-exp and n $\mathbb{G}_2$-exp for updating $\boldsymbol{F}_\ell$ and $\boldsymbol{H}$, and R $\mathbb{G}_t$-exp for updating P. In the recursive reduction phase, the prover complexity is linear to witness length $n(4d-2)$. Then, the total prover complexity is $O(ndR)$, which is a overwhelming term.

Verifier's Complexity: Since the verifier computes $n(4d-2)R$ $\mathbb{G}_2$-exp for updating $\boldsymbol{F}_\ell$ too, the verifier's complexity is also $O(ndR)$

aAggMEA.

The complexity of aAggMEA is $O(R+\log d)$, $O(dR)$, and $O(dR)$ for communication prover and verifier cost, respectively [18].

	Communication	$\mathcal{P}$'s computation	$\mathcal{V}$'s computation
Row Reduction	$O(R)\lvert\mathbb{G}_t\rvert$	$O(mnd+nd^2R)E_1$	$O(m)E_1$
Column Reduction	$O(\log n)\lvert\mathbb{G}_2\rvert$	$O(n)E_2$	$O(n)E_2$
APMEA	$O(\log ndR)\lvert\mathbb{G}_t\rvert$	$O(nd^2R)E_1$	$O(ndR)E_2$
aAggMEA	$O(R+\log d)\lvert\mathbb{G}_t\rvert$	$O(d^2R)E_2$	$O(dR)E_2$
Total(TENET)	$O(R+\log nd)$	$O(mnd+nd^2R)$	$O(m+ndR)$

Table 2. Complexity Table of TENET

$|\mathbb{G}_i|$: size of group elements in $\mathbb{G}_i$, E_i : group exponentiation on $\mathbb{G}_i$

Parameter Setting. When choosing appropriate parameters on TENET, we can achieve sublogarithm communication and sublinear verifier.

Parameter Setting.: Let $N=mn$ be a length of witness vectors. Set the column size and row size as $n=2^{\sqrt{\log N}}$ and $m=\frac{N}{n}$, respectively. Then, define dividing factor as $2d=2^{\sqrt{\log m}}$. Then, the round number of row reduction $R=\log_{2d} m=\sqrt{\log m}$. Let us put all factors from the above results.

Communication.: The communication cost is $O(R+\log nd)=O(\sqrt{\log N})$

[1] The complexity for computing $\boldsymbol{v}'_\ell$ is $O(ndR^2)$. However, in TENET, the prover sets $\boldsymbol{v}_{s,\ell}=\mathbf{1}$ for all distinct s,ℓ. For this reason, the exponentiation of the redundant terms can be omitted.

Prover's Complexity.: The prover's complexity is $O(N \cdot 2^{\sqrt{\log m}})$. Since m is smaller than N, we have rough bound $O(N \cdot 2^{\sqrt{\log N}})$.

Verifier's Complexity.: For simplicity, we focus on rough bound using substitution $\sqrt{\log m}$ with $\sqrt{\log N}$. Then, the verifier's complexity is $O\big(\frac{N}{2^{\sqrt{\log N}}} + \sqrt{\log N} \cdot 4^{\sqrt{\log N}}\big)$; the term d is substituted with $2^{\sqrt{\log N}}$. The left-term $\frac{N}{2^{\sqrt{\log N}}}$ is larger scale than the right-term $\sqrt{\log N} \cdot 4^{\sqrt{\log N}}$. For this reason, we can conclude that the verifier complexity is $O\big(\frac{N}{2^{\sqrt{\log N}}}\big)$, which is smaller than $O\big(\frac{N}{\log N}\big)$.

References

1. Masayuki Abe, G Fuchsbauer, Jens Groth, K Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. *Journal of Cryptology*, 29(2):363–421, 2016.
2. Arasu Arun, Chaya Ganesh, Satya V. Lokam, Tushar Mopuri, and Sriram Sridhar. Dew: Transparent constant-sized zksnarks. Cryptology ePrint Archive, Report 2022/419, 2022. https://eprint.iacr.org/2022/419.pdf.
3. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed σ-protocol theory for lattices. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II*, pages 549–579. Springer, 2021.
4. E Ben-Sasson, A Chiesa, and N Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*, LNCS, pages 31–60. Springer, 2016.
5. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2006*, LNCS, pages 327–357. Springer, 2016.
6. Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. In *https://eprint.iacr.org/2019/1021*, 2019.
7. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy 2018*, pages 315–334. IEEE, 2018.
8. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 677–706. Springer, 2020.
9. Benedikt Bünz, Mary Maller, Pratyush Mishra, and Noah Vesely. Proofs for inner pairing products and applications. *IACR Cryptology ePrint Archive*, 2019:1177, 2019.
10. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 738–768. Springer, 2020.
11. HeeWon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. *IEEE Access*, 10:42067–42082, 2022.
12. Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In *PKC 2020*, volume 12110 of *LNCS*, pages 527–557. Springer, 2020.
13. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

14. Ariel Gabizon, Zachary J Williamson, and Oana 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.
15. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18:186–208, 1989.
16. Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, pages 305–326. Springer, 2016.
17. 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.
18. Sungwook Kim, Hyeonbum Lee, and Jae Hong Seo. Efficient zero-knowledge arguments in discrete logarithm setting: Sublogarithmic proof or sublinear verifier. In *ASIACRYPT 2022*, pages 403–433. Springer, 2023.
19. 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.
20. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II*, pages 1–34. Springer, 2021.
21. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah 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.
22. Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *IEEE Symposium on Security and Privacy 2018*, pages 926–943. IEEE, 2018.
23. Zibo Zhou, Zongyang Zhang, Hongyu Tao, Tianyu Li, and Boyu Zhao. Efficient inner product arguments and their applications in range proofs. *IET Information Security*, 2023.

A Appendix

A.1 Proof of Theorem 1

Proof. (*completeness*) If $m = 1$, completeness holds by perfect completeness of $\mathsf{BP_{IP}}$. Consider the case $m > 1$.

$$\begin{aligned} P' &= P \cdot \nu \cdot e(\mu, U) = P \cdot \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H}) \cdot e(\boldsymbol{w}^{\boldsymbol{x}}, U) \\ &= \boldsymbol{E}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}}, \boldsymbol{H}) \cdot e(u, U)^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \cdot \boldsymbol{E}(\widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H}) \cdot e(\boldsymbol{w}^{\boldsymbol{x}}, U) \\ &= \boldsymbol{E}(\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}} \circ \widehat{\boldsymbol{v}^{\boldsymbol{x}}}, \boldsymbol{H}) \cdot e(u^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle} \cdot \boldsymbol{w}^{\boldsymbol{x}}, U) \end{aligned}$$

Now, we claim that $\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}} \circ \widehat{\boldsymbol{v}^{\boldsymbol{x}}} = \overrightarrow{\boldsymbol{g}'^{\boldsymbol{a}'}} \circ \overrightarrow{\boldsymbol{h}'^{\boldsymbol{b}'}}$ and $u^{\langle \boldsymbol{a},\boldsymbol{b}\rangle} \cdot \boldsymbol{w}^{\boldsymbol{x}} = u^{\langle \boldsymbol{a}',\boldsymbol{b}'\rangle}$. From the prover's computation, we achieve the following equations:

$$\begin{aligned}\overrightarrow{\boldsymbol{g}^{\boldsymbol{a}}} \circ \overrightarrow{\boldsymbol{h}^{\boldsymbol{b}}} \circ \widehat{\boldsymbol{v}^{\boldsymbol{x}}} &= \left(\circ_{i\in I_d} \overrightarrow{\boldsymbol{g}_i^{\boldsymbol{a}_i}} \circ \overrightarrow{\boldsymbol{h}_i^{\boldsymbol{b}_i}} \right) \circ \left(\circ_{i,j\in I_d \wedge i\neq j} (\overrightarrow{\boldsymbol{g}_i^{\boldsymbol{a}_j}} \circ \overrightarrow{\boldsymbol{h}_j^{\boldsymbol{b}_i}})^{x^{j-i}} \right) \\ &= \circ_{i,j\in I_d} (\overrightarrow{\boldsymbol{g}_i^{\boldsymbol{a}_j}} \circ \overrightarrow{\boldsymbol{h}_j^{\boldsymbol{b}_i}})^{x^{j-i}} \\ &= \overrightarrow{\left(\circ_{i\in I_d} \boldsymbol{g}_i^{x^{-i}} \right)^{(\sum_{j\in I_d} x^j \boldsymbol{a}_j)}} \circ \overrightarrow{\left(\circ_{j\in I_d} \boldsymbol{h}_j^{x^j} \right)^{(\sum_{i\in I_d} x^{-i} \boldsymbol{b}_i)}} \\ &= \overrightarrow{\boldsymbol{g}'^{\boldsymbol{a}'}} \circ \overrightarrow{\boldsymbol{h}'^{\boldsymbol{b}'}}\end{aligned}$$

$$\begin{aligned}u^{\langle \boldsymbol{a},\boldsymbol{b}\rangle} \cdot \boldsymbol{w}^{\boldsymbol{x}} &= \prod_{i\in I_d} u^{\langle \boldsymbol{a}_i,\boldsymbol{b}_i\rangle} \cdot \prod_{i,j\in I_d \wedge i\neq j} u^{\langle \boldsymbol{a}_j x^j, \boldsymbol{b}_i x^{-i}\rangle} = \prod_{i,j\in I_d} u^{\langle \boldsymbol{a}_j x^j, \boldsymbol{b}_i x^{-i}\rangle} \\ &= u^{\langle \sum_{j\in I_d} \boldsymbol{a}_j x^j, \sum_{i\in I_d} \boldsymbol{b}_i x^{-i}\rangle} = u^{\langle \boldsymbol{a}',\boldsymbol{b}'\rangle}\end{aligned}$$

From the equation $P' = \boldsymbol{E}(\overrightarrow{\boldsymbol{g}'^{\boldsymbol{a}'}} \circ \overrightarrow{\boldsymbol{h}'^{\boldsymbol{b}'}}, \boldsymbol{H}) \cdot e(u,U)^{\langle \boldsymbol{a}',\boldsymbol{b}'\rangle}$, the updated instance-witness pair $(\boldsymbol{g}', \boldsymbol{h}', \boldsymbol{H}, u, U, P'; \boldsymbol{a}', \boldsymbol{b}')$ belongs to the relation $\mathcal{R}$
(*witness extended emulation*) In order to show the computational witness extended emulation, we construct an expected polynomial time extractor whose goal is to extract the witness using a polynomially bounded tree of accepting transcripts. If so, we can apply the general forking lemma [5].

The case $(m = 1)$ is clear because $\mathsf{BP}_{\mathsf{IP}}$ has witness extended emulation [7]. Let us focus on the case $(m > 1)$. We prove that, for each recursive step on input $(\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{H}, u, U, P)$, we can efficiently extract from the prover witness vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ under the DLR assumption, whose instance is the CRS $\boldsymbol{g} \parallel \boldsymbol{h} \parallel u$ on $\mathbb{G}_1$ and $\boldsymbol{H} \parallel u$ on $\mathbb{G}_2$. First, the extractor runs the prover to obtain $\boldsymbol{v} \in \mathbb{G}_1^{n\times 2d(2d-1)}$ and $\boldsymbol{w} \in \mathbb{G}_1^{2d(2d-1)}$. At this point, the extractor rewinds the prover $12d-5$ times and feeds $12d-5$ non-zero challenges x_t such that all x_t^2 are distinct. Then, the extractor obtains $12d-5$ pairs $\boldsymbol{a}'_t$ and $\boldsymbol{b}'_t$ such that for $t \in [12d-5]$,

$$P \cdot \prod_{s\in J_d} (\boldsymbol{E}(\boldsymbol{v}_s, \boldsymbol{H}) e(w_s, U))^{x_t^s} = P'_t = \boldsymbol{E}\left(\bigcirc_{i\in I_d} \overrightarrow{(\boldsymbol{g}_i^{x_t^{-i}})^{\boldsymbol{a}'_t}} \circ \overrightarrow{(\boldsymbol{h}_i^{x_t^{i}})^{\boldsymbol{b}'_t}}, \boldsymbol{H} \right) e\left(u^{\langle \boldsymbol{a}'_t, \boldsymbol{b}'_t\rangle}, U \right) \tag{1}$$

where $\bigcirc_{j-i=s} \boldsymbol{v}[i,j] = \boldsymbol{v}_s \in \mathbb{G}_1^n$, $\bigcirc_{j-i=s} w[i,j] = w_s \in \mathbb{G}_1$.[2]

The left-hand side (LHS) of Eq. (1) has exponentiation of x_t, and its degree takes even integers between $-4n+2$ and $4n-2$. Our $4n+1$ distinct challenges x_t determine P. Then, the extractor can compute $\boldsymbol{v}_P, w_P$ such that $P = \boldsymbol{E}(\boldsymbol{v}_P, \boldsymbol{H}) e(w_P, U)$. By q-pairing assumption whose instance is the CRS

[2] Once $\boldsymbol{v}_s$ and $\boldsymbol{w}_s$ are constructed, the extractor extracts the witness using them. In the extract process, the extractor does not decompose $\boldsymbol{v}_s$ to multi-$\boldsymbol{v}[i,j]$. That is, it does not affect soundness to substitute sending $\boldsymbol{v} \in \mathbb{G}_1^{n\times 2d(2d-1)}$ with $\bar{\boldsymbol{v}} \in \mathbb{G}_1^{4d-2}$ in Sec. 3.3.

$\boldsymbol{H} \parallel U$ on $\mathbb{G}_2$, we can separate the $\boldsymbol{H}$ and U terms. Then, we obtain two equations:

$$\boxed{\boldsymbol{H}\text{ correspondence}} \;:\; \boldsymbol{v}_P \circ \Big(\bigcirc_{s\in J_d} \boldsymbol{v}_s^{x_t^s} \Big) = \bigcirc_{i\in I_d} \overrightarrow{\big(\boldsymbol{g}_i^{x_t^{-i}}\big)^{\boldsymbol{a}_t'}} \circ \overrightarrow{\big(\boldsymbol{h}_i^{x_t^i}\big)^{\boldsymbol{b}_t'}} \tag{2}$$

$$\boxed{U\text{ correspondence}} \;:\; w_P \cdot \prod_{s\in J_d} w_s^{x_t^s} = u^{\langle \boldsymbol{a}_t', \boldsymbol{b}_t'\rangle} \tag{3}$$

for all $t \in [12d-5]$.

The extractor knows all the exponents $x_t^{j-i}, x_t^{-i}, x_t^{j}, \boldsymbol{a}_t'$, and $\boldsymbol{b}_t'$ in Eq. (2) from $4d-2$ distinct challenges. There are $4d-1$ distinct powers of x_t^2 in the LHS in Eq. (2). Thus, by using the inverse matrix of M and elementary linear algebra in the public exponents of the first $4d-1$ equalities in Eq. (2), the extractor can find the exponent matrices $\{\boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r}\}_{r\in I_d}$ and $\{\boldsymbol{a}_{s,r}, \boldsymbol{b}_{s,r}\}_{r\in I_d}$ for $s \in J_d$ satisfying

$$\boldsymbol{v}_P = \bigcirc_{r\in I_d} \overrightarrow{\boldsymbol{g}_r^{\boldsymbol{a}_{P,r}}} \circ \overrightarrow{\boldsymbol{h}_r^{\boldsymbol{b}_{P,r}}}, \quad \boldsymbol{v}_s = \bigcirc_{r\in I_d} \overrightarrow{\boldsymbol{g}_r^{\boldsymbol{a}_{s,r}}} \circ \overrightarrow{\boldsymbol{h}_r^{\boldsymbol{b}_{s,r}}} \tag{4}$$

We claim that concatenation of submatrices $\boldsymbol{a}_{P,r}, \boldsymbol{b}_{P,r} \in \mathbb{Z}_p^{m'\times n}$ are valid witnesses.

Combine Eq. (4) with Eq. (2):

$$\begin{aligned}
\boldsymbol{v}_P \circ \Big(\bigcirc_{s\in J_d} \boldsymbol{v}_s^{x_t^s} \Big) &= \bigcirc_{r\in I_d} \overrightarrow{\boldsymbol{g}_r^{\boldsymbol{a}_{P,r}}} \circ \overrightarrow{\boldsymbol{h}_r^{\boldsymbol{b}_{P,r}}} \circ \Big(\bigcirc_{s\in J_d} \overrightarrow{\boldsymbol{g}_r^{\boldsymbol{a}_{s,r}}} \circ \overrightarrow{\boldsymbol{h}_r^{\boldsymbol{b}_{s,r}}} \Big)^{x_t^s} \\
&= \bigcirc_{r\in I_d} \overrightarrow{\boldsymbol{g}_r^{\boldsymbol{a}_{P,r}+\sum_{s\in J_d} \boldsymbol{a}_{s,r} x_t^s}} \circ \overrightarrow{\boldsymbol{h}_r^{\boldsymbol{b}_{P,r}+\sum_{s\in J_d} \boldsymbol{b}_{s,r} x_t^s}} \\
&= \bigcirc_{r\in I_d} \overrightarrow{\boldsymbol{g}_r^{\boldsymbol{a}_t' x_t^{-r}}} \circ \overrightarrow{\boldsymbol{h}_r^{\boldsymbol{b}_t' x_t^{r}}}
\end{aligned} \tag{5}$$

By discrete logarithm relation assumption, we can separate exponents. For all $t \in [12d-5]$ and $r \in I_d$, we obtain

$$\boxed{\boldsymbol{g}_r\text{ exponentiation}} \;:\; \boldsymbol{a}_{P,r} + \sum_{s\in J_d} \boldsymbol{a}_{s,r} x_t^s = \boldsymbol{a}_t' x_t^{-r} \tag{6}$$

$$\boxed{\boldsymbol{h}_r\text{ exponentiation}} \;:\; \boldsymbol{b}_{P,r} + \sum_{s\in J_d} \boldsymbol{b}_{s,r} x_t^s = \boldsymbol{b}_t' x_t^{r} \tag{7}$$

Let both Eq. (6) and Eq. (7) be multiplied by x_t^r and x_t^{-r} respectively. Then, both equations have degrees of x_t range between $6d-3$ and $-6d+3$ according to $r \in I_d$ and $s \in J_d$, and it holds for all $t \in [12d-5]$. $12d-5$ distinct challenges $\{x_t\}$ determine polynomials $f, g : \mathbb{Z}_p \to \mathbb{Z}_p^{m\times n}$ satisfying the following equations:

$$\boldsymbol{a}_{P,r} X^r + \sum_{s\in J_d} \boldsymbol{a}_{s,r} X^{s+r} = f(X), \quad \boldsymbol{b}_{P,r} X^{-r} + \sum_{s\in J_d} \boldsymbol{b}_{s,r} X^{s-r} = g(X) \tag{8}$$

for all $r \in I_d$. Notice that the RHSs of Eq. (8) do not depend on the choice of r. Since the possible value of r is between $-2d+1$ and $r = 2d-1$, the polynomials $f(X)$ and $g(X)$ take degrees between $-2d+1$ and $2d-1$. Then, we obtain the following equations:

$$\boldsymbol{a}'_t = \sum_{r \in I_d} \boldsymbol{a}_{P,r} x_t^r, \quad \boldsymbol{b}'_t = \sum_{r \in I_d} \boldsymbol{b}_{P,r} x_t^{-r} \tag{9}$$

In a similar way to obtain exponent vectors $\boldsymbol{a}_{P,r}$ $\boldsymbol{b}_{P,r}$, the extractor can obtain exponents $c_P, c_s \in \mathbb{Z}_p$ such that $w_P = u^{c_P}$ and $w_s = u^{c_s}$. In the RHS in Eq. (2), let us put the results of Eq. (9). Then, we obtain the following equation:

$$u^{c_P} \cdot \prod_{s \in J_d} u^{c_s x_t^s} = u^{\langle \boldsymbol{a}'_t, \boldsymbol{b}'_t \rangle} = \prod_{i,j \in I_d} u^{\sum_{i,j \in I_d} \langle \boldsymbol{a}_{P,j}, \boldsymbol{b}_{P,i} \rangle x_t^{j-i}} \tag{10}$$

The exponents equation $c_P + \sum_{s \in J_d} c_s x_t^s = \sum_{i,j \in I_d} \langle \boldsymbol{a}_{P,j}, \boldsymbol{b}_{P,i} \rangle x_t^{j-i}$ holds by DLR assumption. The $8n-3$ distinct values determine the coefficient of the equation. Therefore, the emulator extracts valid witness $\boldsymbol{a}_P, \boldsymbol{b}_P$, which satisfies $c_P = \sum_{i \in I_d} \langle \boldsymbol{a}_{P,i}, \boldsymbol{b}_{P,i} \rangle = \langle \boldsymbol{a}_P, \boldsymbol{b}_P \rangle$. □