Interactive Proofs for Rounding Arithmetic

SHUO CHEN¹, (Senior Member, IEEE), JUNG HEE CHEON²,³, (Member, IEEE), DONGWOO KIM⁴, (Associate Member, IEEE), AND DAEJUN PARK⁵

¹Microsoft Research, Redmond, WA 98052, USA
²Department of Mathematical Sciences, Seoul National University, Seoul 08826, South Korea
³IMDARC, Seoul National University, Seoul 08826, South Korea
⁴Western Digital Research, Milpitas, CA 95035, USA
⁵Andreessen Horowitz, Menlo Park, CA 94025, USA

Corresponding author: Dongwoo Kim (Dongwoo.Kim@wdc.com)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korean Government Ministry of Science and ICT (MSIT) under Grant NRF-2022R1A5A6000840.

ABSTRACT Interactive proofs are a type of verifiable computing that secures the integrity of computations. The need is increasing as more computations are outsourced to untrusted parties, e.g., cloud computing platforms. Existing techniques, however, have mainly focused on exact computations, but not approximate arithmetic (e.g., floating-point or fixed-point arithmetic). This makes it hard to apply them to certain types of computations (e.g., machine learning or financial applications) that inherently require approximate arithmetic. In this paper, we present an efficient interactive proof system for arithmetic circuits with rounding gates that can represent approximate arithmetic. The main idea is to reduce the rounding gate into a small sub-circuit without rounding, and reuse the machinery of the Goldwasser, Kalai, and Rothblum’s protocol (also known as the GKR protocol) and its recent refinements. Specifically, we shift the algebraic structure from a field to a ring to better deal with the notion of “digits”, and generalize the original GKR protocol over a ring. Then, we reduce the rounding operation to a low-degree polynomial over a ring, and develop a novel, optimal circuit construction of an arbitrary polynomial to transform the rounding polynomial to an optimal circuit representation. Moreover, further optimization on the proof generation cost for rounding is presented employing a Galois ring. Our experimental results show the efficiency of our protocol for approximate arithmetic, e.g., the implementation performed two orders of magnitude better than the existing system for a nested 128 by 128 matrix multiplication of depth 12 on the 16-bit fixed-point arithmetic.

INDEX TERMS Approximate arithmetic, interactive proof, verifiable computing.

I. INTRODUCTION

Verifiable computing (VC) [2], [3], [4], [5] aims to ensure the integrity of computations performed by an untrusted party. In the cloud computing era, as more computationally heavy tasks are delegated to the cloud, VC is considered as a compelling solution for proving their security. The existing literature has demonstrated the feasibility of several basic primitives, such as addition, multiplication, comparisons [6], set operations [7], and key-value store retrieval [8]. Using these primitives, VC was shown feasible for a number of tasks, including matrix multiplication [9], [10], [11], certain SQL-like queries [12], and state-machine updates [13].

The existing literature, however, has mainly focused on exact computations. For example, they deal with verifying $1.11 \times 2.22 = 2.4642$, but not $1.11 \times 2.22 \approx 2.46$, although $[1.11 \times 2.22]_2 = 2.46$ (where $[\cdot]_2$ denotes rounding to two decimal places). Not supporting approximate arithmetic (e.g., fixed-point or floating-point arithmetic), the existing techniques are hard to apply to certain types of computations (e.g., machine learning) that require approximate arithmetic. In particular, an approach to deal with approximate arithmetic in the existing VC systems over a finite field $\mathbb{F}_p$ is to use the integer scaling method.¹ Specifically, in the integer scaling method, fractional number inputs are multiplied by some

¹We also compare other approaches using boolean circuit representation or commitment scheme in Section VII-E.
scaling factor to be regarded as an element in \( \mathbb{F}_p \) (e.g., from 1.23 to 123), and the size of prime \( p \) is set to be bigger than all intermediate values during the computation. Then, plain multiplication over \( \mathbb{F}_p \) can be used for the approximate arithmetic multiplication, where the computation results need to be interpreted as fractional numbers by dividing them by their accumulated scaling factor. Note, however, that in this approach, the bitsize of intermediate values grows exponentially in the depth \( d \) of an arithmetic circuit, and thus the bitsize of \( p \) should be exponential in \( d \). Therefore, this integer scaling method incurs roughly \( \Theta(2^d) \) cost blowup, which is infeasible in practice.

### A. OUR MAIN RESULT

#### 1) PROBLEM FORMULATION: VERIFIABLE COMPUTING FOR FIXED-POINT ARITHMETIC

Suppose we are given an arithmetic circuit on fixed-point arithmetic with \( \eta \) fractional bits. For simplicity of description, we assume that all inputs and intermediate values during computations are contained in \([0, 1]\), i.e., unsigned fixed-point numbers with no integer bits. Then, the fixed-point arithmetic can be translated to an arithmetic over \( \mathbb{Z}_{2^{2\eta}} \) (i.e., a ring of integers modulo \( 2^{2\eta} \)) with an additional rounding operation, as follows: (i) Every input \( x \in [0, 1] \) is multiplied by \( 2^\eta \) to be regarded as an element of \( \mathbb{Z}_{2^{2\eta}} \); (ii) The fixed-point addition and multiplication translate to usual addition and multiplication over \( \mathbb{Z}_{2^{2\eta}} \) followed by rounding, respectively; (iii) The rounding operation follows each multiplication to extract \( \eta \) most significant bits, i.e., \( x \mapsto \lfloor x/2^\eta \rfloor \).

Now, the problem of verifiable computing for fixed-point arithmetic can be reduced to verifiable computing for arithmetic circuit over \( \mathbb{Z}_{2^{2\eta}} \) with the rounding gates.

#### 2) OUR MAIN RESULT

We present an interactive proof protocol for an arithmetic circuit with the rounding gates, whose complexity is described in the following theorem.

**Theorem 1:** Let \( C \) be a layered arithmetic circuit over \( \mathbb{Z}_{2^{2\eta}} \) where the multiplication gate performs regular multiplication followed by rounding, i.e., \((x, y) \mapsto \lfloor xy/2^\eta \rfloor \). (Thus, \( C \) corresponds to a fixed-point arithmetic circuit with \( \eta \) fractional bits only.) Let us fix \( \eta \). Let \( S \) be the size of \( C \), \( d \) be the depth of \( C \), and \( n \) be the number of inputs. Then, there is an interactive proof protocol \((\mathcal{P}, \mathcal{V})\) for the function \( C : \mathbb{Z}_{2^{2\eta}}^n \rightarrow \mathbb{Z}_{2^{2\eta}} \) where the time complexity of \( \mathcal{P} \) is \( O(dS) \), the time complexity of \( \mathcal{V} \) is \( O(n + d \log d \log dS)^{iii} \), the communication cost is \( O(d \log d \log dS) \), and the soundness is \( O((d \log d \log dS)^2) \). Here the unit cost is \( M(\lambda)M(d) \) where \( M(\ell) \) denotes the cost of an arithmetic operation on \( \ell \)-bit elements (or polynomials of degree \( \ell \)). Note that we do not take into account the \( \eta \) factor in the asymptotic costs, since \( \eta \) is fixed to a small constant.

Our protocol is based on Goldwasser, Kalai, and Rothblum’s interactive proof system (GKR protocol) \([4]\) and its recent refinements \([9], [15], [16]\). The costs of the latest GKR protocol variant \([16]\) (that do not support fixed-point arithmetic) are \( O(S), O(n + d \log S)^{iii} \), and \( O(d \log S) \) for \( \mathcal{P}, \mathcal{V}, \) and communication, respectively. Thus, the additional cost to support fixed-point arithmetic in our protocol is roughly quadratic in the depth of the circuit.

In Table 1, we compare the asymptotic complexity of our protocol with the integer scaling method applied on top of the latest GKR variant. The integer scaling method incurs exponential overhead in circuit depth to deal with fixed-point arithmetic, as mentioned earlier.

#### 3) EXPERIMENTAL RESULT

We conducted experiments to quantify the performance of our protocol. In a moderate laptop, for \( 2^{12} \) number of 16-bit rounding operations, the proof generation took a second, while the proof verification took less than a millisecond. We also experimentally show that our protocol is much more efficient than the integer scaling method. Given a nested \( 128 \times 128 \) matrix multiplication of depth 12 over fixed-point numbers with 16-bits below the decimal point, our refinement took 3 minutes to generate a proof for each matrix multiplication, while the integer scaling method took 2.5 hours for the same task (Section VII). The gap between the two will increase exponentially as the depth of multiplication increases (e.g., the depth often increases to hundreds or thousands in neural network training).

|                | Integer Scaling Method | Ours (Theorem 1) |
|----------------|------------------------|-----------------|
| Circuit eval.  | \( M(\lambda^2)O(S) \) | \( M(d)O(dS) \) |
| Proof gen.     | \( M(\lambda^2)O(S) \) | \( M(\lambda)M(d)O(dS) \) |
| Verification   | \( M(\lambda^2)O(n + d \log S)^{iii} \) | \( O(n + d \log d \log dS) \) |
| Soundness      | \( O((d \log S)/2^{\lambda^2}) \) | \( O((d \log d \log dS)/2^\lambda) \) |

\[ \text{TABLE 1. Comparison of our method to integer scaling method, see Theorem 1 for further description.} \]

\[ ^{iii} \text{The main number of } S \text{ is a positive integer.} \]

\[ ^{\dagger} \text{The most significant bits are extracted considering the output of multiplication as } 2\eta \text{-bit element. Indeed, the proper rounding is } x \mapsto \lfloor x/2^\eta \rfloor \text{ which is easily expressed with } \lfloor . \rfloor \text{ as } x \mapsto \lfloor (x + 2^{1-\eta})/2^\eta \rfloor \text{,} \]

\[ ^{\ddagger} \text{We do not take into account the } \mathcal{V}'s \text{ cost for computing wiring predicate on top of the latest GKR variant.} \]

**B. OUR APPROACH AND INDIVIDUAL SUB-RESULTS**

#### 1) OUR GOAL AND DESIGN RATIONALE

The goal is to construct an interactive proof system for fixed-point arithmetic circuits, i.e., arithmetic circuits with rounding gates. The idea is to represent each rounding gate by a small sub-circuit without rounding, and reuse the machinery of the GKR protocol on it. Specifically, we consider an arithmetic circuit over a ring \( \mathbb{Z}_{p^e} := \mathbb{Z}/p^e\mathbb{Z} \) (i.e., integers in a base \( p \) system of \( e \) digits) with the (floor) rounding operation \((x) \mapsto \lfloor x/p \rfloor\), where \( p \) is a prime and \( e \) is a positive integer. (Note that proper rounding, \( \lfloor x/p \rfloor \), can be represented using floor rounding, i.e., \( \lfloor x/p \rfloor = \lfloor (x + p^{-1})/p \rfloor \)).
reason of our introduction of this ring (instead of the usual finite field) is that the rounding operation can be represented much more efficiently (than in the usual finite fields). Below we explain each of our main technical developments.

2) REDUCING A ROUNDBING OPERATION TO A COMBINATION OF RING OPERATIONS AND DIVISION OPERATIONS (SECTION IV) We present a sub-circuit representation of a rounding gate over the base $p$ system, i.e., over $\mathbb{Z}_{ep}$. At first, we employ the lowest-digit-removal polynomial [17], denoted by $ldr$, which sets the least significant digit to zero, i.e., $ldr : x \mapsto [x/p] \cdot p$. Thus, we can represent the floor rounding operation by $ldr(x)/p$. The key observation is that the degree of $ldr$ is small (less than $ep$), and that the division-by-$p$ operations are naturally verifiable in the paradigm of GKR protocol. We also provide an optimal arithmetic circuit for evaluating $ldr$ and proving the result efficiently as we explain below.

3) OPTIMAL CIRCUIT CONSTRUCTION FOR ARBITRARY UNIVARIATE POLYNOMIAL (SECTION V) In the GKR protocol (as well as our one), a computation of interest needs to be represented in the form of an arithmetic circuit, and the performance of the protocol is substantially affected by the structure of the circuit. To this end, we devised a novel, optimal circuit construction of an arbitrary polynomial for the GKR protocol. The constructed circuit is regular with depth $O(\log d)$ and size $O(d)$ where $d$ is the degree of the polynomial, and it is optimal in that the proof generation complexity is also $O(d)$. Moreover, in case that the same polynomial is evaluated on $m$ parallel inputs, our circuit construction yields a circuit of size $O(md)$ and depth $O(\log d)$, for which the proof generation cost is $O(m\sqrt{d} + d)$, which is sublinear in the circuit size (i.e., the proof generation is even faster than the circuit evaluation!), while the previously best known result is linear [16]. This improvement of the proof generation cost is critical, since such a single-polynomial-multiple-inputs computation is common in data-parallel computing as well as neural network training (e.g., the activation function of each layer is pointwisely applied to a weight vector/matrix).

To achieve this, we carefully redesign Paterson-Stockmeyer algorithm [18] (for polynomial evaluation) into a regular circuit, and manage the (costly) constant multiplication part by one linear-sum gate, $(x_1, \cdots, x_n) \mapsto a_1 x_1 + \cdots + a_n x_n$ which can be efficiently proved and verified via GKR protocol.

4) GENERALIZATION OF THE GKR PROTOCOL OVER A RING (SECTION III) While the original GKR protocol is valid over a finite field, since the domain $\mathbb{Z}_{ep}$ we consider is no longer a field for $e > 1$, we identify a minimal modification to the original protocol to admit a ring (Section III-B), and present its construction for a specific family of rings, i.e., $\mathbb{Z}_{q\ell}$ and its extension rings.

Specifically, the GKR protocol is based on the Sum-Check protocol that in turn is based on the Schwartz-Zippel lemma. However, the Schwartz-Zippel lemma does not hold for a ring in general. To extend the original protocol, we first employ the generalized Schwartz-Zippel lemma [19] over a ring, which restricts the (randomness) sampling set to a subset of the domain such that the difference between any two elements of the subset is not a zero divisor. Then, we show that the Sum-Check protocol as well as the GKR protocol can be extended over a ring by restricting the (verifier’s randomness) sampling set to a subset satisfying the aforementioned property. Moreover, we further identify a stronger condition for the sampling set (Remark 1), the “unit difference” property [20], that is, that the difference between any two elements of the sampling set has an inverse. This stronger condition allows us to employ all cost reduction techniques [9], [14], [16] proposed for the original GKR protocol to our extended protocol.

Hence, the extended protocol enjoys the same computational complexity for each participant with the original one, provided that the unit difference property holds for the sampling set $A$. However, the soundness probability becomes bigger (i.e., worse) than that of the original. It is proportional to $1/|A|$, where the denominator is the size of the sampling set $A$, while it was the size $|\mathbb{F}|$ of the entire domain field for the original protocol. Note, however, that for practical purposes, the soundness probability can be quickly improved by simply having multiple prover-verifier pairs in parallel, which does not affect the overall throughput.

5) OPTIMIZATION OF PROOF GENERATION COST FOR ROUNDBING OPERATION (SECTION VI) Consider an approximate computation on $\mathbb{Z}_{ep}$. The underlying ring $\mathbb{Z}_{ep}$ can be replaced by another ring $\mathbb{Z}_{q\ell}$ with a much smaller prime $q \simeq \sqrt{p}$, via base conversion, that is, converting numbers in the base-$p$ system to the corresponding numbers in the base-$q$ system. Here the advantage of employing $\mathbb{Z}_{q\ell}$ is that the degree of the rounding polynomial in $\mathbb{Z}_{q\ell}$ is much smaller than that of $\mathbb{Z}_{ep}$, which in turn significantly reduces the proof generation cost for rounding. However, employing $\mathbb{Z}_{q\ell}$ leads to sacrificing the soundness of the protocol. To mitigate this dilemma, we proposed a technique that allows us to employ $\mathbb{Z}_{q\ell}$ without compromising the soundness, by exploiting an interesting property of a Galois ring.

Specifically, we employ a Galois ring, $\mathbb{Z}_{q\ell}[\sqrt{\ell}]/f(t)$, where $f(t)$ is a monic irreducible polynomial, in the proof generation and verification phases, while we keep using $\mathbb{Z}_{q\ell}$ in the circuit evaluation phase. This allows us to employ a smaller prime $q \sim \sqrt{p}$ where $d$ is the degree of $f(t)$.

ivThe converted number may be marginally different from the original, but such an inaccuracy is acceptable in approximate computation such DNN training.
Employing a smaller prime leads to further reducing the size of the rounding circuit, since the degree of the lowest digit removal polynomial drastically decreases from $ep$ into $ed \approx ed \sqrt{b}$. Note that the soundness probability is not compromised at all with the smaller prime $q$, because the extension ring yields a sampling set of similar size, $q^d \approx p$, to that of the original one (Theorem 5).

However, there is a cost overhead when employing a Galois ring, since the operations on a Galois ring become more expensive as its degree $d$ increases. Thus, having a too small prime $q$ may offset the aforementioned cost benefit. Nevertheless, one can find an optimal $q$ given a set of parameters, and our experiment showed that two orders of magnitude performance improvement can be made by finding such a sweet spot (Section VII).

C. RELATED WORK

The problem of delegating computation with securing integrity has been extensively studied in both theory and practice perspectives. Here we mainly focus on the general-purpose protocols and systems that aim to be practical.

1) INTERACTIVE PROOFS: GKR PROTOCOL AND ITS REFINEMENTS

Goldwasser, Kalai, and Rothblum [4] proposed an interactive proof protocol (also known as GKR protocol) that runs in polynomial time. For a layered arithmetic circuit of size $S$ and depth $d$, the prover of their protocol runs in time poly($S$), and the verifier runs in time poly($d, \log S$).

Several refinements of the GKR protocol have been proposed to improve the cost of the protocol, especially the prover’s cost. Cormode, Mitzenmacher, and Thaler [9] presented a refinement of the GKR protocol (hereafter, CMT) that allows the prover to run in $O(S \log S)$. Thaler [14] further improved the protocol, which allows the prover to run in $O(S)$ for a circuit with a “highly” regular wiring pattern. Subsequently, it has been shown that the prover’s cost can be reduced when a circuit is composed of many parallel copies of subcircuits. Specifically, the prover’s cost is reduced to $O(S \log S_v)$ in [12] and [15], and further reduced to $O(S + S_v \log S_v)$ in [21], where $S_v$ is the size of a subcircuit. Recently, Xie et al. [16] proposed a refinement that allows the prover to run in $O(S)$ for an arbitrary circuit. Although being asymptotically equivalent, Thaler’s refinement [14] performs better than Xie et al.’s [16] for a “highly” regular circuit.

On the other hand, substantial efforts have been made to support more operations than the plain field arithmetic. Vu et al. [6] proposed an extension of CMT that supports inequalities by augmenting a circuit with additional verification logic and auxiliary inputs to be fed by the prover. However, their approach suffers from a significant overhead of the verifier due to the irregularity of their augmented circuit, which needs to be amortized by batching verifications (i.e., verifying the same circuit against many different inputs at the same time) for practical purposes. Zhang et al. [12] improved this by combining CMT with a verifiable polynomial delegation scheme, and showed that an arithmetic circuit with auxiliary inputs can be verified efficiently without batching strategies.

There are other lines of refinement work such as supporting “streaming” verifiers [22], [23] that run in a limited space; employing hardware accelerators such as ASICs and GPUs [21], [24], [25]; and supporting zero-knowledge arguments [16], [26], [27].

Note, however, that no existing interactive proof systems support a verifiable rounding operation efficiently, to the best of our knowledge, which is critical to deal with an approximate arithmetic circuit with a large depth. Though Vu et al. [6]’s approach with auxiliary input can support rounding operation in principle, it forces verifier’s cost to be linear in the number of rounding operations, since the verifier must check the auxiliary input which is at least as many as the number of rounding operations multiplied by the bitsize of rounding input. Therefore, it does not provide efficient verification of rounding operations. Zhang et al. [12], [28] and Wahby et al. [26]’s approach resolves this problem using commitment scheme. However, due to the use of commitment, their system became an argument that is secure only against computationally bounded dishonest prover, and it does not provide post-quantum security due to the specific polynomial commitment scheme based on pairing groups. Also, since the computational cost for commitment is higher than plain operations of interactive proofs, performance of our system for rounding operation can be better or comparable to their’s when the bitsize of rounding operation is not large (see Discussion VII-E).

2) ARGUMENTS: NON-INTERACTIVENESS AND ZERO-KNOWLEDGE

Argument systems are different from interactive proofs in that they are secure only against computationally bounded dishonest provers. Employing cryptographic primitives, they can provide versatile properties such as non-interactivity, public verifiability, and zero-knowledge proofs. However, the use of expensive cryptographic primitives incurs a significant overhead to the prover’s cost.

There have been substantial efforts [3], [29], [30], [31] of developing argument systems based on probabilistically checkable proofs (PCPs) [32], [33], especially ones called “short” PCPs. Although being asymptotically similar to their counterparts (that we will explain below), the PCP-based arguments involve large constants, being too expensive to be practical.

On the other hand, there have been much efforts on developing argument systems without using the short PCPs. Setty et al. [11], [34], [35] proposed argument systems based on linear PCPs [36], where their systems were shown to achieve a practical performance in the batch verification.

\footnote{Although, in theory, the existing work can support rounding by degenerating to much verbose Boolean circuits, it is highly inefficient to implement such Boolean circuits in practice.}
setting. Gennaro et al. [37] introduced quadratic arithmetic programs (QAPs), a novel efficient encoding of computations for verifiable computing and even for zero-knowledge succinct non-interactive arguments (zkSNARKs). Much of improvements have been proposed [10], [38], [39], [40] to make the proving time practical and several works have been proposed via various techniques along with various trade-offs in prover time, verifier time, and supporting functionalities (universality, without trusted setup, etc) [41], [42], [43], [44], [45], [46], [47], [48], [49], [50]. Still, such argument systems provide a prover with quasi-linear cost or inefficient verifier asymptotically. All of those work, however, are applicable only to computations over a finite field, and is not efficient for representing fixed point arithmetic accompanying rounding operations.

There also has been substantial work [13], [38], [39], [51], [52] to extend the coverage of verifiable computing to a more generalized form of computations. Essentially, they developed a “compiler” that translates C-like programs (with e.g., memory accesses and control flows) into corresponding arithmetic circuits (or algebraic constraints). However, their approaches often do not efficiently scale, due to the blowup in the size of generated circuits. On the other hand, [34], [35] presented an encoding of rational numbers in a finite field, but still did not support rounding, suffering from the same problem (i.e., the exponential blowup of the field size) with the integer scaling method described in Section VII.

D. TOWARD VERIFIABLE AI

1) MOTIVATION

This paper is motivated by our vision about verifiable AI computation. Specifically, consider a deep neural network (DNN) training task: it is a computation that takes a set of samples, and produces an output model represented as one or more matrices. The computation often takes hours or even days. Should the training set be poisoned or the training machine(s) be compromised, the output model would have potentially devastating hidden behaviors [53]. Unlike programming bugs or malicious code, compromised AI models are extremely difficult to detect, because the models are nothing but some matrices. However, if verifiable AI computation is achieved, we will be able to trust a model by only trusting the fundamental mathematics, not any other factors such as human operators, program, or platform doing the training.

AI computations are many orders of magnitude heavier and involve more challenging operations than the aforementioned primitives in the VC literature, so it could be a long journey to fully realize the vision. Specifically, DNN training processes mainly consist of an overwhelmingly large amount of computing matrix multiplication and a relatively small amount of computing various non-linear functions such as ReLU, max-pooling, and softmax, where all the operations are performed using approximate arithmetic such as fixed-point or floating-point arithmetic. Here the following fundamental operations are required for DNN training computations: (1) approximate arithmetic, (2) comparison (for ReLU and max-pooling), and (3) natural exponentiation $e^x$ (for softmax and sigmoid). While the comparison operation was shown to be feasible in [6], and $e^x$ can be approximated as a (piecewise) polynomial, approximate arithmetic is not supported in existing VC schemes (e.g., acknowledged in [6] and [34]), to the best of our knowledge.\textsuperscript{vii}

2) APPLICATION TO VERIFIABLE AI

We believe that this work is an important step toward the vision of verifiable AI computations. Specifically, the DNN training iterates the forward and backward passes over the sequence of layers, where each layer computation (in both forward and backward passes) consists of matrix multiplication and nonlinear function application on approximate arithmetic. Without the ability of rounding, the number of digits of the computation results will keep increasing and exceed the limit. Thus the existing VC approaches are not capable in the AI space. Our approach gives a theoretical feasibility for these computations. In addition, it also sheds light on the real-world performance – as shown in Section VII, matrix multiplication on the fixed-point arithmetic can be efficiently supported by our scheme.

Among the nonlinear functions, the ReLU and maxpooling functions can be represented in an (approximate) arithmetic circuit by using the comparison operation [6], [12]. The sigmoid and tanh functions were shown to be effectively approximated as a polynomial [55] with achieving a sufficient accuracy, while such a polynomial can be efficiently represented in a circuit by using our optimal circuit construction. The softmax function requires to compute the natural exponentiation function $e^x$, which can be also approximated as a polynomial for $x \leq 0$, using the input normalization [54], as mentioned in Section I.

Moreover, multiple iterations can be “squashed” [21] into a wide and shallow circuit by laying identical subcircuits of a single iteration side by side. This squashing can drastically reduce the depth of a circuit, which can significantly improve the protocol’s performance [15], [21] at the cost of communication overheads. Finally, the protocol performance can be further improved by using hardware accelerators such as GPUs [15], [25] and ASICs [21], [24].

3) OTHER APPLICATIONS OF INDIVIDUAL RESULTS

As mentioned in the introduction, the individual technical results that we developed for the verifiable rounding operation have their own applications as well. First, our generalized GKR protocol can be used in other settings where rounding is not necessarily involved. For example, a ring $\mathbb{Z}_{q^f}$ has a

\textsuperscript{vii}In particular, it is a well-known practice to use an input normalization for $e^x$ [54] to avoid overflow when computing softmax or sigmoid, in which case $x \leq 0$, and thus a (piecewise) polynomial approximation of $e^x$ for $x \leq 0$ can provide a sufficient precision since $e^x$ is converging in the negative domain.\textsuperscript{vi}

\textsuperscript{vi}Although the technique used for comparison [6] has a potential to be used for rounding, their approach inherently introduces a significant overhead which is not involved in our approach. See Section I-C for more details.
nice property that addition and multiplication on \( \mathbb{Z}_p \) are equivalent to that of the \( e \)-bit machine integer arithmetic when \( p = 2 \), including the “wrapping-around” behavior in case of overflow (e.g., “4 + 4 \equiv 0” in both \( \mathbb{Z}_{2^3} \) and the 3-bit (unsigned) machine integer arithmetic). Thanks to this property, for certain computations that inherently require the modular arithmetic (e.g., ones in cryptography implementations), one can construct arithmetic circuits of such computations at no extra cost.\(^{\text{viii}}\) Note that to admit such computations with the original GKR protocol, one needs to additionally develop a circuit representation of the modulo reduction, i.e., \( x \mapsto x \mod 2^e \), which incurs additional overheads in protocol performance due to the circuit size blowup.

On the other hand, our optimal circuit construction is applicable to the original GKR protocol (and its variants) as well, since it is not specific to the underlying algebraic structure. That is, when a given computation involves evaluation of certain polynomials, our circuit construction scheme can be used to optimize the protocol performance.

II. PRELIMINARIES

In this section, we review a number of basic concepts about verifiable computing (VC). In a VC scenario, a customer delegates a computation to an untrusted platform, and wants to be assured that the computational result is correct. The untrusted platform is called the Prover \( \mathcal{P} \), and the customer is called the Verifier \( \mathcal{V} \). As a motivating scenario of our work, \( \mathcal{V} \) submits a DNN training job to \( \mathcal{P} \), which may take days to run. The goal of \( \mathcal{V} \) is to verify the correctness of the computational result provided by \( \mathcal{P} \), without re-running the same job by \( \mathcal{V} \) itself.

Next, we delve a little deeper into the type of VC we discuss in this paper, the interactive proof protocol.

A. INTERACTIVE PROOF PROTOCOL

We start with a definition of the interactive proof protocol for a function \( f \) as follows.

**Definition 1 (Interactive Proof Protocol for \( f \) [9], [14]):** Consider a prover \( \mathcal{P} \) and a verifier \( \mathcal{V} \) who wishes to compute a function \( f: X \rightarrow Y \). For an input \( x \in X \) chosen by \( \mathcal{V} \), \( \mathcal{P} \) gives the claimed output \( y \) to \( \mathcal{V} \). Then, they exchange a sequence of messages and \( \mathcal{V} \) accepts or rejects.

- **Completeness.** For all inputs \( x \in X \), if \( \mathcal{P} \) follows the protocol and \( y = f(x) \), then \( \mathcal{V} \) accepts \( \mathcal{V} \) accepts \( \{1\} \).
- **\( \delta \)-Soundness.** For all inputs \( x \in X \), if \( y \neq f(x) \), then \( \mathcal{V} \) rejects with probability \( \delta \).

We will call \( \delta \) the soundness probability bound. If \( \mathcal{P} \) and \( \mathcal{V} \) exchange \( r \) messages in total, we say the protocol has \( \lceil r/2 \rceil \) rounds.

\(^{\text{viii}}\) In this case, the optimization via a Galois ring (Section VI) is needed to secure a sampling set that is large enough for the protocol soundness. Moreover, the same technique is applicable to a more general ring \( \mathbb{Z}_n \) for an arbitrary integer \( n \) by using the Chinese remainder theorem, i.e., reducing operations on \( \mathbb{Z}_n \) to that of \( \prod_i \mathbb{Z}_{p_i} \) where \( \prod_i p_i^{e_i} \) is the prime factorization of \( n \). Note that the modular arithmetic on \( \mathbb{Z}_n \) is commonly used in, e.g., the lattice-based cryptography [56].

The GKR protocol is an interactive proof protocol for the evaluation of a layered arithmetic circuit over a finite field \( \mathbb{F} \) where the circuit is composed of addition and multiplication gates (over \( \mathbb{F} \)) of fan-in \( 2 \).

B. SCHWARTZ-ZIPPEL LEMMA

We first recall the Schwartz-Zippel lemma, the Sum-Check protocol, and the Multilinear Extension which constitute the GKR protocol.

**Lemma 1 (Schwartz-Zippel [57]):** Let \( \mathbb{F} \) be a field, and \( f : \mathbb{F}^n \rightarrow \mathbb{F} \) be an \( n \)-variate nonzero polynomial of total degree \( \sum d_i \) for each variable \( D \). Then on any finite set \( A \subseteq \mathbb{F} \) with \( |A| \leq |\mathbb{F}| \),

\[
\mathbb{P} \left[ f(\vec{x}) = 0 \right] \leq \frac{|D|}{|A|^{\delta}}.
\]

Note that the lemma implies that two different polynomials can coincide at only tiny fractions of points. It contributes to soundness of the following sum-check protocol and thus the GKR protocol.

C. SUM-CHECK PROTOCOL

The Sum-Check protocol [58] is an interactive proof protocol for a specific summation function \( S(f) \) over a field \( \mathbb{F} \) as follows.

**Theorem 2: (Sum-Check Protocol [58])** Let \( \mathbb{F} \) be a finite field. Let \( f : \mathbb{F}^n \rightarrow \mathbb{F} \) be an \( n \)-variate polynomial of degree at most \( d < |\mathbb{F}| \) in each variable. The Sum-Check protocol is an interactive proof protocol with soundness \( |\mathbb{F}|^{-\frac{\delta}{|\mathbb{F}|}} \) for the function:

\[
S(f) := \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} f(x_1, x_2, \ldots, x_n).
\]

**Protocol description:** the protocol proceeds in \( n \) rounds. In the first round, \( \mathcal{P} \) sends the value \( S(f) \), and a polynomial

\[
f_1(t) := \sum_{(x_2, x_3, \ldots, x_n) \in \{0,1\}^{n-1}} f(t, x_2, x_3, \ldots, x_n).
\]

\( \mathcal{V} \) checks if \( f_1(0) + f_1(1) = S(f) \), and rejects otherwise. In the \( i \)-th (\( 2 \leq i \leq n \)) round, \( \mathcal{V} \) chooses \( r_{i-1} \) randomly from \( \mathbb{F} \), and sends it to \( \mathcal{P} \). In response, \( \mathcal{P} \) sends a polynomial

\[
f_i(t) := \sum_{(x_{i+1}, \ldots, x_n) \in \{0,1\}^{n-i}} f(r_1, \ldots, r_{i-1}, t, x_{i+1}, \ldots, x_n).
\]

\( \mathcal{V} \) checks if \( f_{i-1}(r_{i-1}) = f_i(0) + f_i(1) \), and rejects otherwise. After the final \( v \)-th round, \( \mathcal{V} \) accepts if \( f_v(r_v) = f(r_1, r_2, \ldots, r_v) \) for a random element \( r_v \in \mathbb{F} \), and rejects otherwise.

**Proof:** For soundness condition, see Appendix B-A. □

Note that the soundness of sum-check protocol is based on Schwartz-Zippel lemma (Lemma 1); if dishonest \( \mathcal{P}' \) sent wrong value \( S'(f) \neq S(f) \), he must send a polynomial \( f'_1(t) \neq f_1(t) \) resulting in \( f'_v(r_v) \neq f(v) \) with high probability. Repeatedly, the lemma forces \( \mathcal{P}' \) to send polynomials \( f'_1(t) \neq f_1(t) \) and to be rejected (at the final round) with high probability.

Note that the sum-check protocol enable \( \mathcal{V} \) to reduce the verification task on \( S(f) \) to that on the evaluation of \( f \) on one random point. It is the core functionality of sum-check in the GKR protocol.
D. MULTILINEAR EXTENSION (MLE)

Lemma 2 (Multilinear Extension [9]): Given a function $V : \{0, 1\}^n \rightarrow F$, there exists a unique multilinear polynomial $\tilde{V}(\tilde{x}) : \{0, 1\}^n \rightarrow F$ extending $V$, i.e., $\tilde{V}(x) = V(x)$ for all $x \in \{0, 1\}^n$. We call $V$ the multilinear extension (MLE) of $V$ over $F$.

The existence of multilinear extension $\tilde{V}$ is guaranteed from the following construction.

$$\tilde{V}(x_1, x_2, \ldots, x_n) := \sum_{\tilde{b} \in \{0, 1\}^n} V(\tilde{b}) \cdot \prod_{i=1}^n [(1 - b_i)(1 - x_i) + b_i x_i] .$$

The uniqueness of multilinear extension is also straightforward, see Appendix B-B. In GKR protocol, the output of each layer in the circuit give rise to the unique multilinear extension.

E. GKR PROTOCOL

Now we describe the GKR protocol\(^{15}\) which is an interactive proof protocol for the evaluation of a layered arithmetic circuit over a finite field $F$. We only give an overview of the protocol, and a detailed description can be found in [4], [15] or in Section III-B.

1) OVERVIEW

Assume we are given a layered arithmetic circuit (over $F$) of input $n$, depth $d$, of size $S$, and of fan-in 2. Each layer is composed of gates outputing addition or multiplication of two inputs. The layers are numbered in a way that output layer is 0, input layer is $d$, and gates of $i$-th layer take as input the output of gates in $i + 1$-th layer. Let $S_i$ denotes the size of $i$-th layer, and assume it is a power of 2, i.e., $S_i = 2^i$ for simplicity. We can number each gate of $i$-th layer with a binary string in $\{0, 1\}^i$, and it defines a function $V_i : \{0, 1\}^i \rightarrow F$ relating the given binary string to output of the corresponding gate. Let $\tilde{V}_i$ be the multilinear extension (MLE) of $V_i$, then there exists an interesting relation between MLEs defined from adjacent layers as follows [59]: (We omit the vector notation, e.g., $\tilde{z}$, $\tilde{\omega}_1$, $\tilde{\omega}_2$)

$$\tilde{V}_i(z) = \sum_{(\omega_1, \omega_2) \in \{0, 1\}^{S_{i+1}}} [(\text{add}_i(z, \omega_1, \omega_2)(\tilde{V}_{i+1}(\omega_1) + \tilde{V}_{i+1}(\omega_2))]$$

$$+ \sum_{(\omega_1, \omega_2) \in \{0, 1\}^{S_{i+1}}} [(\text{mult}_i(z, \omega_1, \omega_2)(\tilde{V}_{i+1}(\omega_1) \cdot \tilde{V}_{i+1}(\omega_2))]$$

where $\text{add}_i$ (or $\text{mult}_i$) is a MLE of a function which is 1 only if the input binary strings indicate an addition (or multiplication) gate and its corresponding two gates providing inputs, and 0 otherwise. We call $\text{add}_i$ (and $\text{mult}_i$) wiring predicates [9].

Now, the GKR protocol proceeds in layer by layer starting from the output layer. $V$ having an output of the circuit, gets a claim $\tilde{V}_0(z_0) = v_0$ evaluating $\tilde{V}_0$ on random point $z_0$. Then, she reduces the claim to $\tilde{V}_1(z_1) = v_1$ executing the sum-check protocol on the relation of MLEs we described. Continuing this process layer by layer, she finally gets a claim that $\tilde{V}_d(z_d) = v_d$, and checks if it is correct by evaluating $V_0$ defined with her inputs.

2) COMPUTATIONAL COMPLEXITY AND SOUNDNESS

A number of refinements [6], [9], [14], [16], [21] on GKR protocol have reduced the computational complexity cost of the protocol. Finally, the cost of $P$ and $V$ in the number of operations over $F$, and the communication cost $C$ in the number of elements of $F$ are as follows [16]:

$$P : O(S), \quad V : O(n + d \log S), \quad C : O(d \log S) \quad (2)$$

where $S$ and $d$ denote the size and depth of the circuit, respectively, and $n$ denotes the number of inputs. The soundness probability $\lambda_s$ is bounded by $(7d \log S + \log n)/|F|$.

We note that the prover’s cost can be broken down into the circuit evaluation cost and the proof generation cost, and we will show certain circuits for which the proof generation cost is smaller than the circuit evaluation cost (Section V).

F. NOTATION AND COST MODEL

In this paper, $Z$, $Z_N$, and $F$ denote the ring of integers, the ring of integers modulo a positive integer $N$, and a finite field, respectively. Also, all logarithms are base 2. The (computational) cost of $P$ or $V$ is measured by the number of arithmetic operations over the corresponding domain, such as $F$ or $Z_N$. The communication cost is measured by the number of elements of the corresponding domain.

III. GENERALIZATION OF GKR PROTOCOL OVER A RING

In this section, we show that the GKR protocol can be applied to an arithmetic circuit over a ring, a more general algebraic structure than a field. Throughout this paper, we refer a ring $R$ to a finite commutative ring with the multiplicative identity 1. It is similar to a field in that it has two operations, i.e., addition and multiplication that is distributive over addition, an additive identity 0, and a multiplicative identity 1. It also has an additive inverse for every element, but does not necessarily have a multiplicative inverse, in contrast to a field. A zero divisor of a ring $R$ is an element $x \in R$ which divides 0, i.e., there exists a nonzero element $y \in R$ such that $xy = 0$. An integral domain is a ring that has no zero divisors other than 0. Typical examples of ring are $Z$ (integers) and $Z_N$ (integers modulo $N$). Note that $Z$ is an integral domain, and $Z_N$ is a field if $N$ is a prime, but is not even an integral domain otherwise.

\(^{15}\)An n-variate polynomial $f(x_1, \ldots, x_n) : \mathbb{F}^n \rightarrow \mathbb{F}$ is called multilinear if it is linear in each variable, e.g., $f(x_1, x_2, x_3) = x_1 x_2 x_3 + x_2 x_3 + x_3$.

\(^{16}\)In particular, we describe the simplest form [59] by Thaler.
A. SCHWARTZ-ZIPPEL LEMMA AND SUM-CHECK PROTOCOL OVER A RING

Since the original GKR protocol is based on the Schwartz-Zippel lemma (Lemma 1), the starting point of generalization is also the lemma. Here we exploit more generalized form given by Bishnoi et al. [19] as follows.

Lemma 3 (Generalized Schwartz-Zippel [19]): Let $R$ be a ring, and $f : R^n \rightarrow R$ be an $n$-variate nonzero polynomial of total degree (the sum of degrees of each variable) $D$ over $R$. Let $A \subseteq R$ be a finite set with $|A| \geq D$ such that $\forall x \neq y \in A$, $x - y \in R$ is not a zero divisor. Then, $Pr_{x \in A^n}[f(x) = 0] \leq \frac{D}{|A|}$. We will call $A$ a sampling set.

Proof: Appendix B-C

This lemma guarantees that the identity check of a polynomial over $R$ can be done similarly as in a field if we sample the random points from a sampling set $A \subseteq R$.

Example 1: Let $R = \mathbb{Z}_{pq}$ for an odd prime $p$, and $A = \{0, 1, 2, \ldots, p - 2, p - 1\}$. Then, $A$ is the sampling set of Lemma 3, since $\forall x \neq y \in A, x - y \in \{-p + 1, \ldots, -1, 1, \ldots, p - 1\}$ is not a zero divisor. Note that zero divisors of $R$ are exactly the nonzero multiples of $p$. The set $A$ is maximal in that $a \in A$ implies $a + np \notin A$ for any nonzero integer $n$.

Now we can naturally extend the Sum-Check protocol (Theorem 2) over $R$, only restricting the random points chosen by $\mathcal{V}$.

Theorem 3 (Generalized Sum-Check Protocol): Let $R$ be a ring, $f : R^n \rightarrow R$ be an $n$-variate polynomial of degree at most $d$ in each variable. Let $A \subseteq R$ be a sampling set of Lemma 3 such that $d < |A|$. Then, the Generalized Sum-Check protocol where $\mathcal{V}$ chooses each random point $r_i$ from $A$, is an interactive proof protocol with soundness $\frac{nd}{|A|}$ for the function:

$$S(f) := \sum_{x_1 \in \{0, 1\}} \cdots \sum_{x_n \in \{0, 1\}} f(x_1, x_2, \ldots, x_n).$$

Proof: Appendix B-D.

Note that the soundness probability is $\frac{nd}{|A|}$ in contrast to $\frac{nd}{|R|}$ in Theorem 2.

Remark 1 (Additional Condition for Efficient Specification of $f(x_1)$): In the $i$-th round of the Sum-Check protocol, (honest) $\mathcal{P}$ should provide

$$f_i(t) := \sum_{(a_1, \ldots, a_n) \in \{0, 1\}^{n-i}} f(r_1, \ldots, r_{i-1}, t, a_{i+1}, \ldots, a_n)$$

to $\mathcal{V}$. While the $f_i(t)$ is specified by evaluations of it on $\deg f + 1$ distinct points from $A$, the distinct points must satisfy the condition that all of their differences have inverses in $R$ for Lagrange interpolation to be available. It is a stronger condition than that of $A$. Note that, in all specific rings we use in this paper, the sampling set $A$ also satisfies that stronger condition.

Example 2: Let $R = \mathbb{Z}_{pq}$ for an odd prime $p$, and $A = \{0, 1, 2, \ldots, p - 2, p - 1\}$ as Example 1. Then, $A$ also satisfies the stronger condition mentioned above, i.e., $\forall x \neq y \in A, x - y$ has a multiplicative inverse in $R$. It follows from the fact that all elements of $R = \mathbb{Z}_{pq}$ other than multiples of $p$ have a multiplicative inverse in $R = \mathbb{Z}_{pq}$.

B. GKR PROTOCOL OVER A RING

Now we present a generalized GKR protocol over $R$. We can see that the original GKR protocol can be applied to an arithmetic circuit over $R$ by restricting random points required in the protocol to the sampling set $A$ of Lemma 3. Below we clarify and validate the modification made in each step of the protocol.

1) MULTILINEAR EXTENSION AND INITIAL STEP

We first need to ensure that the existence and uniqueness (Lemma 2) of Multilinear Extension (MLE) $\tilde{V} : R^\mu \rightarrow R$ extending a function $V : \{0, 1\}^\mu \rightarrow R$. It follows from the fact that the proof of Lemma 2 is valid in $R$ since it exploits only properties (i.e., commutativity and distributivity of addition and multiplication, and existence of the multiplicative inverse 1) that hold in $R$ as well. At the initial step, $\mathcal{V}$ reduces the task of checking output values to that of checking $\tilde{V}_0(z_0) = v_0$ where $\tilde{V}_0$ is a MLE of the output values. In the original protocol, the reduction is valid by Lemma 1. In the generalized protocol, the reduction is valid by Lemma 3, provided that $\mathcal{V}$ samples the random point $z_0$ from the set $A$ of Lemma 3.

2) APPLYING SUM-CHECK PROTOCOL

We already have shown that the Sum-Check protocol is valid in $R$ as well by Theorem 3. Therefore, reducing the task of checking $\tilde{V}_i(z_i) = v_i$ to that of checking both $\tilde{V}_{i+1}(a^*_1) = v_{i+1,1}$ and $\tilde{V}_{i+1}(a^*_2) = v_{i+1,2}$ can be done using the generalized Sum-Check protocol. Note that $\mathcal{V}$ samples each random point from the set $A$ in the generalized Sum-Check protocol.

3) REDUCTION TO VERIFICATION AT A SINGLE POINT AND FINAL STEP

Reducing the task of checking both $\tilde{V}_{i+1}(a^*_1) = v_{i+1,1}$ and $\tilde{V}_{i+1}(a^*_2) = v_{i+1,2}$ to that of checking $\tilde{V}_{i+1}(z_{i+1}) = v_{i+1}$ requires the generalized Schwartz-Zippel lemma (Lemma 3), and $\mathcal{V}$ must evaluate the polynomial $h(t) := \tilde{V}_{i+1}(l(t))$ on $t_{i+1}$ that is randomly sampled from the set $A$, to compute $\tilde{V}_{i+1}(z_{i+1}) = h(t_{i+1})$. Finally, $\mathcal{V}$ having $\tilde{V}_d(z_d) = v_d$ checks if it is correct by evaluating the MLE $\tilde{V}_d$ of the input values on $z_d$ by herself.

4) COMPLEXITY AND SOUNDNESS

Note that the computational cost of the generalized protocol is the same with that of the original protocol (Equation 2) except that the cost is measured by the number of operations or elements of $R$ instead of $F$. The cost reduction techniques [6], [9], [14], [16] proposed in refinements of GKR protocol are also applicable if $R$ satisfies the additional condition introduced in Remark 1.

\[xii\]If $x \in \mathbb{Z}_{pq}$ is not a multiple of $p$, $\gcd(x, p^2) = \gcd(x, p) = 1$, and $ax + by^2 = 1$ for some $a, b \in \mathbb{Z}$, i.e., $a \pmod{p^2} \in \mathbb{Z}_{pq}$ is a multiplicative inverse of $x$. 
Soundness of the generalized GKR protocol follows from that of the generalized Sum-Check protocol. Hence, it has the same soundness with the original one except that \(|P|\) is substituted by \(|A|\) (see following Theorem 4).

**Theorem 4 (GKR Protocol Over \(R\)):** Let \(R\) be a finite ring, and \(C : R^d \to R\) be an arithmetic circuit of size \(S\), depth \(d\), with \(n\) inputs over \(R\). Let \(A\) be the sampling set of \(R\) in Lemma 3. The generalized GKR protocol is an interactive proof protocol for the evaluation of \(C\) with soundness \((7d \log S + \log n)/|A|\). The cost of the generalized GKR protocol is the same with that of the original protocol described in Section II-E, if \(A\) satisfies the stronger condition of Remark 1. The complexity of the generalized GKR protocol is the same with that of the original protocol described in Section II-E.

**IV. VERIFIABLE ROUNDING OPERATION**

In this section, we explain how to support the rounding operation on top of the generalized GKR protocol described in Section III. As explained in Section I, we consider an approximate arithmetic circuit over a ring \(\mathbb{Z}_{\text{ap}}\) (i.e., integers in the base-\(p\) system) where \(p\) is a prime and \(e > 1\), and the rounding gate that performs the (floor) rounding: \(x \mapsto \lfloor x/p \rfloor\).xiij Like closely related previous work [4], [9], [14], we assume that the given circuit is layered. For the simplicity of the presentation, we also assume that the given circuit is structured to have rounding layers each of which consists solely of rounding gates, while the other layers have only addition and multiplication gates.xiv

The idea is to replace each rounding gate with a combina-

tion of plain arithmetic gates, and use our generalized GKR protocol over \(\mathbb{Z}_{\text{ap}}\). Specifically, we employ a low-degree polynomial \(\text{ldr}(x)\) such that \(\lfloor x/p \rfloor = \text{ldr}(x)/p\), where \(\text{ldr}(x)\) can be represented as a circuit over addition and multiplication gates. (Later, in Section V, we will provide an optimal circuit construction for arbitrary polynomials including \(\text{ldr}(x)\).)

Then, the reduced circuit can be replaced with the circuit of \(\text{ldr}(x)\) followed by a division-by-\(p\) gate, \(x \mapsto x/p\). Below we will explain what is the polynomial \(\text{ldr}(x)\), and how to verify the division-by-\(p\) gate in our generalized GKR protocol.

**A. LOWEST-DIGIT-REMOVAL POLYNOMIAL OVER \(\mathbb{Z}_{\text{ap}}\)**

Chen and Han [17] recently showed the existence of a polynomial over \(\mathbb{Z}_{\text{ap}}\) that sets the input’s lowest-digit to zero. They also provided an exact construction of such polynomial.\[\text{ldr}(x) = x - (x \mod p) \mod p^e,\]

where \((x \mod p) \in \{0, 1, \ldots, p - 1\}\).

**B. VERIFICATION OF DIVISION-BY-\(p\) LAYER**

As mentioned earlier, the rounding operation \((t \mapsto \lfloor t/p \rfloor)\) can be represented as \(t \mapsto \text{ldr}(t)/p\). Here the problem is that division is not admitted in an arithmetic circuit over a ring (thus not in the generalized GKR protocol over a ring) in general. However, in \(\text{ldr}(x)/p\), the division is always well-defined, since the result of \(\text{ldr}(x)\) is guaranteed to be a multiple of \(p\), where \(p\) is constant. Also, as mentioned earlier, the given circuit is assumed to have a separate rounding layer that consists solely of rounding gates. Thus, the reduced circuit will have a separate division-by-\(p\) layer that also consists solely of the division-by-\(p\) gates, and we have the following equation:

\[\tilde{V}(z) = \tilde{V}_{i+1}(z)/p\]

where \(\tilde{V}_i\) denotes the MLE of outputs (and inputs, resp.) of the division-by-\(p\) layer. Now, in the generalized GKR protocol, the verifier verifies the outputs of the division-by-\(p\) layer by reducing the verification task of \(\tilde{V}_i(r)\) to, the verification task of \(\tilde{V}_{i+1}(r) = pv\). This reduction enjoys perfect soundness, since for \(\tilde{V}_i(r) \neq \tilde{V}_{i+1}(r)\), we have \(\tilde{V}_{i+1}(r) = p\tilde{V}_i(r) \neq p\tilde{V}_i(r) \equiv \tilde{V}_{i+1}(r) \mod p^e\).

**Remark 2 (Modulus Change at Division-by-\(p\) Layer):**

Note that the codomain of \(\tilde{V}_i\) is \(\mathbb{Z}_{p^e-1}\), while the codomain of \(\tilde{V}_{i+1}\) is \(\mathbb{Z}_{ap}\). That is, the outputs of each rounding layer should be regarded as an element of \(\mathbb{Z}_{p^e-1}\) while the inputs are elements of \(\mathbb{Z}_{ap}\). This is because \(t = ap + b \in \mathbb{Z}_{ap}\) represents \((ap + b) + np^e \in \mathbb{Z}\) for some \(n \in \mathbb{Z}\) where \(0 \leq b < p\), while \([t/p] \equiv a + np^e-1 \in \mathbb{Z}\) is represented by \(a \in \mathbb{Z}_{p^e-1}\).

**V. OPTIMAL CIRCUIT CONSTRUCTION FOR POLYNOMIAL EVALUATION**

In this section, we present a novel, optimal circuit construction for proof and verification of polynomial evaluation with the GKR protocol. The circuit has an optimal depth, and is regular so that a prover (and a verifier) enjoys an optimal cost (and high efficiency) when proving (and verifying) the circuit via the GKR protocol. It has an additional advantage when applied to the parallel evaluation of the same polynomial on multiple inputs, in which case, once a prover has evaluated the circuit, the proof generation cost becomes sublinear in the size of the circuit (i.e., the proof generation is much faster than even the circuit evaluation!), which is better than the previously best known results [16], [21].

**A. OVERVIEW OF OUR CIRCUIT CONSTRUCTION**

Our circuit construction is inspired by the Paterson-

Stockmeyer algorithm [18] evaluating a polynomial \(g(t)\) of
Here we note two properties of the above evaluation method. Specifically, for a given polynomial \( g(t) = \sum_{t=0}^{N} a_{t} t^{i} \), our circuit is constructed to first compute \( \sqrt{N} \) sub-polynomials \( g_{k} \)'s (for \( 1 \leq k \leq \sqrt{N} \)) where \( g_{k}(t) = \sum_{j=1}^{k} a_{j+\sqrt{N}(k-1)} t^{j} \), and then compute \( a_{0} + \sum_{k=1}^{\sqrt{N}} g_{k}(t) \cdot t^{\sqrt{N}(k-1)} \), which gives \( g(t) \). For example, for a polynomial \( g(t) = a_{0} + a_{1} t + \cdots + a_{16} t^{16} \) of degree 16, the constructed circuit (as shown in Figure 1) computes the polynomial as follows:

\[
\begin{align*}
& a_{0} + ((a_{4} t + \cdots + a_{4} t^{4}) + (a_{8} t + \cdots + a_{8} t^{4}) t^{4}) \\
& \quad + ((a_{9} t + \cdots + a_{12} t^{4}) + (a_{13} t + \cdots + a_{16} t^{4}) t^{4}) \cdot t^{8}
\end{align*}
\]

Here we note two properties of the above evaluation method that contributes to our optimal circuit construction. First, not all powers of \( t \) are needed, but only, for example, \( t, t^{2}, t^{3}, t^{4}, \) and \( t^{8} \) are. In general, only \( (\sqrt{N} + \log \sqrt{N}) \) powers of \( t \) are computed, that is, \( t, t^{2}, \cdots, t^{\sqrt{N}}, t^{2\sqrt{N}}, t^{4\sqrt{N}} \). For \( N = 2^{16} \), the constructed circuit (as shown in Figure 1) computes the powers of \( t \) with a regular wiring pattern. These properties contribute to reducing the circuit size, and increasing the circuit regularity.

Now we describe certain observations that led us to our circuit construction. The first observation is that the GKR protocol admits any efficiently computable gate with fan-in >2 without affecting the asymptotic complexity of the protocol, as long as the fan-in is constant. Also, the GKR protocol can admit a layer that solely consists of the linear-sum gates, computes the square-sum \( g_{k} \)’s, and the fused multiply-add gate, \((x, y, z) \mapsto xy + z\), for the summation of \( g_{k} \)’s. This yields a circuit of width \( 2 \sqrt{N} \) and depth \( (3 + \log N) \) with a regular wiring pattern.

Figure 1 shows our circuit construction of a single polynomial \( g(t) \). The circuit is composed of four parts. The first part referred to as \( \text{polygen} \), consisting of log \( \sqrt{N} \) layers with multiplication gates, takes as input \( t \) and computes its powers, \( t, t^{2}, \cdots, t^{\sqrt{N}} \). The second part referred to as \( \text{eval} \), consisting of a single layer over the linear-sum gates, computes the sub-polynomials \( g_{k}(t) \)’s. The third part referred to as \( \text{unify} \), consisting of log \( \sqrt{N} \) layers over the fused multiply-add gates, computes the summation of the sub-polynomials, \( g(t) - a_{0} \). Note that the \( \text{unify} \) part also computes the square-powers, \( t^{2\sqrt{N}}, t^{4\sqrt{N}}, t^{8\sqrt{N}}, \cdots, t^{N/2} \) by the side of the main computation, where the same multiply-add gate is used along with introducing dummy gates, to achieve a regular wiring pattern. The last part referred to as \( \text{extract} \), consisting of a single layer of a constant-addition gate, computes the final result \( g(t) \). More details and a precise definition of our circuit construction are provided in Appendix A.

\footnote{For the simplicity of the presentation, let \( N = 2^{2^{2a}} \) be the smallest power of four such that \( N \geq \deg(g) \).}

\[
\text{FIGURE 1. Our circuit construction of a polynomial of degree 16, } g(t) = \sum_{i=0}^{16} a_{i} t^{i}. \text{ The value of each gate denotes the output of the gate, where } g_{k} = \sum_{j=1}^{k} a_{j+4(k-1)} t^{j}. \text{ The green arrow denotes the linear-sum gate wiring. The gates computing zero are dummy gates that are added to achieve a regular wiring pattern and thus admit an optimal prover and an efficient verifier. The presence of the dummy gates does not affect the asymptotic cost.}
\]

In case that multiple inputs need to be evaluated on the same polynomial, our circuit construction simply puts multiple copies of the same circuit shown in Figure 1 side-by-side. This yields a circuit that has a larger width \( O(M \sqrt{N}) \) but the same depth \( O(\log N) \), where \( M \) is the number of inputs.

### B. COST ANALYSIS

Let us consider the case of multiple inputs being evaluated on the same polynomial. The following lemma shows the complexity of the GKR protocol (precisely, the variants [14] or [16, Section 3]) on our circuit construction for such a case. (The complexity for the single-input case is an instance of that of the single-polynomial-multiple-inputs case.)

**Lemma 5 (Complexity of Protocol on Our Circuit Construction):** Let \( C \) be a circuit generated by our construction for the case of \( M \) inputs being evaluated on the same polynomial of degree \( N \). Then, the complexity of the GKR protocol on \( C \) is as follows:

- Circuit evaluation: \( O(MN) \)
- Proof generation: \( O(M \sqrt{N} + N) \)
- Verification: \( O(M + \log N \log MN) \)
- Communication: \( O(\log N \log MN) \)
- Soundness: \( O((\log N \log MN)/A) \)

where \( A \) is the size of the sampling set, and the verification cost excludes the offline precomputation cost \( O(N) \). The complexity for the single input case is simply the one having \( M = 1 \) in the above.

**Proof:** Appendix B-E.

Here we note that our proof generation cost is better than the previously best known result. Specifically, let \( C \) be the circuit described in Lemma 5, and \( C' \) be a circuit that is equivalent to \( C \) with the same size \( O(MN) \) and the same depth \( O(\log N) \), but is constructed in a standard way.
VI. COST OPTIMIZATION FOR ROUNдинG OPERATION

In this section, we present an optimization technique that can significantly reduce the prover’s cost for the rounding layers described in Section IV.

A. GALOIS RING OVER \( \mathbb{Z}_p^e \) AND SAMPLING SET

A Galois ring \( \mathbb{Z}_p^e \) for a monic irreducible polynomial \( f(t) \in \mathbb{Z}_p[p] \) is a natural generalization of the Galois field \( \mathbb{GF}(p^d) \) over a finite field \( \mathbb{F}_p \). The representation of elements and operations in \( \mathbb{Z}_p^e[t]/(f(t)) \) is similar to that of \( \mathbb{GF}(p^d) \) modulo the difference between \( \mathbb{Z}_p^e \) and \( \mathbb{F}_p \). Let \( d \) be the degree of \( f(t) = t^d + f_{d-1}t^{d-1} + \ldots + f_0 \), where \( f_i \in \mathbb{Z}_p \). Then, the dimension of \( \mathbb{Z}_p^e[t]/(f(t)) \) is \( d \), and each element is represented as a \( d \)-dimensional tuple in \( \mathbb{Z}_p^d \) whose standard basis corresponds to \( 1, t, t^2, \ldots, t^{d-1} \). Thus, the addition corresponds to the component-wise addition in \( \mathbb{Z}_p^d \), and the multiplication by an element \( a = (a_0, a_1, \ldots, a_{d-1}) \) corresponds to the matrix multiplication by its corresponding matrix according to the multiplication rule \( t \cdot (a_0, a_1, \ldots, a_{d-1}) = (0, a_0, a_1, \ldots, a_{d-2}) + a_{d-1} \cdot (f_0, f_1, \ldots, f_{d-1}) \).

A nice property of the Galois ring \( \mathbb{Z}_p^e[t]/(f(t)) \) is that every nonzero element whose coefficients are in \( \{-(p-1), \ldots, -1, 0, 1, \ldots, p-1\} \) is invertible, which leads to the following theorem.

Theorem 5 [60]: Let \( p \) be an odd prime, \( e \) be a positive integer, and \( R \) be a Galois ring \( \mathbb{Z}_p^e[t]/(f(t)) \) of dimension \( d \). Then, all nonzero elements in \( \{(a_0 + a_1t + \ldots + a_{d-1}t^{d-1}) | a_i \in \{-(p-1), p-1\} \cap \mathbb{Z} \} \subseteq R \), are invertible (hence are not zero-divisors in \( R \). Therefore, the subset \( A = \{(a_0 + a_1t + \ldots + a_{d-1}t^{d-1}) | a_i \in \{0, p-1\} \cap \mathbb{Z} \} \subseteq R \) is a valid sampling set for the generalized Schwartz-Zippel lemma (Lemma 3) as well as the generalized GKR protocol (Theorem 4).

Note that the cardinality of the sampling set \( A \) in Theorem 5 is \( p^d \gg p \), which is maximal.\(^{\text{xvi}}\) Moreover, \( A \) satisfies the additional condition of Remark 1.

1) IRREDUCIBLE POLYNOMIALS IN \( \mathbb{Z}_p^e[t] \)

To construct a Galois ring \( \mathbb{Z}_p^e[t]/(f(t)) \), we need an irreducible polynomial in \( \mathbb{Z}_p^e[t] \). Indeed, there exist many irreducible polynomials \( f(t) \in \mathbb{Z}_p^e[t] \) for any degree \( d \), and a sparse polynomial (where most of its coefficients are zero) is desired for the efficiency of multiplication in \( \mathbb{Z}_p^e[t]/(f(t)) \).

\(^{\text{xvi}}\) A set containing more than \( p^d \) elements has distinct elements \( x \) and \( y \) such that \( x - y = (n_0 p, n_1 p, \ldots, n_{d-1} p) \in \mathbb{Z}_p^e[t]/(f(t)) \) by the Pigeonhole principle where \( n_i \)'s are integers, and \( (n_0 p, n_1 p, \ldots, n_{d-1} p) \) is a zero-divisor.

We provide examples of such sparse irreducible polynomials (Lemma 6) in Appendix B-G.

Lemma 6: Let \( p \) be a prime number. All of the following polynomials are irreducible in \( \mathbb{Z}_p \):\n
1) \( \Phi_4(x) = x^2 + 1 \) when \( p \equiv 3 \mod 4 \).
2) \( \Phi_5(x) = x^4 + x^3 + x^2 + x + 1 \) when \( p \equiv 2 \mod 5 \).
3) \( \Phi_9(x) = x^6 + x^3 + 1 \) when \( p \equiv 2 \) or \( 5 \mod 9 \).
4) \( x^3 - a \) for some \( a \) when \( p \equiv 1 \mod 3 \).
5) \( x^4 - 2 \) when \( p \equiv 5 \mod 8 \).
6) \( x^4 - 3 \) when \( p \equiv 5 \mod 12 \).

Proof: Appendix B-G \( \square \)

B. OPTIMIZATION OF PROVER’S COST FOR ROUNдинG LAYERS

Now we explain how to optimize the prover’s cost for the rounding layers. Let \( C_p \) be a given approximate arithmetic circuit over \( \mathbb{Z}_p^e \), and \( q \) be a prime such that \( p \simeq d^q \). First, we convert \( C_p \) to an approximately equivalent circuit \( C_q \) over \( \mathbb{Z}_q^e \), by the base-\( p \)-to-base-\( q \) conversion, where each base-\( p \) rounding gate (\( x \mapsto \lfloor x/p \rfloor \)) in \( C_p \) is replaced with \( d \)-consecutive base-\( q \) rounding gates (\( x \mapsto \lfloor x/q \rfloor \)) in \( C_q \). Then, we apply the generalized GKR protocol over a Galois ring \( \mathbb{Z}_q^e[t]/(f(t)) \) where \( f(t) \) is a monic irreducible polynomial of degree \( d \). Here, we employ the sampling set given in Theorem 5, whose cardinality is \( q^d \simeq p \), which affects the soundness. Moreover, in the process of the protocol, we have the circuit evaluation to be performed over \( \mathbb{Z}_q^e \), and the proof generation and the verification to be conducted over \( \mathbb{Z}_q^e[t]/(f(t)) \). This is valid, since \( \mathbb{Z}_q^e[t]/(f(t)) \) naturally embeds \( \mathbb{Z}_q^e \) as constant terms.

Now we analyze the complexity of the protocol for a rounding layer that consists of \( r \) rounding gates. First, note that the degree of the rounding polynomial (ldr) of \( C_p \) is \( ep \), while that of \( C_q \) is \( deq \simeq \lfloor \sqrt[3]{p} \rfloor \), which is much smaller than \( ep \) for some \( d \). On the other hand, the cost of the individual addition (and multiplication) operation in \( \mathbb{Z}_q^e[t]/(f(t)) \) is \( O(d) \) (and \( O(d^2) \), resp.) times larger than that of \( \mathbb{Z}_p^e \). Based on these facts and Lemma 5, the complexity of the optimized protocol on \( C_p \) and the optimized protocol on \( C_q \) can be summarized as follows (the two are equivalent when \( d = 1 \)):

| \( C_p \) | \( C_q \) |
|---|---|
| Circuit eval. | \( O(ep) \) | \( 2d^2 \lfloor \sqrt[3]{p} \rfloor \) |
| Proof gen. | \( O(ep) \) | \( 32d^4 \lfloor \sqrt[3]{p} \rfloor + 70d^8 \sqrt{d} \lfloor \sqrt[3]{p} \rfloor \) |
| Verification | \( O(log^2 ep) \) | \( d^3 \log \lfloor \sqrt[3]{p} \rfloor \log \lfloor \sqrt[3]{p} \rfloor \) |
| Soundness | \( O(log^2 ep) \) | \( d \log \lfloor \sqrt[3]{p} \rfloor \log \lfloor \sqrt[3]{p} \rfloor \) |

Here the optimization problem is to find \( d \) such that the costs for \( C_q \) are minimized. In particular, given \( p \), the term \( d^4 \lfloor \sqrt[3]{p} \rfloor \) is minimized to \( (e \log p)/4! \), which is much smaller than \( p \), when \( d = \lfloor \log p \rfloor / 4 \), where \( e \) is Euler’s number. In Section VII, we will present an experimental result where two orders of magnitude cost reduction was made by finding a proper \( d \).
VII. EXPERIMENTAL RESULTS

We present experimental results that quantify the efficiency of our scheme. Specifically, we conducted experiments that show how efficiently our scheme support rounding, and how effective the optimization technique is. Also, to show the importance of rounding, we compare our scheme (with rounding) to the original GKR protocol (without rounding) on deeply nested matrix multiplications. We consider matrix multiplication since it is a well-experimented subject considered by all of the existing GKR protocol variants, making it easier to compare with them. More importantly, matrix multiplication constitutes about 90% of DNN training workloads [61].

A. EXPERIMENTAL SETUP

We implemented our generalized GKR protocol\(^{xvii}\) over a ring \(R = \mathbb{Z}_{p^e}[t]/(f(t))\) where \(f(t)\) is a monic irreducible polynomial over \(\mathbb{Z}_{p^e}\). The modulo operations of \(\mathbb{Z}_{p^e}\) are implemented using the Montgomery modular multiplication [62]. The code is written in C++ using the GMP library, and compiled with the LLVM GCC compiler 9.1.0 (with -O3). The source code is available.\(^{xviii}\) All the experiments were performed on a laptop machine with Intel Core i5 CPU running MacOS (64-bit) at 2.9GHz processor and 8GB memory. Throughout this section, we report the verification cost excluding the cost of evaluating MLE of input/output layers, since they are not involved in verifying rounding layers placed in the middle of a circuit.

B. EFFECTIVENESS OF OPTIMIZATION VIA GALOIS RING

To show the effectiveness of the optimization technique described in Section VI, we instantiated our scheme with different Galois rings and compared their performance. Specifically, given an original ring, \(R_1 = \mathbb{Z}_{p^e}(65537)\), we took two Galois rings, \(R_2 = \mathbb{Z}_{(271)^{14}}[t]/(x^2 + 1)\) and \(R_3 = \mathbb{Z}_{(17)^{28}}[t]/(x^4 - 3)\), where \(|R_1| \sim |R_2| \sim |R_3| \sim 2^{112}\). Then, we instantiated our optimized protocol (Section VI) with the three different rings, and experimented with them for a rounding layer that consists of \(2^{14}\) rounding gates, where each rounding gate performs, roughly speaking, the 16-bit rounding, i.e., truncating the least-significant 16 bits.\(^{xix}\)

Figure 2 shows the performance of the protocol over the different rings. The circuit evaluation cost drastically decreases as the dimension of a Galois ring increases. This is because the size of the rounding circuit for \(R_3\) is much smaller than that of \(R_1\), since the size depends on \(ep\). However, the proof generation cost is not the case, since the cost of individual ring operations quadratically increases as the dimension of a Galois ring increases, thus it offsets the benefit of a smaller rounding circuit when the dimension is too high. In our experimental setup, the protocol over \(R_2\) of dimension two performed best in generating proofs. On the other hand, the verification cost increases as the dimension of a Galois ring increases, since the verification cost logarithmically depends on the rounding circuit size, thus the benefit of a smaller rounding circuit is insignificant, but the cost of individual ring operations dominates. In general, the optimal dimension varies depending on the set of parameters of the protocol and the characteristics of computation of interest. Also, we note that the circuit evaluation cost does not involve the cost overhead of individual operations of a Galois ring, since the circuit evaluation is performed over a base ring \(\mathbb{Z}_{p^e}\) instead of its Galois ring \(\mathbb{Z}_{p^e}[t]/(f(t))\), as mentioned in Section VI. This is why the proof generation cost is bigger than the circuit evaluation cost when the dimension is greater than one, although our optimal circuit construction offers the proof generation cost that is asymptotically smaller than the circuit evaluation cost, as described in Section V-B.

C. EFFICIENCY OF VERIFIABLE ROUNDING OPERATION

To quantify the efficiency of our scheme for rounding, we applied our scheme for a single rounding layer that consists of multiple rounding gates. Specifically, we consider our generalized GKR protocol over \(R_2 = \mathbb{Z}_{(271)^{14}}[t]/(x^2 + 1)\), and the rounding operation \(x \mapsto [x/(271^2)]\), roughly the

| \(p\) | \(\log p\) | \(e\) | \(f(t)\) | \(d\) | \(\lambda_s\) |
| --- | --- | --- | --- | --- | --- |
| \(R_1\) | 65537 | ~16 | 7 | N/A | 1 | 0.020 |
| \(R_2\) | 271 | ~8 | 14 | \(x^2 + 1\) | 2 | 0.011 |
| \(R_3\) | 17 | ~4 | 28 | \(x^4 - 3\) | 4 | 0.007 |

\(^{xvii}\) Specifically, the generalization was made on top of Thaler’s variant [14], since we considered Thaler’s variant to compare ours to the original GKR protocol as explained in Section VII-D.

\(^{xviii}\) https://anonymous.4open.science/r/GKR_approx_code-35AF

\(^{xix}\) More precisely, each rounding gate takes as input \(x\), and outputs \([x/65537]\), \([x/(271^2)]\), and \([x/(17^4)]\), respectively, for each \(R_1\), \(R_2\), and \(R_3\).
16-bit rounding. Figure 3 shows the performance of our protocol for a rounding layer of various sizes, from $2^8$ to $2^{19}$. As described in Section V-B, the cost of circuit evaluation and proof generation is linear in the number of rounding gates, while the cost of verification and communication is logarithmic in the number of rounding gates. We also note that the verification becomes even faster than the native evaluation (i.e., performing the rounding operation directly in the native processor, without going through the arithmetic circuit) when the number of rounding gates is more than $2^{18}$.

Moreover, we tested the performance of our scheme for rounding on various rounding bits. In Figure 4, the proof generation and circuit evaluation time for rounding, as well as verification time is described for varying rounding bits from 8bits to 32bits. We can see that the prover time depends linearly on the rounding bit-size while the verifier time remains almost constant, which coincide with theoretical estimation since the process of rounding $8n$ bits is composed of $n$ repeated evaluation of low digit removal polynomial each of which performs 8bit rounding (the verifier time remains constant since it depends logarithmically on the circuit size).

D. COMPARISON TO ORIGINAL GKR PROTOCOL ON Rounding Arithmetic

Now we compare our protocol (that supports rounding) to the original GKR protocol (that does not support rounding) on deeply nested matrix multiplications. The most important value of rounding is that it controls the number of digits within the limit of the underlying system, which is especially necessary for AI computations. This is the most fundamental advancement of our approach, compared to the original GKR.

Moreover, in order to understand the end-to-end performance of our approach, we conducted a performance comparison with the original GKR as follows.

We considered the Thaler [15]'s implementation for the original GKR protocol since it shows the best performance for matrix multiplication among other variants (e.g., [16, 21]). To be a fair comparison, we modified the Thaler’s implementation to employ the same GMP library we used in our protocol implementation.$^x$

1) DATASET AND COMPUTATION

For comparison, we consider a nested multiplication of depth $n, (\cdots ((M^2)^2)^2 \cdots)^2 = M^{2^n},$ where $M$ is a 128 $\times$ 128 matrix whose elements are fixed-point numbers with 16 fractional bits (i.e., 16 bits below the decimal point), and no overflow occurs during the computation.$^{xx}$

As an input, we randomly generated numbers in $[0, 1]$ with 16bit fractional bits.

In the original GKR protocol (over a finite field $\mathbb{Z}_q$) that does not support rounding, the above nested multiplication over the fixed-point numbers is represented as the integer-scaled nested multiplication, i.e., $(\cdots ((2^{16}M)^2)^2 \cdots)^2 = (2^{16})^{2^n}M^{2^n}$. This means that the prime $q$ must be taken to be larger than $(2^{16})^{2^n}$, that is, the bit-size of field elements (in $\mathbb{Z}_q$) exponentially grows in the multiplication depth $n$. In contrasts, in our protocol (over a ring $\mathbb{Z}_{p^e}$), the nested multiplication is represented as the integer-scaled nested multiplication with rounding, i.e., $[\cdots ((\lfloor (2^{16}M^2)^2 \rfloor)^2 \cdots)^2] \simeq 2^{16}M^{2^n}$, where $\lfloor \cdot \rfloor$ denotes $x \mapsto \lfloor x/(2^{16}) \rfloor$. Thus $p^e$ can be only larger than $2^{16}$.

$2^{16n}$ (the additional term $2^{16n}$ is due to the modulus change by rounding as described in Remark 2). That is, the bit-size of ring elements (in $\mathbb{Z}_{p^e}$) is linear in the multiplication depth.

In our experiment, we considered nested matrix multiplications of depth $n = 6, 8, 10,$ and $12$. Depending on the multiplication depth, we took different sized fields or rings. That is, for the original GKR protocol over $\mathbb{Z}_q$, we took the smallest Mersenne prime $q > (2^{16})^{2^n}$, i.e., $(2^{1279} - 1), (2^{14253} - 1), (2^{190993} - 1), \text{ and } (2^{86243} - 1)$, respectively, while for our protocol over $\mathbb{Z}_{p^e}[t]/(t^2 + 2)$, we took $p = 271 \simeq 2^{8}$ and the smallest $e$ such that $p^{e} > (2^{16})^{n+1}$, i.e., $p^e = 271^{14}, 271^{18}, 271^{22}, \text{ and } 271^{26}$, respectively, for each multiplication depth $n = 6, 8, 10,$ and $12$.

$^x$While we experimented with matrix multiplication, we considered Thaler’s general-purpose machinery instead of the special-purpose scheme for matrix multiplication, for the generality of experimental results.

$^{xx}$For simplicity, we consider $M$ such that the elements of $M$ and $M^{2^n}$ are positive fixed-point numbers less than 1, i.e., being represented in 16 bits.
In Figure 5, we compare the performance of our protocol to that of Thaler's on nested matrix multiplication of different depths. To highlight the net effect of rounding, we report the cost for a single matrix multiplication in the context of different multiplication depths. The domain of each protocol is chosen to be large enough to admit a given multiplication depth. That is, our protocol is over \( \mathbb{Z}_{2^{271}}[t]/(t^3 + 2) \), where \( e = 14, 18, 22, \) and \( 26 \), respectively. Thaler's is over \( \mathbb{Z}_q \), where \( q = (2^{1279} - 1), (2^{1425} - 1), (2^{19957} - 1), \) and \((2^{6243} - 1)\), respectively. The performance of Thaler's on the multiplicative depth 12 is extrapolated. The soundness probability of our protocol in Figure 5 is set to \( 2^{-20} \) which is much worse than that of Thaler's compared one (\( \approx 2^{-1200} \) or less). However, optimizing ours to have similar soundness condition could be an overkill. In fact, our soundness can be quickly improved by running iterations, e.g., four prover-verifier pairs in parallel will achieve \( (2^{-14})^4 \approx 2^{-56} \times 10^{-16} \) soundness \(^{xxiii}\) which is similar to the soundness probability \( (2^{-45} \text{ to } 2^{-20}) \) of existing verifiable computing systems.

We can compare our result with GKR protocol on boolean representation, i.e., representing all operations in bitwise so that rounding is also representable efficiently. In fact, our method incurs blow up of cost quadratic in multiplicative depth and rounding bits for each rounding gate, while boolean representation incurs that quadratic in input bits for each multiplication gate. Therefore, our method is efficient when number of rounding gates is smaller than that of the multiplication gates, i.e., when lazy rounding strategy is applicable. Also, our method can be applied with (asymptotic) cost reduction technique derived from high regular wiring pattern, e.g. Thaler [14]'s protocol for matrix multiplication or ours for polynomial evaluation. In contrasts, it seems quite hard to apply them for Boolean representation of such circuit.

We can also compare our result with recent work [12], [26] which combine commitment scheme with interactive proof. In these cases, prover’s cost for rounding gate is dominated by that of the commitment with as many messages as bitsize of the input. Though it seems (asymptotically) efficient than ours in the bitsize, the cost of commitment shows that ours can be better or comparable in some cases. The experimental result of [12] on commitment implies that proving rounding gate with their commitment would at least \( 0.11 \times \) rounding bits (ms) per gate, \(^{xxiv}\) resulting in \( 2^{14} \times 16 \times 0.11 \text{ (ms)} \approx 29 \text{ (sec)} \) for \( 2^{14} \) gates, which is about 7 times costly than our prover (4 sec) in Figure 3. On the other hand, the commitment in [26] \(^{xxv}\) is asymptotically costly in verifier cost than that of ours.

We finally note that there is still room for improvement of ours. In particular, the individual operations of a Galois ring can be further improved, e.g., they can be broken down into multiple independent subroutines, being suitable for parallelization [64], [65]. It will drastically reduce the overhead of increasing the dimension of a Galois ring, which in turn will allow us to employ a much smaller prime \( p \), further improving the overall performance of the protocol.

**VIII. CONCLUSION**

We presented a verifiable computing scheme that supports rounding which is essential for approximate computations. Based on the (latest variant of) GKR protocol that is most efficient in generating proofs among existing verifiable computing protocols, our scheme consists of the following elements: generalization of the GKR protocol over a ring, reduction of the rounding operation to a low-degree polynomial in a ring, optimal circuit construction of arbitrary polynomials, and optimization of proof generation for rounding via a Galois ring. We implemented our scheme, and presented

---

\(^{xxiii}\)For comparison, \( 10^{-16} \) is the uncorrectable bit error rate of a typical hard disk [63].

\(^{xxiv}\)Some argument systems [10], [39], [40] offers roughly \( 2^{-128} \) soundness, but it relies on non-falsifiable cryptographic assumptions.

\(^{xxv}\)On Amazon EC2 e4.8Xlarge running Linux Ubuntu 14.04, with 60GB of RAM, Intel Xeon E5-2666v3 CPUs with 36 virtual cores at 2.9 GHz.

---
experimental results that show the efficiency of our scheme for approximate computations. For example, ours performed two orders of magnitude better than the existing GKR protocol for a nested matrix multiplication of depth 12 on the 16-bit fixed-point arithmetic.

A. LIMITATIONS AND FUTURE WORKS

Though we have reduced the proving complexity of rounding operation significantly, it is not sufficient for being practical. For example, while it requires roughly 1 sec for proving $2^{12}$ rounding gates (with 16-bit fractional bits), it is roughly $10^5 \times$ slower than evaluation of rounding gates, whose overhead can be problematic when proving lots of rounding gates. Still, we remark that this overhead may be unavoidable for interactive proof systems where representation of rounding operations with arithmetic operations is necessary (and our representation is the simplest one possible).

On the other hand, if we allow argument systems utilizing cryptographic commitment schemes, rounding operations can be represented with bit representations, which results in (significant) overhead proportional to the bit length due to the cryptographic schemes. We remark that, such overhead is inevitable in a finite field for which most commitment schemes are supporting.

One interesting future work will be combining commitment schemes over rings (e.g., $\mathbb{Z}_2$ or $\mathbb{Z}_{2^m}$) with our solution. Very recently, some works [66], [67] have come up with such commitment schemes or $\mathbb{Z}K$ proof systems, and combining our result with one of them may result in more efficient and practical solution for verifiable computation over approximate numbers (with rounding operations).

We finally remark that our solution — verifiable computation over rings — has been found out to be applicable in encrypted computation. More precisely, computations over the ciphertexts encrypted by (fully) homomorphic encryption [68], [69] can be represented concisely with ring operations, and one can prove and verify the computation over ciphertexts with better flexibility (hence better efficiency by choosing more efficient parameters) utilizing our solution. We refer to [70] for the detail.

APPENDIX A

CIRCUIT CONSTRUCTION FOR VERIFIABLE POLYNOMIAL EVALUATION

A. NOTATION

Assume we are given a polynomial $g$ over a finite ring $\mathbb{Z}_{2^m}$. (Our representation is also valid with a polynomial over a finite field $\mathbb{F}_p$.) Let us fix $N = 2^m$ to denote the smallest power of four such that $N \geq \text{deg}(g)$. Let us index each layer where the input layer is indexed by 0.\textsuperscript{33} Let us also index each gate in a layer where the left-most gate is indexed by 0, and the index value is represented in the binary form. We write $\tilde{V}_i$ to denote the MLE of the output values of the $i^{th}$ layer as usual. For the simplicity of the presentation, we assume that the number of inputs denoted by $M = 2^m$ is a power of two, in multi-input case. We write $\beta_i(x, y) : \mathbb{Z}_{2^m} \times \mathbb{Z}_{2^m} \rightarrow \mathbb{Z}_{2^m}$ to denote the MLE of $B_i(x, y) : \{0, 1\}^x \times \{0, 1\}^y \rightarrow \{0, 1\}$ where $B_i(x, y)$ is the comparison function that returns 1 if $x = y$, and 0 otherwise. We write $\tilde{I}_s = (1, 1, \ldots, 1) \in \{0, 1\}^t$, and $\chi_k(x) := B_k(x, \tilde{I}_s) : \{0, 1\}^t \rightarrow \{0, 1\}$. We omit $s$ when it is obvious.

B. DESCRIPTION

Now we present the circuit representation for the polynomial $g(t) = \sum_{i=0}^{N-1} a_i t^i$. The circuit is composed of four parts, each of which is called polygen, eval, unify, and extract, respectively, as illustrated in Figure 1. We note that, as we will explain below, the eval and unify layers consist of two sub-circuits placed in parallel, where the left-hand side sub-circuit computes the sub-polynomials $g_i$ and $g_i \tilde{t}$, while the right-hand side one computes the power terms $t^i$. Although the two sub-circuits compute different types of values, we design them to have the identical wiring pattern by introducing the dummy gates (i.e., the gates computing zero), so that the overall circuit becomes regular, allowing the verifier to be efficient. Here, the dummy gates affect only the width of the circuit, not the depth, and thus their effect on the verifier’s cost is negligible, i.e., asymptotically zero, as the verifier’s cost is logarithmically proportional to the circuit width. We first describe the single-input case (Figure 1).

The polygen part corresponds to the sub-circuit between the layers 1 and $n$, where for each $i^{th}$ layer, the input values are $\{t^i\}_{j=1}^{t-1}$, and the output values are $\{t^j\}_{j=1}^{t-1}$. Now we have the following relation between $\tilde{V}_{i+1}$ and $\tilde{V}_i$ (for $0 \leq i < n$) as follows.

$$\tilde{V}_{i+1}(z) = \tilde{V}_i(z_0)(1-z_0) + z_0 \tilde{V}_i(\bar{1})$$

where $z = (z_0, z_1, \ldots, z_l)$, $z_0 = (z_1, z_2, \ldots, z_l)$, $\bar{1} = (1, 1, \ldots, 1)$, and $V_0 = t$. The validity of this equation is derived from the fact that both sides of the equation are MLEs in $z$ agreeing on $[0, 1]^{t+1}$, and the uniqueness of MLE (Lemma 2) that holds for an arbitrary ring (Section III-B). Recall that the gate index value is represented in a bit vector, e.g., $\tilde{V}_2(0, 0) = t$, $\tilde{V}_2(0, 1) = t^2$, $\tilde{V}_2(1, 0) = t^3$, and $\tilde{V}_2(1, 1) = t^4$ denote the output value of the first, the second, the third, and the fourth gate of the second layer, respectively, as shown in Figure 1.

The eval layer, i.e., the $(n+1)^{th}$ layer, produces $2\sqrt{n}$ output values which consists of $g_1(t), \ldots, g_{\sqrt{n}}(t)$, 0, 0, 0, 0, 0, $\sqrt{n}$, from the input values $\{t^j\}_{j=1}^{\sqrt{n}}$. Each $g_k$ (for $1 \leq k \leq n$) is a polynomial of degree at most $\sqrt{n}$, defined by $g_k(t) = \sum_{j=1}^{\sqrt{n}} a_{j+k} \sqrt{(k-1)^t}$. The zeros are the outputs of dummy gates as explained earlier. Now we have the following
• For each polygon layer, $\tilde{V}_i(w, z) : \mathbb{Z}_p^{m_i} \times \mathbb{Z}_{p^r} \rightarrow \mathbb{Z}_{p^r} \ (0 \leq i < n - 1)$,
  \[
  \tilde{V}_{i+1}(w, z) = \sum_{q \in (0, 1)^n} \beta(w, q) \tilde{V}_i(q, z_\mathcal{i})[(1 - z_\mathcal{i}) + z_\mathcal{i} \tilde{V}_i(q, \hat{I}_\mathcal{i})]
  \]
  where $z = (z_0, z_1, \ldots, z_i) \in \mathbb{Z}_{p^r}^{i+1}$, and $z_\mathcal{i} = (z_1, z_2, \ldots, z_i) \in \mathbb{Z}_{p^r}^i$.

• For the eval layer, $\tilde{V}_{n+1}(w, z) : \mathbb{Z}_p^{m_n} \times \mathbb{Z}_p^{n+1} \rightarrow \mathbb{Z}_p^n$,
  \[
  \tilde{V}_{n+1}(w, z) = \sum_{q \in (0, 1)^n} \alpha(z, q) \cdot \tilde{V}_n(w, q).
  \]

• For each unify layer, $\tilde{V}_{j+1}(w, z) : \mathbb{Z}_p^{m_j} \times \mathbb{Z}_p^{n+1-j} \rightarrow \mathbb{Z}_p^n \ (n + 1 \leq j \leq 2n)$,
  \[
  \tilde{V}_{j+1}(w, z) = \sum_{q \in (0, 1)^n} \beta(w, q) [\tilde{V}_j(q, z_0) + \tilde{V}_j(q, z_1) \cdot \tilde{V}_j(q, \hat{1}_{2n-j})].
  \]

• For the extract layer, $\tilde{V}_{2n+2}(w) : \mathbb{Z}_p^{m_2} \rightarrow \mathbb{Z}_p^2$,
  \[
  \tilde{V}_{2n+2}(w) = \tilde{V}_{2n+1}(w, 0) + a_0.
  \]

![FIGURE 6. Construction of (sub-)circuit representation of a polynomial evaluation that consists of $M = 2^m$ inputs. Here we consider operations over $\mathbb{Z}_{p^e}$, and the polynomial in the form of $g(t) = \sum_{i=0}^N a_i t^i$ for the smallest $N = 2^m \geq \deg(g)$. We write $1_k = (1, \ldots, 1) \in \mathbb{Z}_{p^e}^k$, and $a(x, q) : \mathbb{Z}_{p^e}^{N+1} \times \mathbb{Z}_{p^e}^k \rightarrow \mathbb{Z}_{p^e}$ to denote the MLE of a boolean hypercube function $A(x) : \{0, 1\}^{2n+1} \rightarrow \mathbb{Z}_{p^e}$ that represents $(a_0, a_1, \ldots, a_{N}, 0, \ldots, 0, 1) \in \mathbb{Z}_{p^e}^{2n+1}$. For example, $A(0, 0, 0, \ldots, 0) = a_1$, $A(0, 1, \ldots, 1) = a_{2n}$, and $A(1, 1, \ldots, 1) = 1$. We also write $\hat{a}(x, q) : \mathbb{Z}_{p^e}^{N} \times \mathbb{Z}_{p^e}^k \rightarrow \mathbb{Z}_{p^e}$ to denote the MLE of a comparison function $B(x, y) : \{0, 1\}^m \times \{0, 1\}^m \rightarrow \{0, 1\}$ where $B(x, y)$ returns 1 if $x = y$, and 0 otherwise.](https://example.com/figure6.png)

The multi-input case with $M = 2^m$ number of inputs follows naturally from single-input case described so far (see Figure 6).

**APPENDIX B
PROOFS OF THEOREMS AND LEMMAS**

**A. PROOF OF THEOREM 2**

Proof: [Sketch] Completeness directly follows from the protocol description. The main idea for showing soundness can be summarized as follows (See [4] or [58] for the full proof). Assume that a (dishonest) $P'$ sends an incorrect result $S(t') \neq S(t')$ to $V$. Let us distinguish the values claimed by $P'$ from the values which would be claimed by an honest $P$ by adding the prime (′) symbol. Then $f_1(t') \neq f_1(t)$. Otherwise, $V$ will reject immediately by checking if $S(f_1(t')) = f_1(t')$. When $V$ chooses a random $r_1$ from $\mathbb{F}$, by the Schwartz-Zippel lemma (Lemma 1), $f_1(r_1) \neq f_1(r_1)$ with the high probability $1 - \frac{d}{|\mathbb{F}|}$ since $f_1(t)$ is a polynomial of degree at most $d$. If $f_1(r_1) \neq f_1(r_1)$, $P'$ must send $f_2(t') \neq f_2(t)$ because of the same reasoning as before. Continuing this, $P'$ must send $f_s(t') \neq f_s(t') = f(r_1, \ldots, r_s, t)$, and will be rejected with the high probability by $V$ who finally checks if $f_s(t') = f(r_1, \ldots, r_s, t)$ for a randomly chosen $r_s$ in $\mathbb{F}$. The soundness probability bound is derived from the probability $1 - \left(1 - \frac{d}{|\mathbb{F}|}\right)^m$ that at least one of the above high probability events does not occur during the protocol.

**B. PROOF OF LEMMA 2**

Proof: Uniqueness follows from the observation that any multilinear polynomial $\tilde{V}(x_1, x_2, \ldots, x_\mu)$ can be
represented by $\sum_{b \in \{0, 1\}^n} C(b) x_b$, where $x_b := \prod_i x_{i,b_i}$ with $I := \{ i \mid b_i = 1 \}$, and $C(b) \in \mathbb{F}$ is a coefficient corresponding to each monomial $x_b$. Then, $C(b)$ is uniquely determined by $\bar{V}(b)$ for $b \in \{0, 1\}^n$. Specifically, for a zero vector $0$, $\bar{V}(0) = \bar{V}(\bar{0})$. For an elementary vector $e_i$ whose $i$-th component is 1 and all others are 0, $C(e_i) = \bar{V}(e_i) - \bar{V}(\bar{0})$. For a vector $e_{i,j} \in \{0, 1\}^n$ ($i \neq j$) whose $i$-th and $j$-th components are 1 and all others are 0, $C(e_{i,j}) = \bar{V}(e_{i,j}) - C(e_i) - C(e_j) - \bar{V}(\bar{0})$. Continuing this process with increasing the weight of each vector $b \in \{0, 1\}^n$, we can see that every $C(b)$ is uniquely determined by $\bar{V}(b)$ for $b \in \{0, 1\}^n$.

\section*{C. Proof of Lemma 3}

\textbf{Proof:} It follows from the induction on the number of variables $n$ as the original Schwartz-Zippel lemma (Lemma 1), provided that it holds in the single variable case. Let $a_1 \in A$ be a root of $f(t)$. By the division algorithm with a monic polynomial $(t - a_1)$, $f(t) = (t - a_1)f_1(t)$ and the degree of $f_1(t)$ is less than that of $f(t)$. Note that another root, if exists, $a_2 \in A$ ($a_2 \neq a_1$) must be a root of $f_1(t)$ since $(a_2 - a_1)$ is not a zero divisor and $f(a_2) = 0$. Then, the division algorithm with a monic polynomial $(t - a_2)$ on $f_1(t)$ gives $f(t) = (t - a_1)(t - a_2)f_2(t)$ and the degree of $f_2(t)$ is less than that of $f_1(t)$. Continuing this process, we conclude that $f(t)$ cannot have more roots in $A$ than the degree of $f(t)$.

\section*{D. Proof of Theorem 3}

\textbf{Proof:} The proof is almost the same as that of the original Sum-Check protocol. The generalized Schwartz-Zippel lemma (Lemma 3) implies that any two distinct univariate polynomials of degree $\leq d$ over $A$ agree on at least $d$ points among $A$. Following the proof of the original Sum-Check protocol (Theorem 2), the soundness probability of the generalized Sum-Check protocol is bounded by $\frac{n^d}{|A|^d}$.

\section*{E. Proof of Lemma 5 (Prover Cost)}

\textbf{Proof:} We follow the notation of Appendix A. The circuit representation is composed of four parts: \textit{polygen}, \textit{eval}, \textit{unify}, \textit{extract}, and \textit{division} as described before, and the depth is $2n + 3 = O(\log N)$. We first estimate the cost of $\mathcal{P}$ for evaluating the circuit. It is simply $M$ times of the cost for evaluating the circuit of a single polynomial evaluation, and we only estimate the single case (Fig. 1). The $i$-th layer in \textit{polygen} requires $2^{i-1}$ multiplications resulting in $O(2^{2n})$ total for \textit{polygen} part. The \textit{eval} layer requires $O(2^n \cdot 2^{n+1}) = O(2^{2n})$ operations, since evaluating each $f_i(t)$ given $\{t\}^{2n}$ requires $O(2^n)$ operations. The $j$-th layer in \textit{unify} requires $2^{2n+3-j}$ operations resulting in $O(2^{2n})$ total for \textit{unify} part. Since \textit{extract} and \textit{division} part is of negligible cost, the total cost for evaluation is $O(2^n + 2^{2n} + 2^{2n}) = O(N)$, resulting in $O(NM)$ for $M$ rounding gates.

Now we estimate the cost of $\mathcal{P}$ for proving the evaluation given all output of gates in the circuit. We assume Thaler [15]'s Reusing Work reducing $\mathcal{P}$'s cost for evaluating all $\beta_m(w, p)$, $\bar{V}(q)$, and $\alpha(r)$ values required for sum-check to be only $O(2^m)$, $O(2^n)$, and $O(2^t)$ respectively, where $m$, $n$, and $t$ are the number of variables constituting $p$, $q$, and $r$, respectively. Thus, for estimation of the cost, it suffices to count the number of variables appear in summands of the relation of MLEs in multi rounding case (Figure 6).

In \textit{polygen} part, reducing from $\bar{V}_{i+1}$ to $\bar{V}_i$ requires $O(2^n + 2^{m+1})$ cost for sum-check, and additional $O(i \cdot 2^j)$ cost for reducing to single point, resulting in total $O(2^{m+n} + n \cdot 2^n)$ cost. In \textit{eval} layer, sum-check requires $O(2^{2n+1} + 2^{m+n})$ cost. In \textit{unify} part, reducing from $\bar{V}_{i+1}$ to $\bar{V}_i$ requires $O(2^n + 2^{m+2n+2-i})$ cost for sum-check, and additional $O((2n - 1) \cdot 2^{n+2-j})$ cost for reducing to single point, resulting in total $O(2^{m+n} + n \cdot 2^n)$ cost. The \textit{extract} and \textit{division} layer doesn't affect $\mathcal{P}$'s cost since it does not require sum-check. Overall, the cost of proving is $O(2^{m+n} + 2^{2n} + 2^{m+n} + n \cdot 2^n)$ which is $O(\sqrt{NM} + N)$.

\section*{F. Proof of Lemma 5 (Verifier Cost)}

\textbf{Proof:} We follow the notation of Appendix A. Note that $\alpha(z, q)$ can be precomputed in cost $O(N)$, using memoization [6], and will not be considered in the following estimation. Each $\beta_m$ can be evaluated in cost $O(m)$ due to its simple form [15, Section 4.3.1], without affecting the asymptotic cost of $\mathcal{V}$. Also, as the original GKPR protocol, $\mathcal{V}$'s cost for the initial an final step is $O(M \log M)$, since there are $O(M)$ input and output.

Now, we can estimate the cost of $\mathcal{V}$ based on that in the sum-check (Theorem 2). Recall that in sum-check, the cost of $\mathcal{V}$ depends on the number of variables managed by summation. In \textit{polygen} layers, reducing from $\bar{V}_{i+1}$ to $\bar{V}_i$ requires $\mathcal{V}$ to perform $O(m)$ operations for sum-check, and $O(i)$ for reducing to single point. Therefore, the cost for \textit{polygen} layers is $O(mn + n^2)$. In \textit{eval} layer, $O(n)$ cost is required. In \textit{unify} layers, reducing from $\bar{V}_{i+1}$ to $\bar{V}_i$ requires $\mathcal{V}$ to perform $O(n)$ operations for sum-check, $O(2n + 2 - j)$ for reducing to single point, resulting in $O(mn + n^2)$ cost total. Since the cost for \textit{extract} and \textit{division} layers are negligible, the total cost of $\mathcal{V}$ without initial and final step is $O(mn + n^2) = O(\log N \log MN)$. The bound of soundness probability and communication cost can be estimated similarly.

\section*{G. Proof of Lemma 6}

We use following Lemma whose proof can be found in [71].

Lemma 7 [71, Lemma 5.9]: An $n$-th cyclotomic polynomial $\Phi_n$ of degree $\phi(n)$ is irreducible if and only if $n$ is a primitive root modulo $n$ (i.e., $n$ does not divide $n$), and its multiplicative order modulo $n$ is $\phi(n)$, where $\phi$ is the Euler’s totient function.

\footnote{There is a procedure computing the inverse of each component $z_i$ of $z$ for efficient computation of $\beta(z, p)$, but we can deviate from it without asymptotic increase of the cost. More precisely, \textit{C}i[[\gamma_i+1, \ldots, p_n]] in \textit{C}i, as equation (7)) (Full ver. of [14]) can be calculated by \textit{C}jCj[1, \{p_{j+1}, \ldots, p_n\}] + (1 - \gamma_j)Cj[0, \{p_{j+1}, \ldots, p_n\}] without $z_j^{-1}$.}

\footnote{In fact, we perform two consecutive processes of reducing to single point, $\bar{V}_{i}(p, z, 0)$ to $\bar{V}_{i}(p, 0, 0)$, then $\bar{V}_{i}(p, 0, 0)$ & $\bar{V}_{i}(p, 1, \ldots, 1)$ to $\bar{V}(p, 0, 0)$.}
Proof of Lemma 6: (i), (ii), (iii) directly follows from the above lemma and the fact that each prime $p$ is a primitive root modulo 4, 5, or 9, respectively.

More algebraic proof can be found in [72].

For (iv), note that if $x^3 - a$ is reducible, it has monic factor and $x^3 - a$ has a solution in $\mathbb{Z}_p$. We show that there exists an $a$ such that $x^3 - a$ has no solution in $\mathbb{Z}_p$. This is equivalent to the claim that the function $t \mapsto x^3 : \mathbb{Z}_p \to \mathbb{Z}_p$ is not injective. Note that the multiplicative group $\mathbb{Z}_p^\times$ of $\mathbb{Z}_p$ has order $p - 1$, and the order is multiple of 3 when $p \equiv 1 \mod 3$. Now, by Sylow theorem, there exists a group of order 3 in $\mathbb{Z}_p^\times$, and there exists at least 3 elements in $\mathbb{Z}_p$ whose cube is 1. Therefore, the claim follows.

(v), (vi) follows from general irreducibility results on quartic polynomials [73, Theorem 3.(iv)].

Note that above Lemma 7 implies that we can find many irreducible cyclotomic polynomials (with few non-zero coefficients) of higher degree, if needed.

\[
\text{ACKNOWLEDGMENT}
\]

This work is a revised version of the thesis “Verifiable Computing for Approximate Arithmetic” [1] with additional experiments, source codes distribution, and further discussions on limitations and future works. This work was done when Dongwoo Kim was with the Department of Mathematical Sciences, Seoul National University.

\section*{REFERENCES}

[1] D. Kim, “Verifiable computing for approximate arithmetic,” Ph.D. dissertation, Dept. Math. Sci., Seoul Nat. Univ., Seoul, South Korea, 2020.

[2] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy, “Checking computations in polylogarithmic time,” in Proc. 23rd Ann. ACM Symp. Theory Comput. (STOC), 1991, pp. 21–32.

[3] S. Micali, “CS proofs,” in Proc. 35th Ann. Symp. Found. Comput. Sci., 1994, pp. 436–453.

[4] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, “Delegating computation: Interactive proofs for muggles,” in Proc. 14th Ann. ACM Symp. Theory Comput., May 2008, pp. 113–122.

[5] R. Gennaro, C. Gentry, and B. Parno, “Non-interactive verifiable computation: Outsourcing computation to untrusted workers,” in Proc. Ann. Cryptol. Conf. Berlin, Germany: Springer, 2010, pp. 465–482.

[6] V. Vu, S. Setty, A. J. Blumberg, and M. Walfish, “A hybrid architecture for interactive verifiable computation,” in Proc. IEEE Symp. Secur. Privacy, May 2013, pp. 223–237.

[7] A. E. Kosba, D. Papadopoulos, C. Papamanthou, M. F. Sayed, E. Shi, and N. Triandopoulos, “TRUEFOOT: Faster verifiable set computations,” in Proc. 23rd USENIX Conf. Secur. Symp. Berkeley, CA, USA: USENIX Association, 2014, pp. 765–780.

[8] S. Setty, S. Angel, T. Gupta, and J. Lee, “Proving the correct execution of concurrent services in zero knowledge,” in Proc. 12th USENIX Conf. Syst. Princ., May 2013, pp. 238–252.

[9] J. Kilian, “A note on efficient zero-knowledge proofs and arguments (extended abstract),” in Proc. 24th Ann. ACM Symp. Theory Comput. (STOC), 1992, pp. 723–732.

[10] J. Ben-Sasson and M. Sudan, “Short PCPs with polylog query complexity,” SIAM J. Comput., vol. 38, no. 2, pp. 551–607, Jan. 2008.

[11] J. Ben-Sasson, A. Chiesa, D. Genkin, and E. Tromer, “On the concrete efficiency of probabilistically-checkable proofs,” in Proc. 45th Ann. ACM Symp. Theory Comput. (STOC), 2013, pp. 585–594.

[12] S. Arora and S. Safra, “Probabilistic checking of proofs: A new characterization of NP,” J. ACM, vol. 45, no. 1, pp. 70–122, Jan. 1998.

[13] S. Arora and R. Motwani, M. Sudan, and M. Szegedy, “Proof verification and the hardness of approximation problems,” J. ACM, vol. 45, no. 3, pp. 501–555, May 1998.

[14] S. T. Setty, V. Vu, N. Pampalia, B. Braun, A. J. Blumberg, and M. Walfish, “Taking proof-based verified computation a few steps closer to practicality,” in Proc. USENIX Secur. Symp., 2012, pp. 253–268.

[15] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno, and M. Walfish, “Resolving the conflict between generality and practicality in verified computation,” in Proc. 8th ACM Eur. Conf. Conf. Comput. Syst. (EuroSys), 2013, pp. 71–84.

[16] Y. Ishai, E. Kushilevitz, and R. Ostrovsky, “Efficient arguments without short PCPs,” in Proc. 22nd Ann. IEEE Conf. Comput. Complex. (CCC), Jun. 2007, pp. 278–291.
R. Gennaro, C. Gentry, B. Parino, and M. Raykova, “Quadratic span programs and succinct NIZKs without PCPs,” in Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 2013, pp. 626–645.

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. Adv. Cryptol. (CRYPTO). Berlin, Germany: Springer, 2013, pp. 90–108.

E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Succinct non-interactive zero knowledge for a von Neumann architecture,” in Proc. USENIX Secur. Symp., 2014, pp. 781–796.

J. Groth, “On the size of pairing-based non-interactive arguments,” in Proc. 35th Annu. Int. Conf. Adv. Cryptol., vol. 9666, May 2016, pp. 326–346.

I. Giacomelli, J. Madsen, and C. Orlandi, “ZKBoo: Faster zero-knowledge for Boolean circuits,” in Proc. 25th USENIX Secur. Symp. (USENIX Security), 2016, pp. 1069–1083.

E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, “Scalable, transparent, and post-quantum secure computational integrity,” Cryptol. ePrint Arch., Int. Assoc. Cryptograph. Res., Tech. Rep. 2018/046, 2018.

E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, “Aurora: Transparent succinct arguments for RICS,” Cryptol. ePrint Arch., Int. Assoc. Cryptograph. Res., Tech. Rep. 2018/282, 2018.

B. Bunz, 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), San Francisco, California, USA, May 2018, pp. 315–334.

S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, “Liger: Lightweight sublinear arguments without a trusted setup,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secure, Oct. 2017, pp. 2087–2104.

S. Setty, “Spartan: Efficient and general-purpose zkSNARKs without trusted setup,” Cryptol. ePrint Arch., Int. Assoc. Cryptograph. Res., Tech. Rep. 2019/550, 2019.

A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, “Marlin: Preprocessing zkSNARKs with universal and updatable SRS,” in Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 2020, pp. 738–768.

R. Bhaduria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang, “Liger++: A new optimized sublinear IOP” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2020, pp. 2025–2038.

A. Kosba, D. Papadopoulos, C. Papamanthou, and D. Song, “MIRAGE: Succinct arguments for randomized algorithms with applications to universal zk-SNARKs,” in Proc. 29th USENIX Secur. Symp. (USENIX Security), 2020, pp. 2129–2146.

A. Chiesa, D. Ojha, and N. Spooner, “Fractal: Post-quantum and transparent recursive proofs from holography,” in Fractal: Post-quantum and transparent recursive proofs from holography, in Fractal: Post-quantum and transparent recursive proofs from holography, Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 2020, pp. 769–793.

E. Ben-Sasson, A. Chiesa, D. Genkin, and E. Tromer, “Fast reductions from RAMs to delegatable succinct constraint satisfaction problems,” in Proc. 4th Conf. Innov. Theor. Comput. Sci. (ITCS), 2013, pp. 401–414.

R. S. Wahby, S. Setty, Z. Ren, A. J. Blumberg, and M. Walfish, “Efficient RAM and control flow in verifiable outsourced computation,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2015, pp. 1–25.

X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” 2017, arXiv:1712.05526.

T. Vieira. (Feb. 11, 2014). Exp-Normalize Trick. [Online]. Available: https://timvieira.github.io/blog/post/2014/02/11/exp-normalize-trick/

E. Hesamifard, H. Takabi, and M. Ghasemi, “CryptoDL: Deep neural networks over encrypted data,” 2017, arXiv:1711.05189.

J. S. Milne, “Fields and Galois theory (V4. 60),” in Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 2021, pp. 528–558.

J. T. Schwartz, “Fast probabilistic algorithms for verification of polynomial identities,” J. ACM, vol. 27, no. 4, pp. 701–717, Oct. 1980.

C. Lund, L. Fortnow, H. Karloff, and N. Nisan, “Algebraic methods for interactive proofs systems,” J. ACM, vol. 39, no. 4, pp. 859–868, Oct. 1992.

J. Thaler, “A note on the GKR protocol,” Tech. Rep., 2015.

B. R. McDonald, Finite Rings With Identity, vol. 28. New York, NY, USA: Marcel Dekker, 1974.

P. Warden. (Apr. 20, 2015). Why GEMM is at the Heart of Deep Learning. [Online]. Available: https://petewarden.com/2015/04/20/
JUNG HEE CHEON (Member, IEEE) received the B.S. and Ph.D. degrees in mathematics from KAIST, in 1991 and 1997, respectively. He was with ETRI, Brown University, and ICU. Since 2003, he has been a Professor with the Department of Mathematical Sciences, Seoul National University, Seoul, South Korea, where he has been working concurrently as the Director of the Industrial and Mathematical Data Analytics Research Center, since 2017. His research interests include computational number theory and cryptology and their applications to practical problems.

Dr. Cheon served as a Program Committee for over 40 international conferences, including Crypto, Eurocrypt, and Asiacrypt. He was the PC Co-Chair of ANTS-XI and Asiacrypt, from 2015 to 2016. He is an Associate Editor of Designs, Codes and Cryptography and the Journal of Communications and Networks. He received the Best Paper Award in Asiacrypt 2008 and Eurocrypt 2015. He was a recipient of the POSCO TJ Park Prize for Science, in 2019.

DONGWOO KIM (Associate Member, IEEE) received the B.S. and Ph.D. degrees in mathematical sciences from Seoul National University, Seoul, South Korea, in 2013 and 2020, respectively. Since 2020, he has been a Principal Engineer in security and cryptography at Western Digital Research, Milpitas, CA, USA. Before that, he has been a Researcher at the Industrial and Mathematical Data Analytics Research Center, Seoul National University. His research interests include the improvement of homomorphic encryption, verifiable computation, and other cryptographic primitives for practical applications.

DAEJUN PARK received the B.S. and M.S. degrees from Seoul National University and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign. He is currently a Senior Blockchain Security Engineer at a16z crypto, developing formal methods and tools for web3 security to help portfolio companies in particular and the web3 community in general to raise their security bar. Prior to joining a16z crypto, he was the Director of Formal Verification at Runtime Verification, a research-based startup offering formal verification tools and services, where he led a team of formal verification engineers and security auditors for smart contracts and consensus protocols. Early in his career, he was a Founding Member of another tech startup Sparrow, where he designed and implemented static program analysis tools that detect memory safety errors for embedded systems software.

Dr. Park’s work on Language-Parametric Program Verification received the Distinguished Paper Award at OOPSLA’16.