IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 3, 2022, accepted April 8, 2022, date of publication April 18, 2022, date of current version April 26, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3167806

Bulletproofs+: Shorter Proofs for a Privacy-Enhanced Distributed Ledger

HEEWON CHUNG[1], KYOOHYUNG HAN[2], CHANYANG JU[3], MYUNGSUN KIM[4], AND JAE HONG SEO[3]

[1]Desilo Inc., Seoul 06100, Republic of Korea
[2]Samsung SDS Research and Development Center, Seoul 06765, Republic of Korea
[3]Department of Mathematics & Research Institute for Natural Sciences, Hanyang University, Seoul 04763, Republic of Korea
[4]Department of Mathematics, Gachon University, Seongnam 13120, Republic of Korea

Corresponding author: Jae Hong Seo (jaehongseo@hanyang.ac.kr)

The work of Jae Hong Seo and Chanyang Ju was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) (A Study on Cryptographic Primitives for Succinct Non-interactive Argument of Knowledge (SNARK), 50%) under Grant 20210007270012002; and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT), 50%, under Grant 2020R1C1C1A0100696812. The work of Myungsun Kim was supported by the IITP Grant funded by the Korea Government (MSIT) (A Study on Cryptographic Primitives for SNARK) under Grant 2021-0-00727.

ABSTRACT This paper presents a new short zero-knowledge argument for the range proof and arithmetic circuits without a trusted setup. In particular, it can achieve the shortest proof size of the proof system categories without a trusted setup. More specifically, when proving that a committed value is a positive integer less than 64 bits, except for negligible error in the 128-bit security parameter, the proof size is 576 bytes long, which is 85.7% the size of the previous shortest proof due to Bünz *et al.* (Bulletproofs, IEEE Security and Privacy 2018). Similarly, circuit satisfiability can be proven with less communication overhead. Nevertheless, computational overheads in both proof generation and verification are comparable with those of Bulletproofs. Bulletproofs is established as one of the important privacy-enhancing technologies for a distributed ledger due to its trustless feature and short proof size. In particular, it has been implemented and optimized in various programming languages for practical usage by independent entities since it was proposed. The essence of Bulletproofs is based on the logarithmic inner product argument with no zero-knowledge. This paper revisits Bulletproofs from the viewpoint of the first sublinear zero-knowledge argument for linear algebra due to Groth (CRYPTO 2009) and then propose Bulletproofs+, an improved variety of Bulletproofs. The main component is the *zero-knowledge weighted inner product argument (zk-WIP)* which enables to reduce both the range proof and the arithmetic circuit proof. It already has zero-knowledge properties, there is no additional information when reducing zk-WIP, and it incurs a minimal transmission cost during the reduction process. Note that zk-WIP has all characteristics of the inner product argument, such as an aggregating range proof and batch verification; thus, Bulletproofs+ is superior to Bulletproofs in all aspects.

INDEX TERMS Zero-knowledge proofs, range proofs, arithmetic circuit.

I. INTRODUCTION

Adistributed ledger is a database that is consensually shared and synchronized across multiple nodes without a trusted administrator. The blockchain is one type of distributed ledger, where the database consists of linked blocks, called chains, and cryptocurrency, such as Bitcoin [1], is a representative application of the blockchain. The benefit of a distributed ledger is that it is immutable and that any independent observer can verify its validity without the aid of a trusted third party. The transparency of the natural realization of distributed ledgers often causes a data privacy issue since all information is public. For instance, all transaction details, including the sender, the receiver, and the amount transferred, are public in Bitcoin.

Noninteractive zero-knowledge proofs (NIZKs) enable the data owner to generate proof to convince observers of the validity of the data without disclosing it. Range proofs are

The associate editor coordinating the review of this manuscript and approving it for publication was SK Hafizul Islam.

special NIZKs for membership in a predetermined interval. That is, the prover first commits to a value using a commitment scheme and then proves that a committed value lies in a given interval. The range proof has broad applications that include blockchain-based cryptocurrencies in particular. For example, using a range proof, each transaction can be confidentially transferred without disclosing the amount transferred by including only the zero-knowledge proof validity of the transaction [2].

Due to its distributed and transparent nature, a short NIZK without a trusted setup is highly desired in the context of a distributed ledger. Bünz *et al.* [3] proposed a short NIZK without a trusted setup, called Bulletproofs, on the basis of the techniques of Bootle *et al.* [4]. Bulletproofs provides the shortest proof size, which is indeed incomparably shorter than the other range proof systems when a trusted setup is undesired. In fact, [4] achieved the first logarithmic communication complexity, and Bulletproofs improve Bootle *et al.*'s protocol so that the proof size is reduced by a factor of 3 and the protocol is suitable for proving statements on committed values. Hoffmann, Klooß, and Rupp [5] improved Bulletproofs to efficiently cover more expressive relations than rank 1 constraint systems. Even though [5] presents a more generic approach than Bulletproofs, they failed to reduce the proof size in specific arguments such as range proofs. Recently, Bünz, Fisch, and Szepieniec [6] devised a novel polynomial commitment scheme based on the class group and proposed the first succinct NIZK without trusted setup, called Supersonic, on the basis of their polynomial commitment scheme. Although Supersonic has strengths in both low verification costs and a short proof size, its minimum proof size is at least $\times$ 6 that of Bulletproofs for the 128-bit security level, and the gap becomes larger when increasing the security level. Bulletproofs are established as an important privacy-enhancing technology for distributed ledgers due to its trustless features and short proof size. In particular, it has been implemented and optimized in various programming languages for practical usage by independent entities: Java [7], C [8], C++ [9], Rust [10], [11], Go [12], Haskell [13], among others.

This paper presents, Bulletproofs+, an improved variety of Bulletproofs with a shorter proof size. That is, Bulletproofs+ achieves the shortest proof size in the NIZK category without a trusted setup. We compare the proof size of the range proof protocol of Bulletproofs+ with that of Bulletproofs in Table 1 for typical data types on a scale from 8 bits to 64 bits. The proof size of our range proof is $\times 0.8 \sim 0.857$ of that of Bulletproofs. Note that computational overheads in both proof generation and proof verification in Bulletproofs+ are comparable to those of Bulletproofs. To achieve a shorter proof size, we revisits Bulletproofs from the viewpoint of the first sublinear zero-knowledge argument for linear algebra due to Groth [14]. Bulletproofs employ the inner product argument without zero-knowledge as the essential ingredient. In [14] and Bulletproofs+, the main ingredient is the *zero-knowledge weighted inner product argument (zk-WIP),* which enables to reduce both the range proofs and the arithmetic circuit proofs. Briefly, Bulletproofs mask committed values with random numbers to give a zero-knowledge property to the inner product argument and to reduce to the inner product argument, committed values become random polynomials of degree 1 and it makes 5 elements for reduction. In contrast, zk-WIP already has zero-knowledge properties, our protocol needs no random number anymore, and it enables the reduction to zk-WIP by only one element. Therefore, the benefit of reducing zk-WIP is the minimal transmission cost during the reduction process, which makes the overall proof size of Bulletproofs+ smaller than that of Bulletproofs. Furthermore, like Bulletproofs, Bulletproofs+ has additional extensions, such as aggregating range proofs and batch verification.

A. OUR APPROACH

1) WHY WEIGHTED INNER PRODUCT?

The inner product argument based on a homomorphic commitment scheme such as a generalization of the Pedersen commitment [14], [15] is employed as a core building block for more complicated relations such as linear algebra equations, range relations, and circuit satisfiability [3], [4], [14]. More precisely, Groth [14] proposed efficient reductions from the advanced arguments to the inner product argument, and Bootle *et al.* [4] and Bünz *et al.* [3] improved Groth's result in terms of the communication overhead by imposing more interactions between the prover and the verifier. However, this is not a major burden in the random oracle model [16] since this approach can be converted into the noninteractive argument through the Fiat-Shamir heuristic [17] in the random oracle model.

In fact, when Groth proposed the reduction from the advanced argument for linear algebra equations, he used the weighted inner product (WIP) argument as well as the inner product argument as ingredient protocols. For a constant vector $\boldsymbol{c} \in \mathbb{Z}_p^n$, the WIP with respect to $\boldsymbol{c}$, denoted by $\odot_{\boldsymbol{c}}$, is defined as

$$\begin{aligned} \odot_{\boldsymbol{c}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^n &\to \mathbb{Z}_p \\ (\boldsymbol{a}, \boldsymbol{b}) &\mapsto \langle \boldsymbol{a}, (\boldsymbol{c} \circ \boldsymbol{b}) \rangle, \end{aligned}$$

where $\langle\ \rangle$ denotes the standard inner product and $\circ$ denotes the componentwise product (a.k.a. the Hadamard product).

At the heart of the reductions to the WIP argument is the batch processing of several equations so that the communication overhead is reduced. For example, the Hadamard product equation between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, denoted by $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \in \mathbb{Z}_p^n$, is a set of n equations, and this equation can be converted by imposing a random integer y into the equation

$$\langle \boldsymbol{a}, \left((y, y^2, \ldots, y^n) \circ \boldsymbol{b}\right) \rangle = \langle \boldsymbol{c}, (y, y^2, \ldots, y^n) \rangle \in \mathbb{Z}_p. \quad (1)$$

The prover can thus convince the verifier of the original Hadamard product equation $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{c} \in \mathbb{Z}_p^n$ by convincing (1) for randomly chosen y. Both sides of (1) can be directly considered as the WIP with respect to the coefficient vector

TABLE 1. Logarithmic zero-knowledge range proofs.

Data Size	Proof Size (bytes) [3]	Proof Size (bytes) Ours	Proof Size (bytes) ratio*	Applicable Data Types
8 bits	480	384	0.800	age
32 bits	608	512	0.842	position, zip code
64 bits	672	576	0.857	balance, transaction amount

* Bulletproofs+/Bulletproofs ratio

$(y, y^2, \ldots, y^n)$. Therefore, an efficient proof protocol for the WIP is necessary for this approach.

2) LOGARITHMIC ZERO-KNOWLEDGE ARGUMENT FOR WEIGHTED INNER PRODUCT

Groth [14] proposed zk-WIP with linear communication overhead, which is an ingredient protocol for more advanced arguments for linear algebra equations. Subsequent works [3], [4] employed an inner product argument *without zero-knowledge* as an ingredient protocol, but zero-knowledgeness for advanced relations such as circuit satisfiability is achieved by the reduction to the inner product argument. Wahby *et al.* [18] presented a logarithmic zero-knowledge argument for the inner product between a hidden vector and a public vector, which is different from the (weighted) inner product between two hidden vectors in [14]. Hoffmann, Klooß, and Rupp [5] proposed a zero-knowledge argument for the inner product between two hidden vectors satisfying certain constraints, which was called the *almost* zero-knowledge proof protocol by the authors. To blind witness vectors, they used random vectors depending on the witness, which constrains the witness to some degree. To our knowledge, there is no specific construction for a logarithmic WIP proof protocol with full zero-knowledge, where both WIP input vectors are perfectly hidden.

The starting point of this work is logarithmic inner product arguments whose the main component is the following equality as well as the bilinearity of the inner product. For the sake of simplicity, let n be an even number $n = 2\widehat{n}$ for some integers $\widehat{n}$ and $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2), \boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2) \in \mathbb{Z}_p^{\widehat{n}} \times \mathbb{Z}_p^{\widehat{n}}$. Then, one can obtain

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \langle \boldsymbol{a}_1, \boldsymbol{b}_1 \rangle + \langle \boldsymbol{a}_2, \boldsymbol{b}_2 \rangle. \tag{2}$$

That is, an inner product can be represented by the sum of two *half-length* inner products. This property is essential for reduction to a half-length inner product, which leads to logarithmic communications. The WIP is also a bilinear map and satisfies a similar property to (2) when $\boldsymbol{c}$ is the Vandermonde vector, e.g., $\boldsymbol{c} = (y, \ldots, y^n) \in \mathbb{Z}_p^n$.

$$\boldsymbol{a} \odot_{(y,\ldots,y^n)} \boldsymbol{b} = \boldsymbol{a}_1 \odot_{(y,\ldots,y^{\widehat{n}})} \boldsymbol{b}_1 + (y^{\widehat{n}} \cdot \boldsymbol{a}_2) \odot_{(y,\ldots,y^{\widehat{n}})} \boldsymbol{b}_2. \tag{3}$$

Let us give an intuition for a logarithmic WIP argument w.r.t. $(y, \ldots, y^n) \in \mathbb{Z}_p^n$. Assume that the prover commits to vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{Z}_p^{\widehat{n}}$ and integers $c_L = \boldsymbol{a}_1 \odot_{(y,\ldots,y^{\widehat{n}})} \boldsymbol{b}_2$, $c = \boldsymbol{a} \odot_{(y,\ldots,y^n)} \boldsymbol{b}$, $c_R = (y^{\widehat{n}} \cdot \boldsymbol{a}_2) \odot_{(y,\ldots,y^{\widehat{n}})} \boldsymbol{b}_1 \in \mathbb{Z}_p$ and aims to convince the verifier of the relation $c = \boldsymbol{a} \odot_{(y,\ldots,y^n)} \boldsymbol{b}$. The bilinearity of the WIP and (3) guarantee that the following equation holds for a random challenge e.

$$\begin{aligned}&(e\boldsymbol{a}_1 + e^{-1}y^{\widehat{n}}\boldsymbol{a}_2) \odot_{(y,\ldots,y^{\widehat{n}})} (e\boldsymbol{b}_2 + e^{-1}\boldsymbol{b}_1) \\ &= e^2\boldsymbol{a}_1 \odot_{(y,\ldots,y^{\widehat{n}})} \boldsymbol{b}_2 + \boldsymbol{a} \odot_{(y,\ldots,y^n)} \boldsymbol{b} + e^{-2}(y^{\widehat{n}}\boldsymbol{a}_2) \odot_{(y,\ldots,y^{\widehat{n}})} \boldsymbol{b}_1\end{aligned} \tag{4}$$

In our protocol, the verifier can calculate the commitments to $(e\boldsymbol{a}_1 + e^{-1}y^{\widehat{n}}\boldsymbol{a}_2)$ and $(e\boldsymbol{b}_2 + e^{-1}\boldsymbol{b}_1)$ and inputs of the WIP of the left-hand side in (4) with the aid of the prover. Let $\widehat{c} := e^2c_L + c + e^{-2}c_R$. Then, the commitment to $\widehat{c}$ can be publicly calculated using the homomorphic property of an underlying commitment scheme, and this calculation can be used as the result of the WIP when taking $(e\boldsymbol{a}_1 + e^{-1}y^{\widehat{n}}\boldsymbol{a}_2)$ and $(e\boldsymbol{b}_2 + e^{-1}\boldsymbol{b}_1)$ as input. Thus, the equality between (4) and $\widehat{c}$ for randomly chosen e guarantees the equality between each coefficient of a power of e of the right-hand side in (4) and that of $\widehat{c}$, so that we have $c = \boldsymbol{a} \odot_{(y,\ldots,y^n)} \boldsymbol{b}$. Therefore, a WIP proof w.r.t. $(y, \ldots, y^n)$ between n-dimensional vectors is reduced to a WIP proof w.r.t $(y, \ldots, y^{\widehat{n}})$ between $\widehat{n}$-dimensional vectors.

The commitment to the hidden vector of length n, which is the input of the reduction, and each transmission sent by the prover during the reduction is blinded by random group elements chosen by the prover so that the witness is perfectly hidden from the viewpoint of the verifier. Using the discrete logarithms of such random group elements, the prover updates the blinding factor in the commitment to the new hidden vector of length $\widehat{n}$, which is the output of the reduction.

Constant communication is sufficient for each reduction step, and a number of rounds of $O(\log_2(n))$ is sufficient for reducing to a dimension 1 WIP proof protocol. For the final step of the zk-WIP proof protocol, we devise a variant of the Schnorr protocol, which requires 2 group elements and 3 field elements. Therefore, the proposed zk-WIP protocol requires communication of $2\log_2(n) + 5$ field or group elements.

3) ONE ROUND REDUCTION FOR BULLETPROOFS-LIKE PROTOCOLS

Bünz *et al.* proposed a short zero-knowledge argument called Bulletproofs, which includes an aggregate range proof protocol with logarithmic size in the witness size and an arithmetic circuit proof protocol with logarithmic size in the circuit size. Both aggregate range proof and arithmetic circuit proof protocols are built on their inner product proof protocol.

We show that when our zk-WIP proof protocol is used as an ingredient protocol, one commit-and-challenge round is sufficient to reduce from advanced protocols such as aggregate range proof and arithmetic circuit proof to the zk-WIP proof protocol. In particular, the prover sends only a group element in our reductions.

Let us explain the idea of the reduction for our single-range proof protocol. The prover's goal is to convince the verifier that witness v belongs to an interval $[0, 2^n - 1]$, equivalently, the prover knows a binary vector of the witness v of length n.

To show a binary vector, the prover begins by committing to $\boldsymbol{a}_L, \boldsymbol{a}_R \in \mathbb{Z}_p^n$ satsifying

$$\boldsymbol{a}_L - \boldsymbol{a}_R = \mathbf{1}^n \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \mathbf{0} \ \wedge \ \langle \boldsymbol{a}_L, \mathbf{2}^n \rangle = v, \tag{5}$$

where $\mathbf{1}^n = (1, \ldots, 1)$ is a vector filled with 1's in all entries and $\mathbf{2}^n = (1, 2, \ldots, 2^{n-1})$ is a vector consisting of powers of 2. If $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ has other than 0 and 1, (5) cannot be satisfied and the verifier can be convinced. To reduce communication overhead, the verifier sends a random challenge $y \in \mathbb{Z}_p$ to utilize WIP. Then, $2n+1$ equations in (5) are batched to a WIP equation. Therefore, we place each term of the left-hand sides of the equations in (5) into a distinct monomial coefficient with variables y and z as follows.

$$\begin{aligned}&\left(\boldsymbol{a}_L - \mathbf{1}^n \cdot z\right) \odot_{(y,\ldots,y^n)} \left(\boldsymbol{a}_R + \mathbf{2}^n \circ (y^n, y^{n-1}, \ldots, y) + \mathbf{1}^n \cdot z\right) \\ &\quad = \boldsymbol{a}_L \odot_{(y,\ldots,y^n)} \boldsymbol{a}_R + y^{n+1} \langle \boldsymbol{a}_L, \mathbf{2}^n \rangle \\ &\qquad + z \cdot (\boldsymbol{a}_L - \boldsymbol{a}_R) \odot_{(y,\ldots,y^n)} \mathbf{1}^n - \zeta(y, z) \in \mathbb{Z}_p, \end{aligned} \tag{6}$$

where $\zeta(y, z) = y^{n+1} z \langle \mathbf{1}^n, \mathbf{2}^n \rangle + z^2 \langle \mathbf{1}^n, \overrightarrow{y}^n \rangle$ and $\overrightarrow{y}^n$ indicates $(y, \ldots, y^n)$. Each term of the right-hand sides in (5) is either constant or the witness v and appears as a distinct monomial coefficient with variables y and z in

$$0 + y^{n+1} v + z \mathbf{1}^n \odot_{(y,\ldots,y^n)} \mathbf{1}^n - \zeta(y, z). \tag{7}$$

Therefore, the remaining part of our range proof protocol is to run the zk-WIP protocol w.r.t. $(y, \ldots, y^n)$ that convinces

$$\begin{aligned}&\left(\boldsymbol{a}_L - \mathbf{1}^n \cdot z\right) \odot_{(y,\ldots,y^n)} \left(\boldsymbol{a}_R + \mathbf{2}^n \circ (y^n, \ldots, y) + \mathbf{1}^n \cdot z\right) \\ &\qquad = y^{n+1} v + z \mathbf{1}^n \odot_{(y,\ldots,y^n)} \mathbf{1}^n - \zeta(y, z). \end{aligned} \tag{8}$$

By the homomorphic property of an underlying commitment scheme, the commitments to inputs and output of the WIP in (8) can be publicly calculated from public parameters and the commitment sent by the prover at the beginning of our range protocol. Therefore, both the prover and the verifier can run the zk-WIP protocol. Similarly, aggregate range proof and arithmetic circuit proof protocols can be reduced to the zk-WIP proof protocol through one commit-and-challenge round.

B. APPLICATIONS

1) BLOCKCHAIN: CONFIDENTIAL TRANSACTIONS, SMART CONTRACTS, AND MORE

Although Bitcoin [1] supports pseudonymity, it does not guarantee perfect privacy [19], [20]. To address the confidentiality issue, Maxwell [2] proposed the concept of the confidential transaction, where every piece of information except validity is hidden, in the UTXO model. Here, the UTXO is an unspent transaction output, and the UTXO model indicates that each transaction should fully spend the outputs of previously unspent transactions. A confidential transaction consists of commitments to a set of inputs and a set of outputs with a Pedersen commitment scheme [15]. Although the homomorphic property of the Pedersen commitment enables the verifier to check whether the sum of inputs is equal to the sum of outputs, the verifier cannot verify whether a sender has enough balance to involved amounts, and thus a sender should provide additional evidence for this. The range proof exactly resolves this problem and is thus essential in confidential transactions.

Monero [21], [22] is a well-known privacy-enhanced blockchain project that employs confidential transactions in the UTXO model. Each transaction in the UTXO model has 2.5 outputs on average. The range proof should be attached for each transaction output in Monero so that on average, 2.5 range proofs are required for each transaction. The size of each transaction with two outputs was reduced from 13 kB to 2.5 kB since Bulletproofs for aggregate range proof were integrated with Monero in 2018. Bulletproofs+ for aggregate range proof is 96 bytes smaller than Bulletproofs in 128-bit security so that when applying Bulletproofs+ instead of Bulletproofs to Monero, two output transactions were further reduced and finally obtained 2.4 kB. Therefore, Bulletproofs+ can save more than 1 MB every day. Beside Monero, a similar effect can be obtained from the other privacy cryptocurrencies such as QusQuis due to Fauzi *et al.* [23] Compared to Monero, QuisQuis makes UTXO sets nonmonotomically growing by introducing a new notion called updatable public keys; however, Bulletproofs still play an essential role in QuisQuis. Thus, Bulletproofs+ can also affect QuisQuis by reducing the transaction size.

Mimblewimble [24]–[26] aimed to resolve privacy and scalability problems in Bitcoin. In the UTXO model, the sums of input transaction values and output transaction values should be the same, apart from a transaction fee, and anyone can obtain a commitment to 0 from the valid transaction. Then, a sender signs a transaction under the commitment to 0 (as the public key), which implies that no money vanished and none was created. Through this, they simplify the structure of a confidential transaction; however, they still require the sender's balance check for the validity of the transaction, and thus, a range proof is indispensable. Grin [27] and Beam [28] are major implementations of Mimblewimble. For a million blocks, 10 million transactions (2 inputs, 2.5 outputs average) and 100,000 unspent outputs, the UTXO size is nearly 520 MB, and among them, almost 517 MB is allocated to the range proofs [27]. UTXO size can be significantly reduced to approximately 90 MB (100 MB) with Bulletproofs+ (Bulletproofs).

There are several attempts to employ range proofs in smart contracts. A confidential transaction is first proposed based on the UTXO model; however, a smart contract platform usually takes an account-based model. To construct a confidential transaction for the smart contract, it should support not only range proofs but also statements on algebraically encoded values to execute arbitrary smart contracts securely. Zether [29] suggests a confidential transaction compatible with a smart contract platform, especially Ethereum [30], called confidential transfers. Additionally, Findora [31] is one of the projects employing Bulletproofs on a smart contract. The main feature of [31] is supporting audits on a confidential transaction, and it enables us to prove more nuanced

statements with selective disclosure. Both [29] and [31] support confidential asset transfer, and range proofs and arithmetic circuit proofs are necessary. Thus, Bulletproofs+ can also enhance the efficiency of the account-based model.

2) RANGE PROOFS

Range proof is an essential tool for resolving privacy issues in digital financial technology, including distributed ledgers, e.g., [32]. Banks perform the process of identifying and verifying the identity of the client when opening an account. Due to regulations such as laundering antimoney and knowing your customer, this process becomes mandatory and causes privacy issues. The zero-knowledge proofs enable this process to be performed without disclosing the customer's private information. Using the range proofs, the client can convince the banks of some relations on the age, zip codes, and GPS position without disclosing the actual information. For instance, the client can prove that the customer's age is over the legal age and that the zip codes and the GPS information are contained in specific ranges to validate the location where the customer stays.

3) VERIFIABLE SHUFFLES

Bulletproofs+ for the arithmetic circuit can be employed to reduce the proof size of applications beyond distributed ledgers. For example, it can be applied to the verifiable shuffle [3], [33]–[36] that takes a list of committed values as input and outputs a permuted list along with the proof of correctness of the permuted list. Although the verifiable shuffle is an important stand-alone protocol, it is also a good building block for many other applications, such as e-voting protocols [33], [37], mix-net [38], privacy-preserving advertisement delivery [39], and solvency proofs [40]. In terms of the proof size, Bulletproofs have the most efficient scheme that increases the proof size logarithmically in the size of the input list. The shuffle can be implemented either by the sorting circuit using $O(n \log_2(n))$ multiplications, where n is the size of the input list [3], or by the permutation circuit and the multiexponentiation circuits in [36].[1] Although Bulletproofs+ reduces only constant term (e.g., 96 bytes for the 128-bit security), for practically large n (e.g., $n < 2^{32}$), the improvement of Bulletproofs+ makes a meaningful difference similar to the range proof case due to logarithmic increasing speed of Bulletproofs' proof size in n.

C. RELATED WORK

1) RANGE PROOFS

Brickell *et al.* [41] first proposed range proofs, and since then, they have received great attention [42]–[49]. Lipmaa [44] presented a range proof protocol that relies on Lagrange's four-square theorem (a.k.a., Bachet's conjecture), which states that any positive integer can be written as a sum of four squares. Groth [50] improved Lipmaa's suggestion by exploiting Legendre's three-square theorem, which states that a positive integer α can be written as a sum of three squares if and only if α is not of the form $4^{n_1}(8n_2 + 7)$ $n_1, n_2 \in \mathbb{Z}$. More recently, Couteau *et al.* [49] suggested a range proof solution based on a weaker assumption than the strong RSA assumption [51].

2) NIZK FOR ARITHMETIC CIRCUITS

In recent years, many improvements in NIZK for circuit satisfiability have been made [52]–[54]. SNARKs are arguments of knowledge that have succinct proof and efficient verifiers. Although SNARKs provide high performance that can meet practical requirements, they inherently and inevitably require a trusted setup to generate the structured reference string (SRS). To address this problem, Groth *et al.* [55] and many subsequent works [**?**], [56]–[58] proposed proof systems relying on the SRS, where SRS is efficiently updatable. Nevertheless, these proof systems with the updatable SRS still require at least one trusted setup at the beginning of the proof system.

3) CONCURRENT WORK

There are three independent and concurrent works that improve Bulletproofs [59]–[61]. Boneh, Fisch, Gabizon and Williamson [59] proposed a simple range proof from a polynomial commitment scheme. To prove $0 \leq v < 2^n$ with the zero-knowledge property, the prover should transmit $2 \cdot \lceil \log_2(n+2) \rceil + 2$ elements in $\mathbb{G}$ and 5 elements in $\mathbb{Z}_p$. This communication certainly costs less than Bulletproofs; however, it still requires at least one more element than our range proof. Moreover, since the construction is based on a polynomial commitment scheme that needs to choose a prime p larger than n, a prover can only claim the same interval once a polynomial commitment scheme is determined. In contrast, our range proof scheme supports an arbitrary n; thus, there is no restriction for the prover.

Attema and Cramer [60] focused on reconciling Bulletproofs with the theory of Σ-protocols. A prover needs to prove quadratic equations for a range proof; however, Σ-protocols are appropriate for proving arbitrary linear relations, and thus, Bulletproofs require reinvention with the quadratic constraint, which may cause some technical difficulties. To resolve this issue, the authors employed an arithmetic secret sharing-based technique that enables the linearization of all nonlinear statements while preserving the same communication reduction. More precisely, a communication cost for the range proof is $2 \cdot \lceil \log_2(2n+3) \rceil$ elements in $\mathbb{G}$ and 5 elements in $\mathbb{Z}_p$; thus, Bulletproofs+ still remains the transparent range proof with the smallest proof size.

Couteau, Klooß, Lin, and Reichle [61] proposed a new range proof with transparent setup from bounded integer commitments. One of the classical approaches for range proofs is based on square decomposition and to merge it, they suggest a method for transforming a commitment scheme over a finite field to a commitment scheme that enables to commit a bounded integer and to prove relations efficiently.

[1] One can use Bulletproofs to design the permutation argument and the multiexponentiation argument in [36] to achieve the logarithmic proof size of verifiable shuffles.

As a result, they can propose several new instantiations of range proof paradigms. More specifically, they suggest a range proof scheme based on discrete logarithm problem, lattice problem and standard class group assumptions. In lattice-based range proofs, they can improve over the state of the art in a batch setting when at least a few dozen range proofs are required. Under the standard class group assumptions, they can propose the first concrete efficient commitment scheme, which does not require any trusted setup. Compared to Bulletproofs, they can also reduce the proof size about 15% (see Table 1 of [61]) and it varies from about 40 bytes to 120 bytes depending on the parameter. However, the proof size for our scheme is always less than 96 bytes than Bulletproofs for any parameter and our proof size is still less than that of [61] in some parameter settings. Moreover, they do not provide a scheme for arithmetic circuit, but Bulletproofs+ can prove the arithmetic circuit.

D. ORGANIZATION

We provide definitions of assumptions, homomorphic commitment schemes, and zero-knowledge arguments in Section II. In Section III, we present a main building block protocol, the zero-knowledge argument for the WIP without a trusted setup. We propose short zero-knowledge arguments for the range proof and the arbitrary arithmetic circuits in the following sections, Section IV and Section V, respectively. Finally, we provide the performance of the proposed protocols for the specific parameters in Section VI.

II. PRELIMINARIES

We begin by defining some basic notations to be used when defining the preliminary concepts in the following subsections. More specific notations that are useful for describing and analyzing the proposed proof systems are provided in Section II-C. For any algorithm A, $y = A(x; r)$ denotes that y is the output of A on input x with randomness r. When using uniform randomness, we use a shortened notation $y \leftarrow A(x)$, meaning that randomness r is chosen at random outside A, and we set $y = A(x; r)$. For any set S, $x \xleftarrow{\$} S$ denotes uniform random sampling of x from S. Throughout the paper, λ denotes the security parameter, and it is written in unary form when it is used as the algorithm input. For a function $f : \mathbb{N} \to [0, 1]$, f is negligible if $f(\lambda) = \lambda^{-\omega(1)}$ and f is overwhelming when $f(\lambda) = 1-\lambda^{-\omega(1)}$. Here, $negl(\lambda)$ denotes a negligible function.

A. HOMOMORPHIC COMMITMENTS

A (noninteractive) commitment scheme consists of two algorithms Gen and Com. Gen is called the key generation algorithm that takes the security parameter and outputs the commitment key ck. The message space M_{ck}, the randomness space R_{ck}, and the commitment space C_{ck} are specified in ck. The commitment algorithm Com combined with the commitment key ck specifies a commitment function $\mathsf{Com}_{\mathsf{ck}} : \mathsf{M}_{\mathsf{ck}} \times \mathsf{R}_{\mathsf{ck}} \to \mathsf{C}_{\mathsf{ck}}$ that takes $m \in \mathsf{M}_{\mathsf{ck}}$ and outputs a commitment $\mathsf{com} \in \mathsf{C}_{\mathsf{ck}}$ using randomness $r \in \mathsf{R}_{\mathsf{ck}}$. To commit to a message $m \in \mathsf{M}_{\mathsf{ck}}$, the sender selects $r \xleftarrow{\$} \mathsf{R}$ and computes the commitment $\mathsf{com} = \mathsf{Com}_{\mathsf{ck}}(m; r)$. We define several properties of the commitment scheme.

Definition 1 (Homomorphic Commitments): A homomorphic commitment scheme is a (noninteractive) commitment scheme that has a homomorphic property such that

$$\mathsf{Com}(m_1; r_1) +_{\mathsf{C}_{\mathsf{ck}}} \mathsf{Com}(m_2; r_2) = \mathsf{Com}(m_1 +_{\mathsf{M}_{\mathsf{ck}}} m_2; r_1 +_{\mathsf{R}_{\mathsf{ck}}} r_2),$$

for all $m_1, m_2 \in \mathsf{M}_{\mathsf{ck}}$ and $r_1, r_2 \in \mathsf{R}_{\mathsf{ck}}$, where $+_{\mathsf{C}_{\mathsf{ck}}}$, $+_{\mathsf{M}_{\mathsf{ck}}}$ and $+_{R_{\mathsf{ck}}}$ define operations in C_{ck}, M_{ck} and R_{ck}, respectively.

Definition 2 (Hiding Commitments): A commitment scheme is hiding if for all nonuniform polynomial-time interactive adversaries $\mathcal{A}$ the following probability is smaller than or equal to $negl(\lambda)$ for some negligible function $negl(\lambda)$, as shown at the bottom of the page, where the probability goes over the randomness used in $\mathcal{A}$ and Gen and the choice of b and r. We say the scheme is perfectly hiding if $negl(\lambda) = 0$.

Definition 3 (Binding Commitments): A commitment scheme is binding if for all nonuniform polynomial-time interactive adversaries $\mathcal{A}$, the following probability is smaller than or equal to $negl(\lambda)$ for some negligible function $negl(\lambda)$.

$$\Pr\left[\begin{array}{c|c} \mathsf{Com}(m_0; r_0) = \mathsf{Com}(m_1, r_1) & \mathsf{ck} \leftarrow \mathsf{Gen}(1^\lambda); \\ \wedge m_0 \neq m_1 & (m_0, m_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right]$$

where the probability exceeds the randomness used in $\mathcal{A}$ and Gen. We say the commitment scheme is perfectly binding if $negl(\lambda) = 0$.

A generalized Pedersen commitment scheme is extensively used in this work. We explain here how the generalized Pedersen commitment was implemented. Let $\mathsf{M}_{\mathsf{ck}} = \mathbb{Z}_p^n$, $\mathsf{R}_{\mathsf{ck}} = \mathbb{Z}_p$ and $\mathsf{C}_{\mathsf{ck}} = \mathbb{G}$, where $ck = (\mathbb{G}, p, g, g_1, \ldots, g_n)$ and $g, g_i \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, n$. To commit to a message vector $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, one computes $\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{m}; r) := g^r \prod_{i=1}^n g_i^{m_i}$, where $r \xleftarrow{\$} \mathbb{Z}_p$. The generalized Pedersen commitment scheme is perfectly hiding since g is a generator of the cyclic group, and thus, the random blinding factor g^r is uniformly distributed over the cyclic group. If the discrete logarithm assumption holds on $\mathbb{G}$, then the Pedersen commitment scheme is computationally binding [14], [15]. An important fact is that the generalized Pedersen commitment is a homomorphic commitment, i.e., for all $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{Z}_p^n$

$$\left|\frac{1}{2} - \Pr\left[b' = b \;\middle|\; \begin{array}{c} \mathsf{ck} \leftarrow \mathsf{Gen}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}); b \xleftarrow{\$} \{0, 1\}; \\ r \xleftarrow{\$} \mathsf{R}_{\mathsf{ck}}; \mathsf{com} = \mathsf{Com}(m_b; r); b' \xleftarrow{\$} \mathcal{A}(\mathsf{com}) \end{array}\right]\right|$$

and $r, r' \in \mathbb{Z}_p$,

$$\mathsf{Com}_{\mathsf{ck}}(\boldsymbol{m}; r) \cdot \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{m}'; r') = \mathsf{Com}_{\mathsf{ck}}(\boldsymbol{m} + \boldsymbol{m}'; r + r')$$

holds.

B. ZERO-KNOWLEDGE ARGUMENTS OF KNOWLEDGE

We consider arguments consisting of three interactive probabilistic polynomial-time algorithms $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ in the *common random string model*. $\mathcal{K}$ is called the common reference string generator, which takes the security parameter 1^λ as input and outputs the common reference string σ. In this paper, the common reference string is a public key for the generalized Pedersen commitment scheme, that is, uniformly chosen group elements.[2] $\mathcal{P}$ and $\mathcal{V}$ are called the prover and the verifier, respectively. For the sake of simplicity, in this paper, we do not explicitly describe $\mathcal{K}$ but assume that the common reference string is given as common input to both $\mathcal{P}$ and $\mathcal{V}$. At the end of the interaction, the verifier $\mathcal{V}$ accepts (equivalently outputs 1) or rejects (equivalently outputs 0).

We prove that the proposed protocol, Bulletproofs+, is a zero-knowledge argument of knowledge. Informally, the goal of the prover in this protocol is to convince the verifier of witness knowledge that guarantees some statement holds, without disclosing the witness. Let $\mathcal{R} \subset \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be a polynomial time verifiable ternary relation. Given the common reference string σ, we call w a witness for a statement x if $(\sigma, x, w) \in \mathcal{R}$. We define a corresponding reference string-dependent language L_σ as the set of statements x that has a witness w such that $(\sigma, x, w) \in \mathcal{R}$. That is,

$$L_\sigma = \{\, x \mid \exists w \text{ such that } (\sigma, x, w) \in \mathcal{R} \,\}$$

and if $\sigma = \emptyset$, then this is the same as the standard notion of NP languages.

Definition 4 (Argument of Knowledge): The triple $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge *for relation $\mathcal{R}$ if it satisfies the completeness and witness-extended emulation as defined below.*

Definition 5 (Perfect Completeness): $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ has perfect completeness *if for all nonuniform polynomial-time interactive adversaries $\mathcal{A}$,*

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

Definition 6 (Computational Witness-Extended Emulation): We say that $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ has witness-extended emulation *if for all deterministic polynomial provers $\mathcal{P}^*$ if there exists an expected polynomial time emulator $\mathcal{E}$ such that for all nonuniform polynomial time interactive adversaries $\mathcal{A}$ there exists a negligible function $negl(\lambda)$ such that the gap between the following two probabilities is smaller than $negl(\lambda)$.*

$$\Pr\left[\mathcal{A}(tr) = 1 \;\middle|\; \begin{array}{c} \sigma \leftarrow \mathcal{K}(1^\lambda); (x, s) \leftarrow \mathcal{A}(\sigma); \\ tr \leftarrow \langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x)\rangle \end{array}\right] \text{ and}$$

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

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

In the definition of witness-extended emulation, the value s can be regarded as 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, so we call an argument $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ satisfying Definition 6 and Definition 5 an argument of knowledge (of witness w).

C. NOTATION

Let p denote a prime of length λ. In our protocol, we use several sets $\mathbb{G}, \mathbb{Z}_p, \mathbb{Z}_p^*, \mathbb{G}^n, \mathbb{Z}_p^n$ and several binary operations over them. Let $\mathbb{G}$ denote a group of orders p, $\mathbb{Z}_p$ denote the ring of integers modulo p, and $\mathbb{Z}_p^*$ denote $\mathbb{Z}_p \backslash \{0\}$. For a group $\mathbb{F}$, $\mathbb{F}^n$ denotes the n-dimensional product group, and hence, it also denotes vector spaces of dimension n over $\mathbb{F}$. $\mathbb{Z}_p^{n\times m}$ denotes the set of matrices with n rows and m columns over $\mathbb{Z}_p$. An element in Cartesian product set $\mathbb{F} \in \{\mathbb{G}^n, \mathbb{Z}_p^n, \mathbb{Z}_p^{n\times m}\}$ is denoted by bold letters, i.e.,. $\boldsymbol{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$, $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$, and $\boldsymbol{B} \in \mathbb{Z}_p^{n\times m}$. We often consider a vector $\boldsymbol{a}$ in $\mathbb{Z}_p^n$ as a row matrix in $\mathbb{Z}_p^{1\times n}$, and its transpose vector, which is the corresponding column vector, is denoted by $\boldsymbol{a}^\top$.

We define notations for several binary operators among the above defined sets. For two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, the inner product between $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as $\boldsymbol{a} \cdot \boldsymbol{b}^\top = \sum_{i=1}^n a_i \cdot b_i \in \mathbb{Z}_p$ and denoted by $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$. The componentwise multiplication (a.k.a., the Hadamard product) between $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted by $\boldsymbol{a} \circ \boldsymbol{b}$, i.e., $\boldsymbol{a} \circ \boldsymbol{b} = (a_1 \cdot b_1, \ldots, a_n \cdot b_n) \in \mathbb{Z}_p^n$. For $\boldsymbol{a} \in \mathbb{Z}_p^n$ and $\boldsymbol{g} \in \mathbb{G}^n$, the multiexponentiation $\prod_{i=1}^n g_i^{a_i} \in \mathbb{G}$ is denoted by $\boldsymbol{g}^{\boldsymbol{a}}$. For a scalar $c \in \mathbb{Z}_p$ and a vector $\boldsymbol{a} \in \mathbb{Z}_p^n$, the scalar multiplication is denoted by $c \cdot \boldsymbol{a} \in \mathbb{Z}_p^n$, i.e., $c \cdot \boldsymbol{a} = (c \cdot a_1, \ldots, c \cdot a_n)$.

For an integer $y \in \mathbb{Z}_p^*$, we use two vector notations $\overrightarrow{y}^n$ and $\overleftarrow{y}^n$ to denote $(y, y^2, \ldots, y^n)$ and $(y^n, y^{n-1}, \ldots, y^1)$, respectively. In addition, we use two constant vectors $(1, \ldots, 1), (1, 2, \ldots, 2^{n-1}) \in \mathbb{Z}_p^n$, denoted by $\mathbf{1}^n$ and $\mathbf{2}^n$, respectively. Then, the following equality holds.

$$\overrightarrow{y}^n \circ \overleftarrow{y}^n = y^{n+1} \cdot \mathbf{1}^n \tag{9}$$

For a ternary relation $\mathcal{R}$, we use the format {(Public Input; Witness) : $\mathcal{R}$} to denote the relation $\mathcal{R}$ using specific public input and witness.

[2]The public key (or commitment key) of the Pedersen commitment scheme can be chosen as a random string. Therefore, we are in the common random string model, and even in the plain model if we let the verifier choose the random string.

III. ZERO-KNOWLEDGE WEIGHTED INNER-PRODUCT ARGUMENT

Groth [14] proposed the zero-knowledge argument for the weighted inner product (zk-WIP) and used it to build *square-root* zero-knowledge arguments for linear algebra equations. This paper proposes an improved zk-WIP argument and use it to build short zero-knowledge arguments for range proofs and arithmetic circuits.

For a constant vector $\boldsymbol{c}$, the weighted inner product (WIP) with respect to $\boldsymbol{c}$ is defined as

$$\begin{aligned}\odot_{\boldsymbol{c}} : \mathbb{Z}_p^n \times \mathbb{Z}_p^n &\to \mathbb{Z}_p \\ (\boldsymbol{a}, \boldsymbol{b}) &\mapsto \langle \boldsymbol{a}, (\boldsymbol{c} \circ \boldsymbol{b}) \rangle.\end{aligned}$$

The most interesting special case is that $\boldsymbol{c} = \overrightarrow{y}^n$ for an integer $y \in \mathbb{Z}_p^*$, which is also mainly used in [14], since it has useful properties. This paper designs arguments for range proofs and arithmetic circuits based on the zk-WIP argument with respect to $\overrightarrow{y}^n$. For simplicity, we use the notation $\odot_y$ instead of $\odot_{\overrightarrow{y}^n}$. Note that if $y = 1$, then $\odot_y$ is equivalent to the inner product. Even after the map is defined with $y > 1$, it can be utilized like the inner product by computing $\boldsymbol{a} \odot_y (\boldsymbol{b} \circ \overleftarrow{y}^n)$ and one can verify that

$$\boldsymbol{a} \odot_y (\boldsymbol{b} \circ \overleftarrow{y}^n) = y^{n+1} \cdot \langle \boldsymbol{a}, \boldsymbol{b} \rangle \tag{10}$$

and this propoerty is used when the prover needs to perform the inner product between $\boldsymbol{a}$ and $\boldsymbol{b}$ after fixing $y > 1$.

This paper proposes a zero-knowledge argument for the WIP w.r.t. $\overrightarrow{y}^n$ relation. In particular, group-based homomorphic commitment scheme is employed as a building block so that the relation necessarily involves group elements. In addition, *compressed representation* is used in the sense that the witness and the WIP result are committed together into a group element.[3] More precisely, we propose a zero-knowledge proof system for the following relation:

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n, \alpha \in \mathbb{Z}_p) \\ : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^{\boldsymbol{a} \odot_y \boldsymbol{b}} h^{\alpha} \end{array} \right\}$$

The WIP w.r.t $\overrightarrow{y}^n$ and its simplest case, inner product, have an interesting property, which leads to logarithmic communication cost when combined with homomorphic commitment schemes. The WIP w.r.t $\overrightarrow{y}^n$ can be replaced with the sum of two WIPs with half-lengths. When n is an even number, $n = 2\widehat{n}$, let $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2), \boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2) \in \mathbb{Z}_p^{\widehat{n}} \times \mathbb{Z}_p^{\widehat{n}}$. Then,

$$\boldsymbol{a} \odot_y \boldsymbol{b} = \boldsymbol{a}_1 \odot_y \boldsymbol{b}_1 + (y^{\widehat{n}} \cdot \boldsymbol{a}_2) \odot_y \boldsymbol{b}_2 \tag{11}$$

Thus, using the homomorphic property of the homomorphic commitment scheme and (11), zk-WIP can be reduced w.r.t. $\overrightarrow{y}^n$ to two zk-WIPs w.r.t. $\overrightarrow{y}^{\widehat{n}}$. However, this reduction does not lead to proof size diminution, and we need an additional technique to batch two $\widehat{n}$-length arguments to an $\widehat{n}$-length argument. To ensure this end, we impose an additional round and use a random challenge given from the verifier so that the proposed protocol can achieve logarithmic communication cost in the length of vector n. More precisely, the prover of the zk-WIP w.r.t. $\overrightarrow{y}^n$ transmits $2 \cdot \lceil \log_2(n) \rceil + 2$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_p$. The computational costs of both the prover and the verifier are linear in n.

[3]The witness and the inner-product result are separately committed in [4], [14], but those are committed together in [3]. For short proof size, we follow the representation of [3].

$$\boxed{\mathsf{zk\text{-}WIP}_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h, P; \boldsymbol{a}, \boldsymbol{b}, \alpha)}$$

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h, P \in \mathbb{G}; \boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n, \alpha \in \mathbb{Z}_p) \\ : P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^{\boldsymbol{a} \odot_y \boldsymbol{b}} h^{\alpha} \end{array} \right\}$$

$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, P; \boldsymbol{a}, \boldsymbol{b}, \alpha)$
$\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, P)$

If $n = 1$:

$\mathcal{P}$: $r, s, \delta, \eta \xleftarrow{\$} \mathbb{Z}_p$ and computes
$A = \boldsymbol{g}^r \boldsymbol{h}^s g^{r \odot_y \boldsymbol{b} + s \odot_y \boldsymbol{a}} h^{\delta} \in \mathbb{G}$,
$B = g^{r \odot_y s} h^{\eta} \in \mathbb{G}$.

$\mathcal{P} \to \mathcal{V}$: A, B

$\mathcal{V}$: $e \xleftarrow{\$} \mathbb{Z}_p^*$

$\mathcal{P} \leftarrow \mathcal{V}$: e

$\mathcal{P}$: computes
$r' = r + \boldsymbol{a} \cdot e \in \mathbb{Z}_p$
$s' = s + \boldsymbol{b} \cdot e \in \mathbb{Z}_p$
$\delta' = \eta + \delta \cdot e + \alpha \cdot e^2 \in \mathbb{Z}_p$

$\mathcal{P} \to \mathcal{V}$: r', s', δ'

$\mathcal{V}$: outputs *Accept* iff the following equality holds
$P^{e^2} A^e B = \boldsymbol{g}^{r' \cdot e} \boldsymbol{h}^{s' \cdot e} g^{r' \odot_y s'} h^{\delta'} \in \mathbb{G}$

else $(n > 1)$: Let $\widehat{n} = \frac{n}{2}$, $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2)$, $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2)$, $\boldsymbol{g} = (\boldsymbol{g}_1, \boldsymbol{g}_2)$, and $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$, where $\boldsymbol{a}_i$'s, $\boldsymbol{b}_i$'s, $\boldsymbol{g}_i$'s, and $\boldsymbol{h}_i$'s are of the same length $\widehat{n}$.

$\mathcal{P}$: $d_L, d_R \xleftarrow{\$} \mathbb{Z}_p$ and computes
$c_L = \boldsymbol{a}_1 \odot_y \boldsymbol{b}_2 \in \mathbb{Z}_p$
$c_R = (y^{\widehat{n}} \cdot \boldsymbol{a}_2) \odot_y \boldsymbol{b}_1 \in \mathbb{Z}_p$
$L = \boldsymbol{g}_2^{(y^{-\widehat{n}} \cdot \boldsymbol{a}_1)} \boldsymbol{h}_1^{\boldsymbol{b}_2} g^{c_L} h^{d_L} \in \mathbb{G}$
$R = \boldsymbol{g}_1^{(y^{\widehat{n}} \cdot \boldsymbol{a}_2)} \boldsymbol{h}_2^{\boldsymbol{b}_1} g^{c_R} h^{d_R} \in \mathbb{G}$

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

$\mathcal{V}$: $e \xleftarrow{\$} \mathbb{Z}_p^*$

$\mathcal{P} \leftarrow \mathcal{V}$: e

$\mathcal{P}$ and $\mathcal{V}$: compute
$\widehat{\boldsymbol{g}} = \boldsymbol{g}_1^{e^{-1}} \circ \boldsymbol{g}_2^{e \cdot y^{-\widehat{n}}} \in \mathbb{G}^{\widehat{n}}$
$\widehat{\boldsymbol{h}} = \boldsymbol{h}_1^{e} \circ \boldsymbol{h}_2^{e^{-1}} \in \mathbb{G}^{\widehat{n}}$
$\widehat{P} = L^{e^2} P R^{e^{-2}} \in \mathbb{G}$

$\mathcal{P}$: computes
$\widehat{\boldsymbol{a}} = \boldsymbol{a}_1 \cdot e + (\boldsymbol{a}_2 \cdot y^{\widehat{n}}) \cdot e^{-1} \in \mathbb{Z}_p^{\widehat{n}}$
$\widehat{\boldsymbol{b}} = \boldsymbol{b}_1 \cdot e^{-1} + \boldsymbol{b}_2 \cdot e \in \mathbb{Z}_p^{\widehat{n}}$
$\widehat{\alpha} = d_L \cdot e^2 + \alpha + d_R \cdot e^{-2} \in \mathbb{Z}_p$

$\mathcal{P}$ and $\mathcal{V}$: run $\mathsf{zk\text{-}WIP}_{\overrightarrow{y}^{\widehat{n}}}(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, g, h, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, \widehat{\alpha})$.

FIGURE 1. Zero Knowledge Argument for WIP relation.

A. ZERO-KNOWLEDGE ARGUMENT FOR WIP

This section describes the zero-knowledge argument for WIP w.r.t. $\overrightarrow{y}^n$, denoted by $\mathsf{zk\text{-}WIP}_{\overrightarrow{y}^n}(\boldsymbol{x}; \boldsymbol{y})$, where $\boldsymbol{x}$ is the input of $\mathcal{V}$ and $(\boldsymbol{x}; \boldsymbol{y})$ is the input of $\mathcal{P}$. In the proposed protocol, the verifier starts with the public parameters including group generators $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n$, $g, h \in \mathbb{G}$ and $P = \boldsymbol{g}^{\boldsymbol{a}} \boldsymbol{h}^{\boldsymbol{b}} g^c h^{\alpha}$ a

commitment to vectors $\boldsymbol{a}, \boldsymbol{b}$ and their weighted WIP $c = \boldsymbol{a} \odot_y \boldsymbol{b}$, where $\boldsymbol{a}$ $\boldsymbol{b}$ are witnesses. The prover takes as input $\boldsymbol{g}, \boldsymbol{h}, g, h, P$ and $\boldsymbol{a}, \boldsymbol{b}, \alpha$.

For the sake of simplicity, assume that n is a power of 2 and let $\widehat{n} = n/2$. When $n > 1$, protocol $\mathsf{zk\text{-}WIP}_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h, P; \boldsymbol{a}, \boldsymbol{b}, \alpha)$ is a reduction from length-n argument to $\widehat{n}$-length argument. In the case of $n > 1$, the prover begins by choosing random integers $d_L, d_R \xleftarrow{\$} \mathbb{Z}_p$, computes

$$\begin{aligned} c_L &= \boldsymbol{a}_1 \odot_y \boldsymbol{b}_2 \in \mathbb{Z}_p \\ c_R &= (y^{\widehat{n}} \cdot \boldsymbol{a}_2) \odot_y \boldsymbol{b}_1 \in \mathbb{Z}_p \\ L &= \boldsymbol{g}_2^{(y^{-\widehat{n}} \cdot \boldsymbol{a}_1)} \boldsymbol{h}_1^{\boldsymbol{b}_2} g^{c_L} h^{d_L} \in \mathbb{G} \\ R &= \boldsymbol{g}_1^{(y^{\widehat{n}} \cdot \boldsymbol{a}_2)} \boldsymbol{h}_2^{\boldsymbol{b}_1} g^{c_R} h^{d_R} \in \mathbb{G}, \end{aligned}$$

and sends L and R to the verifier. Next, the verifier chooses and sends a random challenge e to the prover. Then, both the prover and the verifier compute

$$\begin{aligned} \widehat{\boldsymbol{g}} &= \boldsymbol{g}_1^{e^{-1}} \circ \boldsymbol{g}_2^{e \cdot y^{-\widehat{n}}} \in \mathbb{G}^{\widehat{n}} \\ \widehat{\boldsymbol{h}} &= \boldsymbol{h}_1^{e} \circ \boldsymbol{h}_2^{e^{-1}} \in \mathbb{G}^{\widehat{n}} \\ \widehat{P} &= L^{e^2} P R^{e^{-2}} \in \mathbb{G}, \end{aligned}$$

and the prover additionally computes

$$\begin{aligned} \widehat{\boldsymbol{a}} &= \boldsymbol{a}_1 \cdot e + (\boldsymbol{a}_2 \cdot y^{\widehat{n}}) \cdot e^{-1} \in \mathbb{Z}_p^{\widehat{n}}, \\ \widehat{\boldsymbol{b}} &= \boldsymbol{b}_1 \cdot e^{-1} + \boldsymbol{b}_2 \cdot e \in \mathbb{Z}_p^{\widehat{n}}, \\ \widehat{\alpha} &= d_L \cdot e^2 + \alpha + d_R \cdot e^{-2} \in \mathbb{Z}_p. \end{aligned}$$

Last, the above $\widehat{n}$-length vectors $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}}$ have a relation with c_L, c_R, and c as follows:

$$\begin{aligned} \widehat{\boldsymbol{a}} \odot_y \widehat{\boldsymbol{b}} &= \boldsymbol{a}_1 \odot_y \boldsymbol{b}_2 e^2 + \boldsymbol{a}_1 \odot_y \boldsymbol{b}_1 + (y^{n/2} \cdot \boldsymbol{a}_2) \\ &\quad \odot_y \boldsymbol{b}_2 + y^{\widehat{n}} \boldsymbol{a}_2 \odot_y \boldsymbol{b}_1 e^{-2} \\ &= c_L \cdot e^2 + c + c_R \cdot e^{-2} \end{aligned}$$

Using the above relation, the following equality can be verified through a simple calculation.

$$\widehat{P} = L^{e^2} P R^{e^{-2}} = \widehat{\boldsymbol{g}}^{\widehat{\boldsymbol{a}}} \widehat{\boldsymbol{h}}^{\widehat{\boldsymbol{b}}} g^{\widehat{\boldsymbol{a}} \odot_y \widehat{\boldsymbol{b}}} h^{\widehat{\alpha}}.$$

Thus, the above interaction between the prover and the verifier shows that the argument for WIP relation with n-length vectors can be reduced to the same argument with half-length vectors. More precisely, the components shared by the prover and the verifier at the end of interaction $(\widehat{\boldsymbol{g}}, \widehat{\boldsymbol{h}}, g, h, \widehat{P}; \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}, \widehat{\alpha})$ is composed of the same zk-WIP argument with a half-length $\widehat{n}$, which is the desired reduction from an argument related to $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathbb{Z}_p^n$ to an argument related to $\widehat{\boldsymbol{a}}$ and $\widehat{\boldsymbol{b}} \in \mathbb{Z}_p^{\widehat{n}}$. Here, the prover sends only two group elements for each reduction, so that totally, it requires only logarithmic communication cost in n.

When $n = 1$, it becomes a variant of the Schnorr protocol such that logarithms of input bases fulfill a specific quadratic relation, which yields both constant communication and computation costs.

The full description of our zero-knowledge argument for the WIP relation is provided in Fig.1. Theorem 1 is the security statement for zk-WIP protocol and its proof is relegated to Appendix.

Theorem 1: Let y be a constant in $\mathbb{Z}_p^*$. The zero-knowledge argument for WIP is presented in Fig. 1 has perfect completeness, perfect honest verifier zero-knowledge and computational witness-extended emulation.

IV. RANGE PROOFS

This section describes our zero-knowledge argument protocols for single-range proof in Section IV-A and aggregate range proof in Section IV-B.

A. SINGLE-RANGE PROOF PROTOCOL

Consider the following group-based range relation such that witness is committed using the Pedersen commitment scheme.

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h, V \in \mathbb{G}; v, \gamma \in \mathbb{Z}_p) \\ : V = g^v h^\gamma \wedge v \in [0, 2^n - 1] \end{array} \right\}$$

Here, V is a commitment to the witness v that lies in an interval $[0, 2^n - 1]$ for some predetermined parameter n. $\boldsymbol{g}$ and $\boldsymbol{h}$ are vectors of group $\mathbb{G}$ generators, but their usage is ambiguous in the above relation. In fact, these are parameters of the generalized Pedersen commitment scheme. v can be represented as an n-bit string $\boldsymbol{a}_L$, and it is committed using $\boldsymbol{g}$ and $\boldsymbol{h}$ in our range proof protocol. Then, the goal of the range protocol is to prove the knowledge of $\boldsymbol{a}_L$ and an additional vector $\boldsymbol{a}_R$ satisfying

$$\boldsymbol{a}_R = \boldsymbol{a}_L - \mathbf{1}^n \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \mathbf{0} \ \wedge \ \langle \boldsymbol{a}_L, \mathbf{2}^n \rangle = v, \quad (12)$$

which implies that the prover knows a binary vector of length n of witness v. Since each $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ has n components, (12) consists of $2n + 1$ equations. To handle multiple equations at once in a sublinear manner in n, this paper follows the technique dating back to Groth [14] such that it batches equations by computing the inner product with $\overrightarrow{y}^n$ for a random challenge y given from the verifier.

Applying the batching technique, (12) becomes a product relation between three values, the witness $\boldsymbol{a}_L, \boldsymbol{a}_R$ and the challenge y used in the batching technique. Bünz *et al.* [3] presented a proof system for the relations in (12) based on their inner-product argument. Their inner-product protocol does not support the zero-knowledgeness property so that the openings of the Pedersen commitments are revealed to the verifier. Hence, the reduction process should introduce exponentiation-level blinding factors to hide openings, which is rather cumbersome to handle, so that it requires several interactions and transmission of 5 elements in $\mathbb{Z}_p$ and 2 group elements in $\mathbb{G}$.

zk-WIP protocol w.r.t. $\overrightarrow{y}^n$ is a tailored protocol for proving a product relation between two hidden vectors $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ and the challenge $\overrightarrow{y}^n$ with zero-knowledge. Consequently, an optimal reduction to the zk-WIP protocol can be obtained

$\{(\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^n, g, h, V \in \mathbb{G}; v, \gamma \in \mathbb{Z}_p) : V = g^v h^\gamma \wedge v \in [0, 2^n - 1]\}$

$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, V; v, \gamma)$
$\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, V)$

$\mathcal{P}$: $\alpha \xleftarrow{\$} \mathbb{Z}_p$, sets $\boldsymbol{a}_L \in \{0,1\}^n$ such that $\langle \boldsymbol{a}_L, \boldsymbol{2}^n \rangle = v$.
computes $\boldsymbol{a}_R = \boldsymbol{a}_L - \boldsymbol{1}^n \in \mathbb{Z}_p^n$ and $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h^\alpha \in \mathbb{G}$.
$\mathcal{P} \to \mathcal{V}$: A
$\mathcal{V}$: $y, z \xleftarrow{\$} \mathbb{Z}_p$
$\mathcal{V} \to \mathcal{P}$: y, z
$\mathcal{P}$ and $\mathcal{V}$: compute

$$\widehat{A} = A \cdot \boldsymbol{g}^{-\boldsymbol{1}^n \cdot z} \cdot \boldsymbol{h}^{\boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z} \cdot V^{y^{n+1}} \cdot g^{\langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle \cdot z - \langle \boldsymbol{1}^n, \boldsymbol{2}^n \rangle \cdot y^{n+1} z - \langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle \cdot z^2} \in \mathbb{G}$$

$\mathcal{P}$: computes

$$\widehat{\boldsymbol{a}}_L = \boldsymbol{a}_L - \boldsymbol{1}^n \cdot z \in \mathbb{Z}_p^n$$
$$\widehat{\boldsymbol{a}}_R = \boldsymbol{a}_R + \boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z \in \mathbb{Z}_p^n$$
$$\widehat{\alpha} = \alpha + \gamma \cdot y^{n+1} \in \mathbb{Z}_p$$

$\mathcal{P}$ and $\mathcal{V}$: run zk-WIP$_{\overrightarrow{y}^n}(\boldsymbol{g}, \boldsymbol{h}, g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$.

FIGURE 2. Range Proof for $v \in [0, 2^n - 1]$.

in the sense that the prover transmits only a group element in one move during the reduction phase.

Let us explain our reduction to the WIP protocol. The prover begins by sending a commitment $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h^\alpha \in \mathbb{G}$ to vectors $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ with a random α, and the verifier returns random challenges $y, z \in \mathbb{Z}_p$. Next, both the prover and the verifier can compute $\widehat{A}$ as follows.

$$A \boldsymbol{g}^{-\boldsymbol{1}^n z} \boldsymbol{h}^{\boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n z} V^{y^{n+1}} g^{\langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle z - \langle \boldsymbol{1}^n, \boldsymbol{2}^n \rangle y^{n+1} z - \langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle z^2} \tag{13}$$

Here, all the exponents are combinations of the challenges y and z, so that $\widehat{A}$ is publicly computable. Finally, both the prover and the verifier run the WIP argument w.r.t $\overrightarrow{y}^n$ on input $(\boldsymbol{g}, \boldsymbol{h}, g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$, where

$$\widehat{\boldsymbol{a}}_L = \boldsymbol{a}_L - \boldsymbol{1}^n \cdot z \in \mathbb{Z}_p^n$$
$$\widehat{\boldsymbol{a}}_R = \boldsymbol{a}_R + \boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z \in \mathbb{Z}_p^n$$
$$\widehat{\alpha} = \alpha + \gamma \cdot y^{n+1} \in \mathbb{Z}_p.$$

One can easily check that the above defined $\widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R$, and $\widehat{\alpha}$ are exponents with bases $\boldsymbol{g}, \boldsymbol{h}$, and h of $\widehat{A}$, respectively. See the proof of Theorem 2 for completeness.

Now, let us explain why the above reduction correctly works. Let β be the exponent with the base g in A, which is set to be 0 by the honest prover. Then, the WIP argument guarantees that the exponent with the base g of $\widehat{A}$, which is defined as

$$\beta + v y^{n+1} + \langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle z - \langle \boldsymbol{1}^n, \boldsymbol{2}^n \rangle y^{n+1} z - \langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle z^2 \tag{14}$$

by (13), is equal to $\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_R$. It can be written as

$$\begin{aligned}
&(\boldsymbol{a}_L - \boldsymbol{1}^n \cdot z) \odot_y (\boldsymbol{a}_R + \boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z) \\
&\quad = \boldsymbol{a}_L \odot_y \boldsymbol{a}_R + \boldsymbol{a}_L \odot_y (\boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z) \\
&\quad\quad - (\boldsymbol{1}^n \cdot z) \odot_y \boldsymbol{a}_R - (\boldsymbol{1}^n \cdot z) \odot_y (\boldsymbol{2}^n \circ \overleftarrow{y}^n + \boldsymbol{1}^n \cdot z) \\
&\quad = \boldsymbol{a}_L \odot_y \boldsymbol{a}_R + y^{n+1} \cdot \langle \boldsymbol{a}_L, \boldsymbol{2}^n \rangle + (\boldsymbol{a}_L - \boldsymbol{a}_R) \odot_y (\boldsymbol{1}^n \cdot z) \\
&\quad\quad - \langle \boldsymbol{1}^n, \boldsymbol{2}^n \rangle y^{n+1} z - \langle \boldsymbol{1}^n, \overrightarrow{y}^n \rangle z^2
\end{aligned} \tag{15}$$

where the fourth equality holds due to (10). Then, since $\boldsymbol{a}_L$, $\boldsymbol{a}_R$, and v are committed to seeing the challenges y and z, we expect that each coefficient of distinct monomials in (14) is equal to the corresponding coefficient in (15). Therefore, (12) should satisfy.

The full description of our range proof protocol is provided in Figure 2. The prover sends only one group element $A \in \mathbb{G}$ for this reduction to the WIP argument of length-n vectors. In total, the prover in the aggregate range proof protocol transmits $2 \cdot \lceil \log_2(n) \rceil + 3$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_p$. The computational cost of both the prover and the verifier is linear in n.

Theorem 2 states the security for range proof protocol and its proof is relegated to Appendix.

Theorem 2: The zero-knowledge argument for the range proof presented in Figure 2 has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.

B. AGGREGATING RANGE PROOFS

This paper shows that our range proof can be extended to support aggregate range proof as in [3]. That is, when the prover needs to perform $m > 1$ range proofs simultaneously, the proof size increases only logarithmically in m, so that this work can achieve the shortest proof size even in performing multiple range proofs. More precisely, the relation for aggregating range proofs can be presented by generalizing a single-range proof relation as follows.

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^{m \cdot n}, g, h \in \mathbb{G}, \boldsymbol{V} \in \mathbb{G}^m; \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m) \\ : V_j = g^{v_j} h^{\gamma_j} \wedge v_j \in [0, 2^n - 1] \text{ for } j \in [1, m] \end{array} \right\}$$

For $j \in [1, m]$, let $\boldsymbol{d}_j := (\underbrace{0, \ldots, 0}_{(j-1) \cdot n}, \boldsymbol{2}^n, \underbrace{0, \ldots, 0}_{(m-j) \cdot n})$. The prover commits to $\boldsymbol{a}_L \in \{0,1\}^{m \cdot n}$, which is the concatenation of all of the bits for v_j's and satisfies $\langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j$ for all $j \in [1, m]$, and $\boldsymbol{a}_R := \boldsymbol{a}_L - \boldsymbol{1}^{mn}$. More precisely, the prover sends a commitment $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h^\alpha \in \mathbb{G}$ to vectors $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ with a random α. The difference between a single-range proof is that $\boldsymbol{a}_L$ is the concatenation of all binary vectors of $v_j's$; thus, the length is mn. Then, the prover's goal is to prove the knowledge of $\boldsymbol{a}_L$ and $\boldsymbol{a}_R$ satisfying the following relations: for all $j \in [1, m]$,

$$\boldsymbol{a}_R = \boldsymbol{a}_L - \boldsymbol{1}^n \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{0} \ \wedge \ \langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j$$

Although the aggregate range proof requires more relations to convince the verifier than the single-range proof, the batching technique used in the single-range proof can be suitably extended. For the challenge z given from the verifier, let $\boldsymbol{d} := \sum_{j=1}^m z^{2j} \cdot \boldsymbol{d}_j$, and then m relations $\langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j \ \forall j \in [1, m]$ can be batched to a single relation $\langle \boldsymbol{a}_L, \boldsymbol{d} \rangle = \sum_{j=1}^m z^{2j} \cdot v_j$. Here, we use even powers of z since z is already reserved

$$\left\{ \begin{array}{l} (\boldsymbol{g}, \boldsymbol{h} \in \mathbb{G}^{m\cdot n}, g, h \in \mathbb{G}, \boldsymbol{V} \in \mathbb{G}^m; \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m) \\ : V_j = g^{v_j} h^{\gamma_j} \wedge v_j \in [0, 2^n - 1] \text{ for } j \in [1, m] \end{array} \right\}$$

$\mathcal{P}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, \boldsymbol{V}; \boldsymbol{v}, \boldsymbol{\gamma})$
$\mathcal{V}$'s input: $(\boldsymbol{g}, \boldsymbol{h}, g, h, \boldsymbol{V})$

$\mathcal{P}$: for $j \in [1, m]$, let $\boldsymbol{d}_j := (\underbrace{0, \ldots, 0}_{(j-1)\cdot n}, \boldsymbol{2}^n, \underbrace{0, \ldots, 0}_{(m-j)\cdot n})$.
chooses $\alpha \xleftarrow{\$} \mathbb{Z}_p$, sets $\boldsymbol{a}_L \in \{0,1\}^{m\cdot n}$ s.t. $\langle \boldsymbol{a}_L, \boldsymbol{d}_j \rangle = v_j$.
computes $\boldsymbol{a}_R = \boldsymbol{a}_L - \boldsymbol{1}^{mn} \in \mathbb{Z}_p^{mn}$ and $A = \boldsymbol{g}^{\boldsymbol{a}_L} \boldsymbol{h}^{\boldsymbol{a}_R} h^{\alpha} \in \mathbb{G}$
$\mathcal{P} \to \mathcal{V}$: A
$\mathcal{V}$: $y, z \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{V} \to \mathcal{P}$: y, z
$\mathcal{P}$ and $\mathcal{V}$: let $\boldsymbol{d} := \sum_{j=1}^m z^{2j} \cdot \boldsymbol{d}_j$, and computes

$$\widehat{A} = A \cdot \boldsymbol{g}^{-\boldsymbol{1}^{mn}\cdot z} \boldsymbol{h}^{\boldsymbol{d}\circ \overleftarrow{\boldsymbol{y}}^{mn} + \boldsymbol{1}^{mn}\cdot z} \cdot \left(\prod_{j=1}^m V_j^{z^{2j}}\right)^{y^{mn+1}} \cdot g^{\langle \boldsymbol{1}^{mn}, \overrightarrow{\boldsymbol{y}}^{mn}\rangle \cdot z - \langle \boldsymbol{1}^{mn}, \boldsymbol{d}\rangle \cdot y^{mn+1} z - \langle \boldsymbol{1}^{mn}, \overrightarrow{\boldsymbol{y}}^{mn}\rangle \cdot z^2} \in \mathbb{G}$$

$\mathcal{P}$: computes

$$\widehat{\boldsymbol{a}}_L = \boldsymbol{a}_L - \boldsymbol{1}^{mn} \cdot z \in \mathbb{Z}_p^{mn}$$
$$\widehat{\boldsymbol{a}}_R = \boldsymbol{a}_R + \boldsymbol{d} \circ \overleftarrow{\boldsymbol{y}}^{mn} + \boldsymbol{1}^{mn} \cdot z \in \mathbb{Z}_p^{mn}$$
$$\widehat{\alpha} \ = \alpha + \sum_{j=1}^m z^{2j} \cdot \gamma_j \cdot y^{mn+1} \in \mathbb{Z}_p$$

$\mathcal{P}$ and $\mathcal{V}$: run zk-WIP$_{\overrightarrow{\boldsymbol{y}}^{mn}}(\boldsymbol{g}, \boldsymbol{h}, g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$.

FIGURE 3. Aggregate Range Proof for $v_1, \ldots, v_m \in [0, 2^n - 1]$.

for convincing the other equations. All the other parts of the protocol are essentially the same as the single-range proof protocol.

The full description of our aggregate range proof protocol is provided in Figure 3. The prover sends only one group element $A \in \mathbb{G}$ for this reduction to the zk-WIP argument of length-mn vectors. In total, the prover in the aggregate range proof protocol transmits $2 \cdot \lceil \log_2(m) + \log_2(n) \rceil + 3$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_p$. The computational cost of both the prover and the verifier is linear in mn.

Theorem 3 covers the security statement for the proposed aggregate range proof portocol and its proof is relegated to Appendix.

Theorem 3: The zero-knowledge argument for the range proof presented in Figure 3 has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.

V. ZERO-KNOWLEDGE ARGUMENT FOR ARITHMETIC CIRCUITS

As another application of the zk-WIP argument, we presents a zero-knowledge proof system for arbitrary arithmetic circuits. Bootle *et al.* [4] presented a conversion from an arbitrary arithmetic circuit with n multiplication gates into a certain relation containing a Hadamard product with some linear constraints, which is formally described below. Bünz *et al.* [3] 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 following relation is for zero-knowledge argument for arithmetic circuits, which is exactly the same as that in Bulletproofs [3], and we also restrict $\boldsymbol{W}_V \in \mathbb{Z}_p^{Q\times m}$ to be of rank m, as in Bulletproofs.

$$\left\{ \begin{pmatrix} \boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{G}^n, \boldsymbol{V} \in \mathbb{G}^m, \ g, h \in \mathbb{G}, \\ \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O \in \mathbb{Z}_p^{Q\times n}, \boldsymbol{W}_V \in \mathbb{Z}_p^{Q\times m}, \\ \boldsymbol{c} \in \mathbb{Z}_p^Q; \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O \in \mathbb{Z}_p^n, \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m \end{pmatrix} \middle| \begin{array}{l} \forall j = g^{v_j} h^{\gamma_j} \forall j \in [1, m] \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O \\ \wedge \ \boldsymbol{W}_L \boldsymbol{a}_L^\top + \boldsymbol{W}_R \boldsymbol{a}_R^\top + \boldsymbol{W}_O \boldsymbol{a}_O^\top = \boldsymbol{W}_V \boldsymbol{v}^\top + \boldsymbol{c}^\top \end{array} \right\}$$

As in our range proof protocol, our goal for the arithmetic circuit proof is to reduce to the zk-WIP argument, and as a result, only one element is sufficient for reducing zk-WIP in terms of the prover. The whole description of our arithmetic circuit proof is given in Fig. 4. For a concise description, we introduce the notation used in Fig. 4. For an integer $z \in \mathbb{Z}_p$, $\tilde{\boldsymbol{z}}^Q$ denotes a vector $(z, z^3, z^5, \ldots, z^{2Q-1}) \in \mathbb{Z}_p^Q$ consisting of odd powers of z for some predetermined Q. For matrices $\boldsymbol{W} \in \mathbb{Z}_p^{Q\times n}$, $\mathsf{T}_{\boldsymbol{W}}^{(y,z)}$ denotes $(y^{-1}, y^{-2}, \ldots, y^{-n}) \circ (\tilde{\boldsymbol{z}}^Q \boldsymbol{W})$. That is, when y, z are challenges given from the verifier, $\mathsf{T}_{\boldsymbol{W}}^{(y,z)}$ is a publicly computable value.

First, the prover sends $A = \boldsymbol{g}_1^{\boldsymbol{a}_L} \boldsymbol{g}_2^{\boldsymbol{a}_O} \boldsymbol{h}_1^{\boldsymbol{a}_R} h^{\alpha}$, which is a commitment to $\boldsymbol{a}_L, \boldsymbol{a}_O, \boldsymbol{a}_R$ with a random α. Then, the prover's goal is to convince that $\boldsymbol{a}_L, \boldsymbol{a}_O, \boldsymbol{a}_R$ and v_j's satisfy the following relations.

$$\begin{aligned} V_j = g^{v_j} h^{\gamma_j} \forall j \in [1, m] \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O \\ \wedge \ \boldsymbol{W}_L \boldsymbol{a}_L^\top + \boldsymbol{W}_R \boldsymbol{a}_R^\top + \boldsymbol{W}_O \boldsymbol{a}_O^\top = \boldsymbol{W}_V \boldsymbol{v}^\top + \boldsymbol{c}^\top \end{aligned}$$

Next, the verifier chooses y and z randomly and sends them to the prover, and both the prover and the verifier compute $\widehat{A}$ as follows.

$$A \boldsymbol{g}_1^{\mathsf{T}_{\boldsymbol{W}_R}^{(y,z)}} \boldsymbol{h}_1^{\mathsf{T}_{\boldsymbol{W}_L}^{(y,z)}} \boldsymbol{h}_2^{y^{-n}(\mathsf{T}_{\boldsymbol{W}_O}^{(y,z)} - \boldsymbol{1}^n)} \boldsymbol{V}^{\tilde{\boldsymbol{z}}^Q \boldsymbol{W}_V} g^{\tilde{\boldsymbol{z}}^Q \cdot \boldsymbol{c}^\top + \mathsf{T}_{\boldsymbol{W}_R}^{(y,z)} \odot_y \mathsf{T}_{\boldsymbol{W}_L}^{(y,z)}} \tag{16}$$

and run the protocol for the zk-WIP w.r.t. $\overrightarrow{\boldsymbol{y}}^{2n}$ on input $((\boldsymbol{g}_1, \boldsymbol{g}_2), (\boldsymbol{h}_1, \boldsymbol{h}_2), g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$, where

$$\widehat{\boldsymbol{a}}_L = (\boldsymbol{a}_L + \mathsf{T}_{\boldsymbol{W}_R}^{(y,z)}, \ \boldsymbol{a}_O) \tag{17}$$
$$\widehat{\boldsymbol{a}}_R = (\boldsymbol{a}_R + \mathsf{T}_{\boldsymbol{W}_L}^{(y,z)}, \ y^{-n}(\mathsf{T}_{\boldsymbol{W}_O}^{(y,z)} - \boldsymbol{1}^n))$$
$$\widehat{\alpha} \ = \alpha + \tilde{\boldsymbol{z}}^Q \boldsymbol{W}_V \boldsymbol{\gamma}^\top \tag{18}$$

One can easily check that the above defined $\widehat{\boldsymbol{a}}_L$, $\widehat{\boldsymbol{a}}_R$, and $\widehat{\alpha}$ are exponents with bases $(\boldsymbol{g}_1, \boldsymbol{g}_2)$, $(\boldsymbol{h}_1, \boldsymbol{h}_2)$, and h of $\widehat{A}$, respectively. The completeness of our suggestion is described in Theorem 4.

Now, let us explain why the above reduction works correctly. Let β be the exponent with the base g in A, which is set to be 0 by the honest prover. Similarly, let $\boldsymbol{a}_P$ be the exponent with base $\boldsymbol{h}_2$ in A, which is set to $\boldsymbol{0}$ by the honest prover. Then, the WIP argument guarantees that the exponent with the base g of $\widehat{A}$, which is defined as

$$\beta + \tilde{\boldsymbol{z}}^Q \boldsymbol{W}_V \boldsymbol{v}^\top + \tilde{\boldsymbol{z}}^Q \cdot \boldsymbol{c}^\top + \mathsf{T}_{\boldsymbol{W}_R}^{(y,z)} \odot_y \mathsf{T}_{\boldsymbol{W}_L}^{(y,z)} \tag{19}$$

by (16), is equal to $\widehat{\boldsymbol{a}}_L \odot_y \widehat{\boldsymbol{a}}_R$ and so is equal to

$$(\boldsymbol{a}_L + \mathsf{T}_{\boldsymbol{W}_R}^{(y,z)}, \boldsymbol{a}_O) \odot_y (\boldsymbol{a}_R + \mathsf{T}_{\boldsymbol{W}_L}^{(y,z)}, \boldsymbol{a}_P + y^{-n}(\mathsf{T}_{\boldsymbol{W}_O}^{(y,z)} - \boldsymbol{1}^n)) \tag{20}$$

$$\left\{ \left(\begin{array}{l} \boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{G}^n, \boldsymbol{V} \in \mathbb{G}^m,\ g, h \in \mathbb{G}, \\ \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O \in \mathbb{Z}_p^{Q \times n}, \boldsymbol{W}_V \in \mathbb{Z}_p^{Q \times m}, \\ \boldsymbol{c} \in \mathbb{Z}_p^Q; \boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O \in \mathbb{Z}_p^n, \boldsymbol{v}, \boldsymbol{\gamma} \in \mathbb{Z}_p^m \end{array} \right) \right\}$$
$$: \begin{array}{l} V_j = g^{v_j} h^{\gamma_j} \forall j \in [1, m] \ \wedge \ \boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O \\ \wedge \boldsymbol{W}_L \boldsymbol{a}_L^\top + \boldsymbol{W}_R \boldsymbol{a}_R^\top + \boldsymbol{W}_O \boldsymbol{a}_O^\top = \boldsymbol{W}_V \boldsymbol{v}^\top + \boldsymbol{c}^\top \end{array}$$

$\mathcal{P}$'s input: $(\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{h}_1, \boldsymbol{h}_2\ \boldsymbol{V},\ g, h,\ \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O,\ \boldsymbol{W}_V,\ \boldsymbol{c}$; $\boldsymbol{a}_L, \boldsymbol{a}_R, \boldsymbol{a}_O, \boldsymbol{v}, \boldsymbol{\gamma})$

$\mathcal{V}$'s input: $(\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{h}_1, \boldsymbol{h}_2\ \boldsymbol{V},\ g, h,\ \boldsymbol{W}_L, \boldsymbol{W}_R, \boldsymbol{W}_O,\ \boldsymbol{W}_V,\ \boldsymbol{c})$

$\mathcal{P}$: chooses $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $A = \boldsymbol{g}_1^{\boldsymbol{a}_L} \boldsymbol{g}_2^{\boldsymbol{a}_O} \boldsymbol{h}_1^{\boldsymbol{a}_R} h^\alpha \in \mathbb{G}$.

$\mathcal{P} \to \mathcal{V}$: A

$\mathcal{V}$: $y, z \xleftarrow{\$} \mathbb{Z}_p$

$\mathcal{V} \to \mathcal{P}$: y, z

$\mathcal{P}$ and $\mathcal{V}$: compute

$$\widehat{A} = A \cdot \boldsymbol{g}_1^{\mathsf{T}_{\boldsymbol{W}_R}^{(y,z)}} \cdot \boldsymbol{h}_1^{\mathsf{T}_{\boldsymbol{W}_L}^{(y,z)}} \cdot \boldsymbol{h}_2^{y^{-n}(\mathsf{T}_{\boldsymbol{W}_O}^{(y,z)} - \boldsymbol{1}^n)} \cdot \boldsymbol{V}^{\boldsymbol{z}^Q \cdot \boldsymbol{W}_V} \cdot g^{\boldsymbol{z}^Q \cdot \boldsymbol{c}^\top + \mathsf{T}_{\boldsymbol{W}_R}^{(y,z)} \odot_y \mathsf{T}_{\boldsymbol{W}_L}^{(y,z)}} \in \mathbb{G}.$$

$\mathcal{P}$: computes

$$\widehat{\boldsymbol{a}}_L = (\boldsymbol{a}_L + \mathsf{T}_{\boldsymbol{W}_R}^{(y,z)},\ \boldsymbol{a}_O) \in \mathbb{Z}_p^{2n}$$
$$\widehat{\boldsymbol{a}}_R = (\boldsymbol{a}_R + \mathsf{T}_{\boldsymbol{W}_L}^{(y,z)},\ y^{-n}(\mathsf{T}_{\boldsymbol{W}_O}^{(y,z)} - \boldsymbol{1}^n)) \in \mathbb{Z}_p^{2n}$$
$$\widehat{\alpha} = \alpha + \boldsymbol{z}^Q \boldsymbol{W}_V \boldsymbol{\gamma}^\top \in \mathbb{Z}_p$$

$\mathcal{P}$ and $\mathcal{V}$: run zk-WIP$_{\overline{y}^{2n}}((\boldsymbol{g}_1, \boldsymbol{g}_2), (\boldsymbol{h}_1, \boldsymbol{h}_2), g, h, \widehat{A}; \widehat{\boldsymbol{a}}_L, \widehat{\boldsymbol{a}}_R, \widehat{\alpha})$

FIGURE 4. Zero knowledge argument for arithmetic circuit.

by (17) and (18). $\boldsymbol{a}_L, \boldsymbol{a}_R$, and $\boldsymbol{v}$ are committed to first disclose the challenges y and z, so that we expect that each coefficient of distinct monomials in (19) is equal to the corresponding coefficient in (20). This enables to convince the verifier of several relations given in (12). For example, there are no y only terms in (19), but $\boldsymbol{a}_L \odot_y \boldsymbol{a}_R + y^n \cdot \boldsymbol{a}_O \odot_y \boldsymbol{a}_P - \boldsymbol{a}_O \odot_y \boldsymbol{1}^n$ are y only terms in (20), so that one can obtain a relation

$$\boldsymbol{a}_L \odot_y \boldsymbol{a}_R + y^n \cdot \boldsymbol{a}_O \odot_y \boldsymbol{a}_P - \boldsymbol{a}_O \odot_y \boldsymbol{1}^n = \boldsymbol{0},$$

which implies the desired relation $\boldsymbol{a}_L \circ \boldsymbol{a}_R = \boldsymbol{a}_O$. We relegate the detailed calculations for checking the soundness to the proof of Theorem 4.

The full description of our arithmetic circuit proof protocol is provided in Fig. 4. The prover sends only one group element $A \in \mathbb{G}$ for this reduction to the zk-WIP argument of length-$2n$ vectors. In total, the prover of the aggregate range proof protocol transmits $2 \cdot \lceil \log_2(n) \rceil + 5$ elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_p$. The computational cost of the prover and the verifier is linear in n.

Theorem 4 covers the security statement for our zero-knowledge proof protocol for arithmetic circuits and its proof is relegated to Appendix.

Theorem 4: The zero-knowledge argument presented in Fig. 4 has perfect completeness, perfect honest verifier zero-knowledge and computational witness extended emulation.

VI. EVALUATION

Experimental Setup: Except where noted, our experiments were conducted on a MacBook Pro (Retina, 13-inch, Late 2013) with an Intel i5 CPU and 8 GB DDR4 memory. For a fair comparison with optimized Bulletproofs implementation, our protocols are implemented in Rust using the curve25519-dalek library for ECC operations [62] and compared with the January 2020 git version of the Bulletproofs implementation by Valence *et al.* [11], which is, to our knowledge, one of the most optimized Bulletproofs implementations. For more details, we use $\mathbb{F}_p = \mathbb{F}_{2^{255}-19}$ and point arithmetic in affine Niels coordinates, and both implementations for Bulletproofs and Bulletproofs+ are expected to have 128-bit security [63].

A. PRACTICAL OPTIMIZATIONS

1) REDUCTION TO SINGLE MULTIEXPONENTIATION

Let g and h be the generators used in the final round of the protocol and e_i be the challenge from the i-th round. In the last round, the verifier computes $\mathsf{g}^{r' \cdot e} \mathsf{h}^{s' \cdot e}$. To avoid computing $\widehat{\boldsymbol{g}}$ and $\widehat{\boldsymbol{h}}$ in every round, we rewrite these final generators g, h as a single multiexponentiation, using recursion unrolling as in [3]. This substantially reduces the computational overhead on the verifier side since a single multiexponentiation can be done much faster than multiplying the results of individual exponentiations.

$$\mathsf{g} = \prod_{i=1}^n g_i^{s_i} \in \mathbb{G} \text{ and } \mathsf{h} = \prod_{i=1}^n h_i^{s_i'} \in \mathbb{G},$$

where $\boldsymbol{s} = (s_1, \ldots, s_n), \boldsymbol{s}' = (s_1', \ldots, s_n') \in \mathbb{Z}_p^n$ depend on only the challenges $(e_1, \ldots, e_{\log_2(n)})$. The scalars $s_1, \ldots, s_n$ and $s_1', \ldots, s_n'$ can be computed by the following simple calculations:

$$s_i = (1/y^{i-1}) \cdot \prod_{j=1}^{\log_2(n)} e_j^{b(i,j)} \text{ and } s_i' = \prod_{j=1}^{\log_2(n)} e_j^{-b(i,j)}$$

where

$$b(i,j) = \begin{cases} 1 & \text{the } j\text{-th bit of } i-1 \text{ is 1} \\ -1 & \text{otherwise} \end{cases}$$

Then, the entire verification check in the zk-WIP argument protocol given in Figure 1 reduces to a single multiexponentiation as follows:

$$\begin{aligned} & \boldsymbol{g}^{r' e \cdot \boldsymbol{s}} \boldsymbol{h}^{s' e \cdot \boldsymbol{s}'} g^{r' \odot s'} h^{\delta'} \\ & \stackrel{?}{=} \left(P \cdot \prod_{j=1}^{\log_2(n)} L_j^{e_j^2} R_j^{e_j^{-2}} \right)^{e^2} \cdot A^e \cdot B \\ & = P^{e^2} \cdot \left(\prod_{j=1}^{\log_2(n)} L_j^{e^2 \cdot e_j^2} R_j^{e^2 \cdot e_j^{-2}} \right) \cdot A^e \cdot B. \end{aligned}$$

2) REUSE IN SCALARS

As an additional optimization, our implementation uses the dynamic programming paradigm to reduce the number of exponentiations in $\mathbb{F}_p$ that cover a large part of the computing scalars. For example, consider an exponent of base g while computing $\widehat{\boldsymbol{A}}$ in our range proof protocol (see Figure 2).

We inductively obtain y^i by multiplying y by y^{i-1}, and this result is reused in computing the last term of the exponent of base g. Consequently, we can obtain a resulting scalar for the g only with $n+5$ multiplications in $\mathbb{F}_p$ where we do not count the exponentiations of base 2 as $2 \ll y$. We apply the same technique to the implementations of other argument protocols.

3) BATCH VERIFICATION

The batch verification technique in [3] is applicable to Bulletproofs+. Informally, batch verification collapses two independent exponentiations $g^a \stackrel{?}{=} 1$ and $g^b \stackrel{?}{=} 1$ into a single exponentiation $g^{a \cdot r+b} \stackrel{?}{=} 1$ by picking a random value in $\mathbb{F}_p$. Similarly, in our WIP-based argument protocols, the verifier needs to test whether $\widehat{\boldsymbol{A}}$ is correctly computed and invoke the WIP verifier with a reduced proof. Because the bases in both computations are equivalent, we can utilize the batch verification technique to reduce CPU times at the verifier as in Bulletproofs.

B. EXPERIMENTAL RESULTS

We implement our protocols with the above optimizations, and in what follows, we present the results. The experimental results of the range arguments are summarized in Table 2 for each of our metrics. We use three metrics: 1) the size of a proof in bytes, 2) the total CPU time at the prover in milliseconds, and 3) the total CPU time at the verifier. For this purpose, we use Rust's benchmark tests that run our benchmark many times and take the average. When demonstrating CPU times, we omit the total number of iterations made by the `test` crate.

1) PROOF SIZE

As shown in the previous sections, Bulletproofs+ prover for the range proof transmits $2 \cdot \lceil \log_2(m) + \log_2(n) \rceil + 6$ field or group elements, which are 3 elements smaller than that of Bulletproofs. In general, the Bulletproofs+ prover for an arbitrary arithmetic circuit sends $2 \cdot \lceil \log_2(n) \rceil + 8$ field or group elements that are 5 elements smaller than that of Bulletproofs. In our experimental parameter setting, Bulletproofs+ for the range proof and arithmetic circuit proof always save 96 bytes and 160 bytes, respectively, compared with Bulletproofs, regardless of input size.

2) PROVER's CPU TIME ON THE AGGREGATE RANGE PROOF

Table 2 and its graph on the prover's time in Figure 5 show that a Bulletproofs+ prover is slightly faster than that of Bulletproofs. When aggregating more proofs, the ratio between speeds tends to increase in our experimentation. For instance, in the case of a range argument for a single 32-bit secret, our range argument protocol runs 9.7% faster than that of Bulletproofs; additionally, in the case of 64 × 32-bit secrets, our protocol runs 26.8% faster than that of Bulletproofs.

Comparing our WIP prover with a prover in the inner product argument of Bulletproofs, our WIP prover has to perform more operations and exponentiations for blinding factors to achieve zero-knowledgeness. These computational overheads are insignificant compared to heavy multiexponentiations performed in the two provers. The most influential computation is performed during the process of reduction from the aggregate range proof to ingredient protocols (WIP for ours and inner product for Bulletproofs). In fact, the benefit of using WIP is a simpler process in the reduction than the approach used in Bulletproofs. More precisely, Bulletproofs require more multiexponentiations in the reduction process, contrary to a single multiexponentiation in Bulletproofs+.

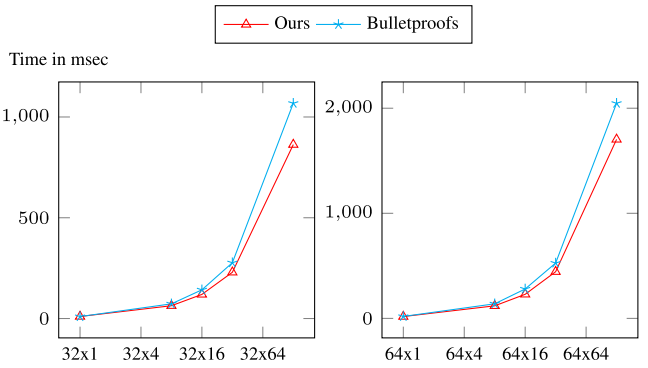

FIGURE 5. CPU times for proving proofs.

Bulletproofs+ can gain computational cost for the prover over Bulletproofs by reducing the number of multi-exponentiations. Specifically, because WIP prover and an IP prover in Bulletproofs both require to invoke a single multi-exponentiation $2\log_2(n)$ times, there is no difference in CPU-time for the provers. However, different from our range argument protocol, the prover in Bulletproofs needs to compute $S = \boldsymbol{g}^{\boldsymbol{s}_L}\boldsymbol{h}^{\boldsymbol{s}_R}h^{\rho}$ that requires to run a single multi-exponentiation of size $2n+1$ two times. Furthermore, the prover has to compute double exponentiations two times. As a result, total invocations of multi-exponentiations amounts to $2\log_2(n)+2$, in addition to two double exponentiations. On the contrary, Bulletproofs+ performs a single multi-exponentiation of size $n+3$ during computing $\widehat{\boldsymbol{A}}$. Bulletproofs+ also needs to perform a single multi-exponentiation to commit to secrets as $\boldsymbol{A}$, of size $2n+1$. Therefore, Bulletproofs+ performs a single multi-exponentiation of smaller size among one of multi-exponentiations, and Bulletproofs+ removes two double exponentiations.

3) VERIFIER's CPU TIME ON THE AGGREGATE RANGE PROOF

Figure 6 shows that our proposal is comparable to Bulletproofs in terms of the verifier's computational cost. Both of the verification costs in Bulletproofs and Bulletproofs+ are dominated by a single multiexponentiation. In fact, Bulletproofs+ requires a multiexponentiation for computing $\widehat{A}$ during the reduction process. We note that $\widehat{A}$ is taken as input to the WIP protocol so that it eventually becomes

TABLE 2. Comparison summary of proof size and timing in aggregate range proofs with Bulletproofs.

Parameters	Proof size (bytes)		Prover time (msec)		Verifier time (msec)	
	[3]	Ours	[3]	Ours	[3]	Ours
32x1	608	512	9.79	10.06	1.41	1.39
32x8	800	704	72.84	63.58	7.19	7.29
32x16	864	768	142.41	119.65	13.53	12.72
32x32	928	832	276.84	229.46	21.93	22.37
32x128	1,056	960	1,068	862.98	80.78	81.04
64x1	672	576	18.40	17.26	2.43	2.57
64x8	864	768	138.56	119.44	12.40	12.59
64x32	992	896	279.79	228.99	22.06	22.19
64x64	1056	960	526.36	444.03	40.53	41.96
64x128	1,120	1,024	2,047	1,703.2	155.53	160.79

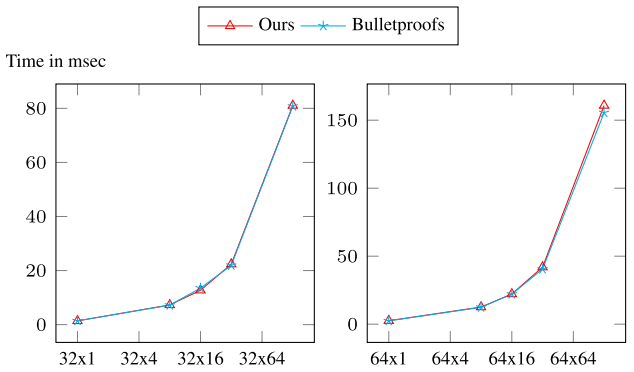

FIGURE 6. CPU times for verifying proofs.

a component in the multiexponentiation performed in the WIP argument protocol. Thus, the verifier can delay the $\widehat{A}$ computation and extend the technique for reduction to single multiexponentiation in Section VI-A. Finally, we obtain a single multiexponentiation.

The difference on the size of the single multi-exponentiation between Bulletproofs and Bulletproofs+ may cause a slight difference between the verification speeds. In fact, a similar technique of batching multiple multi-exponentiations is already used in Bulletproofs to merge two multi-exponentiations. As mentioned above, our implementation extensively exploits batch verification as well as the single multi-exponentiation technique. Thus although our WIP argument ensures the zero-knowledge property, there is no difference in verifier's time from that of Bulletproofs. The main factor to make our verifier slightly slower (precisely, on average 3.3%) than of Bulletproofs is the size of exponents for multiexponentiations.

VII. CONCLUSION

This paper describes a new short zero-knowledge argument for range proofs and arithmetic circuits without a trusted setup. These proof systems are one of the widely employed proof systems in the blockchain and the proof size is one of the major criteria to be contained in a block. We can achieve the shortest proof size of the proof system categories while preserving the proof cost and the verification cost. Furthermore, we implement Bulletproofs+ and we can show that the proving time and the verification time are comparable to Bulletproofs.

Limitations and Future Work: This paper achieves the smallest proof size in the proof system category without a trusted setup. However, the computational cost for the verifier is still $O(n)$ where n is the circuit size, meaning that the verifier also should execute a circuit as large as the prover's circuit. It causes some inefficiency problems for the verifier depending on the applications such as deep learning and succinct blockchain. Recently, there are a couple of papers that can achieve $O(\log_2 n)$ for the verifier. However, these protocols are not transparent [56], [64], some cannot achieve $O(\log_2 n)$ proof size [65], or, some require multiexponentiations of large size [6] that still causes inefficiency problem when n is small. Hence, one can raise the following natural and meangingful question:

Is there a way to achieve $O(\log_2 n)$ at the same time as the proof size and the verifier cost?

and if there is a protocol that can achieve these properties, Bulletproofs-style proof system can be employed in the various applications.

ACKNOWLEDGMENT

This work is done before Heewon Chung joins Desilo Inc. and Kyoohyung Han joins Samsung SDS.

REFERENCES

[1] S. Nakamoto. (2009). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf
[2] G. Maxwell. (2016). *Confidential Transactions*. [Online]. Available: https://people.xiph.org/~greg/confidential_values.txt
[3] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 315–334.
[4] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science. Berlin, Germany: Springer, 2016, pp. 327–357.
[5] M. Hoffmann, M. Klooß, and A. Rupp, "Efficient zero-knowledge arguments in the discrete log setting, revisited," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2093–2110.
[6] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent SNARKs from dark compilers," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 12105. Cham, Switzerland: Springer, 2020, pp. 677–706.
[7] B. Bünz. *BulletproofLib*. Accessed: Mar. 1, 2022. [Online]. Available: https://github.com/bbuenz/BulletProofLib

[8] A. Poelstra, P. Wuille, and G. Maxwell. *secp256k1-zkp*. Accessed: Mar. 1, 2022. [Online]. Available: https://github.com/apoelstra/secp256k1-mw/tree/bulletproofs

[9] Monero Project. *Monero*. Accessed: Mar. 1, 2022. [Online]. Available: https://github.com/monero-project/monero/tree/master/src/ringct

[10] *Mimblewimble*. Accessed: Mar. 1, 2022. [Online]. Available: https://github.com/mimblewimble/rust-secp256k1-zkp

[11] H. de Valence, C. Yun, and O. Andreev. (2018). *A Pure-Rust Implementation of Bulletproofs Using Ristretto*. [Online]. Available: https://github.com/dalek-cryptography/Bulletproofs

[12] ING Bank. *Bulletproofs*. Accessed: May 4, 2020. [Online]. Available: https://github.com/ing-bank/zkrp/tree/master/bulletproofs

[13] A Inc. *Bulletproofs*. Accessed: Mar. 1, 2022. [Online]. Available: https://github.com/adjoint-io/bulletproofs

[14] J. Groth, "Linear algebra with sub-linear zero-knowledge arguments," in *Proc. CRYPTO*, in Lecture Notes in Computer Science. Berlin, Germany: Springer, 2009, pp. 192–208.

[15] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 576. Berlin, Germany: Springer, 1991, pp. 129–140.

[16] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. 1st ACM Conf. Comput. Commun. Secur. (CCS)*, 1993, pp. 62–73.

[17] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 263. Berlin, Germany: Springer, 1987, pp. 186–194.

[18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zkSNARKs without trusted setup," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 926–943.

[19] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science), vol. 7859. Berlin, Germany: Springer, 2013.

[20] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science), vol. 7859. Berlin, Germany: Springer, 2013.

[21] S.-F. Sun, M. Au, J. Liu, and T. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero," in *Proc. ESORICS*, in Lecture Notes in Computer Science, vol. 10493. Cham, Switzerland: Springer, 2017, pp. 456–474.

[22] T. H. Yuen, S. feng Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "RingCT 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science), vol. 12059. Berlin, Germany: Springer, 2020, p. 508.

[23] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "Quisquis: A new design for anonymous cryptocurrencies," in *Proc. ASIACRYPT*, in Lecture Notes in Computer Science, vol. 11921. Cham, Switzerland: Springer, 2019, pp. 649–678.

[24] T. E. Jedusor. (2016). *Mimblewimble*. [Online]. Available: https://scalingbitcoin.org/papers/mimblewimble.txt

[25] A. Poelstra. (2016). *Mimblewimble*. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf

[26] G. Fuchsbauer, M. Orrù, and Y. Seurin, "Aggregate cash systems: A cryptographic investigation of mimblewimble," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 11476. Cham, Switzerland: Springer, 2019, pp. 657–689.

[27] *Grin*. Accessed: Mar. 2, 2022. [Online]. Available: https://grin.mw

[28] *Beam*. Accessed: Mar. 2, 2022. [Online]. Available: https://beam.mw

[29] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2020, pp. 423–443.

[30] G. Wood. (2014). *Ethereum*. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[31] *Findora*. Accessed: Mar. 2, 2022. [Online]. Available: https://findora.org

[32] NIST. *Post Quantum Cryptography*. Accessed: Mar. 2, 2022. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography

[33] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proc. 8th ACM Conf. Comput. Commun. Secur. (CCS)*, 2001, pp. 116–125.

[34] J. Groth, "A verifiable secret shuffe of homomorphic encryptions," in *Proc. PKC*, in Lecture Notes in Computer Science. Berlin, Germany: Springer, 2003, pp. 145–160.

[35] J. Groth and Y. Ishai, "Sub-linear zero-knowledge argument for correctness of a shuffle," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2008, pp. 379–396.

[36] S. Bayer and J. Groth, "Efficient zero-knowledge argument for correctness of a shuffle," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 7237. Berlin, Germany: Springer, 2012, pp. 263–280.

[37] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Proc. CRYPTO*. Berlin, Germany: Springer, 2001, pp. 368–387.

[38] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*. Boston, MA, USA: Springer, 1983, pp. 199–203.

[39] M. Backes, A. Kate, M. Maffei, and K. Pecina, "ObliviAd: Provably secure and practical online behavioral advertising," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 257–271.

[40] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 720–731.

[41] E. F. Brickell, D. Chaum, I. B. Damgård, and J. van de Graaf, "Gradual and verifiable release of a secret," in *Proc. CRYPTO*, in Lecture Notes in Computer Science. Berlin, Germany: Springer, 1988, pp. 156–166.

[42] A. Chan, Y. Frankel, and Y. Tsiounis, "Easy come—Easy go divisible cash," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 1403. Berlin, Germany: Springer, 1998, pp. 561–575.

[43] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 1807. Berlin, Germany: Springer, 2000, pp. 431–444.

[44] H. Lipmaa, "On diophantine complexity and statistical zero-knowledge arguments," in *Proc. ASIACRYPT*, in Lecture Notes in Computer Science, vol. 2894. Berlin, Germany: Springer, 2003, pp. 398–415.

[45] J. Camenisch, R. Chaabouni, and A. shelat, "Efficient protocols for set membership and range proofs," in *Proc. ASIACRYPT*, in Lecture Notes in Computer Science, vol. 5350. Berlin, Germany: Springer, 2008, pp. 234–252.

[46] J. Groth, "Efficient zero-knowledge arguments from two-tiered homomorphic commitments," in *Proc. ASIACRYPT*, in Lecture Notes in Computer Science, vol. 7073. Berlin, Germany: Springer, 2011, pp. 431–448.

[47] R. Chaabouni, H. Lipmaa, and B. Zhang, "A non-interactive range proof with constant communication," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 179–199.

[48] A. González and C. Ráfols, "New techniques for non-interactive shuffle and range arguments," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 9696. Cham, Switzerland: Springer, 2016, pp. 427–444.

[49] G. Couteau, T. Peters, and D. Pointcheval, "Removing the strong RSA assumption from arguments over the integers," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 10211. Cham, Switzerland: Springer, 2017, pp. 321–350.

[50] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security*. Berlin, Germany: Springer, 2005, pp. 467–482.

[51] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 9696. Berlin, Germany: Springer, 1997, pp. 480–494.

[52] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct NIZKs without PCPs," in *Proc. EUROCRYPT*. Berlin, Germany: Springer, 2013, pp. 626–645.

[53] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *Proc. CRYPTO*. Berlin, Germany: Springer, 2013, pp. 90–108.

[54] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. EUROCRYPT*. Berlin, Germany: Springer, 2016, pp. 305–326.

[55] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers, "Updatable and universal common reference strings with applications to zk-SNARKs," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2018, pp. 698–728.

[56] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings," in *Proc. ACM SIGSAC Conf. Comput. Commun.*, 2019, pp. 2111–2128.

[57] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zKSNARKs with universal and updatable SRS," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 12105. Cham, Switzerland: Springer, 2020, pp. 738–768.

[58] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 11694. Cham, Switzerland: Springer, 2019, pp. 733–764.
[59] D. Boneh, B. Fisch, A. Gabizon, and Z. Williamson. *A Simple Range Proof From Polynomial Commitments*. Accessed: Mar. 2, 2022. [Online]. Available: https://hackmd.io/@dabo/B1U4kx8XI
[60] T. Attema and R. Cramer, "Compressed Σ-protocol theory and practical application to plug & play secure algorithmics," in *Proc. CRYPTO*, in Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2020, pp. 513–543.
[61] G. Couteau, M. Klooß, H. Lin, and M. Reichle, "Efficient range proofs with transparent setup from bounded integer commitments," in *Proc. EUROCRYPT*, in Lecture Notes in Computer Science, vol. 12698. Cham, Switzerland: Springer, 2021, pp. 247–277.
[62] I. A. Lovecruft and H. de Valence. (2019). *Curve25519-Dalek Version 2.0.0*. [Online]. Available: https://docs.rs/curve25519-dalek/2.0.0/curve25519_dalek/
[63] *Elliptic Curves for Security*, IEFT, document RFC 7748, 2016. [Online]. Available: https://tools.ietf.org/html/rfc7748
[64] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptol. ePrint Arch., Aztec, London, U.K., Tech. Rep. 2019/953, 2019. [Online]. Available: https://ia.cr/2019/953
[65] S. Setty, "Spartan: Efficient and general-purpose zkSNARKs without trusted setup," in *Proc. CRYPTO*, in Lecture Notes in Computer Science, vol. 12172, D. Micciancio and T. Ristenpart, Eds. Cham, Switzerland: Springer, 2020, pp. 704–737.

HEEWON CHUNG received the B.S. degree in mathematics from the Korea Advance Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2009, and the M.S. and Ph.D. degrees in mathematics from Seoul National University, Seoul, Republic of Korea, in 2017.

From 2016 to 2017, he was a Research Assistant with the Agency for Science, Technology, and Research (A*STAR), Singapore. From 2018 to 2019, he was a Manager with Korea Telecom, Republic of Korea. From 2020 to 2021, he was a Postdoctoral Researcher at Hanyang University, Seoul. Since 2022, he has been a Cryptographic Researcher at Desilo Inc., Seoul. His research interests include resolving the scalability problem and confidential transactions in blockchain using zero-knowledge proofs, especially, verifiable computation (VC) proofs and SNARKs.

KYOOHYUNG HAN received the B.S. and Ph.D. degrees in mathematical sciences from Seoul National University, Seoul, South Korea, in 2013 and 2019, respectively.

Since 2020, he has been a Senior Researcher at Samsung SDS. His current research interests include privacy-enhancing computation, including homomorphic encryption, secure multiparty computation, zero-knowledge proof, and its application to real-world services.

CHANYANG JU was born in Incheon, Republic of Korea, in 1994. She received the B.S. degree in mathematics from Myongji University, Yongin, Republic of Korea, in 2013. She is currently pursuing the Ph.D. degree in applied mathematics with Hanyang University, Seoul, Republic of Korea.

Her research interests include zero-knowledge proof (ZKP) and verifiable computation (VC).

MYUNGSUN KIM received the B.S. degree in computer science and engineering from Sogang University, Seoul, South Korea, in 1994, the M.S. degree in computer science and engineering from Information and Communications University (ICU), Daejeon, in 2002, and the Ph.D. degree in mathematics from Seoul National University (SNU), Seoul, in 2012. He is currently an Assistant Professor with the Department of Mathematics, Gachon University. Previously, he was working with the Department of Information Security, University of Suwon. His research interests include efficient constructions of cryptographic algorithms and their practical applications to real-world solutions.

JAE HONG SEO received the B.S. degree in mathematics from Korea University, in 2004, and the Ph.D. degree from the Department of Mathematical Science, Seoul National University, in 2011.

From 2011 to 2013, he was with the National Institute of Communications and Technology, Japan. From 2013 to 2018, he was with Myongji University, South Korea. Since 2018, he has been an Associate Professor with the Department of Mathematics, Hanyang University, South Korea. His research interests include computational algorithms, cryptology, and their applications to deep learning and blockchain.

Dr. Seo has served as a Program Committee Member for IACR conferences and workshops. He was a recipient of the Sangsan Young Mathematician Award, Korean Mathematical Society, in 2012, and the Young Scientist Award for Excellence, Elsevier&NRF, in 2018. He was the Program Committee (Co-)Chair of APKC 2018 and ICISC 2019.