이학 박사 학위논문

Verifiable Computing for Approximate Arithmetic

(근사 연산에 대한 계산 검증 연구)

2020년 2월

서울대학교 대학원

수리과학부

김동우

Verifiable Computing for Approximate Arithmetic

(근사 연산에 대한 계산 검증 연구)

지도교수 천정희

이 논문을 이학 박사 학위논문으로 제출함

2019년 10월

서울대학교 대학원

수리과학부

김동우

김동우의 이학 박사 학위논문을 인준함

2019년 12월

위 원 장	김 명 환	(인)
부 위 원 장	천 정 희	(인)
위 원	이 향 숙	(인)
위 원	현 동 훈	(인)
위 원	윤 아 람	(인)

Verifiable Computing for Approximate Arithmetic

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Dongwoo Kim

Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Sciences
Seoul National University

February 2020

Abstract

Verifiable Computing for Approximate Arithmetic

Dongwoo Kim

Department of Mathematical Sciences

The Graduate School

Seoul National University

Verifiable Computing (VC) is a complexity-theoretic method to secure the integrity of computations. The need is increasing as more computations are outsourced to untrusted parties, e.g., cloud 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, data analysis, and scientific computation) that inherently require approximate arithmetic.

In this thesis, we present an efficient interactive proof system for arithmetic circuits with rounding gates that can represent approximate arithmetic. The main idea is to represent the rounding gate into a small subcircuit, 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 represent the rounding operation by 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, we further optimize the proof generation cost for rounding by employing a Galois ring. We provide experimental results that show the efficiency of our system for approximate arithmetic. For example, our implementation performed two orders of magnitude better than the existing system for a nested 128×128 matrix multiplication of depth 12 on the 16-bit fixed-point arithmetic.

Key words: Verifiable Computing, Approximate Arithmetic
Student Number: 2013-20228

Contents

Chapter 1

Introduction

Outsourcing computation has becoming omnipresent in recent technologies such as cloud computing and distributed computing, since it can facilitate effective distribution and utilization of computational power and storage beyond the physical limits of various devices. However, an inherent problem is that due to errors or corruptions caused by communications, mistake of a delegatee, or a malicious adversary, the result of delegated computation is not always guaranteed to be correct. To resolve this problem, various solutions exploiting several delegatees [CRR11, CLS12] or trusted hardwares [SSW10] have been proposed, but they relied on imperfect assumptions that at least one delegatee or the trusted hardware is always correct. The fundamental solution to this problem: securing integrity of delegated computation, has been studied and proposed under the name *verifiable computing* in the realm of complexity theory and cryptography.

1.1 Verifiable Computing

Given a computation to be outsourced, we call a delegator a verifier, and a delegatee a prover. In Verifiable Computing (VC) scheme [GGP10] (or protocol [IKO07, GKR08]), the prover provides a proof that his claimed result is correct, and the verifier given that proof and the result, efficiently checks if the result is correct or not. The fascinating property of VC is that prover can not deceive the verifier with wrong result, and that verifier can check the correctness with much less computational cost than the cost of executing the computation by itself. With this properties, VC is regarded as a genuine solution to secure the integrity of outsourced computation.

After splendid theoretical constructions [BFLS91, Mic94, IKO07, GKR08, GGP10, GGPR13], existing literature has demonstrated the feasibility of several basic primitives, such as addition, multiplication, comparisons [VSBW13], set operations [KPP+14], and key-value store retrieval [SAGL18]. Using these primitives, VC was shown feasible for a number of tasks, including matrix multiplication [Tha13a, PHGR13, SBV+13], certain SQL-like queries [ZGK+17], and state-machine updates [BFR+13].

However, all existing VC targets computations represented by an arithmetic circuit over a finite field which captures all NP problems in theory (with boolean circuit), but incurs significant blowup of size when representing many kinds of computations in practice. An approximate arithmetic which will be described more precisely in the next subsection is one of such computations suffering substantial blowup of cost when transformed to an arithmetic circuit over a finite field, and has been excluded from the domain of practical verifiable computation.

1.2 Verifiable Approximate Arithmetic

The existing VC 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 $\lfloor 1.11 \times 2.22 \rceil_2 = 2.46$ (where $\lfloor \cdot \rceil_2$ 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 diverse types of computations (e.g., machine learning, data analysis, and scientific computation) 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. Specifically, in the integer scaling method, fractional number inputs are multiplied by some 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 $O(2^d)$ cost blowup, which is far from being practical.

1.2.1 Problem: Verification of Rounding Arithmetic

We can formalize the approximate arithmetic which is a target of our study as follows. Suppose we are given an arithmetic circuit on fixed-point arithmetic with η 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 $\mathbb{Z}_{2^{2\eta}} := \mathbb{Z}/2^{2\eta}\mathbb{Z}$ (i.e., a ring of integers modulo $2^{2\eta}$) with an additional *rounding* operation, as follows: (i) Every input $\in [0, 1)$ is multiplied by 2^{η} 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 η most significant bits, i.e., $x \to \lfloor x/2^{\eta} \rceil$.[i] 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.* We note that the difficulty of verifiable approximate arithmetic originates from the *rounding* operation which can not be represented efficiently by a polynomial over a finite filed $\mathbb{F}$, and our study is focused on this *rounding* operation.

1.2.2 Motivation: Verifiable Machine Learning (AI)

We end this section introducing our vision on verifiable machine learning or AI which motivated this study. 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. Unlike programming

[i] Here, the most significant bits are extracted considering the output of multiplication as 2η-bit element. Indeed, the proper rounding is $x \to \lfloor x/2^{\eta} \rceil$ which is easily expressed with $\lfloor \cdot \rceil$ as $x \to \lfloor (x + 2^{\eta-1})/2^{\eta} \rfloor$, and we use $\lfloor \cdot \rceil$ for simple description.

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.

1.3 List of Papers

This thesis contains the results of the following paper.

- [CCKP19] Shuo Chen, Jung Hee Cheon, Dongwoo Kim, Daejun Park: Verifiable Computing for Approximate Computation. IACR Cryptology ePrint Archive, 2019.

Chapter 2

Preliminaries

In this chapter, we review a number of basic concepts about Verifiable Computing (VC). Specifically, we recall interactive proof and interactive argument; Schwartz-Zippel lemma [Sch80] and sum-check protocol [LFKN92]. Finally, we review Goldwasser, Kalai, and Rothblum's interactive proof protocol (the GKR protocol [GKR08]) for an arithmetic circuit over a finite field. The GKR protocol and several refinements [CMT12, VSBW13, Tha13a, XZZ$^+$19] of it constitute the state-of-the-art VC systems based on interactive proof or argument, and our proposed VC system for approximate computation also stems from it.

2.1 Interactive Proof and Argument

We start with the definition of an interactive proof and an interactive argument for a function f as follows.

Definition 1. (Interactive Proof (or Interactive Argument) for f [CMT12, Tha13a]) Consider a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ who wishes to compute a

function $f : X \to Y$. A pair $(\mathcal{P}, \mathcal{V})$ of interactive algorithms is called *interactive proof for* f if the following holds. 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 $y = f(x)$, $\Pr[\mathcal{V} \text{ accepts}] = 1$.
- δ-*Soundness*. For all inputs $x \in X$, and for all (malicious) $\mathcal{P}'$ interacting with $\mathcal{V}$, if $y \neq f(x)$, $\Pr[\mathcal{V} \text{ accepts}] < \delta$.

We will call δ the soundness probability bound. If $\mathcal{P}$ and $\mathcal{V}$ exchange r messages in total, we say the protocol has $\lfloor r/2 \rceil$ rounds. If the δ-*Soundness* condition only holds for a computationally bounded prover $\mathcal{P}'_b$, we call the pair $(\mathcal{P}, \mathcal{V})$ *interactive argument for* f.

2.2 Sum-Check Protocol

Before we introduce the sum-check protocol, we recall the Schwartz-Zippel lemma as follows.

Lemma 1. *(Schwartz-Zippel [Sch80]) Let $\mathbb{F}$ be a field, and $f : \mathbb{F}^\nu \to \mathbb{F}$ be an ν-variate nonzero polynomial of total degree (the sum of degrees of each variable) D. Then on any finite set $A \subseteq \mathbb{F}$ with $D \leqslant |A|$, $\Pr_{\vec{x} \leftarrow A^\nu}[f(\vec{x}) = 0] \leqslant \frac{D}{|A|}$.*

Note that the lemma implies that two different polynomials can coincide at only tiny fraction of points. It contributes to the soundness of following sum-check protocol and the GKR protocol described later. Now, we introduce the sum-check protocol [LFKN92] as follows.

Theorem 1. *(Sum-Check Protocol [LFKN92]) Let $\mathbb{F}$ be a finite field. Let $f : \mathbb{F}^\nu \to \mathbb{F}$ be an ν-variate polynomial of degree at most $d < |\mathbb{F}|$ in each variable. The Sum-Check protocol is an interactive proof protocol $(\mathcal{P}, \mathcal{V})$ with soundness $\frac{\nu d}{|\mathbb{F}|}$ for the function:*

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

The computational cost[i] *of $\mathcal{P}$ is $O(d2^\nu)O(f)$, and the cost of $\mathcal{V}$ is $O(d\nu) + O(f)$ where $O(f)$ is the cost to evaluate f on one point. The communication cost which counts the number of field elements transferred is $O(d\nu)$.*

Protocol description: The protocol proceeds in n rounds. We explain the case where $\mathcal{P}$ is honest, and the proof shows that if $\mathcal{P}$ is not honest, he can not convince $\mathcal{V}$.

In the first round, $\mathcal{P}$ sends the value $S(f)$, and a polynomial

$$f_1(t) := \sum_{(x_2, x_3, \ldots, x_\nu) \in \{0,1\}^{\nu-1}} f(t, x_2, x_3 \ldots, x_\nu).$$

$\mathcal{V}$ checks if $f_1(0) + f_1(1) = S(f)$, and rejects otherwise.

In the i-th $(2 \leqslant i \leqslant \nu)$ round, $\mathcal{V}$ chooses r_{i-1} randomly from $\mathbb{F}$, and sends it to $\mathcal{P}$. In response, $\mathcal{P}$ sends a polynomial

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

$\mathcal{V}$ checks if $f_{i-1}(r_{i-1}) = f_i(0) + f_i(1)$, and rejects otherwise.

After the final ν-th round, $\mathcal{V}$ chooses r_ν randomly from $\mathbb{F}$, and accepts

[i] The cost counts the number of field operations $(+, \times)$ required.

if $f_\nu(r_\nu) = f(r_1, r_2, \ldots, r_\nu)$, and rejects otherwise.

Proof. The completeness condition and the cost of $\mathcal{P}, \mathcal{V}$, and communication directly follows from the protocol description. The main idea for showing soundness condition can be summarized as follows (see [GKR08] or [LFKN92] for the full proof). Assume that a (dishonest) $\mathcal{P}'$ sends an incorrect result $S(f)' \neq S(f)$ to $\mathcal{V}$. Let us distinguish the values claimed by $\mathcal{P}'$ from the values which would be claimed by an honest $\mathcal{P}$ by adding the prime ($'$) symbol. Then $f_1(t)' \neq f_1(t)$. Otherwise, $\mathcal{V}$ will reject immediately by checking if $S(f)' = f_1(0)' + f_1(1)'$. When $\mathcal{V}$ 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 $f_1(r_1)' \neq f_1(r_1)$, $\mathcal{P}'$ *must* send $f_2(t)' \neq f_2(t)$ because of the same reasoning as before. Continuing this, $\mathcal{P}'$ must send $f_\nu(t)' \neq f_\nu(t) = f(r_1, \ldots, r_{\nu-1}, t)$, and will be rejected with the high probability by $\mathcal{V}$ who finally checks if $f_\nu(r_\nu)' = f(r_1, \ldots, r_\nu)$ for a randomly chosen r_ν in $\mathbb{F}$. The soundness probability bound is derived from the probability $1 - (1 - \frac{d}{|\mathbb{F}|})^\nu$ that at least one of the above high probability events does not occur during the protocol. □

As the proof shows, the soundness of sum-check protocol is based on Schwartz-Zippel lemma (Lemma 1). Note that the sum-check protocol enables $\mathcal{V}$ to *reduce* the verification task on the correctness of $S(f)$ to that on the correctness of evaluation of f on one random point. It is the core utility of sum-check in the following GKR protocol.

2.3 The GKR Protocol

Before introducing the GKR protocol, we recall the multilinear extension (MLE).

Lemma 2. *(Multilinear Extension [CMT12]) Given a function* $V : \{0,1\}^\mu \to \mathbb{F}$*, there exists a unique multilinear polynomial*[ii] $\tilde{V}(\vec{x}) : \mathbb{F}^\mu \to \mathbb{F}$ *extending* V*, i.e.,* $\tilde{V}(\vec{x}) = V(\vec{x})$ *for all* $\vec{x} \in \{0,1\}^\mu$*. We call* $\tilde{V}$ *the* multilinear extension *(MLE) of* V *over* $\mathbb{F}$*.*

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

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

The uniqueness follows from an observation that any multilinear polynomial $\tilde{V}(x_1, x_2, \ldots, x_\mu)$ can be represented by $\sum_{b \in \{0,1\}^\mu} C(b) x_b$, where $x_b := \Pi_{i \in I} x_i$ 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 $\tilde{V}(b)$'s for $b \in \{0,1\}^\mu$. Specifically, for a zero vector $\vec{0}$, $C(\vec{0}) = \tilde{V}(\vec{0})$. For an elementary vector e_i whose i-th component is 1 and all others are 0, $C(e_i) = \tilde{V}(e_i) - C(\vec{0})$. For a vector $e_{i,j} \in \{0,1\}^\mu$ $(i \neq j)$ whose i-th and j-th components are 1 and all others are 0, $C(e_{i,j}) = \tilde{V}(e_{i,j}) - C(e_i) - C(e_j) - C(\vec{0})$. Continuing this process with increasing the weight of each vector $b \in \{0,1\}^\mu$, we can see that every $C(b)$ for $b \in \{0,1\}^\mu$ is uniquely determined by $\tilde{V}(b)$. □

In GKR protocol, the output of each layer in the circuit gives rise to the unique multilinear extension. Now, we describe the GKR protocol which

[ii] An μ-variate polynomial $f(x_1, \ldots, x_\mu) : \mathbb{F}^\mu \to \mathbb{F}$ is called *multilinear* if it is linear in each variable, e.g., $f(x_1, x_2, x_3) = a x_1 x_2 x_3 + b x_2 x_3 + c x_3$.

is an interactive proof protocol for the evaluation of a layered arithmetic circuit over a finite field $\mathbb{F}$.[iii] We only give an overview of the protocol, and a detailed description can be found in [GKR08, Tha13b] or in Section 4.2.2.

Overview of the GKR protocol. Assume we are given a layered[iv] arithmetic circuit (over $\mathbb{F}$) of depth d, of size (the number of gates) S, and of fan-in 2 (i.e., each gate has 2 input). Each layer is composed of addition gates and multiplication gates outputting addition and multiplication of two inputs, respectively. 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., $S_i = 2^{s_i}$ for simplicity. We can number each gate of i-th layer with a binary string in $\{0,1\}^{s_i}$, and it defines a function $V_i : \{0,1\}^{s_i} \to \mathbb{F}$ relating the given binary string to output of the corresponding gate. Let $\tilde{V}_i$ be the MLE of V_i, then there exists an interesting relation between MLEs defined from adjacent layers as follows [Tha15]: (We omit the vector notation, e.g., $\vec{z}$, $\vec{\omega_1}$, $\vec{\omega_2}$ are denoted by z, ω_1, ω_2.)

$$\begin{aligned}\tilde{V}_i(z) = \sum_{(\omega_1,\omega_2)\in\{0,1\}^{2s_{i+1}}} [&\tilde{add}_i(z,\omega_1,\omega_2)(\tilde{V}_{i+1}(\omega_1) + \tilde{V}_{i+1}(\omega_2)) \\ &+ \tilde{mult}_i(z,\omega_1,\omega_2)(\tilde{V}_{i+1}(\omega_1)\tilde{V}_{i+1}(\omega_2))]\end{aligned} \tag{2.3.1}$$

where $\tilde{add}_i$ (or $\tilde{mult}_i$) is a MLE of a function add_i (or $mult_i$) which is 1 only if the input binary strings indicate an addition (or multiplication) gate

[iii] In particular, we describe the recent refinement [CMT12, Tha13a, Tha15] of the GKR protocol that our technical development later will be based on.

[iv] Every circuit can be transformed to layered form increasing the circuit size at most d (depth of the circuit) times.

and its corresponding two gates providing inputs, and 0 otherwise. More precisely, the function $add_i(z, \omega_1, \omega_2)$ is 1 only if the gate indicated by z at i-th layer is an addition gate whose left input and right input are the output of the gates indicated by ω_1 and ω_2 at $i+1$-th layer, respectively. The function $mult_i(z, \omega_1, \omega_2)$ is defined similarly for the multiplication gate. We call $\tilde{add}_i$ (and $\tilde{mult}_i$) *wiring predicates* as in [CMT12].

Now, the GKR protocol proceeds in layer by layer, starting from the output layer. $\mathcal{V}$ 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 (recall that the output layer corresponds to 0-th layer). Then, she reduces this claim to $\tilde{V}_1(r_1) = v_1$ and $\tilde{V}_1(r_2) = v_2$ where r_1 and r_2 are randomly chosen by $\mathcal{V}$ executing the sum-check protocol on the relation of MLEs (equation 2.3.1) we described above.

The claims $\tilde{V}_1(r_1) = v_1$ and $\tilde{V}_1(r_2) = v_2$ can be reduced to one as follows. $\mathcal{V}$ asks $\mathcal{P}$ to send $h(t) := \tilde{V}_1(l(t))$ where $l(t)$ is the line such that $l(0) = r_1$ and $l(1) = r_2$ (note the line is uniquely determined). Then, given $h(t)$, $\mathcal{V}$ checks if $h(0) = v_1$ and $h(1) = v_2$, then samples a random point $r \in \mathbb{F}$ to get the claim $\tilde{V}_1(l(r)) = h(r)$, and proceeds to the next layer with this claim.

Continuing this process layer by layer, $\mathcal{V}$ finally gets a claim that $\tilde{V}_d(z_d) = v_d$, and checks if it is correct by evaluating $\tilde{V}_0$ defined with her inputs.

Wiring predicates. In the process of GKR protocol, $\mathcal{V}$ must evaluate wiring predicates $\tilde{add}_i(z, \omega_1, \omega_2)$ and $\tilde{mult}_i(z, \omega_1, \omega_2)$ by itself. When the circuit is log-space uniform, computing the wiring predicates can be done in $O(poly(\log S))$ cost, which is much less than the circuit evaluation cost $O(S)$. In general circuit, however, the cost of computing the wiring pred-

icates can be $\Omega(S)$. In this case, that high cost of $\mathcal{V}$ can be amortized by batching [VSBW13], i.e., the wiring predicates are computed at once before the GKR protocol, and $\mathcal{V}$ carries the protocol on many input-ouput pairs simultaneously (with the same random values) exploiting that predicate values. In this thesis, we assume that the circuit is highly regular, so that the wiring predicates can be efficiently computable by $O(\log S)$ cost.

Computational cost. In the original GKR protocol, the computational cost $O(poly(S))$ of $\mathcal{P}$ was a main bottleneck. It was improved to $O(S \log S)$ in [CMT12] exploiting sparsity of wiring predicates, and further improved to $O(S)$ in [Tha13a] using reusing work technique motivated by [VSBW13] when the circuit has highly regular wiring patterns. Finally, Xie *et al.* [XZZ+19] showed that the cost of $\mathcal{P}$ can be $O(S)$ for all general circuits using clever bookkeeping method motivated by previous work[v].

Therefore, the computational cost of $\mathcal{P}$ and $\mathcal{V}$, and the communication cost $\mathcal{C}$ in the number of operations or elements over $\mathbb{F}$ are as follows:[vi]

$$\mathcal{P} : O(S), \quad \mathcal{V} : O(n + d \log S), \quad \mathcal{C} : O(d \log S) \tag{2.3.2}$$

where n is the number of input and output values.

We note that $\mathcal{P}$'s cost can be broken down into the circuit evaluation cost and the proof generation cost. They are asymptotically the same in general, but later we will show certain circuits for which the proof generation cost is smaller than the circuit evaluation cost (Section 4.4.3).

[v] In fact, they also exploit Chiesa *et al.* [CFS17]'s approach using random linear combination to reduce $\mathcal{P}$'s cost for *reduction to verification at a single point* step (see Section 4.2.2) to $O(S)$ from $O(S \log S)$.

[vi] We assume that the *wiring predicates* of circuit can be efficiently computable [Tha13a], or the cost can be amortized by batching [VSBW13] or data-parallel computations [Tha13b, WJB+17].

The GKR protocol can be summarized as follows.

Theorem 2.3.1. *(GKR Protocol [GKR08, CMT12, Tha13a, XZZ+19]) Let $C : \mathbb{F}^n \to \mathbb{F}$ be a layered arithmetic circuit over a finite field $\mathbb{F}$. Let S and d be the size and depth of C, respectively, and n be the number of input. The GKR protocol (with recent refinements) is an interactive proof protocol $(\mathcal{P}, \mathcal{V})$ for C with soundness $O(\frac{d \log S + \log n}{|\mathbb{F}|})$. The computational cost of $\mathcal{P}$ and $\mathcal{V}$ is $O(S)$ and $O(n + d \log S)$[vii], respectively, while the communication cost is $O(d \log S)$.*

Proof. The soundness can be similarly proved as that of the sum-check protocol (Theorem 1). For detailed proof, see [GKR08] or [Rot09]. Derivation of cost can be found in [Tha13a] for highly regular circuit, or in [XZZ+19] for a general circuit. □

2.4 Notation and Cost Model

In this paper, $\mathbb{Z}$, $\mathbb{Z}_N$, and $\mathbb{F}$ denote the ring of integers, the ring of integers modulo a positive integer N, and a finite field, respectively. Also, all logarithms are of base 2. When we say the (time) cost of $\mathcal{P}$ or $\mathcal{V}$, it counts the number of arithmetic operations over the corresponding domain, such as $\mathbb{F}$ or $\mathbb{Z}_N$. Similarly, the communication cost measures the number of elements of the corresponding domain. Hereafter, we use MLE for an abbreviation of multilinear extension (Lemma 2), $\mathcal{P}$ for prover, and $\mathcal{V}$ for verifier.

[vii] We assume that the *wiring predicates* of circuit can be efficiently computable [Tha13a], or the cost can be amortized by batching [VSBW13] or data-parallel computations [Tha13b, WJB+17].

Chapter 3

Related Work

The problem of delegating computation with securing integrity has been extensively studied in both theory and practice perspectives. In this chapter, we review some general-purpose protocols and systems that aim to be practical. The systems can be divided into two categories: interactive proofs and (non)-interactive arguments.

3.1 Interactive Proofs

Goldwasser, Kalai, and Rothblum [GKR08] 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 [CMT12] presented a refinement of the GKR porotocol (hereafter, CMT) that allows the prover to run in $O(S \log S)$. Thaler [Tha13a] fur-

ther improved the protocol, which allows the prover to run in $O(S)$ for a circuit with a "sufficiently" 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 $O(S \log S_c)$ in [Tha13b, ZGK+17], and further reduced to $O(S + S_c \log S_c)$ in [WJB+17], where S_c is the size of a subcircuit. Recently, Xie *et al.* [XZZ+19] proposed a refinement that allows the prover to run in $O(S)$ for an arbitrary circuit. Although being asymptotically equivalent, Thaler's refinement [Tha13a] still performs better than Xie *et al.*'s [XZZ+19] for a regular circuit.

On the other hand, substantial efforts have been made to support more operations than the plain field arithmetic. Vu *et al.* [VSBW13] 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.* [ZGK+17] improved this by combining CMT with a verifiable polynomial delegation scheme, and showed that an arithmetic circuit with auxiliary inputs can be efficiently verified.

There are other lines of refinement work such as supporting "streaming" verifiers [CCM09, CTY11] that run in a limited space; employing hardware accelerators such as ASICs and GPUs [WHG+16, WJB+17, TRMP12]; and supporting zero-knowledge proofs [WTS+18, XZZ+19].

Note that, however, 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.[i] Though Vu *et al.* [VSBW13]'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. Therefore, it does not provide efficient verification of rounding operations. Zhang *et al.* [ZGK+17]'s approach resolves this problem using polynomial commitment scheme. However, due to the use of the polynomial commitment, their system became an argument that is secure only against *computationally bounded* dishonest prover. In particular, their system does *not* provide post-quantum security due to the specific polynomial commitment scheme employed in their system. Also, the polynomial commitment scheme is quite slow in practice when the size $|w|$ of witness is large, since the cost of prover is $\Omega(|w| \log^2 |w|)$.

3.2 (Non-)Interactive Arguments

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-interactiveness, 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 [Kil92, Mic94, BSS08, BSCGT13b] of developing argument systems based on probabilistically checkable proofs

[i] 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.

(PCPs) [AS98, ALM+98], 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.* [SMBW12, SVP+12, SBV+13] proposed argument systems based on linear PCPs [IKO07], where their systems were shown to achieve a practical performance in the batch verification setting. Gennaro *et al.* [GGPR13] introduced quadratic arithmetic programs (QAPs), a novel efficient encoding of computations, and proposed a zero-knowledge succinct non-interactive argument system (zkSNARK). Much of improvements have been proposed [PHGR13, BSCG+13, BSCTV14, Gro16], but these argument systems suffer from a trusted setup cost that needs to be amortized to be practically efficient. The trusted setup issue, however, has been largely addressed in recent work [BSBHR18, BSCR+18, BBB+18, WTS+18, AHIV17, Set19]. Still, in these work, prover's cost is quasi-linear $O(|C| \log |C|)$ or verifier is not efficient, in contrasts to the latest refinement [XZZ+19] of GKR protocol providing linear prover cost $O(|C|)$ and efficient verifier (in certain types of circuits).

There also has been substantial work [BSCGT13a, BSCG+13, BSCTV14, BFR+13, WSR+15] 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, [SVP+12,

SBV+13] 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 1.2.

Chapter 4

Interactive Proof for Rounding Arithmetic

In this chapter, we describe our interactive proof for rounding arithmetic. We first show that the GKR protocol can be made valid over a ring which is a more general oboject than a field. Then, from the observation that rounding operation can be efficiently representable by arithmetic in certain rings, we propose an efficient interactive proof for rounding arithmetic.

4.1 Overview of Our Approach and Result

We begin by providing an overview of the technical details of our approach and result. Our goal is to construct an interactive proof for fixed-point arithmetic circuits (i.e., arithmetic circuits with rounding gates). The idea is to reduce the rounding gate into 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), where p is a prime and $e > 1$, and the (floor) rounding operation, $x \mapsto \lfloor x/p \rfloor$. (Note that proper rounding, $\lfloor x/p \rceil$, can be represented using floor rounding, i.e., $\lfloor x/p \rceil = \left\lfloor (x + \frac{p-1}{2})/p \right\rfloor$.) Below we explain each of our main technical developments.

Reducing rounding to a combination of the plain ring operations (Section 4.3). We present a sub-circuit representation of the rounding gate over the base p system. At first, we employ the lowest digit removal polynomial, ldr [CH18]. The polynomial ldr sets the least significant digit to zero, i.e., $\mathsf{ldr} : x \mapsto \lfloor x/p \rfloor \cdot p$, and thus we can have the floor rounding operation by $\mathsf{ldr}(x)/p$. We exploit the fact that ldr is the polynomial whose degree is $< ep$, while the degree of such a polynomial could be as large as p^e if it is generated by using the general interpolation technique.[i] Then, we construct an optimal arithmetic sub-circuit over $\mathbb{Z}_{p^e}$ that computes ldr by using our optimal circuit construction method that we will explain below.

Optimal circuit construction for arbitrary univariate polynomial (Section 4.4). In the GKR protocol (as well as our generalized one), a computation of interest needs to be represented in the form of an arithmetic circuit, and the performance of the protocol could be largely affected by the structure of a circuit. Now that we have the aforementioned rounding polynomial, it is important to carefully construct a circuit of the polynomial to achieve good performance. To this end, we devised a novel, optimal circuit construction of an *arbitrary* polynomial for the GKR protocol. A constructed circuit is regular with depth $O(\log d)$ and size $O(d)$ where d is the degree of the polynomial. The circuit construction is optimal in that

[i] Moreover, when $e > 1$ or p is not a prime, such polynomials may not even exist, where the interpolation techniques are not applicable.

the proof generation complexity is linear in d. Moreover, in case that the same polynomial is evaluated on m 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 faster than even the circuit evaluation!), while the previously best known result is linear [XZZ+19]. 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 analyze the Paterson-Stockmeyer polynomial evaluation method [PS73], and carefully design the circuit by exploiting the linear-sum gate, $(x_1, \cdots, x_n) \mapsto a_1x_1 + \cdots + a_nx_n$, and the fused multiply-add gate, $(x, y, z) \mapsto xy + z$, which can be efficiently verified via the Sum-Check (and GKR) protocol.

Generalization of the GKR protocol over a ring (Section 4.2). While the original GKR protocol is valid over a finite field, since the domain $\mathbb{Z}_{p^e}$ we consider is no longer a field for $e > 1$, we identify a minimal modification to the original protocol to admit a ring (Section 4.2.2), and present its construction for a specific family of rings, i.e., $\mathbb{Z}_{p^e}$ 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 [BCPS18] 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 4.2.1), the "unit difference" property [MP12], that is, that the difference between any two elements of the sampling set has an inverse. This stronger condition allows us to employ the cost reduction technique [Tha13a] proposed for the original GKR protocol to our extended protocol.

The extended protocol enjoys the same complexity with the original, provided that the unit difference property holds for the sampling set A. Specifically, given a circuit of size S and depth D, the prover's cost is $O(S \log S)$,[ii] and the verifier's cost is $O(n + D \log S)$, where $O(n)$ is the additional cost (for generating the multi-linear extension) at the input/output layer, and n is the number of input/output values. The communication cost is $O(D \log S)$. The soundness probability, however, becomes bigger (i.e., worse) than that of the original. That is, it is bounded by $(7D \log S + \log n)/|A|$, where the denominator is the size of the sampling set A, while it was the size of the entire domain for the original protocol. Note that, however, 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.

Optimization of proof generation cost for rounding (Section 4.5). Consider an approximate computation on $\mathbb{Z}_{p^e}$. The underlying ring $\mathbb{Z}_{p^e}$

[ii]The prover's cost becomes $O(S)$ if the generalization is made on top of the latest GKR variant [XZZ+19].

can be replaced by another ring $\mathbb{Z}_{q^{de}}$ with a much smaller prime $q \simeq \sqrt[d]{p}$, via base conversion, that is, converting numbers in the base-p system to the corresponding numbers in the base-q system.[iii] Here the advantage of employing $\mathbb{Z}_{q^{de}}$ is that the size of the rounding polynomial in $\mathbb{Z}_{q^{de}}$ is much smaller than that of $\mathbb{Z}_{p^e}$, which in turn significantly reduces the proof generation cost for rounding. However, employing $\mathbb{Z}_{q^{de}}$ leads to sacrificing the soundness of the protocol. To mitigate this dilemma, we proposed a technique that allows us to employ $\mathbb{Z}_{q^{de}}$ without compromising the soundness, by exploiting an interesting property of a Galois ring.

Specifically, we employ a Galois ring, $\mathbb{Z}_{(q^d)^e}[t]/f(t)$, where $f(t)$ is a monic irreducible polynomial, in the proof generation and verification phases, while we keep using $\mathbb{Z}_{p^e}$ in the circuit evaluation phase. This allows us to employ a smaller prime $q \sim \sqrt[d]{p}$ where d is the degree of $f(t)$. 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 $edq \simeq ed\sqrt[d]{p}$. 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, $q^d \simeq p$, to that of the original one (Theorem 4).

However, there is a cost overhead when employing a Galois ring, since the operations on a Galois ring become more expensive as its dimension 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 5.2.1).

[iii] The converted number may be marginally different from the original, but such an inaccuracy is acceptable in approximate computation such DNN training.

Our Result. We briefly summarize the cost of our interactive proof for a rounding arithmetic circuit. The complexity of our interactive proof protocol for an arithmetic circuit with the rounding gates is described in the following theorem.

Theorem 4.1.1. *Let $C : \mathbb{Z}_{2^{2\eta}}^n \to \mathbb{Z}_{2^{2\eta}}$ be a layered arithmetic circuit over $\mathbb{Z}_{2^{2\eta}}$ where the multiplication gate performs regular multiplication followed by rounding, i.e., $(x, y) \mapsto \lfloor (xy)/2^\eta \rceil$. (Thus, C corresponds to a fixed-point arithmetic circuit with η fractional bits only.) Let us fix η. Let S be the size of C, d be the depth of C, and n be the number of inputs. Then, our interactive proof protocol $(\mathcal{P}, \mathcal{V})$ for C has soundness $O((d \log d \log dS)/2^\lambda)$. The computational cost of $\mathcal{P}$ is $O(dS)$, the cost of $\mathcal{V}$ is $O(n + d \log d \log dS)$*[iv]*, and the communication cost is $O(d \log d \log dS)$. Here the unit cost is $M(\lambda)M(d)$ where $M(\ell)$ denotes the cost of an arithmetic operation on ℓ-bit elements (or polynomials of degree ℓ). Note that we do not take into account the η factor in the asymptotic costs, since η is fixed to a small constant.*

Our protocol is based on Goldwasser, Kalai, and Rothblum's interactive proof system (GKR protocol) [GKR08] and its recent refinements [CMT12, Tha13b, XZZ+19]. The costs of the latest GKR protocol variant [XZZ+19] (that do *not* support fixed-point arithmetic) are $O(S)$, $O(n + d \log S)$[iv], 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.

Below we compare the asymptotic complexity of our protocol with the integer scaling method applied on top of the latest GKR variant. The

[iv] We do not take into account the $\mathcal{V}$'s cost for computing wiring predicate [CMT12], or we assume that the circuit is highly regular [Tha13a].

integer scaling method incurs exponential overhead in circuit depth to deal with fixed-point arithmetic, as mentioned earlier.

	Integer Scaling Method	Ours (Theorem 4.1.1)
Circuit eval.	$M(2^d)O(S)$	$M(d)O(dS)$
Proof gen.	$M(2^d)O(S)$	$M(\lambda)M(d)O(dS)$
Verification	$M(2^d)O(n + d \log S)$	$M(\lambda)M(d)O(n + d \log d \log dS)$
Soundness	$O((d \log S)/2^{2^d})$	$O((d \log d \log dS)/2^{\lambda})$

We also 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 128×128 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 5.3). 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).

4.2 Interactive Proof 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.

Notation and preliminary. Throughout this thesis, 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 $\mathbb{Z}$ (integers) and $\mathbb{Z}_N$ (integers modulo N). Note that $\mathbb{Z}$ is an integral domain, and $\mathbb{Z}_N$ is a field if N is a prime, but is not even an integral domain otherwise.

4.2.1 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.* [BCPS18] as follows.

Lemma 3. *(Generalized Schwartz-Zippel [BCPS18]) Let R be a ring, and $f : R^n \to R$ 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 $|A| \geqslant D$ such that $\forall x \neq y \in A$, $x - y \in R$ is not a zero divisor. Then, $\Pr_{\vec{x} \leftarrow A^n}[f(\vec{x}) = 0] \leqslant \frac{D}{|A|}$. We will call A a* sampling set.

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 $f(t)$. □

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 4.2.1. *Let $R = \mathbb{Z}_{p^e}$ 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 1) over R, only restricting the random points chosen by $\mathcal{V}$.

Theorem 2. *(Generalized Sum-Check Protocol) Let R be a finite ring, $f : R^n \to R$ 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:*

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

Proof. The proof is almost the same as that of the original sum-check protocol. The gerenalized Schwartz-Zippel lemma (Lemma 3) implies that any two distinct univariate polynomials of degree $\leqslant d$ over R agree on at

most d points among A. Following the proof of the original Sum-Check protocol (Theorem 1), the soundness probability of the generalized sum-check protocol is bounded by $\frac{nd}{|A|}$. □

Note that the soundness probability is $\frac{nd}{|A|}$ in contrast to $\frac{nd}{|\mathbb{F}|}$ in Theorem 1.

Remark 4.2.1 (Additional condition for efficient specification of $f_i(t)$)**.** *In the i-th round of the Sum-Check protocol, (honest) $\mathcal{P}$ should provide*

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

to $\mathcal{V}$. While the $f_i(t)$ is specified by evaluations of it on $deg_i f + 1$ 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 4.2.2. *Let $R = \mathbb{Z}_{p^e}$ for an odd prime p, and $A = \{0, 1, \ldots, p-2, p-1\}$ as Example 4.2.1. 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 $R = \mathbb{Z}_{p^e}$ other than multiples of p have a multiplicative inverse in $R = \mathbb{Z}_{p^e}$*[v]*.*

4.2.2 The 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*

[v] If $x \in \mathbb{Z}_{p^e}$ is not a multiple of p, $\gcd(x, p^e) = \gcd(x, p) = 1$, and $ax + bp^e = 1$ for some $a, b \in \mathbb{Z}$, i.e., $a \pmod{p^e} \in \mathbb{Z}_{p^e}$ is a multiplicative inverse of x.

of Lemma 3. Below we clarify and validate the modification made in each step of the protocol.

Multilinear extension & Initial step. We first need to ensure that the existence and uniqueness (Lemma 2) of Multilinear Extension (MLE) $\tilde{V} : R^n \to R$ extending a function $V : \{0,1\}^n \to R$. 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.

Applying sum-check protocol. We already have shown that the Sum-Check protocol is valid in R as well by Theorem 2. Therefore, reducing the task of checking $\tilde{V}_i(z_i) = v_i$ to that of checking both $\tilde{V}_{i+1}(\omega_1^*) = v_{i+1,1}$ and $\tilde{V}_{i+1}(\omega_2^*) = v_{i+1,2}$ 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.

Reduction to verification at a single point & final step. Reducing the task of checking both $\tilde{V}_{i+1}(\omega_1^*) = v_{i+1,1}$ and $\tilde{V}_{i+1}(\omega_2^*) = v_{i+1,2}$ 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.

Complexity & soundness. Note that the computational cost of the generalized protocol is the same with that of the original protocol (Equation 2.3.2) except that the cost is measured by the number of operations or elements of R instead of $\mathbb{F}$. The cost reduction techniques [CMT12, VSBW13, Tha13a, XZZ+19] proposed in refinements of GKR protocol are also applicable if R satisfies the additional condition introduced in Remark 4.2.1.

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 $|\mathbb{F}|$ is substituted by $|A|$ (see following Theorem 3).

Theorem 3. *(GKR protocol over R) Let $C : R^n \to R$ be an arithmetic circuit over a finite ring R. Let S and d be the size and depth of C, respectively, and n be the number of input. Let A be the* sampling set *of R in Lemma 3. The generalized GKR protocol described above is an interactive proof protocol for C with soundness $O(\frac{d \log S + \log n}{|A|})$. The computational cost and communication cost of the generalized GKR protocol is the same as that of the original GKR protocol (Theorem 2.3.1), except that we use number of operations or elements of R for the unit cost.*

4.3 Verifiable Rounding Operation

In this section, we explain how to support the rounding operation on top of the generalized GKR protocol described in Section 4.2.2. As explained in Section 1.2.1, we consider an approximate arithmetic circuit over a ring $\mathbb{Z}_{p^e}$ (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: $x \mapsto \lfloor x/p \rfloor$.[vi] Like closely related previous work [GKR08, CMT12, Tha13a, XZZ+19], 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.[vii]

The idea is to replace each rounding gate with a combination of plain arithmetic gates, and use our generalized GKR protocol over $\mathbb{Z}_{p^e}$. Specifically, we employ a low-degree polynomial $\mathsf{ldr}(x)$ such that $\lfloor x/p \rfloor = \mathsf{ldr}(x)/p$, where $\mathsf{ldr}(x)$ can be represented as a circuit over addition and multiplication gates. (Later, in Section 4.4, we will provide an optimal circuit construction for arbitrary polynomials including $\mathsf{ldr}(x)$.) Then, the rounding gate can be replaced with the circuit of $\mathsf{ldr}(x)$ followed by a division-by-p gate, $x \mapsto x/p$. Below we will explain what is the polynomial $\mathsf{ldr}(x)$, and how to verify the division-by-p gate in our generalized GKR protocol.

4.3.1 Lowest-Digit-Removal Polynomial over $\mathbb{Z}_{p^e}$

Chen and Han [CH18] recently showed the existence of a polynomial over $\mathbb{Z}_{p^e}$ that sets the input's lowest-digit to zero. They also provided an exact construction of such polynomial.

Lemma 4. *(Lowest-digit-removal polynomial [CH18]) Let p be a prime and $e \geqslant 1$ be a positive integer. Then there exists a polynomial $\mathsf{ldr}(x)$ of degree at most $(e-1)(p-1)+1$ such that for every integer $0 \leqslant x < p^e$,*

[vi] As mentioned earlier, the proper rounding, $\lfloor x/p \rceil$, can be represented using the floor rounding, i.e., $\lfloor x/p \rceil = \left\lfloor (x + \frac{p-1}{2})/p \right\rfloor$.

[vii] An arbitrary circuit can be adjusted to satisfy this assumption by adding dummy gates (i.e., a multiplication-by-p gate followed by a rounding gate) for each non-rounding gate.

we have

$$\mathsf{ldr}(x) \equiv x - (x \bmod p) \mod p^e,$$

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

Remark 4.3.1. *In fact,* $\mathsf{ldr}(x)$ *can be represented by* $\sum_{i=0}^{p-1}(x-i)(1-(x-i)^d)$ *where* d *is a positive integer such that* $t^d = 1$ *if* $t \in \mathbb{Z}_{p^e}$ *is not divisible by* p*, and* 0 *otherwise. However, such* d *is quite large* ($p^e - p^{e-1}$)*, and Lemma 4 provides a more compact form of* $\mathsf{ldr}(x)$ *with degree less than* ep*. We also note that the bound of degree is in fact trivial, since every polynomial over* $\mathbb{Z}_{p^e}$ *is reduced to a polynomial of degree* $< ep$ *using the relation* $(x^p - x)^e = 0$ *in* $\mathbb{Z}_{p^e}$*. Refer to [Car64, JPSZ06, BH17] for the characteristic of functions that are representable by a polynomial over* $\mathbb{Z}_{p^e}$ *(or a finite commutative ring with* 1*).*

Remark 4.3.2. *Note that the degree of* $\mathsf{ldr}(x)$ *is small: roughly* logarithmic *in the size of* $\mathbb{Z}_{p^e}$*. It provides us an efficient representation of rounding as a combination of additions and multiplications.*

Example 4.3.1. *([CH18]) For* $e = 2$*, we have:*

$$\mathsf{ldr}(x) = -x(x-1)\cdots(x-p+1)$$

4.3.2 Verification of Division-by-p Layer

As mentioned earlier, the rounding operation ($x \mapsto \lfloor x/p \rfloor$) can be represented as $x \mapsto \mathsf{ldr}(x)/p$. 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 $\mathsf{ldr}(x)/p$, the division is always well-defined, since the result of $\mathsf{ldr}(x)$ 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:

$$\tilde{V}_i(z) = \tilde{V}_{i+1}(z)/p \tag{4.3.1}$$

where $\tilde{V}_i$ (and $\tilde{V}_{i+1}$) 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 $\tilde{V}_i(r) = v$, to the verification task of $\tilde{V}_{i+1}(r) = pv$. This reduction enjoys perfect soundness, since for $\tilde{V}_i'(r) \neq \tilde{V}_i(r)$, we have $\tilde{V}_{i+1}(r) = p\tilde{V}_i(r) \neq p\tilde{V'}_i(r) = \tilde{V'}_{i+1}(r) \pmod{p^e}$.

Remark 4.3.3 (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 $\mathbb{Z}_{p^e}$. 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 $\mathbb{Z}_{p^e}$. This is because $t = ap + b \in \mathbb{Z}_{p^e}$ represents $(ap + b) + np^e \in \mathbb{Z}$ for some $n \in \mathbb{Z}$ where $0 \leqslant b < p$, while $\lfloor t/p \rfloor \equiv a + np^{e-1} \in \mathbb{Z}$ is represented by $a \in \mathbb{Z}_{p^{e-1}}$.*

4.4 Delegation of Polynomial Evaluation in Optimal Cost

In this section, we present a novel, optimal circuit construction of an arbitrary polynomial for 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 *sub-linear* 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 [WJB+17, XZZ+19].

4.4.1 Overview of Our Circuit Construction

Our circuit construction is inspired by the Paterson-Stockmeyer algorithm [PS73] evaluating a polynomial $g(t)$ of degree N in $O(\sqrt{N})$ non-constant multiplications.[viii] Specifically, for a given polynomial $g(t) = \sum_{i=0}^{N} a_i t^i$, our circuit is constructed to first compute $\sqrt{N}$ sub-polynomials g_k's (for $1 \leqslant k \leqslant \sqrt{N}$) where $g_k(t) = \sum_{j=1}^{\sqrt{N}} a_{j+\sqrt{N}(k-1)} t^j$, 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 4.1) computes the polynomial as follows:

$$\begin{aligned} a_0 + &\big((a_1 t + \cdots + a_4 t^4) + (a_5 t + \cdots + a_8 t^4) \cdot t^4\big) \\ &+ \big((a_9 t + \cdots + a_{12} t^4) + (a_{13} t + \cdots + a_{16} t^4) \cdot t^4\big) \cdot t^8 \end{aligned}$$

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, that is, t, t^2, $\cdots$, $t^{\sqrt{N}}$, $t^{2\sqrt{N}}$, $t^{4\sqrt{N}}$, $t^{8\sqrt{N}}$, $\cdots$, $t^{N/2}$, are needed to compute $g(t)$ in the above evaluation method. Also,

[viii] For the simplicity of the presentation, let $N = 2^{2n}$ be the smallest power of four such that $N \geqslant \deg(g)$.

every sub-polynomial g_k is computed using the *same* small subset of powers of t, that is, t, t^2, $\cdots$, $t^{\sqrt{N}}$. 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, $\vec{x} \mapsto \sum a_i x_i$, at no cost overhead, by exploiting its nice evaluation structure, even if its fan-in is not constant (see Section 4.4.2 for more details). These observations give us more flexibility in constructing a circuit, and we utilize the linear-sum gate for the evaluation of 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 4.1 shows our circuit construction of a single polynomial $g(t)$. The circuit is composed of four parts. The first part referred to as *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 *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 *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 *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 *extract*, consisting of a single layer of a constant-addition gate, computes the final result $g(t)$. More details and a precise definition

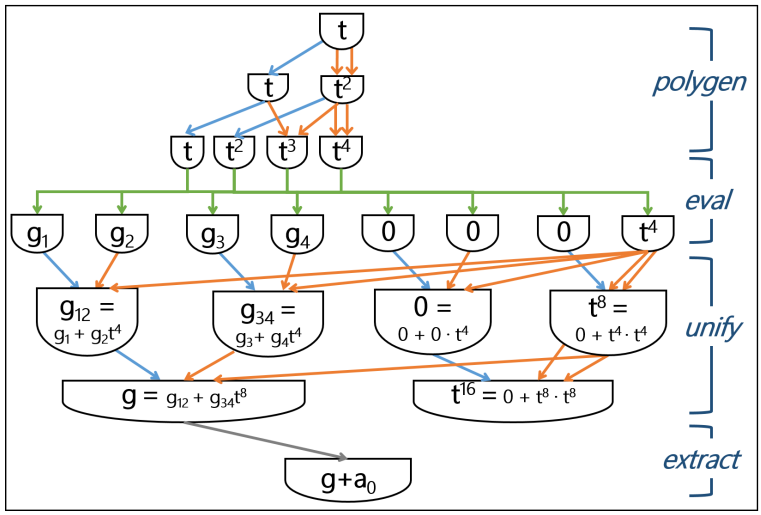


Figure 4.1: Our circuit construction of a polynomial of degree 16, $g(t) = \sum_{i=0}^{16} a_i t^i$. The value of each gate denotes the output of the gate, where $g_k = \sum_{j=1}^{4} a_{j+4(k-1)} t^j$. The green arrow denotes the linear-sum gate wiring. The gates computing zero are dummy gates that 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.

of our circuit construction are provided in Section 4.4.2.

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 4.1 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.

4.4.2 Our Circuit for Polynomial Evaluation

Notation. Assume we are given a polynomial g over a finite ring $\mathbb{Z}_{p^e}$. (Our representation is also valid with a polynomial over a finite field $\mathbb{F}$.) Let

us fix $N = 2^{2n}$ to denote the smallest power of four such that $N \geqslant \deg(g)$. Let us index each layer where the input layer is indexed by 0.[ix] 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 $\beta_s(x, y) : \mathbb{Z}_{p^e}^s \times \mathbb{Z}_{p^e}^s \to \mathbb{Z}_{p^e}$ to denote the MLE of $B_s(x, y) : \{0,1\}^s \times \{0,1\}^s \to \{0,1\}$ where $B_s(x, y)$ is the comparison function that returns 1 if $x = y$, and 0 otherwise. We write $\vec{1}_s = (1, 1, \ldots, 1) \in \{0,1\}^s$, and $\chi_s(x) := B_s(x, \vec{1}_s) : \{0,1\}^s \to \{0,1\}$. We omit s when it is obvious.

Description. Now we present the circuit representation for the polynomial $g(t) = \sum_{i=0}^{N} a_i t^i$. The circuit is composed of four parts, each of which is called *polygen*, *eval*, *unify*, and *extract*, respectively, as illustrated in Figure 4.1. 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 $g_{i,j}$, 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 4.1).

[ix] In the GKR protocol, the output layer is indexed by 0.

The *polygen* part corresponds to the sub-circuit between the layers 1 and n, where for each i^{th} layer, the input values are $\{t^j\}_{j=1}^{2^{i-1}}$, and the output values are $\{t^j\}_{j=1}^{2^i}$. Now we have the following relation between $\tilde{V}_{i+1}$ and $\tilde{V}_i$ (for $0 \leqslant i < n$) as follows.

$$\tilde{V}_{i+1}(z) = \tilde{V}_i(z_{-0})[(1 - z_0) + z_0 \tilde{V}_i(\vec{1})]$$

where $z = (z_0, z_1, \ldots z_i)$, $z_{-0} = (z_1, z_2, \ldots z_i)$, $\vec{1} = (1, 1, \ldots, 1)$, and $\tilde{V}_0 = t$. The validity of this equation is derived from the fact that both sides of the equation are MLEs in z agreeing on $\{0,1\}^{i+1}$, and the uniqueness of MLE (Lemma 2) that holds for an arbitrary ring (Section 4.2.2). 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 4.1.

The *eval* layer, i.e., the $(n+1)^{\text{th}}$ layer, produces $2\sqrt{N}$ output values which consists of $g_1(t), \cdots, g_{\sqrt{N}}(t)$, 0, 0, $\cdots$, 0, $t^{\sqrt{N}}$, from the input values $\{t^j\}_{j=1}^{\sqrt{N}}$. Each g_k (for $1 \leqslant k \leqslant n$) is a polynomial of degree at most $\sqrt{N}$, defined by $g_k(t) = \sum_{j=1}^{\sqrt{N}} a_{j+\sqrt{N}(k-1)} t^j$. The zeros are the outputs of dummy gates as explained earlier. Now we have the following relation between the two MLEs.

$$\tilde{V}_{n+1}(z) = \sum_{q \in \{0,1\}^n} \alpha(z,q) \cdot \tilde{V}_n(q) \text{ where } z = (z_0, z_1, \ldots, z_n),$$

$$\alpha(z,q) := \text{MLE of } \begin{cases} a_{[(z,q)]}, & \text{if } z_0 = 0 \\ \chi_{2n}, & \text{if } z_0 = 1 \end{cases}$$

where $[v]$ denotes the integer value represented by the binary vector v,

e.g., $[(1,1,0,1)] = 13$.

The *unify* part follows the *eval* layer, corresponding to a sub-circuit of depth $\log \sqrt{N}$ from the $(n+2)^{\text{th}}$ layer to the $(2n+1)^{\text{th}}$ layer, as shown in Figure 4.1. Each layer of the *unify* part takes as input, $g_1(t), \ldots, g_i(t), 0, \ldots, 0, t^j$, and produces $g_{1,2}(t), \ldots, g_{i-1,i}(t), 0, \ldots, 0, t^{2j}$, where $g_{k,k+1} = g_k + g_{k+1}t^j$. The final layer of the *unify* part will produce $(g(t) - a_0)$ and t^N. Now we have the following relation between two adjacent MLEs.

$$\tilde{V}_{i+1}(z) = \tilde{V}_i(z, 0) + \tilde{V}_i(z, 1) \cdot \tilde{V}_i(1, 1, \ldots, 1)$$

where $z = (z_0, z_1, \ldots, z_{2n-i})$.

Note that the above equation makes no distinction between the two sub-circuits, i.e., one that computes $g_{1,2}(t), \ldots, g_{i-1,i}(t)$ and another that computes $0, \ldots, t^{2j}$, which significantly reduces the prover's cost that otherwise would have been very large. This is achieved by introducing the dummy gates that compute zero, as explained earlier.

Finally, the *extract* layer, i.e., the $(2n+2)^{\text{th}}$ layer, takes two inputs $(g(t) - a_0)$ and t^N, and simply returns $g(t)$ by adding the constant a_0 to the first input. The relation is as follows:

$$\tilde{V}_{2n+2} = \tilde{V}_{2n+1}(0) + a_0$$

The multi-input case with $M = 2^m$ number of inputs follows naturally from single-input case described so far (see Figure 4.2).

4.4.3 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 proto-

- *polygen* layer, $\tilde{V}_i(w,z) : \mathbb{Z}_{p^e}^m \times \mathbb{Z}_{p^e}^i \to \mathbb{Z}_{p^e}$ $(0 \leqslant i \leqslant n-1)$,

$$\tilde{V}_{i+1}(w,z) = \sum_{q \in \{0,1\}^m} \beta(w,q)\tilde{V}_i(q, z_{-0})[(1-z_0) + z_0\tilde{V}_i(q, \vec{1}_i)]$$

  where $z = (z_0, z_1, \ldots z_i) \in \mathbb{Z}_{p^e}^{i+1}$, and $z_{-0} = (z_1, z_2, \ldots z_i) \in \mathbb{Z}_{p^e}^i$.

- *eval* layer, $\tilde{V}_{n+1}(w,z) : \mathbb{Z}_{p^e}^m \times \mathbb{Z}_{p^e}^{n+1} \to \mathbb{Z}_{p^e}$,

$$\tilde{V}_{n+1}(w,z) = \sum_{q \in \{0,1\}^n} \alpha(z,q) \cdot \tilde{V}_n(w,q)$$

- *unify* layer, $\tilde{V}_{j+1}(w,z) : \mathbb{Z}_{p^e}^m \times \mathbb{Z}_{p^e}^{2n+1-j} \to \mathbb{Z}_{p^e}$ $(n+1 \leqslant j \leqslant 2n)$,

$$\tilde{V}_{j+1}(w,z) = \sum_{q \in \{0,1\}^m} \beta(w,q)[\tilde{V}_j(q,z,0) + \tilde{V}_j(q,z,1) \cdot \tilde{V}_j(q, \vec{1}_{2n+2-j})]$$

- *extract* layer, $\tilde{V}_{2n+2}(w) : \mathbb{Z}_{p^e}^m \to \mathbb{Z}_{p^e}$,

$$\tilde{V}_{2n+2}(w) = \tilde{V}_{2n+1}(w,0) + a_0$$

Figure 4.2: 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 $N = 2^{2n} \geqslant \deg(g)$. We write $\vec{1}_k = (1, \ldots, 1) \in \mathbb{Z}_{p^e}^k$, and $\alpha(z,q) : \mathbb{Z}_{p^e}^{n+1} \times \mathbb{Z}_{p^e}^n \to \mathbb{Z}_{p^e}$ to denote the MLE of a boolean hypercube function $A(x) : \{0,1\}^{2n+1} \to \mathbb{Z}_{p^e}$ that represents $(a_1, \ldots, a_N, 0, \ldots, 0, 1) \in \mathbb{Z}_{p^e}^{2N}$. For example, $A(0,0,\ldots,0) = a_1$, $A(0,1,\ldots,1) = a_N$, and $A(1,1,\ldots,1) = 1$. We also write $\beta(w,p) : \mathbb{Z}_{p^e}^m \times \mathbb{Z}_{p^e}^m \to \mathbb{Z}_{p^e}$ to denote the MLE of a comparison function $B(x,y) : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ where $B(x,y)$ returns 1 if $x = y$, and 0 otherwise.

col (precisely, the variants [Tha13a] or [XZZ+19, 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.

Prover's cost. The circuit representation is composed of four parts; *polygen*, *eval*, *unify*, *extract*, and *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.4.1). The i-th layer in *polygen* requires 2^{i-1} multiplications resulting in $O(2^n)$ total for *polygen* part. The *eval* layer requires $O(2^n \cdot 2^{n+1}) = O(2^{2n})$ operations,

since evaluating each $f_i(t)$ given $\{t^j\}_{j=1}^{2^n}$ requires $O(2^n)$ operations. The j-th layer in *unify* requires 2^{2n+3-j} operations resulting in $O(2^n)$ total for *unify* part. Since *extract* and *division* part is of negligible cost, the total cost for evaluation is $O(2^n + 2^{2n} + 2^n) = O(N)$, 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 [Tha13b]'s Reusing Work reducing $\mathcal{P}$'s cost for evaluating all $\beta_m(w,p)$, $\tilde{V}(q)$, and $\alpha(r)$ values required for sum-check to be only $O(2^m)$, $O(2^s)$, and $O(2^t)$ respectively, where m, s, and t are the number of variables constituting p, q, and r, respectively[x]. 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 4.2).

In *polygen* part, reducing from $\tilde{V}_{i+1}$ to $\tilde{V}_i$ requires $O(2^m + 2^{m+i})$ cost for sum-check, and additional $O(i \cdot 2^i)$ cost for reducing to single point, resulting in total $O(2^{m+n} + n \cdot 2^n)$ cost. In *eval* layer, sum-check requires $O(2^{2n+1} + 2^{m+n})$ cost. In *unify* part, reducing from $\tilde{V}_{j+1}$ to $\tilde{V}_j$ requires $O(2^m + 2^{m+2n+2-j})$ cost for sum-check, and additional $O((2n+2-j) \cdot 2^{2n+2-j})$ cost for reducing to single point[xi], resulting in total $O(2^{m+n}+n\cdot 2^n)$ cost. The *extract* and *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 $O(\sqrt{N}M + N)$.

[x]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, $C^{(j)}[(p_{j+1},\ldots,p_{s_i})]$ in [Tha13b, equation (7)] (Full ver. of [Tha13a]) can be calculated by $r_j C^{(j-1)}[1,p_{j+1},\ldots,p_{s_i}]+(1-r_j)C^{(j-1)}[0,p_{j+1},\ldots,p_{s_i}]$ without z_j^{-1}.

[xi]In fact, we perform two consecutive processes of reducing to single point, $\tilde{V}_j(p,z,1)$ & $\tilde{V}_j(p,z,0)$ to $\tilde{V}_j(p,z,r_1)$, then $\tilde{V}_j(p,z,r_1)$ & $\tilde{V}_j(p,1,1,\ldots,1)$ to $\tilde{V}_j(p,r)$.

Verifier's cost. Note that $\alpha(z, q)$ can be precomputed in cost $O(N)$, using memoization [VSBW13], and will not be considered in the following estimation. Each β_m can be evaluated in cost $O(m)$ due to its simple form [Tha13b, Section 4.3.1], without affecting the asymptotic cost of $\mathcal{V}$. Also, as the original GKR 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 1). Recall that in sum-check, the cost of $\mathcal{V}$ depends on the number of variables managed by summation. In *polygen* layers, reducing from $\tilde{V}_{i+1}$ to $\tilde{V}_i$ requires $\mathcal{V}$ to perform $O(m)$ operations for sum-check, and $O(i)$ for reducing to single point. Therefore, the cost for *polygen* layers is $O(mn + n^2)$. In *eval* layer, $O(n)$ cost is required. In *unify* layers, reducing from $\tilde{V}_{j+1}$ to $\tilde{V}_j$ requires $\mathcal{V}$ to perform $O(m)$ 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 *extract* and *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. □

Remark 4.4.1. *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 (i.e., computing all the powers of t using the exponentiation-by-squaring method, computing all the monomials, and adding all the monomials in a binary tree fashion). Then, the proof generation cost of Giraffe [WJB+17] and Libra [XZZ+19] on C' are $O(MN+N \log N)$ and $O(MN)$, respectively, while ours is $O(M\sqrt{N} + N)$. Their other costs (i.e., circuit evaluation,*

verification, and communication) on C' are the same with ours.

4.5 Cost Optimization

In this section, we present an optimization technique that can significantly reduce the prover's cost for the rounding layers described in Section 4.3.

4.5.1 Galois Ring over $\mathbb{Z}_{p^e}$ and a Sampling Set

A Galois ring $\mathbb{Z}_{p^e}[t]/(f(t))$ over $\mathbb{Z}_{p^e}$ for a monic irreducible polynomial $f(t) \in \mathbb{Z}_p[t]$ is a natural generalization of the Galois field $\mathsf{GF}(p^n)$ 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 $GF(p^n)$ 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 $\mathbb{Z}_{p^e}^d$ whose standard basis corresponds to $1, t, t^2, \ldots, t^{d-1}$. Thus, the addition corresponds to the component-wise addition in $\mathbb{Z}_{p^e}^d$, 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 4. *[McD74] 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 $\{a_0 + a_1 t + \ldots + a_{d-1}t^{d-1} \mid a_i \in [-(p-1), p-1] \cap \mathbb{Z}\} \subseteq R$, are invertible (hence are not zero-divisors) in R. Therefore, the subset $A =$*

$\{a_0 + a_1 t + \ldots + a_{d-1} t^{d-1} \mid a_i \in [0, p-1] \cap \mathbb{Z}\} \subseteq R$ *is a valid* sampling set *for the generalized Schwartz-Zippel lemma (Lemma 3) as well as the generalized GKR protocol (Theorem 3).*

Note that the cardinality of the sampling set A in Theorem 4 is $p^d \gg p$, which is maximal.[xii]. Moreover, A satisfies the additional condition of Remark 4.2.1.

Irreducible Polynomial in $\mathbb{Z}_p[t]$. To construct a Galois ring $\mathbb{Z}_{p^e}[t]/(f(t))$, we need an irreducible polynomial in $\mathbb{Z}_p[t]$. Indeed, there exist many irreducible polynomials $f(t) \in \mathbb{Z}_p[t]$ for any degree d, but a sparse polynomial (where most of its coefficients are zero) is desired for the efficiency of multiplication in $\mathbb{Z}_{p^e}[t]/(f(t))$. Below we provide examples of such a sparse irreducible polynomial. (More irreducible polynomials can be systemically found using Lemma 7 in the following proof.)

Lemma 6. *Let p be a prime number. All of the following polynomials are irreducible in $\mathbb{Z}_p$:*

i. $\Phi_4(x) = x^2 + 1$ *when* $p \equiv 3 \mod 4$.

ii. $\Phi_5(x) = x^4 + x^3 + x^2 + x + 1$ *when* $p \equiv \pm 2 \mod 5$.

iii. $\Phi_9(x) = x^6 + x^3 + 1$ *when* $p \equiv 2$ *or* $5 \mod 9$.

iv. $x^3 - a$ *for some* a *when* $p \equiv 1 \mod 3$.

v. $x^4 - 2$ *when* $p \equiv 5 \mod 8$.

[xii] 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.

vi. $x^4 - 3$ *when* $p \equiv 5 \mod 12$.

Proof. We exploit following Lemma whose proof can be found in [Mil18].

Lemma 7. *[Mil18, Lemma 5.9] An n-th cyclotomic polynomial* Φ_n *of degree* $\varphi(n)$ *is irreducible if and only if p is a primitive root modulo n (i.e., p does not divide n), and its multiplicative order modulo n is* $\varphi(n)$*, where* φ *is the Euler's totient function.*

(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 [Gar07].

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 $\mathbb{Z}_p$ which is equivalent to the claim that the function $t \to t^3 : \mathbb{Z}_p \to \mathbb{Z}_p$ 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 [DLW05, 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. □

4.5.2 Optimization of Prover's Cost for Rounding 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 $p \simeq q^d$. First, we convert C_p to an approximately equivalent circuit C_q over $\mathbb{Z}_{q^{de}}$, by the base-p-to-base-q conversion, where each base-p rounding gate ($x \mapsto \lfloor x/p \rceil$) in C_p is replaced with d-consecutive base-q rounding gates ($x \mapsto \lfloor x/q \rceil$) in C_q. Then, we apply the generalized GKR protocol over a Galois ring $\mathbb{Z}_{q^{de}}[t]/(f(t))$ where $f(t)$ is a monic irreducible polynomial of degree d. Here, we employ the sampling set given in Theorem 4, 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 $\mathbb{Z}_{q^{de}}$, and the proof generation and the verification to be conducted over $\mathbb{Z}_{q^{de}}[t]/(f(t))$. This is valid, since $\mathbb{Z}_{q^{de}}[t]/(f(t))$ naturally embeds $\mathbb{Z}_{q^{de}}$ 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 $deq \simeq de\sqrt[d]{p}$, which is much smaller than ep for some d. On the other hand, the cost of the individual addition (and multiplication) operation in $\mathbb{Z}_{q^{de}}[t]/(f(t))$ 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 unoptimized 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.	$O(epr)$	$2d^2e\sqrt[d]{p}r$
Proof gen.	$O(epr)$	$32d^4e\sqrt[d]{p} + 70d^3\sqrt{de\sqrt[d]{p}r}$
Verification	$O(\log^2 epr)$	$d^3(\log de\sqrt[d]{p})(\log de\sqrt[d]{p}r^{10})$
Soundness	$O(\frac{\log^2 epr}{p})$	$d(\log de\sqrt[d]{p})(\log de\sqrt[d]{p}r^6)/4p$

Here the optimization problem is to find d such that the costs for C_q are minimized. In particular, given p, the term $d^4\sqrt[d]{p}$ is minimized to $((\mathbf{e}\ln p)/4)^4$, which is much smaller than p, when $d = (\ln p)/4$, where $\mathbf{e}$ is Euler's number. In Section 5.2.1, we will present an experimental result where two orders of magnitude cost reduction was made by finding a proper d.

Chapter 5

Experimental Results

We present experimental results that quantify the efficiency of our protocol. Specifically, we conducted experiments that show how efficiently our protocol support rounding, and how effective the optimization technique is. Also, to show the importance of rounding, we compare our protocol (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 [War].

5.1 Experimental Setup

We implemented our generalized GKR protocol[i] 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 oper-

[i]Specifically, the generalization was made on top of Thaler's variant [Tha13a], since we considered Thaler's variant to compare ours to the original GKR protocol as explained in Section 5.3.

ations of $\mathbb{Z}_{p^e}$ are implemented using the Montgomery modular multiplication [Mon85]. The code is written in C++11 using the GMP library, and compiled with the LLVM GCC compiler 9.1.0 (with -O3). 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.

5.2 Verifiable Rounding Operation

We first presents the performance of verifiable rounding operations in our protocol.

5.2.1 Effectiveness of Optimization via Galois Ring

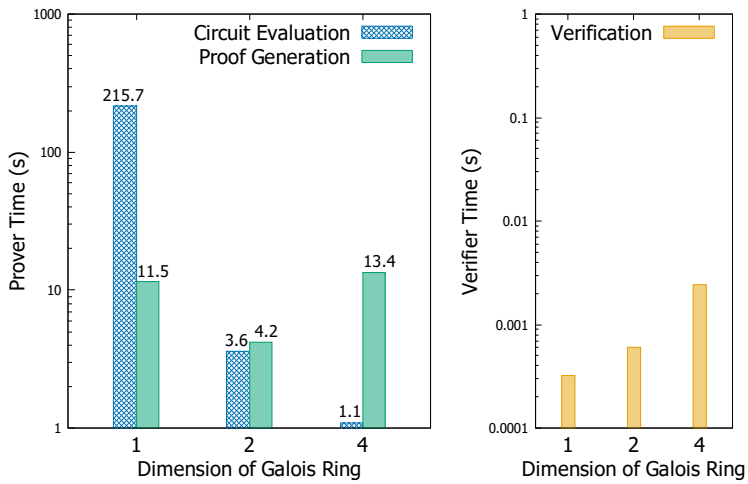
To show the effectiveness of the optimization technique described in Section 4.5.2, we instantiated our scheme with different Galois rings and compared their performance. Specifically, given an original ring, $R_1 = \mathbb{Z}_{(65537)^7}$, we took two Galois rings, $R_2 = \mathbb{Z}_{(271)^{14}}[t]/(t^2+1)$ and $R_3 = \mathbb{Z}_{(17)^{28}}[t]/(t^4-3)$, where $|R_1| \simeq |R_2| \simeq |R_3| \simeq 2^{112}$. Then, we instantiated our optimized protocol (Section 4.5.2) 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.[ii]

Figure 5.1 shows the performance of the protocol over the different

[ii]More precisely, each rounding gate takes as input x, and outputs $\lfloor x/65537 \rceil$, $\lfloor x/(271^2) \rceil$, and $\lfloor x/(17^4) \rceil$, respectively, for each R_1, R_2, and R_3.

	p	$\log p$	e	$f(t)$	d	λ_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

(a) Galois ring structures $\mathbb{Z}_{p^e}[t]/(f(t))$



(b) Performance of protocol over different Galois rings

Figure 5.1: Performance of our protocol over different Galois rings $\mathbb{Z}_{p^e}[t]/(f(t))$, for a rounding layer consisting of 2^{14} gates. The table describes three different rings $R_1 = \mathbb{Z}_{(65537)^7}$, $R_2 = \mathbb{Z}_{(271)^{14}}[t]/(x^2+1)$ and $R_3 = \mathbb{Z}_{(17)^{28}}[t]/(x^4-3)$, where d denotes the dimension of a Galois ring, and λ_s denotes the soundness probability bound of the protocol over the ring. Each rounding gate performs $x \mapsto \lfloor x/(p^d) \rfloor$, i.e., roughly the 16-bit rounding.

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 4.5.2. 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 4.4.3.

5.2.2 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 will consider our generalized GKR protocol over $R_2 = \mathbb{Z}_{(271)^{14}}[t]/(t^2+1)$, and the rounding operation $x \mapsto \lfloor x/(271^2) \rfloor$, roughly the 16-bit rounding. Figure 5.2 shows the performance of our protocol for a rounding layer of various sizes, from 2^8 to 2^{19}. As described in Section 4.4.3, the cost of circuit evaluation and proof generation is linear in the number of rounding gates, while the cost of verification and communi-

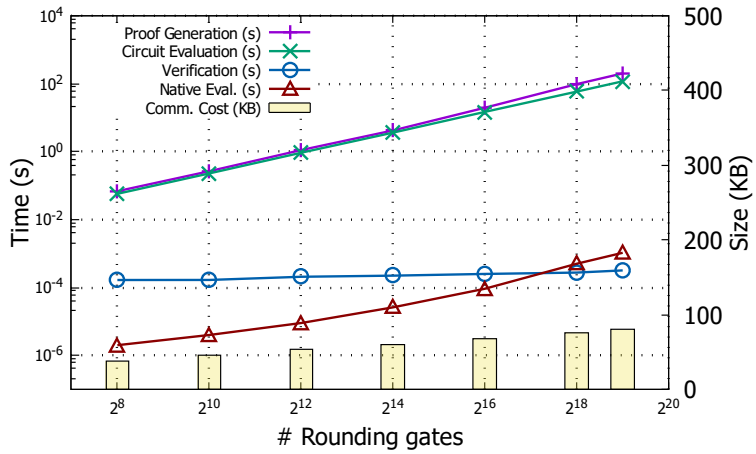


Figure 5.2: Performance of our protocol for a rounding layer of various sizes. The protocol is over $R_2 = \mathbb{Z}_{(271)^{14}}[t]/(x^2+1)$, and the rounding operation is $x \mapsto \lfloor x/(271^2) \rceil$, roughly the 16-bit rounding.

cation 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}.

5.3 Comparison to Thaler's Refinement of GKR Protocol

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 [Tha13b]'s implementation for the original GKR protocol since it shows the best performance for matrix multiplication among other variants (e.g., [WJB+17, XZZ+19]). To be a fair comparison, we modified the Thaler's implementation to employ the same GMP library we used in our protocol implementation.[iii]

Moreover, we consider a nested multiplication of depth n, $(\cdots(M^2)^2\cdots)^2 = M^{2^n}$, where M is a 128×128 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.[iv]

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., $(\cdots(((2^{16}M)^2)^2)^2\cdots)^2 = (2^{16})^{2^n}M^{2^n}$. 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 our protocol (over a ring $\mathbb{Z}_{p^e}$), however, the nested multiplication is represented as the integer-scaled nested multiplication *with rounding*, i.e., $\lfloor(\cdots\lfloor(\lfloor(\lfloor(2^{16}M)^2\rceil)^2\rceil)^2\rceil\cdots)^2\rceil \simeq 2^{16}M^{2^n}$, where $\lfloor\cdot\rceil$ denotes $x \mapsto \lfloor x/(2^{16})\rfloor$. Thus p^e can be only larger than $2^{16}\cdot 2^{16n}$ (the additional term 2^{16n} is due to the modulus change by rounding as described in Remark 4.3.3). That is, the bit-size of ring elements (in $\mathbb{Z}_{p^e}$) is *linear* in the multiplication depth.

[iii] 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.

[iv] 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.

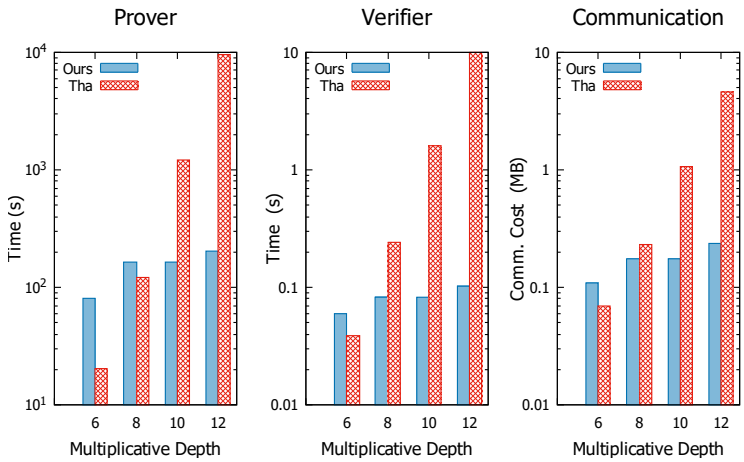


Figure 5.3: Performance comparison of ours to Thaler [Tha13a]'s on a single 128×128 matrix multiplication (over fixed-point numbers with 16 fractional bits) in the context of different multiplication depths. The domain of each protocol is chosen to be large enough to admit a given multiplication depth. That is, our protocol is over $\mathbb{Z}_{(271)^e}[t]/(t^3+2)$, where $e = 14$, 18, 22, and 26, respectively. Thaler's is over $\mathbb{Z}_q$, where $q = (2^{1279}-1)$, $(2^{4253}-1)$, $(2^{19937}-1)$, and $(2^{86243}-1)$, respectively. The performance of Thaler's on the multiplicative depth 12 is extrapolated.

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., $(2^{1279}-1)$, $(2^{4253}-1)$, $(2^{19937}-1)$, and $(2^{86243}-1)$, respectively, while for our protocol over $\mathbb{Z}_{p^e}[t]/(t^3+2)$, 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}, and 271^{26}, respectively, for each multiplication depth $n = 6$, 8, 10, and 12.

In Figure 5.3, 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. That is, the cost for the entire nested multiplication is the one in Figure 5.3 multiplied by the number of matrix multiplications.

Figure 5.3 shows that the cost of Thaler's *exponentially* increases in the multiplication depth, while ours is *linear* in the depth. When the multiplication depth is small (e.g., depth 6), the cost of our protocol could be bigger than Thaler's, due to the overhead of rounding. However, when the multiplication depth is greater than a certain amount (e.g., depth 8), ours is much better than Thaler's (e.g., two orders of magnitude better when depth is 12), and the difference will be exponential as the depth increases. This experimental result confirms that it is critical to support the rounding operation for verifiable computing of an approximate arithmetic circuit with a large multiplication depth.

5.4 Discussion

We want to note that there is still room for improvement of our implementation, since in this work, we have mainly focused on the proof-of-concept evaluation of our approach. In particular, the implementation of the individual operations of a Galois ring can be further improved. While those operations are sequentially executed in our current implementation, they can be easily broken down into multiple independent subroutines, being suitable for parallelization [CT65, FPV+09] or hardware acceleration. This optimization 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.

On the other hand, the soundness probability of our protocol in Figure 5.3 is set to 2^{-14}, which is not high, but sufficient in certain contexts. Moreover, it can be quickly improved by simply running n parallel pairs of the prover and the verifier, which yields $(2^{-14})^n$ soundness, without affecting the throughput performance. For example, running only four prover-verifier pairs in parallel will achieve $2^{-56} < 10^{-16}$ soundness,[v] which is similar to the soundness probability (2^{-45} to 2^{-20} [Tha13b, SVP+12]) of existing verifiable computing scheme experiments.[vi]

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 such as matrix multiplication. Also, our method can be applied with (asymptotic) cost reduction technique derived from higly regular wiring pattern, e.g. Thaler [Tha13a]'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 [ZGK+17, WTS+18] 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 (asymptot-

[v] For comparison, 10^{-16} to 10^{-13} is the uncorrectable bit error rate of a typical hard disk [GvI07].

[vi] Pinocchio [PHGR13] offers roughly 2^{-128} soundness, but it is based on strong cryptographic assumptions.

ically) 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 [ZGK+17] on commitment implies that proving rounding gate with their commitment would require at least $0.11\times$ rounding bits (ms) per gate[vii], resulting in $2^{14} \times 16 \times 0.11$ (ms) $\simeq 29$ (sec) for 2^{14} gates, which is about 7 times costly than our prover (4 sec) in Figure 5.2. On the other hand, the commitment in [WTS+18] [viii] is asymptotically costly in verifier cost than that of ours.

[vii] On Amazon EC2 c4.8xlarge running Linux Ubuntu 14.04, with 60GB of RAM, Intel Xeon E5-2666v3 CPUs with 36 virtual cores at 2.9 GHz.

[viii] It requires verifier's cost to be $O(\sqrt{\#\text{ of gates} \times \text{rounding bits}})$.

Chapter 6

Conclusions

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 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. We end this section introducing our vision and future plan of research from this work.

6.1 Towards 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 Chapter 5, 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 [VSBW13, ZGK+17]. The sigmoid and tanh functions were shown to be effectively approximated as a polynomial [HTG17] 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 $x \leqslant 0$, using the input normalization [Vie].

Moreover, multiple iterations can be "squashed" [WJB+17] 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 [Tha13b, WJB+17] at the cost of communication overheads. Finally, the protocol performance can be further improved by using hardware accelerators such

as GPUs [TRMP12, Tha13b] and ASICs [WHG+16, WJB+17].

6.2 Verifiable Cryptographic Computation

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 $\mathbb{Z}_{p^e}$ has a nice property that addition and multiplication on $\mathbb{Z}_{p^e}$ 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 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.[i] 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., $x \mapsto x \bmod 2^e$, 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.

[i] In this case, the optimization via a Galois ring (Section 4.5.1) 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 $\prod_i \mathbb{Z}_{p_i^{e_i}}$ 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 [Mic11].

Bibliography

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 2087–2104, 2017.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *J. ACM*, 45(1):70–122, 1998.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334, 2018.

[BCPS18] Anurag Bishnoi, Pete L Clark, Aditya Potukuchi, and John R Schmitt. On zeros of a polynomial in a finite grid. *Combinatorics, Probability and Computing*, 27(3):310–333, 2018.

[BFLS91] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32. ACM, 1991.

[BFR+13] Benjamin Braun, Ariel J Feldman, Zuocheng Ren, Srinath Setty, Andrew J Blumberg, and Michael Walfish. Verifying computations with state. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 341–357. ACM, 2013.

[BH17] Balázs Bulyovszky and Gábor Horváth. Polynomial functions over finite commutative rings. *Theoretical Computer Science*, 703:76–86, 2017.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013*, pages 90–108. Springer, 2013.

[BSCGT13a] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from rams to delegatable succinct constraint satisfaction problems. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 401–414. ACM, 2013.

[BSCGT13b] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 585–594, 2013.

[BSCR+18] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for r1cs. Cryptology ePrint Archive, Report 2018/828, 2018.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium*, pages 781–796, 2014.

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[Car64] Leonard Carlitz. Functions and polynomials (mod pn). *Acta arithmetica*, 9(1):67–78, 1964.

[CCKP19] Shuo Chen, Jung Hee Cheon, Dongwoo Kim, and Daejun Park. Verifiable computing for approximate computation. 2019.

[CCM09] Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. Annotations in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 222–234. Springer, 2009.

[CFS17] Alessandro Chiesa, Michael A Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. *arXiv preprint arXiv:1704.02086*, 2017.

[CH18] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In *Advances in Cryptology – EUROCRYPT 2018*, pages 315–337, 2018.

[CLS12] Xiaofeng Chen, Jin Li, and Willy Susilo. Efficient fair conditional payments for outsourcing computations. *IEEE Transactions on Information Forensics and Security*, 7(6):1687–1694, 2012.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

[CRR11] Ran Canetti, Ben Riva, and Guy N Rothblum. Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 445–454. ACM, 2011.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment*, 5(1):25–36, 2011.

[DLW05] Eric Driver, Philip A Leonard, and Kenneth S Williams. Irreducible quartic polynomials with factorizations modulo p. *The American Mathematical Monthly*, 112(10):876–890, 2005.

[FPV+09] Franz Franchetti, Markus Puschel, Yevgen Voronenko, Srinivas Chellappa, and José MF Moura. Discrete fourier transform on multicore. *IEEE Signal Processing Magazine*, 26(6):90–102, 2009.

[Gar07] Paul Garret. Abstract algebra. `http://www-users.math.umn.edu/~garrett/m/algebra/notes/Whole.pdf`, 2007. pp. 100–102.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Annual Cryptology Conference*, pages 465–482. Springer, 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Annual International Conference on the Theory*

and Applications of Cryptographic Techniques, pages 626–645. Springer, 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 113–122. ACM, 2008.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*, pages 305–326, 2016.

[GvI07] Jim Gray and Catharine van Ingen. Empirical measurements of disk failure rates and error rates. *CoRR*, abs/cs/0701166, 2007.

[HTG17] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *CoRR*, abs/1711.05189, 2017.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*, pages 278–291. IEEE, 2007.

[JPSZ06] Jian Jun Jiang, Guo Hua Peng, Qi Sun, and Qi Fan Zhang. On polynomial functions over finite commutative rings. *Acta Mathematica Sinica*, 22(4):1047–1050, 2006.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, 1992.

[KPP+14] Ahmed E. Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, Mahmoud F. Sayed, Elaine Shi, and Nikos Triandopoulos. Trueset: Faster verifiable set computations. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 765–780, Berkeley, CA, USA, 2014. USENIX Association.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.

[McD74] Bernard R McDonald. *Finite rings with identity*, volume 28. Marcel Dekker Incorporated, 1974.

[Mic94] Silvio Micali. Cs proofs. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 436–453. IEEE, 1994.

[Mic11] Daniele Micciancio. Lattice-based cryptography. *Encyclopedia of Cryptography and Security*, pages 713–715, 2011.

[Mil18] James S Milne. Fields and galois theory (v4. 60). *order*, 3:138, 2018.

[Mon85] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.

[PS73] Michael S Paterson and Larry J Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2(1):60–66, 1973.

[Rot09] Guy N Rothblum. *Delegating computation reliably: paradigms and constructions*. PhD thesis, Massachusetts Institute of Technology, 2009.

[SAGL18] Srinath Setty, Sebastian Angel, Trinabh Gupta, and Jonathan Lee. Proving the correct execution of concurrent services in zero-knowledge. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, pages 339–356, Berkeley, CA, USA, 2018. USENIX Association.

[SBV+13] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J Blumberg, Bryan Parno, and Michael Walfish. Resolving the con-

flict between generality and plausibility in verified computation. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 71–84. ACM, 2013.

[Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980.

[Set19] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. Cryptology ePrint Archive, Report 2019/550, 2019.

[SMBW12] Srinath TV Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *NDSS*, volume 1, page 17, 2012.

[SSW10] Ahmad-Reza Sadeghi, Thomas Schneider, and Marcel Winandy. Token-based cloud computing. In *International Conference on Trust and Trustworthy Computing*, pages 417–429. Springer, 2010.

[SVP+12] Srinath TV Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security Symposium*, pages 253–268, 2012.

[Tha13a] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology–CRYPTO 2013*, pages 71–89. Springer, 2013.

[Tha13b] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. *arXiv preprint arXiv:1304.3812*, 2013.

[Tha15] Justin Thaler. A note on the gkr protocol, 2015.

[TRMP12] Justin Thaler, Mike Roberts, Michael Mitzenmacher, and Hanspeter Pfister. Verifiable computation with massively parallel interactive proofs. *arXiv preprint arXiv:1202.1350*, 2012.

[Vie] Tim Vieira. Exp-normalize trick. `https://timvieira.github.io/blog/post/2014/02/11/exp-normalize-trick/`.

[VSBW13] Victor Vu, Srinath Setty, Andrew J Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 223–237. IEEE, 2013.

[War] Pete Warden. Why GEMM is at the heart of deep learning. `https://petewarden.com/2015/04/20/`.

[WHG+16] Riad S Wahby, Max Howald, Siddharth Garg, Abhi Shelat, and Michael Walfish. Verifiable asics. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 759–778. IEEE, 2016.

[WJB+17] Riad S Wahby, Ye Ji, Andrew J Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *Proceedings of the*

2017 ACM SIGSAC Conference on Computer and Communications Security, pages 2071–2086. ACM, 2017.

[WSR+15] Riad S Wahby, Srinath TV Setty, Zuocheng Ren, Andrew J Blumberg, and Michael Walfish. Efficient ram and control flow in verifiable outsourced computation. In *NDSS*, 2015.

[WTS+18] Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 926–943. IEEE, 2018.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. Cryptology ePrint Archive, Report 2019/317, 2019.

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vsql: Verifying arbitrary sql queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 863–880. IEEE, 2017.

국문초록

계산검증 기술은 계산의 무결성을 확보하기 위한 계산 복잡도 이론적 방법이다. 최근 많은 계산이 클라우드 플랫폼과 같은 제3자에게 외주됨에 따라 그 필요성이 증가하고 있다. 그러나 기존의 계산검증 기술은 *비근사* 연산만을 고려했을 뿐, 근사 연산 (부동 소수점 또는 고정 소수점 연산)은 고려하지 않았다. 따라서 본질적으로 근사 연산이 필요한 특정 유형의 계산 (기계 학습, 데이터 분석 및 과학 계산 등)에 적용하기 어렵다는 문제가 있었다.

이 논문은 반올림 게이트를 수반하는 산술 회로를 위한 효율적인 대화형 증명 시스템을 제시한다. 이러한 산술 회로는 근사 연산을 효율적으로 표현할 수 있으므로, 근사 연산에 대한 효율적인 계산 검증이 가능하다. 주요 아이디어는 반올림 게이트를 작은 회로로 변환한 후, 여기에 Goldwasser, Kalai, 및 Rothblum의 프로토콜 (GKR 프로토콜)과 최근의 개선을 적용하는 것이다. 구체적으로, 대수적 객체를 유한체가 아닌 "숫자"를 보다 잘 처리할 수 있는 *환*으로 치환한 후, 환 위에서 적용 가능하도록 기존의 GKR 프로토콜을 일반화하였다. 이후, 반올림 연산을 환에서 차수가 낮은 다항식으로 표현하고, 다항식 연산을 최적의 회로 표현으로 나타내는 새롭고 최적화된 회로 구성을 개발하였다. 또한, 갈루아 환을 사용하여 반올림을 위한 증명 생성 비용을 더욱 최적화하였다. 마지막으로, 실험을 통해 우리의 근사 연산 검증 시스템의 효율성을 확인하였다. 예를 들어, 우리의 시스템은 구현 시, 16 비트 고정 소수점 연산을 통한 깊이 12의 반복된 128×128 행렬 곱셈의 검증에 있어 기존 시스템보다 약 100배 더 나은 성능을 보인다.

주요어휘: 계산 검증, 근사 연산
학번: 2013-20228