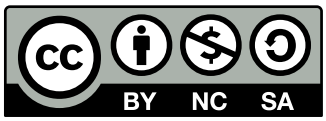
Aalto University
School of Electrical
Engineering

Master's Programme in Master's Programme in Security and Cloud Computing

Onigoroshi: Polynomial Interactive Oracle Proofs for Circuit Satisfiability over Cyclotomic Rings with Automorphism Gates

Shuto Kuriyama

Master's Thesis
2024



Aalto University
School of Electrical
Engineering

Author Shuto Kuriyama

Title Onigoroshi: Polynomial Interactive Oracle Proofs for Circuit Satisfiability over Cyclotomic Rings with Automorphism Gates

Degree programme Master's Programme in Security and Cloud Computing

Major Security and Cloud Computing

Supervisor Prof. Russell W.F. Lai & Prof. Helger Lipmaa

Advisor Prof. Russell W.F. Lai

Date 13 August 2024 **Number of pages** 61+1 **Language** English

Abstract

Lattice-based cryptography is a leading candidate for quantum-secure cryptography, and it provides a wide range of advanced cryptographic primitives. Constructing advanced cryptographic primitives requires proving relations involving cryptographic building blocks. Succinct non-interactive arguments of knowledge (SNARKs) enable the efficient non-interactive verification of such relations with short proofs. The construction of SNARKs can be accomplished by compiling an informatic theoretical object known as a polynomial interactive oracle proof (PIOP) with a compatible polynomial commitment scheme.

For efficient constructions of cryptographic primitives, it is important that the cryptographic building blocks and SNARKs operate over the same mathematical structure. However, most existing SNARKs cannot be directly applied to lattice-based cryptographic building blocks because most SNARKs operate over fields, whereas many lattice-based cryptographic primitives work over cyclotomic rings. Some lattice-based cryptographic primitives involve specific non-arithmetic operations such as ring-automorphism operations (e.g., bootstrapping in Fully Homomorphic Encryption).

We present Onigoroshi: the first polynomial interactive oracle proof over cyclotomic rings that can prove relations involving the ring-addition, ring-multiplication, and ring-automorphism operations. To construct the PIOP, we adapted HyperPlonk into the cyclotomic ring setting. We introduced Automorphism-Check, a new PIOP capable of verifying the consistency between the input and output of an automorphism over a cyclotomic ring.

Keywords SNARK, PIOP, Lattice-based Cryptography, Post Quantum

Preface

I want to thank my supervisor, Prof. Russell W.F Lai, for proposing an exciting research topic and ideas and guiding me throughout the research period. I also want to thank my co-supervisor, Prof. Helger Lipmaa, for helping me familiarise myself with interactive oracle proofs, including (Hyper)Plonk.

I also want to thank Dr. Michael Klooß for having discussions and teaching me mathematical concepts.

Otaniemi, 13 August 2024

Shuto Kuriyama

Contents

Symbols and abbreviations

Symbols

$\xrightarrow{\sim}$	isomorphism map

Abbreviations

IOP	interactive oracle proof
PIOP	polynomial interactive oracle proof
SNARK	succinct non-interactive argument of knowledge

Parameters

ϕ	$\phi = \varphi(z)$ in CRT transformation or a homormophism map.
e	$e = \mathsf{f}/z$ in CRT transformation.
q	prime number

1 Introduction

Constructions of advanced cryptographic primitives often require proving relations involving their cryptographic building blocks. To obtain efficient constructions, it is important that the building blocks and the proof system operate over the same mathematical structure to avoid conversion overheads.

Lattice-based cryptography, one of the top candidates for quantum-secure cryptography, often operates over cyclotomic rings. Motivated by this, we construct the first polynomial interactive oracle proofs (PIOP) for the satisfiability of algebraic circuits over cyclotomic rings involving addition, multiplication, and automorphism gates.

Lattice-based Cryptography. Classic public-key cryptosystems are based on the hardness of the integer factorisation problem or discrete logarithm problem. However, these problems were proved to be solvable in polynomial time using a quantum computer by Peter Shor in 1994 [Sho97]. Motivated by recent advancements in building quantum computers (e.g. [AAB+19, IBM23, IBM21]), cryptography that is secure against both classic and quantum computers have been proposed. One of the leading candidates is lattice-based cryptography.

Lattices began to be used constructively in cryptography since the seminal works of Ajtai [Ajt96] and Regev [Reg05], who respectively introduced the Short Integer Solution (SIS) and Learning with Errors (LWE) and proved their average-case hardness based on the worst-case hardness of lattice problems. To obtain more efficient constructions, subsequent works (e.g. [Mic02, LPR13, LS15]) considered the SIS and LWE problems over algebraic lattices which arise from number fields. Among the most popular are lattices which are constructed from the ring of integers of cyclotomic fields, also known as cyclotomic rings. Apart from ring addition and multiplication, some lattice-based cryptographic constructions over cyclotomic rings utilise automorphisms. Fully homomorphic encryption (FHE) is a notable example [CGGI20, LMK+23a, GPvL23].

When constructing advanced cryptographic primitives, it is often required to generate succinct proofs of relations involving its building blocks, e.g. the FHE homomorphic evaluation algorithm is executed correctly, which are typically expressed as algebraic relations over cyclotomic rings. One way to obtain these succinct proofs is to convert the ring relations into equivalent relations over a finite field and apply a succinct argument system for proving finite field relations. Switching to a non-native algebraic structure, however, introduces additional overheads which could be computationally expensive (e.g. [TW24]). This motivates the construction of succinct argument systems which natively operate over cyclotomic rings.

Succinct Non-Interactive Argument of Knowledge (SNARK). This thesis focuses on a type of argument systems known as succinct non-interactive arguments of knowledge (SNARKs), for algebraic relations over cyclotomic rings. More specifically, a SNARK is an argument system satisfying succinctness, non-interactivity, and knowledge soundness, which we briefly recall below.

An interactive proof system (IP) is a protocol consisting of a prover and a verifier, where given a (mathematical) statement belonging to a set known as the language, the prover interactively convinces the polynomial time verifier that the claimed statement indeed belongs to the language [GMR85, BM88]. Focusing on the class NP, a language can be defined by a polynomial-time computable binary relation R, and a statement in the language is any string x such that there exists a witness w satisfying $(x, w) \in R$. In classic (i.e. pen and paper) proofs, a prover sends the witness of a statement to a verifier, and the verifier accepts or rejects the proof deterministically by evaluating R. Unlike classic proofs where each step can be verified trivially, the verification process in interactive proofs can be probabilistic, which means there is a tiny chance that a false proof could be accepted.

An interactive proof must satisfy two basic properties: completeness and soundness. Completeness ensures that an honest prover can convince the verifier with a very high probability. Soundness guarantees that a malicious adversary fails to convince the verifer with a very high probability. A stronger notion of soundness, known as knowledge soundness, guarantees that if a prover could convince the verifer with a high probability, then it must "know" a witness to the claimed statement. An interactive proof satisfying (knowledge) soundness only against polynomial-time adversaries is also called an interactive argument.

Despite the name, an interactive proof/argument could be non-interactive, meaning that the protocol consists of the prover sending a single message to the verifier. It is very useful property in a situation where a prover and a verifier cannot communicate in real time. A large class of interactive proofs/arguments, more specifically those with public-coin and non-adaptive verifier challenges, such as those considered in this thesis, can be transformed into non-interactivite ones by the Fiat-Shamir Transform [FS87] in the random oracle model.

For an interactive proof/argument to be non-trivial, it has to further satisfy zero-knowledge or succinctness (or both). Zero-knowledge means that the prover reveals nothing to the verifier apart from the veracity of the statement during interactions [GMR85]. Succinctness ensures that both the proof size and the verification time (after preprocessing) are sublinear in the combined size of the statement and the witness.

As alluded above, a SNARK is an argument system which satisfies succinctness, non-interactiveness, and knowledge soundness. SNARKs (zero-knowledge or not) have a broad range of applications, including verifiable outsourced computation [FNP20, KPS18], anounymous cryptocurrencies [BCG+14, PSS19], the so-called zk-rollups [BBCCL21, LLC22], and privacy-preserving machine learning [FQZ+21, LXZ21].

SNARKs from Polynomial Interactive Oracle Proof (PIOP). A common construction method of SNARKs is to compile an informatic theoretical object, known as a polynomial interactive oracle proof (PIOP), by a compatible cryptographic commitment scheme, known as a polynomial commitment scheme (PCS).

An IOP can be seen a generalisation of interactive proof where, instead of reading entire prover messages, the verifier has oracle access to the proof strings sent by the prover throughout the protocol, and may probablistically query them at any point

of interactions [BCS16]. The notion of IOP also generalises a more well-known type of proof system known as probabilistically checkable proofs (PCPs) [BFL91] by introducing interactivity. A polynomial interactive oracle proofs (PIOP) is a particular type of IOP where the prover's messages are evaluation oracles of polynomials.

As hinted above, a PIOP is to be combined with a PCS. Unlike an ordinary commitment scheme, a PCS allows to encode a polynomial as a value known as a commitment, against which one can generate efficiently prove the validity of evaluations of the committed polynomial at any given points.

To turn a PIOP and a PCS into a SNARK, the compilation process involves replacing polynomial oracles in a PIOP with commitments to these polynomials and their evaluation proofs. This yields a public-coin interactive argument which can then be converted into a non-interactive protocol using the Fiat-Shamir transform [FS87].

Various types of PIOPs can be employed in the constructing of SNARKs. Sonic [MBKM19], Marlin [CHM+20], Plonk [GWC19] are categorised as univariate PIOPs. Hyrax, Libra, Spartan, Quarks, Gemini, and Hyperplonk are multilinear PIOPS [WTs+18, XZZ+19, Set20, SL20, BCHO22, CBBZ23].

For efficient constructions of SNARKs, the mathematical structures on which the PIOP and the PCS operate must align. While several lattice-based PCS over cyclotomic rings exist, e.g. [CMNW24, FMN23, CLM23], only one PIOP over cyclotomic rings have been proposed [BCS23]. In [BCS23], they constructed a PIOP for an NP-complete relation known as the Rank-1 Constraint System (R1CS). However, their PIOP does not support relations comprising automorphism operations, which can appear in some lattice-based cryptographic primitives [ABPS24, LMK+23b, GPV23], but only arithmetic operations: addition and multiplication.

1.1 Our Contribution

In this work, we construct Onigoroshi — the first multilinear polynomial interactive oracle proof (PIOP) for relations defined by algebraic circuits over cyclotomic rings, incorporating automorphism gates and addition and multiplication gates. The encoding method in our PIOP is identical to the one in Plonk [GWC19], unlike PIOPs for the Rank-1 Constraint System (R1CS). A cyclotomic ring is defined as $R_q := \mathbb{Z}[\zeta_\mathfrak{f}]$ where $\zeta_\mathfrak{f}$ is the primitive $\mathfrak{f}^{th}$ root of unity. Specifically, our contributions are as follows:

1. The construction of a multilinear PIOP over the ring R_q for plonk relations, which can prove relations defined over R_q.

2. The extension of the aforementioned PIOP to support automorphism gates as well as addition and multiplication gates by introducing a new subcomponent PIOP named Automorphism-Check.

To construct our PIOP, we adapted HyperPlonk [CBBZ23] to the ring R_q setting so that it can prove relations over cyclotomic rings natively. We chose HyperPlonk as it is a multilinear extension of Plonk, one of the most widely adapted PIOP in industry due to its universality (circuit-independent trusted setup), prover's efficiency, and the flexibility provided by custom gates. Moreover, HyperPlonk is based on the boolean

hypercube, eliminating the use of a multiplicative subgroup employed in Plonk, which is non-trivial if there exists over the cyclotomic ring R_q. Our PIOP supports relations over the ring R_q involving the canonical automorphisms of R_q. This allows a native compilation of our PIOP by a lattice-based commitment scheme, thereby equipping the resulting SNARK with post-quantum security. A relation over the ring R_q can be defined as

$$R := \{(\mathbb{x}, \mathbb{w}) \in R_q^{n+m} \mid C(\mathbb{x}, \mathbb{w}) = 0\},$$

where C is an arithmetic circuit over the ring R_q. C naturally comprises ring addition and ring multiplication gates. Our PIOP can support a relation defined by an algebraic circuit C that includes ring automorphism gates in addition to the default addition and multiplication gates. A ring automorphism is an isomorphism that maps from a ring to itself. An automorphism gate takes a ring element $a \in R_q$, applies an automorphism $\psi : R_q \to R_q$, and outputs $\psi(a)$.

Applications Our PIOP can be used to prove the satisfiability of algebraic circuits over the ring R_q that include addition, multiplication, and automorphism gates. We provide more concrete examples of such applications below.

1. A relation over the ring R_q, with or without automorphisms, can arise from lattice-based constructions of advanced cryptographic primitives such as Fully Homomorphic Encryption (FHE) and verifiable computation [BHM20, ABPS24]. Our PIOP can prove such relations natively. Specifically, relations over the ring R_q involving automorphisms arise in the bootstrapping of FHE [LMK+23b, GPV23]. Onigoroshi can be used to equip these schemes with the verifiability of server computations.

2. It can prove the SIMD-satisfiability of arithmetic circuits over a finite field using the CRT transformation, defined in Lemma 3.9. Our PIOP can be applied in a finite field setting (equivalent to HyperPlonk), and it can prove $\varphi(z)$ many statements in parallel. $\varphi(z)$ is a CRT parameter explained in Section 3.1. This can be accomplished by applying the inverse CRT transformation to the multiple statements over a finite field to convert them into a single statement over the ring R_q and then running our PIOP over the ring for that statement.

3. It can be used for proving the satisfiability of layered arithmetic circuits, where each layer consists of $\varphi(z)$ copies of a single circuit over a finite field, with adjacent layers interconnected by a permutation. A permutation can be implemented by an R_q-linear combinations of automorphisms $\mathsf{Aut}(R_q)$ since any $\mathbb{Z}_q$-linear map can be expressed as an R_q-linear combinations of automorphisms $\mathsf{Aut}(R_q)$.

1.2 Open Problems

Onigoroshi does not support verification of non-arithmetic operations; thereby, some operations in lattice-based cryptographic constructions must be proved separately (e.g.,

p-ary decomposition, norm bounds, rounding, and modulus switching). Extending Onigoroshi with a lookup argument [ZGK+22, PK22, BCG+18, EFG22, ZBK+22, GW20, CBBZ23, STW24] would allow us to prove relations involving these non-arithmetic operations efficiently.

Batching Automorphism-Checks for different automorphisms should also be considered since the Automorphism-Check currently has to be performed for each set of different automorphism gates, which affects the complexity of Onigoroshi by the number of automorphisms in circuits as a factor.

Introducing a streaming model [BCHO22] to Onigoroshi would enable a space-efficient version of our PIOP and facilitate various configurations with time and memory tradeoffs for the prover.

1.3 Related Work

The PCP theorem [BFLS91] provided a novel characterisation of proof systems by showing that any NP statement has a probablistically checkable proof (PCP) that can be verified in polylogarithmic time relative to the size of a classic proof. Subsequently, Kilian introduced interactive arguments from PCPs [Kil92] that are succcint. Micali[Mic00] showed how to make these interactive arguments non-interactive by applying the Fiat-Shamir Transform [FS87].

In 2010, Groth introduced the first non-interactive zero-knowledge (NIZK) argument with sublinear-communication in the common reference string (CRS) model for the circuit satisfiablity problem (Circuit-SAT) [Gro10]. The argument has a quadratic CRS size and quadratic prover time. Lipmaa reduced the size of CRS to quasi-linear in the circuit size [Lip12]. Following this, Gennaro et al. proposed a NIZK argument with a linear CRS size and quasi-linear prover time [GGPR13]. Groth16 [Gro16] was proposed as a pairing-based zk-SNARK for arithmetic circuit satisfiability (R1CS) [GGPR13], known for its shortest proof size and fastest verification time. However, it is not universal, requiring a new trusted setup for every circuit.

The aforementioned SNARKs are not post-quantum secure, as they rely on classic hardness assumptions. Consequently, various lattice-based SNARKs have been developed to address this limitation. Initially, lattice-based designated-verifier SNARKs were constructed using linear-only encryption, whose existence is a non-falsifiable assumption, and linear-PCP [BISW17, BISW18, ISW21]. Rinocchio is another lattice-based SNARK based on linear-only encryption, which operates over rings [GNS23]. However, the security assumption of these SNARKs based on the linear-only encryption is broken [DAFS24], although there are no known attacks to their constructions themselves. Later works [ACL+22, CLM23] constructed publicly verifiable SNARKs based on a knowledge assumption called the k-R-ISIS assumption, which is also subsequently cryptanalysed [WW23], although there are no attacks against the SNARK constructions themselves. Another line of work [BLNS20, AL21, ACK21, CLM23] constructed publicly verifiable SNARKs based on the Bulletproofs folding technique [BCC+16, BBB+18] have a quasi-linear time prover, with [CLM23] further achieving a polylogarithmic-time verifier.

Notably, none of those above lattice-based constructions follows the framework of

compiling a PIOP with a polynomial commitment scheme. The only exception is the construction of a lattice-based SNARK in [BCS23], where the construction is based on the compilation of a PIOP for R1CS over rings with a lattice-based polynomial commitment scheme from levelled bilinear modules. In [BCS23], they picked a PIOP over fields from [BCG20] and adapted it to construct a PIOP over rings. They picked the PIOP in [BCG20] since it does not use any non-arithmetic operations. We adapted HyperPlonk to ring settings, which supports the plonk arithmatisation [GWC19] and the custom gates. The PIOP from [BCG20] does not support custom gates. Both achieve linear prover time and sublinear verifier time in the size of a circuit.

2 Technical Overview

Notaions A cyclotomic ring is defined as $R_q := \mathbb{Z}[\zeta_{\mathfrak{f}}]$ where $\zeta_{\mathfrak{f}}$ is the primitive $\mathfrak{f}^{th}$ root of unity. We denote the boolean hypercube as $B_\mu := \{0,1\}^\mu$. $\langle i \rangle_n$ is a n-bit binary representation of an integer i. $[\mathbf{x}] := \sum_{i=0}^{\mu-1} x_i \cdot 2^i$ is an integer representation of some bit representation $\mathbf{x} \in B_\mu$.

We provide a brief overview of the construction of Onigoroshi, which can prove relations over a cyclotomic ring R_q involving automorphisms. Our general strategy is to adapt an existing construction of a PIOP over fields into the ring R_q setting.

We considered plonkish PIOPs as candidates since PIOPs over a ring with the Plonk [GWC19] arithmetisation have not been concretely explored. We say plonkish when a PIOP encodes arithmetic circuits similarly to Plonk. We consider plonkish PIOPs because Plonk is universal (single trusted setup for all circuits of some bounded size), prover efficient, and it supports custom gates. However, Plonk is not an ideal candidate for the ring R_q setting, since it uses a complex field structure, specifically assuming the existence of a multiplicative subgroup of the field, which is a non-trivial object to find in the ring R_q. Moreover, Plonk performs the Fast Fourier Transforms (FFTs) in the protocol, which leads to the quasi-linear prover time.

Alternatively, we consider HyperPlonk [CBBZ23] that works with the boolean hypercube, which is a subset of R_q, instead of a multiplicative subgroup. This simplifies the structure of the protocol, thereby eliminating the need for the FFTs in HyperPlonk, which reduces the prover time into linear in the size of circuit. In the process of adapting HyperPlonk to the ring R_q with automorphism gates setting, we encountered a few challenges. Firstly, Onigoroshi is not perfect complete since denominators (evaluations of a polynomial over B_μ) of the Product-Check described in Section 4.3 have to be invertible over R_q, in other words, they have to be elements in $R_q^\times$. Another difficulty is that the gate identity check described in the following for automorphism gates must separately be performed from that of addition and multiplication gates because automorphism operations are non-arithmetic operations.

(Hyper)Plonk

Plonk is a univariate polynomial IOP that operates over some field $\mathbb{F}$. Given an arithmetic circuit C over $\mathbb{F}$ and a pair consisting of an instance $\mathbb{x} \in \mathbb{F}^n$ and a witness $\mathbb{w} \in \mathbb{F}^m$, the prover encodes the evaluation of the circuit $C(\mathbb{x}, \mathbb{w})$ using a univariate polynomial $\omega \in \mathbb{F}[X]$ and a cyclic multiplicative subgroup $S \subseteq \mathbb{F}$ where the generator of the cyclic group S is denoted by $g \in S$. Suppose there are s many gates in the circuit C, where each gate is fan-in two (two inputs and one output) and is either an addition or multiplication operation. The prover encodes the n-dimensional vector $\mathbb{x}$ as

$$\omega(s^{-i}) \quad \text{for each } i = 1, \ldots, n$$

and encodes the left input, the right input, and the output for each j^{th} gate as

$$\omega(g^{3j}), \quad \omega(g^{3j+1}), \quad \omega(g^{3j+2}),$$

respectively, for all $j = 0, \ldots, s$. The last gate is an additional gate to ensure that the final output of the circuit is zero, such that $C(\mathbb{x}, \mathbb{w}) = 0$. After encoding, the oracle to the polynomial ω, denoted by $[[\omega]]$, and ω itself can be interpreted as a proof string and a witness, respectively. The prover $P(\mathsf{pp}, [[\omega]], \omega)$ convinces the verifier $V(\mathsf{vp}, [[\omega]])$ that the prover correctly encoded the circuit evaluation $C(\mathbb{x}, \mathbb{w})$ by a univariate polynomial and knows this polynomial ω. Plonk comprises two primary polynomial IOPs as subcomponents: zero-check and permutation-check. The prover convinces the verifier by running these polynomial IOPs.

HyperPlonk is a multilinear polynomial variant of Plonk that operates over a field $\mathbb{F}$. The prover encodes an arithmetic circuit C with s many gates using a multilinear polynomial $\omega \in \mathbb{F}[X^{\mu+2}]$ and the boolean hypercube $B_\mu := \{0,1\}^\mu$ where $n+s+1 = 2^\mu$. An instance $\mathbb{x}$ is encoded as

$$\omega(0, 0, \langle i \rangle) \quad \text{for all } i = 0, \ldots, n-1,$$

where $\langle i \rangle$ denotes the μ-bit binary representation of i. The left and right inputs, as well as the output for j^{th} gate are encoded as

$$\omega(0, 0, \langle j \rangle), \quad \omega(0, 1, \langle j \rangle), \quad \omega(1, 0, \langle j \rangle),$$

for all $j = n, \ldots, n+s$. By running the zero-check and permutation-check, the prover convinces the verifier that the circuit evaluation $C(\mathbb{x}, \mathbb{w})$ has been correctly encoded by the multilinear polynomial ω and that the prover knows this polynomial ω.

Onigoroshi

In the construction of Onigoroshi, the protocol first needs to encode an algebraic circuit over the ring R_q, which consists of addition, multiplication, and automorphism gates, using a multilinear polynomial over R_q. An automorphism gate for any automorphism

$$\psi : R_q \to R_q \in \mathsf{Aut}(R_q)$$

takes only the left input $a \in R_q$ and outputs $\psi(a)$. Second, it performs the ring version of Zero-Check, as detailed in Section 4.2, to verify that the multilinear polynomial has correctly encoded gates except for automorphism gates. It ensures that the output of each gate results from the correct binary operation, either addition or multiplication, applied to the left and right inputs (Gate Identity Check). Third, it performs the Automorphism-Check, which is a new check we introduced in the section 4.6, to ensure that automorphism gates have been correctly encoded by the multilinear polynomial. Specifically, this check verifies that the output of each automorphism gate is the evaluation of a corresponding automorphism on the left input. Finally, it performs the ring version of Permutation-Check, detailed in Section 4.5, to ensure the equality between the output of each gate and the corresponding input to another gate. The Figure 1 shows the building blocks of Onigoroshi.

In the following, we summarise the construction steps of Onigoroshi. First, we show how Onigoroshi encodes algebraic circuits over the ring R_q comprising addition,

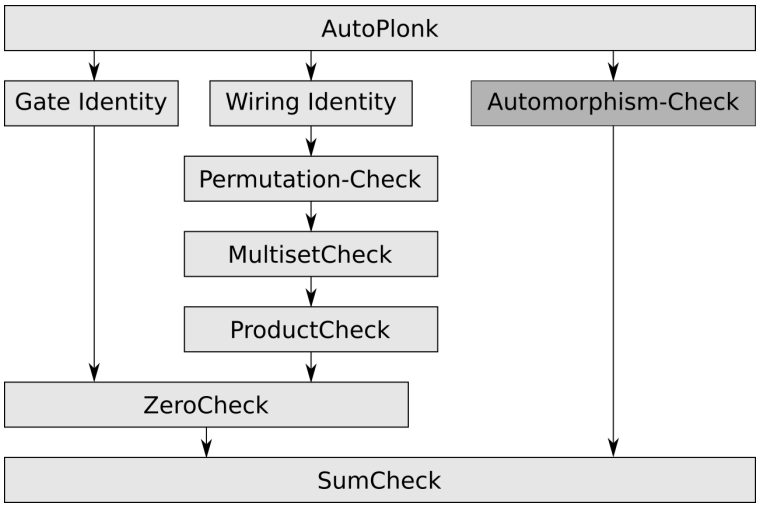


Figure 1: The diagram shows the subcomponents Onigoroshi consists of. Arrows indicate the reduction between two components which means that the top component internally uses the bottom component.

multiplication, and automorphism gates. Second, we describe the gate identity check in which the consistency amongst the left and right inputs and output over a gate operation is verified for each gate except for automorphism gates. Third, we present the gate identity check for automorphism gates, which uses the Automorphism-Check. Finally, we describe the wiring identity check that verifies the consistency between wires in a circuit, i.e., equalities between the output of a gate and the input of the corresponding gate.

2.1 Econding Circuits over the ring R_q with Automorphisms

We outline how Onigoroshi encodes algebraic circuits over the ring R_q and explain the concept of automorphism gates in the following.

Let $C : R_q^{n+m} \to R_q$ be a fan-in algebraic circuit defined over a cyclotomic ring R_q, which takes an instance $\mathbb{x} \in R_q^n$ and a witness $\mathbb{w} \in R_q^m$ as inputs, performs operations using addition, multiplication, and automorphism gates, and then produces a result in R_q. Let s be the number of gates in the circuit. The computation trace of the circuit C can be represented as

$$T := \left\{(L_i, R_i, O_i) \in R_q^3\right\}_{i=0,\ldots,n+s},$$

where (L_i, R_i, O_i) denote the left input, right input, and output of the i^{th} gate, respectively. Suppose $n + s + 1 = 2^\mu$, where n denotes the size of the public input $\mathbb{x}$, s denotes the number of gates, 1 represents the final output of the circuit. The prover encodes T by interpolating a $\mu + 2$-variate multilinear polynomial $\omega \in R_q[X^{\mu+2}]$ such that

$$L_i = \omega(0, 0, \langle i \rangle),$$
$$R_i = \omega(0, 1, \langle i \rangle),$$
$$O_i = \omega(1, 0, \langle i \rangle),$$

for all $i = 0, \ldots, n + s$, where $\langle i \rangle$ denotes the μ-bit binary representation of i.
Using the boolean hypercube $B_\mu := \{0,1\}^\mu$, we can rewrite the encoding as below.

$$\begin{aligned} L_{[\mathbf{x}]} &= \omega(0,0,\mathbf{x}), \\ R_{[\mathbf{x}]} &= \omega(0,1,\mathbf{x}), \\ O_{[\mathbf{x}]} &= \omega(1,0,\mathbf{x}), \end{aligned}$$

for all $\mathbf{x} \in B_\mu$, where $[\mathbf{x}]$ denotes the integer representation of the binary vector $\mathbf{x}$.
Algebraic circuits we consider have addition gates, multiplication gates, and automorphism gates for automorphisms in $\mathsf{Aut}(R_q)$. $\mathsf{Aut}(R_q)$ is identical to the Galois group

$$\mathrm{Gal}(\mathbb{Q}(\zeta_\mathfrak{f})/\mathbb{Q}) = \{\psi_i : \zeta_\mathfrak{f} \mapsto \zeta_\mathfrak{f}^i \mid i \in (\mathbb{Z}/n\mathbb{Z})^\times\},$$

where $\mathbb{Q}(\zeta_\mathfrak{f})/\mathbb{Q}$ is the cyclotomic field extension for the $\mathfrak{f}^{th}$ root of unity in $\mathbb{Q}$. An automorphism gate for ψ takes the left input L_i and outputs $\psi(L_i)$. The right input to an automorphism gate is a dummy. We let S_ψ denote the set of indices of automorphism gates for an automorphism ψ such that $S_\psi \subseteq B_\mu$.

2.2 Gate Indentity Check

In the following, we describe the gate identity check performed in Onigoroshi to ensure that the multilinear polynomial ω has correctly encoded the circuit C. Specifically, it verifies that the output of each gate is the result of the expected gate operation – whether it is addition, multiplication, or custom gate operations – applied to the inputs. Automorphism gates are verified separately, which will be explained later in this section.

Let $S_0, S_1, S_2 : R_q^\mu \to \{0,1\}$ be selector polynomials. In the preprocessing phase, the indexer $\mathcal{I}(\mathsf{gp}, C)$ computes the oracle to these selector polynomials. The prover convinces that the following equation holds using the Zero-Check described in Section 4.2:

$\forall\, \mathbf{x} \in B_\mu \setminus S_\psi$

$$\begin{aligned} S_0(\mathbf{x}) \cdot \Big(\underbrace{\omega(0,0,\mathbf{x})}_{L_{[\mathbf{x}]}} + \underbrace{\omega(0,1,\mathbf{x})}_{R_{[\mathbf{x}]}} \Big) \;+\; S_1(\mathbf{x}) \cdot \Big(\underbrace{\omega(0,0,\mathbf{x})}_{L_{[\mathbf{x}]}} \cdot \underbrace{\omega(0,1,\mathbf{x})}_{R_{[\mathbf{x}]}} \Big) \\ + S_3(\mathbf{x}) \cdot G\Big(\underbrace{\omega(0,0,\mathbf{x})}_{L_{[\mathbf{x}]}}, \underbrace{\omega(0,1,\mathbf{x})}_{R_{[\mathbf{x}]}} \Big) \;-\; \underbrace{\omega(1,0,\mathbf{x})}_{O_{[\mathbf{x}]}} + I(\mathbf{x}) = 0, \end{aligned} \tag{1}$$

where the selector polynomials are evaluated in the following manner:

- for addition gate: $(O_{[\mathbf{x}]} = L_{[\mathbf{x}]} + R_{[\mathbf{x}]})$

$$S_0(\mathbf{x}) = 1, \qquad S_1(\mathbf{x}) = S_2(\mathbf{x}) = 0$$

- for multiplication gate: $(O_{[\mathbf{x}]} = L_{[\mathbf{x}]} \cdot R_{[\mathbf{x}]})$

$$S_1(\mathbf{x}) = 1, \qquad S_0(\mathbf{x}) = S_2(\mathbf{x}) = 0$$

- when $[\mathbf{x}] < n$ or $[\mathbf{x}] = n + s$: $(O_{[\mathbf{x}]} = I(\mathbf{x}))$

$$S_0(\mathbf{x}) = S_1(\mathbf{x}) = S_2(\mathbf{x}) = 0$$

When $[\mathbf{x}] < n$ or $[\mathbf{x}] = n + s$, $\omega(0, 0, \mathbf{x})$ represents the public input or the last output of the circuit, respectively.

2.3 Automorphism-Check

We introduce a new PIOP named Automorphism-Check described in Section 4.6, performed by Onigoroshi to verify the gate identity of automorphism gates.

For the automorphism gates represented by S_ψ, the gate identity is separately verified. The prover needs to convince the verifier that the following holds:

$\forall\, \mathbf{x} \in S_\psi$

$$\psi\big(\underbrace{\omega(0, 0, \mathbf{x})}_{L_{[\mathbf{x}]}}\big) = \underbrace{\omega(1, 0, \mathbf{x})}_{O_{[\mathbf{x}]}}.$$

The Automorphism-Check is built upon another PIOP, Sum-Check, described in Section 4.1.

2.4 Wiring Identity Check

Finally, we explain the wiring identity check performed in Onigoroshi to verify the consistency over wires, i.e., the equalities between the output of each gate and its corresponding input of another gate. In a circuit, the output of each gate is wired to the input of another gate except for the last gate. The corresponding output and input have the same value.

These equality constraints are captured by a permutation $\sigma : B_{\mu+2} \to B_{\mu+2}$ for $B_{\mu+2} := \{0, 1\}^{\mu+2}$. The prover needs to convince the verifier that the following equation holds.

$$\omega(\mathbf{x}) = \omega(\sigma(\mathbf{x})) \quad \forall \mathbf{x} \in B_{\mu+2}. \tag{2}$$

In the preprocessing phase, the indexer $\mathcal{I}(\mathsf{gp}, C)$ computes the oracle to the multilinear polynomial $\sigma^* : B_{\mu+2} \to R_q$ such that $\sigma^*(\mathbf{x}) = [\sigma(\mathbf{x})]$ for all $\mathbf{x} \in B_{\mu+2}$. $\sigma^*(\mathbf{x})$ is the integer representation of $\sigma(\mathbf{x})$.Then, the prover convinces the verifier that the following holds:

$$\Big\{([\mathbf{x}], \omega(\mathbf{x}))\Big\}_{\mathbf{x} \in B_{\mu+2}} = \Big\{(\sigma^*(\mathbf{x}), \omega(\mathbf{x}))\Big\}_{\mathbf{x} \in B_{\mu+2}}. \tag{3}$$

The equality of the multiset (3) implies that (2) holds. The equality (3) can be proved by the Multiset-Check described in Section 4.4.

3 Preliminaries

We denote a finite field with order p by $\mathbb{F}_p$. $\mathbb{Z}_q^*$ is the multiplicative group of integers modulo q. $\mathsf{negl}(x)$ is a negligible function $\mathsf{negl}(x) : \mathbb{N} \to \mathbb{R}$ such that for any positive integer c there exists an integer N such that $\forall x \geq N \;\; |\mathsf{negl}(x)| \leq 1/x^c$. A multiset is an extension of a set that can contain multiple instances for each element.

Polynomial

The multilinear extension is a core tool used for the Zero-Check in Section 4.2.

Definition 3.1 (Extentions)**.** Let $\mathbb{F}$ to be a finite field, and let $f : \{0,1\}^k \to \mathbb{F}$ be a function that maps the k-dimensional hypercube to $\mathbb{F}$. A k-variate polynomial $\widetilde{f}$ over $\mathbb{F}$ is said to be the extension of f if

$$\widetilde{f}(\mathbf{x}) = f(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \{0,1\}^k.$$

The extension $\widetilde{f}$ is said to be multilinear if it has a degree at most 1 for each variable.

Lemma 3.1. Any k-variate function $f : \{0,1\}^k \to \mathbb{F}$ has a unique multilinear extension (MLE) $\widetilde{f}$ over $\mathbb{F}$ such that:

$$\widetilde{f}(\mathbf{X}) = \sum_{\mathbf{b} \in \{0,1\}^k} f(\mathbf{b}) \cdot \mathsf{eq}(\mathbf{b}, \mathbf{X}),$$

where $\mathsf{eq}(\mathbf{Y}, \mathbf{X}) = \prod_{i=1}^{k} (X_i Y_i + (1 - X_i)(1 - Y_i))$.

Number Theory

We provide a list of primary number theoretical definitions used in this work.

Definition 3.2 (Euler's Totient Function)**.** Given a positive integer n, the Euler's totient function φ counts the number of positive integers k less than n that are coprime to n. $\varphi(n)$ is the number of $k \in \mathbb{Z}_{\geq 0}$ such that $1 \leq k \leq n$ and $\gcd(n, k) = 1$. If $p_1, ..., p_m$ are the distinct primes such that $p_i | n$ for all $i \in [m]$, then

$$\varphi(n) = n \left(1 - \frac{1}{p_1}\right) \cdots \left(1 - \frac{1}{p_m}\right).$$

Definition 3.3 (Root of Unity)**.** For any positive integer $\mathfrak{f}$, an $\mathfrak{f}^{th}$ root of unity is a number x satisfying the equation $x^{\mathfrak{f}} = 1$. There are $\mathfrak{f}$ such solutions to the equation.

Definition 3.4 (Primitive Root of Unity)**.** A primitive $\mathfrak{f}^{th}$ root of unity is a number x satisfying $x^k = 1$ for the smallest number k where $k = \mathfrak{f}$. In other words, a primitive $\mathfrak{f}^{th}$ root of unity is an $\mathfrak{f}^{th}$ root of unity that is not an e^{th} root of unity for any $e < \mathfrak{f}$.

The group of primitive residue classes modulo $\mathfrak{f}$ arises in the definition of automorphisms over the ring R_q.

Definition 3.5 (Group of Primitive Residue Classes modulo $\mathfrak{f}$)**.** The multiplicative group $(\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times$ consists of non-negative integers $k \in \{0, \ldots, \mathfrak{f} - 1\}$ modulo $\mathfrak{f}$ such that k is coprime to $\mathfrak{f}$ ($\gcd(k, \mathfrak{f}) = 1$). The order of the group is given by Euler's totient function defined in 3.2: $|(\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times| = \varphi(\mathfrak{f})$. If $\mathfrak{f}$ is a prime, then the group is cyclic.

Algebraic Number Theory

We list elementary definitions in algebraic number theory relevant to the cyclotomic field described in Section 3.1.

The following definition and lemmas imply that factors of a cyclotomic polynomial in Lemma 3.8 are coprime each other.

Definition 3.6 (Irreducible Polynomial)**.** A non-constant polynomial $f \in K[X]$ is irreducible over a field K if f cannot be factored into the product of two non-constant polynomials over K.

Lemma 3.2. An ideal $\langle f \rangle$ generated by an irreducible polynomial $f \in K[X]$ over a field K is a maximal ideal.

Lemma 3.3. Distinct maximal ideals of a ring R are coprime each other.

The following definitions are used in relation to cyclotomic fields (Definition 3.13) and cyclotomic rings (Definition 3.17).

Definition 3.7 (Field Extension)**.** A field K is said to be an extension of a field F, denoted by K/F, if F is a subfield of K. For example, the complex field $\mathbb{C}$ is an extension field of the real field $\mathbb{R}$.

Definition 3.8 (Degree of Field Extension)**.** Let K/F be a field extension. We can see K as a vector space over a field F. The degree of the extension K/F, denoted by $[K : F]$, is the dimension of the vector space K. The degree of a field extension can be finite or infinite.

Definition 3.9 (Number Field)**.** A finite field extension over a rational field $\mathbb{Q}$ is called a number field.

Definition 3.10 (Algebraic Integer)**.** An algebraic integer in a number field K is an element $\alpha \in K$, which is a root of a monic polynomial $f \in \mathbb{Z}[X]$ $(f(\alpha) = 0)$.

Definition 3.11 (Ring of Integers)**.** A set of algebraic integers in a number field K is denoted as $\mathcal{O}_K$ and is called the ring of integers of K.

3.1 Cyclotomic Field

In this section, we first provide the definition of the cyclotomic field, the Galois group containing automorphisms over the cyclotomic field, and the ring of integers of the cyclotomic field, known as the cyclotomic ring. Afterwards, we explain the CRT transformation, which is an isomorphism mapping an element in a cyclotomic ring to multiple slots of elements in a finite field.

Definition 3.12 (Cyclotomic Extension)**.** The cyclotomic extension of any field K is the field extension of the form $K(\zeta)$, where ζ is a root of unity.

Definition 3.13 (Cyclotomic Field)**.** The $\mathfrak{f}^{th}$ cyclotmoic field is a cyclotomic extension $\mathbb{Q}(\zeta_\mathfrak{f}) = \{\sum_{i=0}^{\mathfrak{f}-1} a_i \cdot \zeta_\mathfrak{f}^i \mid a_i \in \mathbb{Q}\}$ of the rational field $\mathbb{Q}$, where $\zeta_\mathfrak{f}$ is the primitive $\mathfrak{f}^{th}$ root of unity.

Definition 3.14 (Group of $\mathfrak{f}^{th}$ Roots of Unity)**.** The group $\mu_\mathfrak{f}$ of different $\mathfrak{f}^{th}$ roots of unity in a field K is a cyclic group.

For example, if $K = \mathbb{C}$, $\mu_2 = \{1, -1\}$ and $\mu_4 = \{1, -1, i, -i\}$. The primitive $\mathfrak{f}^{th}$ roots of unity are the generators of $\mu_\mathfrak{f}$, denoted by $\zeta_\mathfrak{f}$. $\zeta_\mathfrak{f}^k$ is a primitive $\mathfrak{f}^{th}$ root of unity if and only if $\gcd(k, \mathfrak{f}) = 1$ since the order of $\zeta_\mathfrak{f}^k$ is $\mathfrak{f}/\gcd(k, \mathfrak{f})$. This implies that $\mu_\mathfrak{f}$ contains $\phi(\mathfrak{f}) = |(\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times|$ many primitive $\mathfrak{f}^{th}$ roots of unity. We can write $K(\mu_\mathfrak{f})$ for the cyclotomic extension $K(\zeta_\mathfrak{f})$ since $\zeta_\mathfrak{f}$ generates all the elements in $\mu_\mathfrak{f}$.

Automorphisms over the $\mathfrak{f}^{th}$ cyclotomic field form an abelian group called the Galois Group, defined by the group $(\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times$, to which the Galois group is injectively homomorphic.

Definition 3.15 (Automorphisms)**.** An automorphism is an isomorphism map from a mathematical object to itself, including a group, ring, and vector space.

Lemma 3.4. The mapping

$$\mathrm{Gal}(K(\mu_\mathfrak{f})/K) \to (\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times$$

defined by $\psi \mapsto i \mod \mathfrak{f}$ where $\psi(\zeta) = \zeta^i$ for all $\zeta \in \mu_\mathfrak{f}$ is an injective homomorphism. Since $(\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times$ is abelian, the subgroup $\mathrm{Gal}(K(\mu_\mathfrak{f})/K)$ is also abelian. When $K = \mathbb{Q}$, the above map is an isomorphism.

Lemma 3.5 (Automorphisms over Cyclotomic Extensions)**.** Automorphisms over a cyclotomic extension $K(\mu_\mathfrak{f})/K$ form a group called the Galois group, denoted as $\mathrm{Gal}(K(\mu_\mathfrak{f})/K)$. For any $\psi \in \mathrm{Gal}(K(\mu_\mathfrak{f})/K)$, there exists an integer $i \in (\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times$ such that $\psi(\zeta) = \zeta^i$ for all $\zeta \in \mu_\mathfrak{f}$.
For the cyclotomic field extension $\mathbb{Q}(\zeta_\mathfrak{f})/\mathbb{Q}$, where $K = \mathbb{Q}$, the Galois group can be expressed as

$$\mathrm{Gal}(\mathbb{Q}(\zeta_\mathfrak{f})/\mathbb{Q}) = \{\psi_i \mid i \in (\mathbb{Z}/\mathfrak{f}\mathbb{Z})^\times\}, \quad \text{where } \psi_i : \zeta \mapsto \zeta^i,$$

for all $\zeta \in \mu_\mathfrak{f}$.

We define a cyclotomic ring, which is equivalent to the ring of integers of a cyclotomic field.

Definition 3.16 (Cyclotomic Polynomial)**.** The $\mathfrak{f}^{th}$ cyclotomic polynomial $\Phi_\mathfrak{f}(x) \in \mathbb{C}[X]$ is defined as

$$\Phi_\mathfrak{f}(x) = \prod_{\zeta \in \mu_\mathfrak{f}} (x - \zeta)$$

where $\mu_\mathfrak{f}$ is a group of primitive $\mathfrak{f}^{th}$ roots of unity. The degree of $\Phi_\mathfrak{f}$ is defined by $\varphi(\mathfrak{f})$ where φ is the Euler's totient function.

Definition 3.17 (Cyclotomic Ring)**.** The ring of integers of $\mathfrak{f}^{th}$ cyclotomic fields $\mathbb{Q}(\zeta_\mathfrak{f})$ is $R = \mathbb{Z}[\zeta_\mathfrak{f}]$, where $\zeta_\mathfrak{f}$ is the primitive $\mathfrak{f}^{th}$ root of unity. The ring of integer R is often referred to as the cyclotomic ring. We denote the quotient version of the $\mathfrak{f}^{th}$ cyclotomic ring by R_q, defined as

$$R_q := \frac{R}{qR} = \frac{\mathbb{Z}[\zeta_\mathfrak{f}]}{q\mathbb{Z}[\zeta_\mathfrak{f}]}, \tag{4}$$

where $q \in \mathbb{N}$.

Lemma 3.6. The cyclotomic ring R_q is isomorphic to $\mathbb{Z}[X]/\Phi_\mathfrak{f}(X)$ where $\Phi_\mathfrak{f}(X)$ is the $\mathfrak{f}^{th}$ cyclotomic polynomial.

$$R_q \cong \frac{\mathbb{Z}_q[X]}{\Phi_\mathfrak{f}(X)}. \tag{5}$$

Proof. There is a natural homomorphism $\phi : \mathbb{Z}_q[X] \to \mathbb{Z}[\zeta_\mathfrak{f}]$ defined by $\phi(f(X)) = f(\zeta_\mathfrak{f})$ for all $f \in \mathbb{Z}_q[X]$. It can be verified that $\phi(f(X)) + \phi(g(X)) = f(\zeta_\mathfrak{f}) + g(\zeta_\mathfrak{f}) = (f + g)(\zeta_\mathfrak{f}) = \phi(f(X) + g(X))$ for all $f, g \in \mathbb{Z}_q[X]$. Similarly, ϕ also preserves multiplication. The kernel of this homomorphism is given by $\ker(\phi) := \{f \in \mathbb{Z}_q[X] \mid \phi(f) = 0\} = \langle \Phi_\mathfrak{f}(X) \rangle$ as $\phi(\Phi_\mathfrak{f}(X)) = \Phi_\mathfrak{f}(\zeta_\mathfrak{f}) = 0$. By the first isomorphism theorem, $\mathbb{Z}_q[X]/\ker(\phi) \cong \mathrm{Im}(\phi)$. Consequently, $\mathbb{Z}_q[X]/\langle \Phi_\mathfrak{f}(X) \rangle \cong \mathbb{Z}_q[\zeta_\mathfrak{f}] = R_q$. □

The following are required to describe the CRT transformation in Lemma 3.9.

Proposition 3.1. Let $\mathbb{F}_q$ be a finite field of the order q and $f \in \mathbb{F}_q[X]$ be an irreducible polynomial of degree n. Then $\mathbb{F}_q[X]/f(X)$ is a finite field of order q^n.

Theorem 3.1 (Theorem by E.H.Moore [Moo96])**.** Let q be a prime number and n be an integer. For every prime power q^n, there exist finite fields (Galois fields) of order q^n, and they are all isomorphic to each other.

Lemma 3.7 (Chinese Remainder Theorem for Ring of Integers)**.** Let R be the ring of integers and $I = \prod_{i\in[m]} I_i$ where $I_1, ..., I_m$ are pairwise coprime (comaximal) ideals in R. Then, the ring homomorphism $\psi : R \to \bigoplus_{i\in[m]} (R/I_i)$, defined by $\psi(r) = (r + I_1, ..., r + I_m)$, induces a map that defines the following ring isomorphism:

$$\begin{aligned} R/I &\cong \bigoplus_{i\in[m]} (R/I_i) \\ &= R/I_1 \times \cdots \times R/I_m. \end{aligned}$$

Lemma 3.8 ([Con15], Theorem 5.3, [LS18], Theorem 1.1.)**.** Let $\mathfrak{f} = \prod p_i^{f_i}$ for $f_i \geq 1$ and $z = \prod p_i^{g_i}$ for any $1 \leq g_i \leq f_i$, where p_i is a prime number, and these products are prime factorisations of $\mathfrak{f}$ and z. If q is a prime number such that $q \equiv 1 \pmod{z}$ and $ord_\mathfrak{f}(q) = \mathfrak{f}/z$, i.e. $\min\{k \in \mathbb{Z} \mid q^k \equiv 1 \pmod{\mathfrak{f}}\} = \mathfrak{f}/z$, then the $\mathfrak{f}^{th}$ cyclotomic polynomial $\Phi_\mathfrak{f}(X)$ factors as below:

$$\Phi_\mathfrak{f}(X) \equiv \prod_{j=1}^{\varphi(z)} (X^{\mathfrak{f}/z} - r_j) \pmod{q},$$

where $r_j \in \mathbb{Z}_q^*$ are distinct and $X^{\mathfrak{f}/z} - r_j$ are irreducible in the ring $\mathbb{Z}_q[X]$.

We finally describe the CRT transformation, which is an isomorphism between the ring R_q and multiple slots of a finite field.

Lemma 3.9. Under the condition in Lemma 3.8, the cyclotomic ring R_q splits into $\varphi(z)$ many slots of a finite field with order $q^{\mathfrak{f}/z}$ as below.

$$R_q \overset{(4,5,7,8)}{\cong} \mathbb{F}_{q^{\mathfrak{f}/z}}^{\varphi(z)} \tag{6}$$

where $\mathbb{F}_{q^{\mathfrak{f}/z}}$ is a finite field of order $q^{\mathfrak{f}/z}$. We represent the isomorphism transformation described above as CRT, given by:

$$\mathsf{CRT} : R_q \xrightarrow{\sim} \mathbb{F}_{q^e}^{\phi},$$

where $e = \mathfrak{f}/z$ and $\phi = \varphi(z)$.

Proof. By Chinese Remainder Theorem 3.7 and the Lemma 3.8, $\mathbb{Z}_q[X]/\Phi_{\mathfrak{f}}(X)$ splits into $\varphi(z)$ many finite fields:

$$\frac{\mathbb{Z}_q[X]}{\Phi_{\mathfrak{f}}(X)} \cong \bigoplus_{j\in[\varphi(z)]} \frac{\mathbb{Z}_q[X]}{X^{\mathfrak{f}/z} - r_j} = \frac{\mathbb{Z}_q[X]}{X^{\mathfrak{f}/z} - r_1} \times \cdots \times \frac{\mathbb{Z}_q[X]}{X^{\mathfrak{f}/z} - r_{\varphi(z)}}, \tag{7}$$

where $\mathbb{Z}_q[X]/(X^{\mathfrak{f}/z} - r_j)$ is a finite field of order $q^{\mathfrak{f}/z}$ by Proposition 3.1. Consequently, $\mathbb{Z}_q[X]/(X^{\mathfrak{f}/z} - r_j)$ is isomorphic to any finite field $\mathbb{F}_{q^{\mathfrak{f}/z}}$ of order $q^{\mathfrak{f}/z}$ by Theorem 3.1.

$$\mathbb{Z}_q[X]/(X^{\mathfrak{f}/z} - r_j) \cong \mathbb{F}_{q^{\mathfrak{f}/z}}. \tag{8}$$

□

3.2 Schwarz-Zippel-Demillo-Lipton (SZDL) Lemma over R_q

The Schwarz-Zippel-Demillo-Lipton (SZDL) lemma is often used in the soundness analysis of IOPs. We modified the SZDL lemma for the ring R_q setting for our PIOP. First, we provide the SZDL lemma for field settings.

Lemma 3.10 (Schwarz-Zippel-Demillo-Lipton (SZDL) Lemma)**.** Let p and q be distinct μ-variate polynomials over a finite field $\mathbb{F}$ with total degree at most d. The probability that the evaluations of these two polynomials at a randomly chosen point are equal is upper bounded by

$$\Pr_{\mathbf{r}\leftarrow\$\mathbb{F}^\mu}[p(\mathbf{r}) = q(\mathbf{r})] \leq \frac{d}{|\mathbb{F}|}.$$

The following is the SZDL lemma for the ring R_q, which is used for proving the soundness of subcomponent PIOPs

Lemma 3.11. Let $q \equiv 1 \pmod{z}$ and $ord_m(q) = \mathfrak{f}/z$ as in Lemma 3.8. Let f and $g \in R_q[X^\mu]$ be μ-variate polynomials wth total degree at most d such that $f \neq g$. By Lemma 3.9, it follows that $R_q \cong \mathbb{F}_{q^{\mathfrak{f}/z}}^{\varphi(z)}$. Let $e = \mathfrak{f}/z$ and $\phi = \varphi(z)$. Then the following holds,

$$\Pr_{\mathbf{r} \leftarrow\$ R_q^\mu}[f(\mathbf{r}) = g(\mathbf{r})] \leq \frac{d}{|\mathbb{F}_{q^e}|} = \frac{d}{q^e}$$

Proof. Let

$$f(\mathbf{X}) = \sum_{i_1,\ldots,i_\mu=0}^{d} a_{i_1,\ldots,i_\mu} \cdot X_1^{i_1} \cdots X_\mu^{i_\mu},$$

$$g(\mathbf{X}) = \sum_{i_1,\ldots,i_\mu=0}^{d} b_{i_1,\ldots,i_\mu} \cdot X_1^{i_1} \cdots X_\mu^{i_\mu}.$$

Then

$$\begin{aligned}\mathsf{CRT}(f)(\mathbf{X}) &= \mathsf{CRT}\Big(\sum_{i_1,\ldots,i_\mu=0}^{d} a_{i_1,\ldots,i_\mu} \cdot X_1^{i_1} \cdots X_\mu^{i_\mu}\Big) \\ &= \sum_{i_1,\ldots,i_\mu=0}^{d} \mathsf{CRT}(a_{i_1,\ldots,i_\mu}) \odot \mathsf{CRT}(X_1)^{\odot i_1} \odot \cdots \odot \mathsf{CRT}(X_\mu)^{\odot i_\mu},\end{aligned}$$

where $\odot$ is a hadamard product and $\mathbf{x}^{\odot i} = \mathbf{x} \odot \cdots \odot \mathbf{x}$ is a hadamard power i of $\mathbf{x}$. Now let

$$(\alpha_{i_1,\ldots,i_\mu}^{(1)}, \ldots, \alpha_{i_1,\ldots,i_\mu}^{(\phi)}) = \mathsf{CRT}(a_{i_1,\ldots i_\mu}), \quad (x_j^{(1)}, \ldots, x_j^{(\phi)}) = \mathsf{CRT}(X_j) \quad \forall j \in [\mu].$$

Then

$$\begin{aligned}\mathsf{CRT}(f)(\mathbf{X}) &= \left(\sum_{i_1,\ldots,i_\mu=0}^{d} \alpha_{i_1,\ldots i_\mu}^{(1)} (x_1^{(1)})^{i_1} \cdots (x_\mu^{(1)})^{i_\mu}, \ldots, \sum_{i_1,\ldots,i_\mu=0}^{d} \alpha_{i_1,\ldots i_\mu}^{(\phi)} \left(x_1^{(\phi)}\right)^{i_1} \cdots \left(x_\mu^{(\phi)}\right)^{i_\mu}\right) \\ &= \left(F_1(x_1^{(1)}, \ldots, x_\mu^{(1)}), \ldots, F_\phi(x_1^{(\phi)}, \ldots, x_\mu^{(\phi)})\right),\end{aligned}$$

where $F_j(x_1^{(j)}, \ldots, x_\mu^{(j)}) = \sum_{i_1,\ldots,i_\mu=0}^{d} \alpha_{i_1,\ldots,i_\mu}^{(j)} \left(x_1^{(j)}\right)^{i_1} \cdots \left(x_\mu^{(j)}\right)^{i_\mu}$.

Similarly,

$$\mathsf{CRT}(g)(\mathbf{X}) = \left(G_1(x_1^{(1)}, \ldots, x_\mu^{(1)}), \ldots, G_\phi(x_1^{(\phi)}, \ldots, x_\mu^{(\phi)})\right),$$

where

$$(\beta_{i_1,\ldots,i_\mu}^{(1)}, \ldots, \beta_{i_1,\ldots,i_\mu}^{(\phi)}) = \mathsf{CRT}(b_{i_1,\ldots i_\mu}),$$

$$G_j(x_1^{(j)}, \ldots, x_\mu^{(j)}) = \sum_{i_1,\ldots,i_\mu=0}^{d} \beta_{i_1,\ldots,i_\mu}^{(j)} \left(x_1^{(j)}\right)^{i_1} \cdots \left(x_\mu^{(j)}\right)^{i_\mu}.$$

Let $(r_1, ..., r_\mu) = \mathbf{r}$ and $(\gamma_j^{(1)}, ..., \gamma_j^{(\phi)}) = \mathsf{CRT}(r_j) \quad \forall j \in [\mu]$.
If $f(\mathbf{r}) = g(\mathbf{r})$, then $\mathsf{CRT}(f)(\mathbf{r}) = \mathsf{CRT}(g)(\mathbf{r})$. In other words, the following holds:

$$\left(F_1(\gamma_1^{(1)}, \ldots, \gamma_\mu^{(1)}), \ldots, F_\phi(\gamma_1^{(\phi)}, \ldots, \gamma_\mu^{(\phi)})\right) = \left(G_1(\gamma_1^{(1)}, \ldots, \gamma_\mu^{(1)}), \ldots, G_\phi(\gamma_1^{(\phi)}, \ldots, \gamma_\mu^{(\phi)})\right).$$

For any $j \in [\phi]$, the probability that $F_j(\gamma_1^{(j)}, \ldots, \gamma_\mu^{(j)}) = G_j(\gamma_1^{(j)}, \ldots, \gamma_\mu^{(j)})$ is at most $d/|\mathbb{F}_{q^e}| = \frac{d}{q^e}$ by Lemma 3.10. Thus, the probability that $F_j(\gamma_1^{(j)}, \ldots, \gamma_\mu^{(j)}) = G_j(\gamma_1^{(j)}, \ldots, \gamma_\mu^{(j)})$ for all $j \in [\phi]$ is at most $\frac{d}{q^e}$, where $F_j = G_j$ for all $j \in [\phi] \setminus \{k\}$ and $F_k \neq G_k$. Since $f \neq g$, there exists at least one index k such that $F_k \neq G_k$. □

3.3 Algebraic Circuit

We define an algebraic circuit over a cyclotomic ring R_q that comprises addition, multiplication, and automorphism gates.

Definition 3.18 (Arithmetic Circuit)**.** An arithmetic circuit C over a field $\mathbb{F}$ with the set of variables $X = \{x_1, \ldots, x_n\}$ is a directed acyclic graph. Nodes with in-degree 0 are called inputs and are labelled by either a variable in X or an element in $\mathbb{F}$. The remaining nodes are gates labelled either + for addition or $\times$ for multiplication. When each gate of the circuit C has in-degree k, we call C a fan-in k circuit.

Definition 3.19 (Algebraic Circuit over a ring R_q with Automorphism Gates)**.** An algebraic circuit C over a cyclotomic ring R_q is an algebraic circuit with input that are either variables or a ring element from R_q. Its gates can be either addition, multiplication, or automorphism gates. If C is a fan-in k circuit, an automorphism gate takes k inputs and outputs the evaluation of an automorphism ψ applied on the first input. The remaining inputs are dummy inputs.

3.4 Arguments of Knowledge.

We first define an indexed relation, which extends a standard relation by incorporating the index composed of circuit-dependent objects that are preprocessed in the offline phase. Afterwards, we provide the definitions of interactive proofs, arguments, and SNARKs for an indexed relation.

Definition 3.20 (Indexed Relation)**.** An indexed relation R for a circuit C is a set of tuples $(\mathtt{i}, \mathtt{x}, \mathtt{w})$, where $\mathtt{i}$ is the index, $\mathtt{x}$ is the instance, and $\mathtt{w}$ is the witness [CHM+20]. The index $\mathtt{i}$ comprises circuit-dependent objects, which can be computed even before $\mathtt{x}$ is known (offline phase). In an arithmetic circuit, $\mathtt{i}$ can be the description of the circuit, $\mathtt{x}$ can be a partial assignment to gates in the circuit, $\mathtt{w}$ can be the assignment of values to all gates in the circuit that makes all the gates consistent. The corresponding language $L(R)$ is defined as $L(R) = \{(\mathtt{i}, \mathtt{x}) \mid \exists \mathtt{w}\ s.t.\ (\mathtt{i}, \mathtt{x}, \mathtt{w}) \in R)\}$.

Interactive proofs/Arguments for indexed relations are referred to as holographic interactive proofs or arguments if an algorithm, known as the indexer, computes the oracles for the verifier's input (instance), and the verifier has an oracle access to the input rather than receiving it explicitly [CHM+20]. In this work, interactive proofs and arguments are defined in the holographic setting.

Definition 3.21 (Interactive Proof (IP))**.** An interactive proof system (IP) for an indexed relation R with the completeness error $\epsilon_c : \mathbb{N} \to [0,1]$ and the soundness error $\epsilon_s : \mathbb{N} \to [0,1]$ is an interactive protocol $\Pi = (\mathsf{Setup}, \mathcal{I}, P, V)$. In this protocol:

- The randomised algorithm $\mathsf{Setup}(1^\lambda)$ generates parameters gp,
- The indexer $\mathcal{I}(\mathsf{gp}, \mathbb{i})$ deterministically produces public parameters pp and vp for the prover and verifier during the preprocessing phase, which serve as oracles for objects in $\mathbb{i}$,
- The prover $P(\mathsf{pp}, \mathbb{x}, \mathbb{w})$ is a randomised algorithm that generates a proof π,
- The verifier $V(\mathsf{vp}, \mathbb{x}, \pi)$ is a determinitic algorithm that outputs 0 or 1.

The output of the verifier is a random variable denoted by $\langle P(pp, \mathbb{x}, \mathbb{w}), V(vp, \mathbb{x})\rangle$. The entire sequence of messages exchanged between the prover and verifier is referred to as the transcript, denoted by $t = (m_1, ..., m_k)$. An interactive oracle proof satisfies the following properties:

- Completeness: For all $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$,

$$\Pr\left[\langle P(\mathsf{pp}, \mathbb{x}, \mathbb{w}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \end{array}\right] \geq 1 - \epsilon_c(|\mathbb{i}| + |\mathbb{x}|).$$

- (Non-)Adaptive Soundness:
  - Non-Adaptive Case: For all pairs of PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ and for all $(\mathbb{i}, \mathbb{x}) \notin L(R)$, the following holds,

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathbb{i}, \mathbb{x}, \mathsf{st}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \\ \wedge\ (\mathbb{i}, \mathbb{x}) \notin L(R) \end{array} \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \\ \mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{gp}) \end{array}\right] \leq \epsilon_s(|\mathbb{i}| + |\mathbb{x}|).$$

  - Adaptive Case: For all pairs of PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, the following holds,

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathbb{i}, \mathbb{x}, \mathsf{st}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \\ \wedge\ (\mathbb{i}, \mathbb{x}) \notin L(R) \end{array} \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \\ (\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{gp}) \end{array}\right] \leq \epsilon_s(|\mathbb{i}| + |\mathbb{x}|).$$

Adaptive soundness is stronger than non-adaptive soundness as it allows the verifier to choose the index $\mathbb{i}$ and instance $\mathbb{x}$ adaptively after the preprocessing phase, whereas the instance is fixed before the preprocessing phase in the non-adaptive soundness setting. When we say soundness, we refer to the adaptive soundness in this paper. An interactive proof is **public-coin** if every verifier's message is randomly sampled from some public distribution and statistically independent of everything else.

Definition 3.22 (Argument)**.** An argument system for an indexed relation R is an interactive proof system in which its soundness is guaranteed only against adversaries that run in polynomial time. This type of system is also known as computationally sound proof.

Definition 3.23 (Argument of Knowledge)**.** An argument of knowledge for an indexed relation R is an argument system $\Pi = (\mathsf{Setup}, \mathcal{I}, P, V)$ that satisfies knowledge soundness. Knowledge soundness extends the definition of soundness by additionally ensuring that if the prover convinces the verifier, then the prover knows the witness. The protocol satisfies the following properties:

- Completeness: for all $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$,

$$\Pr\left[\langle P(\mathsf{pp}, \mathbb{x}, \mathbb{w}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \end{array}\right] \geq 1 - \epsilon_c(|\mathbb{i}| + |\mathbb{x}|).$$

- Knowledge Soundness (Rewinding): There exists a polynomial-time algorithm called extractor $\mathcal{E}$ such that, given access to any pair of PPT adversarial prover algorithms $(\mathcal{A}_1, \mathcal{A}_2)$, the following holds,

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathbb{i}, \mathbb{x}, \mathsf{st}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \\ \wedge\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin L(R) \end{array} \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \\ (\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{gp}) \\ \mathbb{w} \leftarrow \mathcal{E}^{\mathcal{A}_1, \mathcal{A}_2}(\mathsf{gp}, \mathbb{i}, \mathbb{x}) \end{array}\right] \leq \epsilon_\kappa(|\mathbb{i}| + |\mathbb{x}|).$$

If the extractor $\mathcal{E}$ can learn the witness by querying the oracle for the prover without the prior knowledge of the witness, it implies that the prover must know the witness. Knowledge soundness implies soundness.

Definition 3.24 (Non-interactive Argument)**.** An interactive public-coin argument system can be transformed into a non-interactive one using the Fiat-Shamir transform [FS87]. The transformation is achieved by replacing the verifier's randomly sampled message at the i^{th} round with the evaluation of a random oracle at a point, which is the list of the messages sent by the prover up to the i^{th} round.

Definition 3.25 (Succinct Argument)**.** An argument system $\Pi = (\mathsf{Setup}, \mathcal{I}, P, V)$ for an indexed relation R is succinct, if for all $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$, the communication complexity between P and V (i.e., the size of the transcript) is at most $o(|\mathbb{w}|) \cdot \mathsf{poly}(\lambda)$ and the verification time is at most $o(\mathsf{time}_R(\mathbb{x}, \mathbb{w})) \cdot \mathsf{poly}(\lambda)$, where $\mathsf{time}_R(\mathbb{x}, \mathbb{w})$ is the time required to decide the relation.

In some definitions of succinctness, the verifier's run-time is not required to be succinct.

Definition 3.26 (Succinct Non-interactive Argument of Knowledge)**.** A succinct non-interactive argument of knowledge (**SNARK**) is a non-interactive argument of knowledge that runs in a succinct manner.

3.5 Polynomial IOP.

Interactive oracle proofs (IOPs) combine elements of an interactive proof and a probabilistically checkable proof (PCP) [BCS16]. The verifier has oracle access to the messages sent by the prover rather than reading the messages in full. They serve as a fundamental component in the construction of SNARKs. A polynomial IOP (PIOP) is an IOP in which a proof string consists of a polynomial. PIOPs are often used in the construction of SNARKs, combined with a polynomial commitment scheme.

Definition 3.27 (Interactive Oracle Proof (IOP))**.** An interactive oracle proof (IOP) for an indexed relation R is a public-coin interactive proof system (P, V) that operates as follows. Given $(\mathbb{x}, \mathbb{w})$ as input for P and $\mathbb{x}$ as input for V, they interact over m rounds. In each round, the prover P sends a proof string Π_i, and the verifier V responds with a ramdomly sampled challenge message α_i in each round $i \in [m]$. The verifier can query the received proof strings at any location during the interaction. After the interaction, V accepts or rejects based on the instance $\mathbb{x}$, V's own randomness, and the response to queries made to the proof strings. An IOP satisfies completeness and knowledge soundness:

- Completeness: for all $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$,

$$\Pr\left[\langle P(\mathsf{pp}, \mathbb{x}, \mathbb{w}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \end{array}\right] \geq 1 - \epsilon_c(|\mathbb{i}| + |\mathbb{x}|).$$

- Knowledge Soundness (Rewinding): There exists a polynomial-time algorithm called extractor $\mathcal{E}$ such that, given access to any pair of PPT adversarial prover algorithms $(\mathcal{A}_1, \mathcal{A}_2)$, the following holds,

$$\Pr\left[\begin{array}{c} \langle \mathcal{A}_2(\mathbb{i}, \mathbb{x}, \mathsf{st}), V(\mathsf{vp}, \mathbb{x})\rangle = 1 \\ \wedge\ (\mathbb{i}, \mathbb{x}, \mathbb{w}) \notin L(R) \end{array} \;\middle|\; \begin{array}{c} \mathsf{gp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pp}, \mathsf{vp}) \leftarrow \mathcal{I}(\mathsf{gp}, \mathbb{i}) \\ (\mathbb{i}, \mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{gp}) \\ \mathbb{w} \leftarrow \mathcal{E}^{\mathcal{A}_1, \mathcal{A}_2}(\mathsf{gp}, \mathbb{i}, \mathbb{x}) \end{array}\right] \leq \epsilon_\kappa(|\mathbb{i}| + |\mathbb{x}|).$$

Definition 3.28 (Polynomial IOP (PIOP) with Preprocessing)**.** A polynomial interactive oracle proof (PIOP) with preprocessing (holographic PIOP) for an indexed relation R is a public-coin interactive proof system $(\mathcal{I}, P, V)$ that operates as follows. During the preprocessing phase, the indexer $\mathcal{I}$ takes $\mathbb{i}$ as input and generates a list of oracles to polynomials. Given $(\mathbb{x}, \mathbb{w})$ as input for P and $\mathbb{x}$ as input for V, they interact over m rounds. In each round $i \in [m]$, the prover P sends an oracle to a polynomial f_i, and the verifier V responds with a randomly sampled challenge message α_i. The verifier can query the received polynomials at any point r_i and receive the response $f_i(r_i)$ from the oracle. After the interaction, V accepts or rejects based on the instance $\mathbb{x}$, V's own randomness, and the response to queries made to the polynomials. We denote an oracle to a polynomial f by $[[f]]$.

Soundness and Knowledge Soundness of PIOPs

The following lemma states that if a PIOP satisfies soundness, then it also holds knowledge soundness.

Lemma 3.12 (Sound PIOPs are Knowledge Sound [CBBZ23])**.** Consider σ-sound PIOP for an indexed relation R such that, for all $(\mathbb{i}, \mathbb{x}, \mathbb{w})$, $\mathbb{w}$ consists of only polynomials and $\mathbb{x}$ contains oracles to these polynomials. Then, the PIOP is knowledge sound

Proof. We show that there exists an extractor that outputs $\mathbb{w}^*$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}^*) \in R$ if and only if $(\mathbb{i}, \mathbb{x}) \in L(R)$, which implies that the soundness error is the same as the knowledge error. The extractor queries each oracle to a μ-variate polynomial with degree d in each variable at $(d+1)^\mu$ points to obtain the polynomial inside the oracle. If $(\mathbb{i}, \mathbb{x}, \mathbb{w}^*) \in R$, then $(\mathbb{i}, \mathbb{x}) \in L(R)$ by the definition of the indexed relation. Suppose $(\mathbb{i}, \mathbb{x}) \in L(R)$ but $(\mathbb{i}, \mathbb{x}, \mathbb{w}^*) \notin R$ towards contradiction. Then, there must exist $\mathbb{w}^\dagger \neq \mathbb{w}^*$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}^\dagger) \in R$. However, there cannot be two distinct μ-variate polynomials with degree d that agree on $(d+1)^\mu$ points. Therefore $\mathbb{w}^* = \mathbb{w}^\dagger$ which is a contradiction. The extractor outputs the unique witness for each $(\mathbb{i}, \mathbb{x}) \in L(R)$, thus the soundness and the knowledge soundness are the same. □

4 Base PIOPs over a Cyclotomic Ring R_q

In this section, we presented a list of Polynomial IOPs (PIOPs) that serve as building blocks of our primary PIOP, Onigoroshi, detailed in Section 5. All the PIOPs support the relation defined by any arithmetic circuit over the cyclotomic ring R_q. We adapted the existing PIOPs—Sum-Check, Zero-Check, Product-Check, Multiset-Check, and Permutation-Check, which are used in HyperPlonk [CBBZ23]—to operate over the ring R_q instead of a finite field. Moreover, we introduce a new PIOP termed Automorphism-Check, designed to verify the consistency between input and output under an automorphism $\psi \in \mathsf{Aut}(R_q)$. All the PIOPs discussed in this section can ultimately be reduced to the Sum-Check PIOP.

We denote the boolean hypercube by $B_\mu = \{0,1\}^\mu \subseteq \mathbb{F}^\mu$. A polynomial $f \in R_q^{(\leq d)}[X^\mu]$ is a μ-variate polynomial over R_q with degree in each variable at most d, and $[[f]]$ is the oracle to the polynomial f. In this work, we require that any polynomial $f \in R_q^{(\leq d)}[X^\mu]$ can be expressed as

$$f(\mathbf{X}) := h(\tau_1(\mathbf{X}), \ldots, \tau_c(\mathbf{X})),$$

where h is a $c = O(1)$ variate polynomial, τ_i is a multilinear polynomial for all $i \in [c]$, and f can be evaluated using oracle access to c many multilinear polynomials $(\tau_1, \ldots, \tau_c)$ and the description of h.

4.1 Sum-Check PIOP

The sum-check protocol was first introduced in [LFKN90]. In the protocol, the prover convinces the verifier that the sum of the evaluations of a multilinear polynomial over a hypercube equals a specified value in R_q.

Definition 4.1 (Sum-Check relation over R_q)**.** The relation R_{sum} is the set of tuples $(\mathbb{x}; \mathbb{w}) = ((c, [[f]]); f)$, where $f \in R_q^{(\leq d)}[X^\mu]$ and $\sum_{\mathbf{b} \in B_\mu} f(\mathbf{b}) = c$

Construction

The prover and the verifier have μ round interactions in the protocol. In each round, given a μ-variate polynomial $f \in R_q^{(\leq d)}[X^\mu]$, the prover computes a univariate polynomial and sends the verifier the oracle to the univariate polynomial (See Figure 2). The verifier responds with a random challenge point for the univariate polynomial. At the end of the protocol, the verifier checks the consistency between the multivariate polynomial f and univariate polynomials sent by the prover by querying the oracle to f.

Theorem 4.1. The sum-check PIOP for the relation R_{sum} is perfectly complete and it has knowledge error $\epsilon_{sum}^{d,\mu} := \frac{\mu d}{q^e}$.

Proof. We modified the proof in [T+22] and adapted to R_q setting.

Sum-Check Protocol

- (The first round) The prover computes a univariate polynomial of degree d

$$g_1(X) := \sum_{\mathbf{b} \in B_{\mu-1}} f(X, \mathbf{b})$$

  and send the oracle $[[g_1]]$ to the verifier.

- The verifier checks that $c = g_1(0) + g_1(1)$. Reject otherwise.

- Then the verifier samples a random value $r_1 \leftarrow R_q$, sends it to the prover, and sets $c_1 \leftarrow g_1(r_1)$.

- **for** $i = 2, ..., \mu$:

  - The prover computes a univariate polynomial

$$g_i(X) := \sum_{\mathbf{b} \in B_{\mu-i}} f(r_1, ..., r_{i-1}, X, \mathbf{b})$$

    and send the oracle $[[g_i]]$ to the verifier.

  - The verifier checks that $c_{i-1} = g_i(0) + g_i(1)$. Reject otherwise.

  - Then the verifeir samples a random value $r_i \leftarrow R_q$, sends it to the prover, and sets $c_i \leftarrow g_i(r_i)$.

- The verifier evaluates $f(r_1, ..., r_\mu)$ by querying the oracle $[[f]]$ and accepts if

$$c_\mu = g_\mu(r_\mu) = f(r_1, ..., r_\mu).$$

Figure 2: Sum-Check Protocol

Completeness

If the prover sends the polynomials $g_i(X)$ defined in the protocol for every i round, then the verifier accepts with probability 1.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12. If $c \neq \sum_{\mathbf{b} \in B_\mu} f(\mathbf{b})$, then there has to be at least one round i, where the prover sends the verifier a univariate polynomial $h_i(X)$ such that

$$h_i(X) \neq g_i(X) := \sum_{\mathbf{b} \in B_{\mu-i}} f(r_1, ..., r_{i-1}, X, \mathbf{b})$$

and yet $h_i(r_i) = g_i(r_i)$. The probability that $h_i(r_i) = g_i(r_i)$ is at most $d/|\mathbb{F}_{q^e}| = \frac{d}{q^e}$ by Lemma 3.11. Thus, the probability that the prover sends a univariate polynomial $h_i(X) \neq g_i(X)$ yet $h_i(r_i) = g_i(r_i)$ in round i is at most $\frac{d}{q^e}$. Therefore, the probability that the prover sends a univariate polynomial $h_i(X) \neq g_i(X)$ yet $h_i(r_i) = g_i(r_i)$ at any round is at most $\sum_{j=1}^{\mu} \frac{d}{q^e} = \mu \cdot \frac{d}{q^e}$ by union bound over μ rounds. □

Complexity Analysis

The prover is required to compute μ univariate polynomials in the sum-check protocol. We adapt the dynamic programming algorithm in [Tha13, XZZ+19, CBBZ23] to the ring R_q setting, achieving the prover time linear in 2^μ (See Algorithm 1). The Algorithm 1 takes $(h, \tau_1, \ldots, \tau_c)$ as inputs instead of $f \in R_q^{(\leq d)}[X^\mu]$ such that $f(\mathbf{X}) := h(\tau_1(\mathbf{X}), \ldots, \tau_c(\mathbf{X}))$ and outputs univariate polynomials $g_1, \ldots, g_\mu$. h is a $c = O(1)$ variate polynomial with degree d, τ_i is a multilinear polynomial for all $i \in [c]$, and f can be evaluated using oracle access to c many multilinear polynomials $(\tau_1, \ldots, \tau_c)$ and the description of h. h can be evaluated through an arithmetic circuit with $O(d)$ gates as its number of coefficients is $(d+1)^c = O(d)$.

Algorithm 1: Computing $g_1, \ldots, g_\mu$ [Tha13, XZZ+19]

Data: $(h, \tau_1, \ldots, \tau_c)$
Result: $g_1, \ldots, g_\mu$
For each τ_j computes the table $A_j : \{0,1\}^\mu \to R_q$ for all evaluations over B_μ;
for $i = \mu \ldots 1$ **do**
  For each $\mathbf{b} \in B_{\mu-1}$ and $j \in [c]$, set
  $g^{(j,\mathbf{b})} := (1 - X)A_j[\mathbf{b}, 0] + X \cdots A_j[\mathbf{b}, 1]$;
  Compute $g_{\mathbf{b}} \leftarrow h(g^{(1,\mathbf{b})}(X), \ldots, g^{(c,\mathbf{b})}(X))$;
  $g_i \leftarrow \sum_{\mathbf{b} \in B_{i-1}} g_{\mathbf{b}}(X)$;
  Send $[[g_i]]$ to V;
  Receive a randomly sampled value r_i from V;
  Set $A_j[\mathbf{b}] \leftarrow g^{(j,\mathbf{b})}(r_i)$ for each $\mathbf{b} \in B_{\mu-1}$;
end

Since h has degree d, $g_{\mathbf{b}} \leftarrow h(g^{(1,\mathbf{b})}(X), \ldots, g^{(c,\mathbf{b})}(X))$ can be computed by evaluating h at $d+1$ distinct points (e.g., $\{0, 1, \ldots, d\}$) and interpolating the evaluations. Evaluating h at $d+1$ points is equivalent to evaluating the circuit with $O(d)$ gates on $d+1$ inputs, which takes $O(d^2)$ time. Furthermore, the evaluation of $g^{(j,\mathbf{b})}$ at $d+1$ distinct points takes $d+1$ steps. Consequently, the total amount of time to evaluate $g^{(j,\mathbf{b})}$ for all $j \in [c]$ at $d+1$ distinct points is $c \cdot (d+1)$. Assuming $c \approx d$, computing $g_{\mathbf{b}}$ takes $O(d^2)$ in total and computing g_i takes $\sum_{\mathbf{b} \in B_{i-1}} O(d^2) = O(2^{i-1} \cdot d^2)$. Therefore, the total amount of time to run the Algorithm 1 is

$$\sum_{i=1}^{\mu} O\left(2^{i-1} \cdot d^2\right) = O\left((2^\mu - 1) \cdot d^2\right)$$
$$= O\left(2^\mu d^2\right).$$

The verifier time is $O(\mu)$ because, in each i^{th} round, the verifier responds with a randomly sampled value $r_i \in R_q$ and queries the oracle $[[g_i]]$ at 3 points $\{0, 1, r_i\}$. This results in a total time $\sum_{i=1}^{\mu} O(1) = O(\mu)$. The proof size (the total size of oracles sent by the prover) is $\sum_{i=1}^{\mu}(d+1) = O(\mu d)$, and the witness size, which is the size of $\tau_1, \ldots, \tau_c$ (the description of h is public), is $O(c \cdot 2^{\mu}) = O(2^{\mu})$.

In summary, the complexity of the sum-check PIOP for R_{sum} with regards to $f \in R_q^{(\leq d)}[X^{\mu}]$ is presented below;

- The prover time is $\mathsf{pt}_{sum}^{f} = O\left(2^{\mu} d^2\right)$ R_q-ops.
- The verifier time is $\mathsf{vt}_{sum}^{f} = O(\mu)$.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}_{sum}^{f} = \mu$.
- The query complexity is $\mathsf{qc}_{sum}^{f} = 3\mu + 1$. The verifier queries an oracle at 3 points in each round and once at the end of the protocol.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}_{sum}^{f} = O(\mu d)$.
- The size of the witness is $O(2^{\mu})$.

4.2 Zero-Check PIOP

In the zero-check protocol, the prover convinces the verifier that, given a multivariate polynomial over R_q, the evaluation of the polynomial at any point in hypercube B_{μ} is evaluated to 0. The zero-check is built upon the sum-check protocol.

Definition 4.2 (Zero-Check relation over R_q)**.** The relation R_{zero} is the set of tuples $(\mathbb{x}; \mathbb{w}) = ([[f]], f)$, where $f \in R_q^{(\leq d)}[X^{\mu}]$ and $f(\mathbf{b}) = 0$ for all $\mathbf{b} \in B_{\mu}$.

Construction

The protocol is constructed as below (See Figure 3).

Zero-Check Protocol: Given $(\mathbb{x}; \mathbb{w}) = ([[f]], f)$,

- The verifier sends a random vector $\mathbf{r} \in R_q^{\mu}$ to the prover
- Let F be the multilinear extension of f:

$$F(\mathbf{X}) = f(\mathbf{X}) \cdot \mathsf{eq}(\mathbf{X}, \mathbf{r}),$$

  where $\mathsf{eq}(\mathbf{X}, \mathbf{Y}) := \prod_{i=1}^{\mu}(X_i Y_i + (1 - X_i)(1 - Y_i))$.

- Run the sum-check PIOP to convince prover that $(0, [[F]], F) \in R_{sum}$.

Figure 3: Zero-Check Protocol

Theorem 4.2. The sum-check PIOP for the relation R_{zero} is perfectly complete and it has knowledge error $\epsilon_{zero}^{d,\mu} := \frac{\mu d}{q^e} + \epsilon_{sum}^{d+1,\mu} = O\left(\frac{\mu d}{q^e}\right)$.

Proof. **Completeness**

For $([[f]], f) \in R_{zero}$, F is always zero over B_μ, therefore, the sum-check of F is zero. The completeness follows from that of the sum-check protocol.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12. Let $g \in R_q^{(\leq d)}[X^\mu]$ be a polynomial defined as

$$g(\mathbf{Y}) = \sum_{\mathbf{b} \in B_\mu} f(\mathbf{b}) \cdot \mathsf{eq}(\mathbf{b}, \mathbf{Y}).$$

$([[f]], f) \in R_{zero}$ holds if and only if $g(\mathbf{Y}) = 0$. The right implication is obvious. Conversely, if $g(\mathbf{Y}) = 0$, for any $\mathbf{b}^* \in B_\mu$,

$$g(\mathbf{b}^*) = \sum_{\mathbf{b} \in B_\mu} f(\mathbf{b}) \cdot \mathsf{eq}(\mathbf{b}, \mathbf{b}^*) \quad = f(\mathbf{b}^*) = 0$$

because $\mathsf{eq}(\mathbf{b}, \mathbf{b}^*) = 1$ if $\mathbf{b} = \mathbf{b}^*$ and 0 otherwise. This implies that for any $\mathbf{b}^* \in B_\mu$, $f(\mathbf{b}^*) = 0$.
Thus, for all $([[f]], f) \notin R_{zero}$, the polynomial g is a non-zero polynomial with total degree μ. By Lemma 3.11,

$$\Pr_{\mathbf{r} \leftarrow\$ R_q^\mu}[g(\mathbf{r}) = 0] \leq \frac{\mu}{q^e}.$$

Therefore, the probability that the verifier accepts $[[f]] \notin \mathcal{L}(R_{sum})$ is at most $\frac{\mu}{q^e}$ in addition to the probability that the sum-check protocol accepts when $(0, [[F]], F) \notin R_{sum}$, which is $\frac{\mu}{q^e} + \epsilon_{sum}^{d,\mu}$. □

Complexity Analysis

The complexity of the zero-check PIOP for R_{zero} is presented below;

- The prover time is $\mathsf{pt}_{zero}^f = \mathsf{pt}_{sum}^F = O(2^\mu d^2)$ R_q-ops.
- The verifier time is $\mathsf{vt}_{zero}^f = O(\mu)$.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}_{zero}^f = \mathsf{rc}_{sum}^F = \mu$.
- The query complexity is $\mathsf{qc}_{zero}^f = \mathsf{qc}_{sum}^F = 3\mu + 1$.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}_{zero}^f = \mathsf{pl}_{sum}^F = O(\mu d)$.
- The size of the witness is $O(2^\mu)$.

4.3 ProductCheck PIOP

In the product-check, the prover convinces the verifier that, given a rational function, the product of the evaluations of the rational function over a hypercube takes a specified value in R_q. The product check is built upon the zero-check protocol.

Definition 4.3 (Product-Check relation over R_q)**.** The relation R_{prod} is the set of tuples $(\mathbb{x}; \mathbb{w}) = \big((t, [[f_1]], [[f_2]]); f_1, f_2\big)$, where $f_1, f_2 \in R_q^{(\leq d)}[X^\mu]$, $f_2(\mathbf{b}) \in R_q^\times$ for all $\mathbf{b} \in B_\mu$, and $\prod_{\mathbf{b}\in B_\mu}(f_1/f_2)(\mathbf{b}) = t$. $R_q^\times$ contains all the invertible elements in R_q. In the case where $f_2 = c$ for some constant c, the relation is $(\mathbb{x}; \mathbb{w}) = \big((t, [[f]]); f\big)$ for $f = f_1/c$.

The following lemma is required in the construction of the product-check PIOP.

Lemma 4.1 ([SL20], §5)**.** Let $f_1, f_2 \in R_q[X^\mu]$. Then $t = \prod_{\mathbf{b}\in B_\mu}(f_1/f_2)(\mathbf{b})$ if and only if there exists a multivariate polynomial $v \in R_q[X^{\mu+1}]$ such that:

$$
\begin{aligned}
&v(1, ..., 1, 0) = t,\\
&v(0, \mathbf{b}) = \frac{f_1}{f_2}(\mathbf{b}) \quad \text{and} \quad v(1, \mathbf{b}) = v(\mathbf{b}, 0) \cdot v(\mathbf{b}, 1) \qquad \forall \mathbf{b} \in B_\mu.
\end{aligned}
$$

Proof. We define v as follow to prove the forward implication.

$$
v(1, ..., 1) = 0 \quad \text{and} \quad \forall l \in [\mu],\ \forall \mathbf{b} \in B_{\mu-l}, \quad v(1^l, 0, \mathbf{b}) = \prod_{\mathbf{c}\in B_l}(f_1/f_2)(\mathbf{b}, \mathbf{c}).
$$

Then,

$$
\begin{aligned}
v(0, \mathbf{b}) = \prod_{\mathbf{c}\in B_0}(f_1/f_2)(\mathbf{b}, \mathbf{c}) = (f_1/f_2)(\mathbf{b}) \quad &\text{if } l = 0\\
v(1^l, 0) = \prod_{\mathbf{c}\in B_\mu}(f_1/f_2)(\mathbf{c}) = t \quad &\text{if } l = \mu,
\end{aligned}
$$

When $l > 0$, $\forall \mathbf{b}' \in B_{\mu-l}$,

$$
\begin{aligned}
v(1^l, 0, \mathbf{b}') &= \prod_{\mathbf{c}\in B_l}(f_1/f_2)(\mathbf{b}', \mathbf{c})\\
&= \prod_{\mathbf{c}\in B_{l-1}}(f_1/f_2)(\mathbf{b}', 0, \mathbf{c}) \cdot \prod_{\mathbf{c}\in B_{l-1}}(f_1/f_2)(\mathbf{b}', 1, \mathbf{c})\\
&= v(1^{l-1}, 0, \mathbf{b}', 0) \cdot v(1^{l-1}, 0, \mathbf{b}', 1).
\end{aligned}
$$

Thus, by setting $\mathbf{b} = (1^{l-1}, 0, \mathbf{b}')$: $v(1, \mathbf{b}) = v(\mathbf{b}, 0) \cdot v(\mathbf{b}, 1)$ for all $\mathbf{b} \in B_\mu \setminus \{1^\mu\}$. When $\mathbf{b} = 1^\mu$, $v(1^{\mu+1}) = 0 = v(1^\mu, 0) \cdot v(1^{\mu+1})$. Thus, such v exists.

Next, we prove the reverse implication. We have, by induction, for $l = 0, ..., \mu$,

$$
v(1^l, 0, \mathbf{b}) = \prod_{\mathbf{c}\in B_l} v(0, \mathbf{b}, \mathbf{c})\ \forall \mathbf{b} \in B_{\mu-l}. \tag{9}
$$

The proof of the statement above is given in the end. Now we take $l = \mu$, then $t = v(1^l, 0) = \prod_{\mathbf{c}\in B_\mu} v(0, \mathbf{c}) = \prod_{\mathbf{c}\in B_\mu}(f_1/f_2)(\mathbf{c})$.

We provide the proof of the equation (9) by induction below.
(Base case) When $l = 0$, $v(1^0, 0, \mathbf{b}) = v(0, \mathbf{b})$ and $\prod_{\mathbf{c}\in B_0} v(0, \mathbf{b}, \mathbf{c}) = v(0, \mathbf{b})$.
(Induction Steps) Suppose the equation (9) holds for $l = k$:

$$v(1^k, 0, \mathbf{b}) = \prod_{\mathbf{c}\in B_k} v(0, \mathbf{b}, \mathbf{c}) \ \forall \mathbf{b} \in B_{\mu-k}.$$

For $\mathbf{b} \in B_{\mu-(k+1)}$,

$$\begin{aligned} v(1^{k+1}, 0, \mathbf{b}) &= v(1, 1^k, 0, \mathbf{b}) \\ &= v(1^k, 0, \mathbf{b}, 0) \cdot v(1^k, 0, \mathbf{b}, 1) \quad (\because v(1, \mathbf{b}) = v(\mathbf{b}, 0) \cdot v(\mathbf{b}, 1)) \\ &= \prod_{\mathbf{c}\in B_k} v(0, \mathbf{b}, 0, \mathbf{c}) \cdot \prod_{\mathbf{c}\in B_k} v(0, \mathbf{b}, 1, \mathbf{c}) \\ &= \prod_{\mathbf{c}\in B_{k+1}} v(0, \mathbf{b}, \mathbf{c}). \end{aligned}$$

□

Construction

By Lemma 4.1, there exists a polynomial $v \in R_q^{(\leq 1)}[X^{\mu+1}]$ such that its evaluation at point $(1^\mu, 0)$ equals the product of the evaluations of the rational function f' over B_μ. The prover convinces the verifier that the polynomial v was correctly computed according to its definition by performing the zero-check, as described in 4.2. At the end of the protocol, the verifier checks if the evaluation of v at point $(1^\mu, 0)$ matches as the desired value by querying the oracle $[[v]]$. Figure 4 provides a more detailed description of the protocol.

Theorem 4.3. The product-check PIOP for the relation R_{prod} is perfectly complete, and it has knowledge error $\epsilon_{prod}^{d,\mu} := \epsilon_{zero}^{d',\mu} = O\left(\frac{\mu d'}{q^e}\right)$, where $d' = \max(2, d + 1)$.

Proof. **Completeness**

If the prover generates v honestly,

$$v(0, \mathbf{b}) - f'(\mathbf{b}) = 0, \quad v(1, \mathbf{b}) - v(\mathbf{b}, 0) \cdot v(\mathbf{b}, 1) = 0 \quad \forall \mathbf{b} \in B_\mu$$

Thus, $\hat{f}(\mathbf{b}) = 0$ and $\hat{g}(\mathbf{b}) = 0$ for all $\mathbf{b} \in B_\mu$. Therefore, $([[\hat{h}]], \hat{h}) \in R_{zero}$, and the verifier accepts in the zero-check. By Lemma 4.1, if $\left((t, [[f_1]], [[f_2]],); f_1, f_2\right) \in R_{prod}$, $v(1, ..., 1, 0) = \prod_{\mathbf{b}\in B_\mu}(f_1/f_2)(\mathbf{b}) = t$.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12. For any $\left((t, [[f_1]], [[f_2]]), f_1, f_2\right) \notin R_{prod}$ and any v sent by a dishonest prover, it is either that $v(1, ..., 1, 0) \neq t$ and the verifier rejects or v is computed incorrectly, i.e., $([[\hat{h}]], \hat{h}) \notin R_{zero}$. Therefore, the probability that the verifier accepts is $\max(0, \epsilon_{zero}^{d',\mu+1}) = \epsilon_{zero}^{d',\mu+1}$. □

Product-Check Protocol: Given $(\mathbb{x}; \mathbb{w}) = \big((t, [[f_1]], [[f_2]]); f_1, f_2\big)$, we denote $f' := f_1/f_2$.

- The prover sends the oracle to $v \in R_q^{(\leq 1)}[X^{\mu+1}]$ such that

$$v(0, \mathbf{b}) = f'(\mathbf{b}), \quad v(1, \mathbf{b}) = v(\mathbf{b}, 0) \cdot v(\mathbf{b}, 1) \quad \forall \mathbf{b} \in B_\mu.$$

- Define $\hat{h} \in R_q^{(\leq \max(2,d+1))}[X^{\mu+1}]$ such that

$$\hat{h}(X_0, ..., X_{\mu+1}) := (1 - X_0) \cdot \hat{f}(X_1, ..., X_\mu) + X_0 \cdot \hat{g}(X_1, ..., X_\mu),$$

  where

$$\hat{f}(\mathbf{X}) := v(1, \mathbf{X}) - v(\mathbf{X}, 0) \cdot v(\mathbf{X}, 1), \quad \hat{g}(\mathbf{X}) := f_2(\mathbf{X}) \cdot v(0, \mathbf{X}) - f_1(\mathbf{X}).$$

  Run the Zero-Check PIOP (Section 4.2) for $([[\hat{h}]], \hat{h}) \in R_{zero}$.

- The verifier queries $[[v]]$ at point $(1, ...1, 0) \in B_{\mu+1}$ and checks if the following holds:

$$v(1, ..., 1, 0) = t.$$

Figure 4: Product-Check Protocol

Complexity Analysis

The complexity of the product-check PIOP for R_{prod} is presented below;

- The prover time is $\mathsf{pt}^{f'}_{prod} = \mathsf{pt}^{\hat{h}}_{zero} + 2^\mu = O\big(2^\mu d^2\big)$ R_q-ops. The term 2^μ is the steps for computing v.
- The verifier time is $\mathsf{vt}^{f'}_{prod} = O(\mu)$.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}^{f'}_{prod} = \mathsf{rc}^{\hat{h}}_{zero} + 1 = \mu + 1$.
- The query complexity is $\mathsf{qc}^{f'}_{prod} = \mathsf{qc}^{\hat{h}}_{zero} + 1 = 3\mu + 2$.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}^{f'}_{prod} = \mathsf{pl}^{\hat{h}}_{zero} + 2^\mu = O(2^\mu)$. The term 2^μ is the size of the oracle $[[v]]$.
- The size of the witness is $O(2^\mu)$.

4.4 Multiset-Check PIOP

The multiset-check is a building block of the permutation-check in Section 4.5 and is built upon the product-check in Section 4.2.

Definition 4.4 (Multiset Check relation over R_q)**.** The relation R^l_{mset} for any $l \geq 1$ is the set of tuples

$$(\mathbb{x}; \mathbb{w}) = \Big(([[f_1]], ..., [[f_l]], [[g_1]], ..., [[g_l]]); (f_1, ..., f_l, g_1, ..., g_l)\Big)$$

where $f_i, g_i \in R_q^{(\leq d)}[X^\mu]$ for $1 \leq i \leq l$ and the equality of the following two multisets of tuples holds:

$$\Big\{\big(f_1(\mathbf{b}), ..., f_l(\mathbf{b})\big)\Big\}_{\mathbf{b}\in B_\mu} = \Big\{\big(g_1(\mathbf{b}), ..., g_l(\mathbf{b})\big)\Big\}_{\mathbf{b}\in B_\mu}.$$

Construction (Step 1)

There are two steps to construct the multiset-check protocol. We first build a basic multiset-check protocol for the relation R^1_{mset} and then construct the multiset-check protocol for the general relation R^l_{mset} based on the basic protocol. In the multiset-check protocol for R^1_{mset}, the verifier sends a randomly sampled challenge, and the prover generates two polynomials based on this challenge as well as the witness polynomials provided as input. Then, they perform the product-check (Section 4.3) for these generated polynomials (See Figure 5).

Multiset-Check Protocol for R^1_{mset}: Given $(\mathbb{x}; \mathbb{w}) = \Big(([[f]], [[g]]); (f, g)\Big)$,

- The verifier samples a challenge $r \leftarrow\!\!\$\ R_q$ and sends it to the prover.
- The prover set $f' = r + f$ and $g' = r + g$ and perform the ProductCheck PIOP (Section 4.3) for $\big((1, [[f']], [[g']]); f', g'\big) \in R_{prod}$.

Figure 5: Multiset-Check Protocol for R^1_{mset}

Theorem 4.4. The multiset-check PIOP for the relation R^1_{mset} is perfectly complete, and it has knowledge error $\epsilon^{d,\mu}_{mset,1} := \frac{2^\mu}{q^e} + \epsilon^{d,\mu}_{prod} = O\Big(\frac{2^\mu + \mu\cdot\max(2,d+1)}{q^e}\Big) = O\Big(\frac{2^\mu+\mu d}{q^e}\Big)$.

Proof. **Completeness**

For any $\big(([[f]], [[g]]); (f, g)\big) \in R^1_{mset}$, the following holds:

$$\prod_{\mathbf{b}\in B_\mu} (r + f(\mathbf{b})) = \prod_{\mathbf{b}\in B_\mu} (r + g(\mathbf{b})).$$

Thus, $\prod_{\mathbf{b}\in B_\mu}(r+f(\mathbf{b}))/(r+g(\mathbf{b})) = 1$. Therefore, $\big((1, [[f']], [[g']]); f', g'\big) \in R_{prod}$. The completeness follows from that of the product-check.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12. For any $\left(([[f]],[[g]]);(f,g)\right) \notin R^1_{mset}$, the following holds:

$$F(X) := \prod_{\mathbf{b}\in B_\mu}(X + f(\mathbf{b})) \neq G(X) := \prod_{\mathbf{b}\in B_\mu}(X + g(\mathbf{b})).$$

There are two cases where the verifier still accepts.

1. For a challenge $r \leftarrow\!\$\ R_q$, $F(r) = G(r)$. Since the degree of F and G is 2^μ, the probability that the verifier accepts in this case is, by Lemma 3.11, at most $\frac{2^\mu}{q^e}$.

2. When $F(r) \neq G(r)$, $\left((1,[[r+f]],[[r+g]]);r+f,r+g\right) \notin R^{d,\mu}_{prod}$ as $\prod_{\mathbf{b}\in B_\mu}\frac{r+f(\mathbf{b})}{r+g(\mathbf{b})} \neq 1$. The probability that the verifier accepts in this case is at most $\epsilon^{d,\mu}_{prod}$.

Therefore, the probability that the verifier accepts is at most $\frac{2^\mu}{q^e} + \epsilon^{d,\mu}_{prod}$. □

Construction (Step 2)

Finally, we construct the multiset-check for the relation R^l_{mset} using the base protocol described in Figure 5 as a building block. In the protocol, the verifier sends $l-1$ randomly sampled challenges to the prover. In response, the prover generates two polynomials as linear combinations of l polynomials provided as input. Subsequently, they perform the multiset-check for the relation R^1_{mset} on these two polynomials. The construction detail is described below (See Figure 6).

Multiset-Check Protocol for R^l_{mset} for any $l \geq 1$:
Given $(\mathbb{x};\mathbb{w}) = \Big(([[f_1]],...,[[f_l]],[[g_1]],...,[[g_l]]);(f_1,...,f_l,g_1,...,g_l)\Big)$.

- The verifier samples challenges $r_2,...,r_l \leftarrow\!\$\ R_q$ and sends them to the prover.
- The prover sets $\ddot{f} = f_1 + r_2\cdot f_2 + \cdots + r_l \cdot f_l$ and $\ddot{g} = g_1 + r_2 \cdot g_2 + \cdots + r_l \cdot g_l$, and subsequently run the Multiset-Check PIOP, detailed in Figure 5, for $\left(([[\ddot{f}]],[[\ddot{g}]]);\ddot{f},\ddot{g}\right) \in R^1_{mset}$.

Figure 6: Multiset-Check Protocol for R^l_{mset}

Theorem 4.5. The multiset-check PIOP for the relation R^l_{mset} is perfectly complete, and it has knowledge error $\epsilon^{d,\mu}_{mset,l} := \frac{2^\mu(l-1)}{q^e} + \epsilon^{d,\mu}_{mset,1} = O\left(\frac{2^\mu l+\mu d}{q^e}\right)$.

Proof. **Completeness**

For any $\Big(([[f_1]],...,[[f_l]],[[g_1]],...,[[g_l]]);(f_1,...,f_l,g_1,...,g_l)\Big) \in R^l_{mset}$, the following holds:

$$\Big\{f_1(\mathbf{b})+r_2\cdot f_2(\mathbf{b})+\cdots+r_l\cdot f_l(\mathbf{b})\Big\}_{\mathbf{b}\in B_\mu} = \Big\{g_1(\mathbf{b})+r_2\cdot g_2(\mathbf{b})+\cdots+r_l\cdot g_l(\mathbf{b})\Big\}_{\mathbf{b}\in B_\mu}.$$

Therefore, $\big(([[f']],[[g']]);f',g'\big) \in R^1_{mset}$. The completeness follows from that of the multiset-check for R^1_{mset}.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12.
For any $\Big(([[f_1]],...,[[f_l]],[[g_1]],...,[[g_l]]);(f_1,...,f_l,g_1,...,g_l)\Big) \notin R^l_{mset}$,
the following holds:

$$U := \Big\{\big(f_1(\mathbf{b}),...,f_l(\mathbf{b})\big)\Big\}_{\mathbf{b}\in B_\mu} \neq V := \Big\{\big(g_1(\mathbf{b}),...,g_l(\mathbf{b})\big)\Big\}_{\mathbf{b}\in B_\mu}.$$

There are two cases where the verifier accepts under the above condition,

1. When $\big(([[\ddot{f}]],[[\ddot{g}]]);\ddot{f},\ddot{g}\big) \in R^1_{mset}$, the following holds:

   $$\Big\{f_1(\mathbf{b})+r_2\cdot f_2(\mathbf{b})+\cdots+r_l\cdot f_l(\mathbf{b})\Big\}_{\mathbf{b}\in B_\mu} = \Big\{g_1(\mathbf{b})+r_2\cdot g_2(\mathbf{b})+\cdots+r_l\cdot g_l(\mathbf{b})\Big\}_{\mathbf{b}\in B_\mu}.$$

   Let W be the maximal multiset such that $W \subseteq U$ and $W \subseteq V$. We define $U' := U \setminus W$ and $V' := V \setminus W$. Then $|U'| = |V'| > 0$ and $U' \cap V' = \emptyset$ by $U \neq V$. Hence, there exists an element $\mathbf{u} \in U'$ such that $\mathbf{u} \notin V'$. Let's define $l-1$ variate polynomial $\phi_{\mathbf{u}} \in R_q[X^{l-1}]$ such that $\phi_{\mathbf{u}}(\mathbf{X}) = u_1 + u_2\cdot X_1 + \cdots + u_l \cdot X_{l-1}$. Since the total degree of $\phi_{\mathbf{u}}$ is $l-1$, for any $\mathbf{u} \in U'$ and $\mathbf{v} \in V'$ $(\mathbf{u} \neq \mathbf{v})$, the following holds by Lemma 3.11:

   $$\Pr_{\mathbf{r}\leftarrow\$R_q^{l-1}}[\phi_{\mathbf{u}}(\mathbf{r}) = \phi_{\mathbf{v}}(\mathbf{r})] \leq \frac{l-1}{q^e}.$$

   We need to compute the probability that, given $\mathbf{r} \in R_q^{l-1}$, $\phi_{U'}(\mathbf{r}) = \phi_{V'}(\mathbf{r})$ where $\phi_{U'}(\mathbf{r}) := \{\phi_{\mathbf{u}}(\mathbf{r})\}_{\mathbf{u}\in U'}$. $\phi_{U'}(\mathbf{r}) = \phi_{V'}(\mathbf{r})$ implies that there exists $\mathbf{u} \in U'$ such that $\phi_{\mathbf{u}}(\mathbf{r}) \in \phi_{V'}(\mathbf{r})$. Together with the union bound,

   $$\begin{aligned}
   \Pr_{\mathbf{r}\leftarrow\$R_q^{l-1}}[\phi_{U'}(\mathbf{r}) = \phi_{V'}(\mathbf{r})] &\leq \Pr_{\mathbf{r}\leftarrow\$R_q^{l-1}}[\exists \mathbf{u} \in U' \text{ such that } \phi_{\mathbf{u}}(\mathbf{r}) \in \phi_{V'}(\mathbf{r})] \\
   &\leq \Pr_{\mathbf{r}\leftarrow\$R_q^{l-1}}\left[\bigcup_{\mathbf{v}\in V'}\big(\phi_{\mathbf{u}}(\mathbf{r}) = \phi_{\mathbf{v}}(\mathbf{r})\big)\right] \\
   &\leq \sum_{\mathbf{v}\in V'} \Pr_{\mathbf{r}\leftarrow\$R_q^{l-1}}[\phi_{\mathbf{u}}(\mathbf{r}) = \phi_{\mathbf{v}}(\mathbf{r})] \\
   &= \frac{|V'|(l-1)}{q^e} \leq \frac{2^\mu(l-1)}{q^e}
   \end{aligned}$$

2. When $\left(([[\ddot{f}]], [[\ddot{g}]]); \ddot{f}, \ddot{g}\right) \notin R^1_{mset}$, the probability that the verifier accepts is at most $\epsilon^{d,\mu}_{mset,1}$.

Therefore, in total, the probability that the verifier accepts is at most $\frac{2^\mu(l-1)}{q^e} + \epsilon^{d,\mu}_{mset,1}$. □

Complexity Analysis

The complexity of the multiset-check PIOP for R^l_{mset} with regards to

$$\mathbf{G} = (f_1, \ldots, f_l, g_1, \ldots, g_l)$$

is described below;

- The prover time is $\mathsf{pt}^{\mathbf{G}}_{mset} = \mathsf{pt}^{\ddot{f},\ddot{g}}_{mset,1} = \mathsf{pt}^{f'/g'}_{prod} = O\left(2^\mu d^2\right)$ R_q-ops. Evaluating $\ddot{f}$ and $\ddot{g}$ takes $l \cdot (d+1) = O(d)$ time.
- The verifier time is $\mathsf{vt}^{\mathbf{G}}_{mset} = \mathsf{vt}^{f'/g'}_{prod} = O(\mu)$.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}^{\mathbf{G}}_{mset} = \mathsf{rc}^{f'/g'}_{prod} = \mu + 1$.
- The query complexity is $\mathsf{qc}^{\mathbf{G}}_{mset} = \mathsf{qc}^{f'/g'}_{prod} = 3\mu + 2$.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}^{\mathbf{G}}_{mset} = \mathsf{pl}^{f'/g'}_{prod} = O(2^\mu)$.
- The size of the witness is $2l \cdot O(2^\mu) = O(l \cdot 2^\mu)$.

4.5 Permutation-Check PIOP

The permutation-check is a key protocol to check the consistency amongst wires in an algebraic circuit. It is built upon the multiset-check protocol described in Section 4.4.

Definition 4.5 (Permuation-Check relation over R_q)**.** The indexed relation R_{perm} is the set of tuples

$$(\mathbb{i}; \mathbb{x}; \mathbb{w}) = \left(\sigma; ([[f]], [[g]]); (f, g)\right)$$

where $\sigma : B_\mu \to B_\mu$ is a permutation, $f, g \in R_q^{(\leq d)}[X^\mu]$, and $f(\mathbf{b}) = g(\sigma(\mathbf{b}))$ for all $\mathbf{b} \in B_\mu$.

The following lemma asserts that if the equation (10) is satisfied, it necessarily implies that the permutation-check relation is also satisfied.

Lemma 4.2. Let $f, g \in R_q^{(\leq d)}[X^\mu]$ and $\sigma : B_\mu \to B_\mu$ be a permutation. Let $\iota_{id}, \iota_\sigma \in R_q^{(\leq 1)}[X^\mu]$ such that

$$\iota_{id}(\mathbf{b}) = [\mathbf{b}] := \sum_{i=1}^{\mu} \mathbf{b}_i \cdot 2^{i-1} \quad \text{and} \quad \iota_\sigma(\mathbf{b}) = \iota_{id}(\sigma(\mathbf{b})) \qquad \forall \mathbf{b} \in B_\mu.$$

Then, $g(\mathbf{b}) = f(\sigma(\mathbf{b}))$ for all $\mathbf{b} \in B_\mu$ if only and if

$$\left\{\left(\iota_{id}(\mathbf{b}), f(\mathbf{b})\right)\right\}_{\mathbf{b}\in B_\mu} = \left\{\left(\iota_\sigma(\mathbf{b}), g(\mathbf{b})\right)\right\}_{\mathbf{b}\in B_\mu}. \tag{10}$$

Proof. Let $X := \{(\iota_{id}(\mathbf{b}), f(\mathbf{b}))\}_{\mathbf{b}\in B_\mu}$ and $Y := \{(\iota_\sigma(\mathbf{b}), g(\mathbf{b}))\}_{\mathbf{b}\in B_\mu}$, where $\iota_{id}(\mathbf{b}) = [\mathbf{b}]$ and $\iota_\sigma(\mathbf{b}) = [\sigma(\mathbf{b})]$. We first prove the forward implication.
For any $([\mathbf{b}], f(\mathbf{b})) \in X$, $[\mathbf{b}] = [\sigma(\sigma^{-1}(\mathbf{b}))]$ and $f(\mathbf{b}) = g(\sigma^{-1}(\mathbf{b}))$. By the definition of Y, $\left([\sigma(\sigma^{-1}(\mathbf{b}))], g(\sigma^{-1}(\mathbf{b}))\right) \in Y$, which implies $([\mathbf{b}], f(\mathbf{b})) \in Y$. For any $([\sigma(\mathbf{b})], g(\mathbf{b})) \in Y$, $([\sigma(\mathbf{b}]), g(\mathbf{b})) \in X$ since $g(\mathbf{b}) = f(\sigma(\mathbf{b}))$. Therefore, $X = Y$. Conversely, suppose $X = Y$. For any $\mathbf{b} \in B_\mu$, if we consider $([\sigma(\mathbf{b})], g(\mathbf{b})) \in Y$, there exists $([\mathbf{b}^*], f(\mathbf{b}^*)) \in X$ for $\mathbf{b}^* \in B_\mu$ such that $([\sigma(\mathbf{b})], g(\mathbf{b})) = ([\mathbf{b}^*], f(\mathbf{b}^*))$. Thus, $[\sigma(\mathbf{b})] = [\mathbf{b}^*] \rightarrow \sigma(\mathbf{b}) = \mathbf{b}^*$ and $g(\mathbf{b}) = f(\mathbf{b}^*)$. Combined above, $g(\mathbf{b}) = f(\sigma(\mathbf{b}))$. □

Construction

Given two multivariate polynomials $f, g \in R_q^{(\leq d)}$ and a permutation $\sigma : B_\mu \rightarrow B_\mu$, we define two polynomials $\iota_{id}, \iota_\sigma \in R_q^{(\leq 1)}[X^\mu]$ such that, for any $\mathbf{b} \in B_\mu$, $\iota_{id}(\mathbf{b}) = [\mathbf{b}] := \sum_{i=1}^{\mu} \mathbf{b}_i \cdot 2^{i-1}$ and $\iota_\sigma(\mathbf{b}) = [\sigma(\mathbf{b})]$. It holds that $f(\mathbf{b}) = g(\sigma(\mathbf{b}))$ for all $\mathbf{b} \in B_\mu$ if and only if $\left\{\left(\iota_{id}(\mathbf{b}), f(\mathbf{b})\right)\right\}_{\mathbf{b}\in B_\mu} = \left\{\left(\iota_\sigma(\mathbf{b}), g(\mathbf{b})\right)\right\}_{\mathbf{b}\in B_\mu}$ by Lemma 4.2. Therefore, the permutation-check simply performs the multiset-check on these polynomials (See Figure 7).

Permuation-Check Protocol: Given $(\mathbb{i}; \mathbb{x}; \mathbb{w}) = \left(\sigma; ([[f]], [[g]]); (f, g)\right)$, the indexer generates two oracles $[[\iota_{id}]]$ and $[[\iota_\sigma]]$ for $\iota_{id}, \iota_\sigma \in R_q^{(\leq 1)}[X^\mu]$ such that $\iota_{id}(\mathbf{b}) = [\mathbf{b}]$ and $\iota_\sigma(\mathbf{b}) = [\sigma(\mathbf{b})]$ for all $\mathbf{b} \in B_\mu$.

- Run the Multiset-Check PIOP described in Section 4.4 for

$$\left(([[\iota_{id}]], [[f]], [[\iota_\sigma]], [[g]]), (\iota_{id}, f, \iota_\sigma, g)\right) \in R_{mset}^2$$

Figure 7: Permutation-Check Protocol

Theorem 4.6. The permutation-check PIOP for the relation R_{perm} is perfectly complete, and it has knowledge error $\epsilon_{perm}^{d,\mu} := \epsilon_{mset,2}^{d,\mu} = O\left(\frac{2^\mu + \mu d}{q^e}\right)$.

Proof. **Completeness**

For any $\left(\sigma; ([[f]], [[g]]); (f, g)\right) \in R_{perm}$, the following holds by Lemma 4.2:

$$\left\{\left(\iota_{id}(\mathbf{b}), f(\mathbf{b})\right)\right\}_{\mathbf{b}\in B_\mu} = \left\{\left(\iota_\sigma(\mathbf{b}), g(\mathbf{b})\right)\right\}_{\mathbf{b}\in B_\mu}.$$

Thus, $\left(([[\iota_{id}]], [[f]], [[\iota_\sigma]], [[g]]), (\iota_{id}, f, \iota_\sigma, g)\right) \in R_{mset}^2$. The completeness follows from that of the multiset-check for R_{mset}^2.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12. For any $\big(([[f]],[[g]]);(f,g)\big) \notin R_{perm}$, the knowledge error remains consistent with that of the multiset-check, and is, thus, at most $\epsilon^{d,\mu}_{mset,2}$. □

Complexity Analysis

The complexity of the permutation-check PIOP for R_{perm} with respect to

$$\mathbf{S} := (\iota_{id}, f, \iota_{\sigma}, g)$$

is presented below:

- The prover time is $\mathsf{pt}^{f,g}_{perm} = \mathsf{pt}^{\mathbf{S}}_{mset} = O\big(2^{\mu} d^2\big)$ R_q-ops.
- The verifier time is $\mathsf{vt}^{f,g}_{perm} = \mathsf{vt}^{\mathbf{S}}_{mset} = O(\mu)$.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}^{f,g}_{perm} = \mathsf{rc}^{\mathbf{S}}_{mset} = \mu + 1$.
- The query complexity is $\mathsf{qc}^{f,g}_{perm} = \mathsf{qc}^{\mathbf{S}}_{mset} = 3\mu + 2$.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}^{f,g}_{perm} = \mathsf{pl}^{\mathbf{S}}_{mset} = O(2^{\mu})$.
- The size of the witness is $O(2 \cdot 2^{\mu}) = O(2^{\mu})$.

4.6 Automorphism-Check PIOP

The automorphism-check is a newly introduced PIOP in this work that can prove the consistency between the input and output of automorphism gates in an algebraic circuit over R_q. It is built upon the sum-check protocol described in Section 4.1.

Definition 4.6 (Automorphism-Check relation over R_q)**.** The relation R_{auto} is the set of tuples $(\mathbb{i}, \mathbb{x}; \mathbb{w}) = \big((\psi, s_{\psi}); ([[f]],[[g]]); f, g\big)$, where $\psi \in \mathsf{Aut}(R_q)$ is an automorphism over R_q, $s \in R_q^{(\leq 1)}[X^{\mu}]$ is a polynomial such that:

$$s(\mathbf{b}) = \begin{cases} 1 & \text{if the gate represented by } \mathbf{b} \text{ is the } \psi\text{-automorphism gate} \\ 0 & \text{Otherwise} \end{cases},$$

$f, g \in R_q^{(\leq d)}[X^{\mu}]$, and $\psi\big(s(\mathbf{b}) \cdot f(\mathbf{b})\big) = s(\mathbf{b}) \cdot g(\mathbf{b})$ for all $\mathbf{b} \in B_{\mu}$.

Construction

In this protocol (See Figure 8), we reduce the automorphism-check relation into the following.

$$\sum_{\mathbf{b}\in\mathcal{S}_\psi}\Big(\psi\big(f(\mathbf{b})\big)-g(\mathbf{b})\Big)\cdot\mathsf{eq}(\mathbf{b},\mathbf{r})=0,$$

where $S_\psi \subseteq B_\mu$ is a set such that, for any $\mathbf{b}\in S_\psi$, the gate represented by $\mathbf{b}$ is the ψ-automorphism gate, $\mathsf{eq}(\mathbf{X},\mathbf{Y}) := \sum_{i=1}^{\mu}(X_iY_i+(1-X_i)(1-Y_i))$, and $\mathbf{r}\in R_q$ is a randomly sampled value. By the homomorphic property of ψ, we can rewrite it as:

$$\begin{aligned}&\sum_{\mathbf{b}\in\mathcal{S}_\psi}\Big(\psi\big(f(\mathbf{b})\big)\cdot\psi\Big(eq\big(\psi^{-1}(\mathbf{b}),\psi^{-1}(\mathbf{r})\big)\Big)-g(\mathbf{b})\cdot\mathsf{eq}(\mathbf{b},\mathbf{r})\Big)=0\\ \leftrightarrow\quad &\psi\left(\sum_{\mathbf{b}\in\mathcal{S}_\psi}f(\mathbf{b})\cdot eq\big(\mathbf{b},\psi^{-1}(\mathbf{r})\big)\right)-\sum_{\mathbf{b}\in\mathcal{S}_\psi}g(\mathbf{b})\cdot\mathsf{eq}(\mathbf{b},\mathbf{r})=0.\end{aligned}\tag{11}$$

Since $\mathbf{b}\in\mathbb{Z}_q$, $\psi^{-1}(\mathbf{b})=\mathbf{b}$. We have two sum-check instances in the equation (11) but both sums are taken over $\mathcal{S}_\psi$ rather than B_μ as in the sum-check protocol (Section 4.1). To address this issue, we introduced a polynomial $s_\psi\in R_q^{(\le 1)}[X^\mu]$ such that $s_\psi(\mathbf{b})=1$ if $\mathbf{b}\in\mathcal{S}_\psi$ and 0 otherwise. The modification of the equation (11) is presented below:

$$\psi\left(\sum_{\mathbf{b}\in B_\mu}s_\psi(\mathbf{b})\cdot f(\mathbf{b})\cdot\mathsf{eq}(\mathbf{b},\psi^{-1}(\mathbf{r}))\right)-\sum_{\mathbf{b}\in B_\mu}s_\psi(\mathbf{b})\cdot g(\mathbf{b})\cdot\mathsf{eq}(\mathbf{b},\mathbf{r})=0.\tag{12}$$

The prover and the verifier perform the sum-check for these two instances. Subsequently, the prover sends the evaluations of these two sum-check instances to the verifier, who then verifies the consistency of the equation (12).

Theorem 4.7. The automorphism-check PIOP for the relation R_{auto} is perfectly complete and it has knowledge error $\epsilon_{auto}^{d,\mu} := \frac{\mu}{q^e}+\epsilon_{sum}^{d,\mu}=O\left(\frac{\mu d}{q^e}\right)$.

Proof. **Completeness**

For any $\big((\psi;([[f]],[[g]]);f,g\big)\in R_{auto}$, it always holds that $\psi(a)=b$. By the homomorphic property of ψ,

$$\begin{aligned}\psi(a)&=\sum_{\mathbf{b}\in B_\mu}\psi\big(s_\psi(\mathbf{b})\big)\cdot\psi\big(f(\mathbf{b})\big)\cdot\psi(\mathsf{eq}(\mathbf{b},\psi^{-1}(\mathbf{r})))\\ &=\sum_{\mathbf{b}\in B_\mu}\psi\big(s_\psi(\mathbf{b})\big)\cdot\psi\big(f(\mathbf{b})\big)\cdot eq\big(\psi(\mathbf{b}),\psi(\psi^{-1}(\mathbf{r}))\big)\\ &=\sum_{\mathbf{b}\in B_\mu}s_\psi(\mathbf{b})\cdot\psi\big(f(\mathbf{b})\big)\cdot\mathsf{eq}(\mathbf{b},\mathbf{r})\quad\Big(\because\psi(x)=\psi\Big(\sum_{1\in[x]}1\Big)=\sum_{1\in[x]}\psi(1)=x\ \ \forall x\in\mathbb{Z}_q\Big)\\ &=\sum_{\mathbf{b}\in B_\mu}s_\psi(\mathbf{b})\cdot g(\mathbf{b})\cdot\mathsf{eq}(\mathbf{b},\mathbf{r})\\ &=b.\end{aligned}$$

Automorphism-Check Protocol: Given $(\mathbb{i}, \mathbb{x}; \mathbb{w}) = \big((\psi, s_\psi); ([[f]], [[g]]); f, g\big)$, the indexer generates oracles $[[\psi]]$ and $[[s_\psi]]$ where $s_\psi \in R_q^{\leq 1}[X^\mu]$ is a polynomial such that

$$s_\psi(\mathbf{b}) = \begin{cases} 1 & \text{if the gate represented by } \mathbf{b} \text{ is the } \psi\text{-automorphism gate} \\ 0 & \text{Otherwise} \end{cases}.$$

- The verifier sends a random challenge $\mathbf{r} \leftarrow\!\$\ R_q^\mu$ to the prover.
- The prover computes the following multilinear extensions

$$F(\mathbf{X}) = s_\psi(\mathbf{X}) \cdot f(\mathbf{X}) \cdot \mathsf{eq}(\mathbf{X}, \psi^{-1}(\mathbf{r})),$$
$$G(\mathbf{X}) = s_\psi(\mathbf{X}) \cdot g(\mathbf{X}) \cdot \mathsf{eq}(\mathbf{X}, \mathbf{r}),$$

  where $\mathsf{eq}(\mathbf{X}, \mathbf{Y}) := \sum_{i=1}^{\mu}(X_iY_i + (1 - X_i)(1 - Y_i))$.
- Let

$$a = \sum_{\mathbf{b}\in B_\mu} s_\psi(\mathbf{b}) \cdot f(\mathbf{b}) \cdot \mathsf{eq}(\mathbf{b}, \psi^{-1}(\mathbf{r})),$$
$$b = \sum_{\mathbf{b}\in B_\mu} s_\psi(\mathbf{b}) \cdot g(\mathbf{b}) \cdot \mathsf{eq}(\mathbf{b}, \mathbf{r}),$$

  which can be computed by the prover in an offline manner.
- The prover sends the verifier a and b.
- Run the Sum-Check PIOP described in Section 4.1 for $\big((a, [[F]]); F\big)$ and $\big((b, [[G]]); G\big)$.
- Lastly, the verifier checks $\psi(a) = b$. Reject otherwise.

Figure 8: Automorphism-Check Protocol

The completeness follows from that of sum-check protocol.

Knowledge Soundness

We only need to consider soundness by Lemma 3.12. For any $\big(\psi; ([[f]], [[g]]); f, g\big) \notin R_{auto}$, let $H_1, H_2 \in R_q^{(\leq d)}[X^\mu]$ such that

$$H_1(\mathbf{Y}) = \sum_{\mathbf{b}\in B_\mu\psi} \psi\big(s_\psi(\mathbf{b})\cdot f(\mathbf{b})\big)\cdot\mathsf{eq}(\mathbf{b}, \mathbf{Y}) \quad \text{and} \quad H_2(\mathbf{Y}) = \sum_{\mathbf{b}\in B_\mu\psi} s_\psi(\mathbf{b})\cdot g(\mathbf{b})\cdot\mathsf{eq}(\mathbf{b}, \mathbf{Y}).$$

Then $H_1(\mathbf{Y}) \neq H_1(\mathbf{Y})$ since there exists $\mathbf{b} \in \mathcal{S}_\psi$ such that $\psi(f(\mathbf{b})) \neq g(\mathbf{b})$. Since the total degree of H_1 and H_2 is μ, by Lemma 3.11,

$$\Pr_{\mathbf{r}\leftarrow\!\$ R_q^\mu}[H_1(\mathbf{r}) = H_2(\mathbf{r})] \leq \frac{\mu}{q^e}.$$

Therefore, the probability that the verifier accepts is at most $\frac{\mu}{q^e}$, plus the probability that the sum-check protocol accepts when $(b, [[G]], G) \notin R_{sum}$ and $\psi(a) = b$, resulting in $\frac{\mu}{q^e} + \epsilon_{sum}^{d,\mu}$. □

Complexity Analysis

The complexity of the automorphism-check PIOP for R_{auto} is presented below;

- The prover time is $\mathsf{pt}_{auto}^{f,g} = \mathsf{pt}_{sum}^{F} + \mathsf{pt}_{sum}^{G} = O\left(2^{\mu} d^2\right)$ R_q-ops.
- The verifier time is $\mathsf{vt}_{auto}^{f,g} = O(\mu)$.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}_{auto}^{f,g} = \mathsf{rc}_{sum}^{F} + \mathsf{rc}_{sum}^{G} + 1 = 2\mu + 1$.
- The query complexity is $\mathsf{qc}_{auto}^{f,g} = \mathsf{qc}_{sum}^{F} + \mathsf{qc}_{sum}^{G} + 1 = 6\mu + 3$.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}_{auto}^{f,g} = \mathsf{pl}_{sum}^{F} + \mathsf{pl}_{sum}^{G} + 2 = O(2^{\mu})$.
- The size of the witness is $O(2^{\mu})$.

5 Onigoroshi

In this section, we describe Onigoroshi, the first PIOP that can prove relations defined by algebraic circuits over a cyclotomic ring R_q comprising automorphism gates. Onigoroshi is built upon the zero-check, permutation-check, and automorphism-check described in Section 4.

Definition 5.1 (Onigoroshi indexed relation)**.** Let

$$\mathbf{gp} := (R_q, \ell, \ell_\varsigma, \ell_p, \ell_g, \ell_a, f)$$

be the public parameters for the indexed relation R_{oni}, where $R_q := \mathbb{Z}_q[\mathfrak{f}]$ is a cyclotomic ring for conductor $\mathfrak{f}$,

- $\ell = 2^\mu$ is the number of constraints,
- $\ell_\varsigma = 2^{\nu_\varsigma}$ is the number of selectors per gate,
- $\ell_p = 2^{\nu_p}$ is the number of public inputs,
- $\ell_g = 2^{\nu_g}$ is the number of inputs per gate plus 1 (for the output),
- $\ell_a = 2^{\nu_a} = \varphi(\mathfrak{f}) - 1$ is the number of automorphisms over $R_q := \mathbb{Z}_q[\mathfrak{f}]$ except for the identity map.

$f : R_q[X^\mu]^{\ell_\varsigma + \ell_g + 1} \to R_q[X^\mu]$ is a polynomial map such that:

$$f(\boldsymbol{\kappa}, \boldsymbol{\lambda}, \rho) = \kappa_0 \cdot \left(\lambda_0 + \cdots + \lambda_{\ell_g - 2}\right) + \kappa_1 \cdot \left(\lambda_0 \cdots \lambda_{\ell_g - 2}\right) + \sum_{i=1}^{\ell_\varsigma - 2} \kappa_{i+1} \cdot G_i(\lambda_0, \ldots, \lambda_{\ell_g - 2}) - \lambda_{\ell_g - 1} + \rho,$$

where $\boldsymbol{\kappa} \in R_q[X^\mu]^{l_\varsigma}, \boldsymbol{\lambda} \in R_q[X^\mu]^{l_g}, \rho \in R_q[X^\mu]$ and $G_i : R_q^{\ell_g - 1} \to R_q$ is a custom gate.
The indexed relation R_{oni} is the set of tuples

$$(\mathbb{i}; \mathbb{x}; \mathbb{w}) = \left((\varsigma, s, \sigma, \psi^{(1)}, \ldots, \psi^{(\varphi(\mathfrak{f})-1)}); \left(p, [[\omega]]\right); \omega\right)$$

where $\sigma : B_{\mu+\nu_g} \to B_{\mu+\nu_g}$ is a permutation and $s \in R_q^{(\leq 1)}[X^{\mu+\nu_a}]$ is a polynomial such that, for all $i = 1, \ldots, \varphi(\mathfrak{f}) - 1$:

$$s(\langle i - 1 \rangle_{\nu_a}, \mathbf{b}) = \begin{cases} 1 & \text{if the gate specified by } \mathbf{b} \text{ is the } \psi^{(i)}\text{-automorphism gate} \\ 0 & \text{Otherwise} \end{cases}.$$

$\psi^{(1)}, \ldots, \psi^{(\varphi(\mathfrak{f})-1)} \in \mathsf{Aut}(R_q)$ are all the automorphisms over R_q except for the identity map, $\varsigma \in R_q^{(\leq 1)}[X^{\mu+\nu_\varsigma}], p \in R_q^{(\leq 1)}[X^{\nu_p}], \omega \in R_q^{(\leq 1)}[X^{\mu+\nu_g}]$ are the selector polynomial, public input polynomial, and witness polynomial, respectively. $(\mathbb{i}; \mathbb{x}; \mathbb{w}) \in R_{oni}$ satisfies the followings:

- The gate identity:

$$\hat{f}(\mathbf{b}) = 0 \quad \forall \mathbf{b} \in B_\mu \setminus \bigcup_{i=1}^{\varphi(\mathfrak{f})-1} S_{\psi^{(i)}},$$

where $S_{\psi^{(i)}} \subseteq B_\mu$ is a set containing all the gate indices of $\psi^{(i)}$-automorphism gates and

$$\hat{f}(\mathbf{X}) = f\Big(s\big(\langle 0\rangle_{\nu_s}, \mathbf{X}\big), \ldots, s\big(\langle \ell_s - 1\rangle_{\nu_s}, \mathbf{X}\big),\ \omega\big(\langle 0\rangle_{\nu_g}, \mathbf{X}\big), \ldots, \omega\big(\langle \ell_g - 1\rangle_{\nu_g}, \mathbf{X}\big), p(\mathbf{X})\Big). \tag{13}$$

Onigoroshi performs the zero-check described in 4.2 to verify the aforementioned instance.

- The automorphism gate identity:

$$\Big(\big(\psi^{(i)}, s\big(\langle i-1\rangle_{\nu_a}, \mathbf{X}\big)\big); ([[\omega_{in}]], [[\omega_{out}]]); (\omega_{in}, \omega_{out})\Big) \in R_{auto} \quad \forall i \in A$$

where $A := \{j \in [\varphi(\mathfrak{f}) - 1] \mid S_{\phi^{(j)}} \neq \emptyset\}$,
$\omega_{in} = \omega\big(\langle 0\rangle_{\nu_g}, \mathbf{X}\big)$ and $\omega_{out} = \omega\big(\langle \ell_g - 1\rangle_{\nu_g}, \mathbf{X}\big)$. The definition of the relation R_{auto} is described in Definition 4.6.

- The wiring identity:

$$(\sigma; ([[\omega]], [[\omega]]); (\omega, \omega)) \in R_{perm} \quad \text{(described in Section 4.5)}.$$

- The consistency between the public input and witness polynomial:

$$p(\mathbf{X}) = \omega(0^{\mu+\nu_g-\nu_p}, \mathbf{X}).$$

Construction

In this protocol (See Figure 9), the prover first proves the gate identity by performing the zero-check described in Section 4.2 on the polynomial $\hat{f} \cdot \bar{s}$, where $\hat{f}$ is defined at the equation (13) and $\bar{s} \in R_q^{(\leq 1)}[X^\mu]$ is a polynomial such that $\bar{s}(\mathbf{b}) = 0$ if $\mathbf{b} \in \bigcup_{i=1}^{\varphi(\mathfrak{f})-1} \mathcal{S}_{\psi^{(i)}}$ and 1 otherwise. The gate identity check is performed over all the gates except for automorphism gates, which are specified by $B_\mu \setminus \bigcup_{i=1}^{\varphi(\mathfrak{f})-1} \mathcal{S}_{\psi^{(i)}}$. Subsequently, the protocol performs the automorphism-check described in Section 4.6 for automorphism gates. The wiring identity is verified by performing the permutation-check described in Section 4.5 on the witness polynomial ω. At the end of the protocol, the verifier checks the consistency between the public input polynomial p and witness polynomial

Theorem 5.1. Let $\mathbf{gp} := (R_q, \ell, \ell_s, \ell_p, \ell_g, \ell_a, f)$ be the public parameters of the relation R_{oni} where $\ell_g, \ell_s = O(1)$. Let $d := \deg(f)$ be the total degree of f. Onigoroshi PIOP for the relation R_{oni} described in the Figure 9 is complete with error $\varepsilon = \frac{\phi}{q^e}$, and it has knowledge error $\epsilon_{oni}^{\mathbf{gp}} := \max\left\{\epsilon_{zero}^{d+1,\mu}, \epsilon_{auto}^{d,\mu}, \epsilon_{perm}^{1,\mu+\nu_g}, \frac{\nu_p}{q^e}\right\} = \epsilon_{perm}^{1,\mu+\nu_g} = O\left(\frac{2^{\mu+\mu d}}{q^e}\right)$.

Onigoroshi

Indexer: $\mathcal{I}\left(\mathbf{gp}, \mathbb{i} = (\varsigma, s, \sigma, \psi^{(1)}, \ldots, \psi^{(\varphi(\mathfrak{f})-1)})\right)$ calls the permutation PIOP indexer $([[\iota_{id}]], [[\iota_\sigma]]) \leftarrow \mathcal{I}_{perm}(\sigma)$.

The indexer computes the oracle $[\bar{s}]$, where $\bar{s} \in R_q^{(\leq 1)}[X^\mu]$ is defined as

$$\bar{s}(\mathbf{b}) = \begin{cases} 1 & \left(\sum_{i=1}^{\varphi(\mathfrak{f})-1} s\big(\langle i-1\rangle_{\nu_a}, \mathbf{b}\big) = 0\right) \\ 0 & (otherwise) \end{cases} \quad \forall \mathbf{b} \in B_\mu.$$

$\mathcal{I}$ outputs $\mathbf{vp} = \Big([[\varsigma]], [[\bar{s}]], [[\iota_{id}]], [[\iota_\sigma]], \big([[\psi^{(i)}]]\big)_{i\in A}\Big)$,

where $A := \Big\{j \in [\varphi(\mathfrak{f}) - 1] \mid \sum_{\mathbf{b}\in B_\mu} s\big(\langle j-1\rangle_{\nu_a}, \mathbf{b}\big) \neq 0\Big\}$.

Protocol: The prover $P(\mathbf{gp}, \mathbb{i}, p, \omega)$ and the verifier $V(\mathbf{gp}, p, \mathbf{vp})$ perform the follwoing protocol.

- The prover sends the witness oracle $[[\omega]]$ to the verifier, where $\omega \in R_q^{(\leq 1)}[X^{\mu+\nu_g}]$.
- (Gate Identity) The prover sends the verifier the oracle $[[\hat{f} \cdot \bar{s}]]$, where $\hat{f} \in R_q^{(\leq d)}[X^\mu]$ is defined in the equation (13). Perform the Zero-Check PIOP described in Section 4.2 for $([[\hat{f} \cdot \bar{s}]], \hat{f} \cdot \bar{s}) \in R_{zero}$.
- (Automorphism Gate Identity) For all $i \in A$,
  the prover and verifier perform the Automorphism-Check for

  $$\Big(\big(\psi^{(i)}, s\big(\langle i-1\rangle_{\nu_a}, \mathbf{X}\big)\big); \big([[\omega_{in}]], [[\omega_{out}]]\big); (\omega_{in}, \omega_{out})\Big) \in R_{auto},$$

  where $\omega_{in} = \omega\big(\langle 0\rangle_{\nu_g}, \mathbf{X}\big)$ and $\omega_{out} = \omega\big(\langle \ell_g - 1\rangle_{\nu_g}, \mathbf{X}\big)$.
- (Wiring Identity) The prover and verifier perform the Permutation-Check PIOP described in Section 4.5 for $\Big(\sigma, \big([[\omega]], [[\omega]]\big), (\omega, \omega)\Big) \in R_{perm}$.
- The verifier checks the consistency between the public input polynomial and witness polynomial. It samples $\mathbf{r} \in R_q^{\nu_p}$, queries $[[\omega]]$ at point $(\langle 0\rangle_{\mu+\nu_g-\nu_p}, \mathbf{r})$, and then verifies if $p(\mathbf{r}) = \omega(\langle 0\rangle_{\mu+\nu_g-\nu_p}, \mathbf{r})$.

Figure 9: Onigoroshi Protocol

Proof. **Completeness**

For any $\Big((\varsigma, s, \sigma, \psi^{(1)}, \ldots, \psi^{(\varphi(\mathfrak{f})-1)}); \big(p, [[\omega]]\big); \omega\Big) \in R_{oni}$, the following properties hold by Definition 5.1:

- $([[\hat{f} \cdot \bar{s}]], \hat{f} \cdot \bar{s}) \in R_{zero}$, hence the verifier accepts the gate identity check.

Completeness follows from the completeness of the zero-check PIOP.

- $(\psi^{(i)}, [[\omega]], \omega) \in R_{auto}$ for all $i \in A$, thus the verifier accepts all the automorphism-check. Completeness follows the completeness of the automorphism-check.
- The public input polynomial $p \in R_q^{(\leq 1)}[X^{\nu_p}]$ is identical to $\omega(0^{\mu+\nu_g-\nu_p}, \mathbf{X}) \in R_q^{(\leq 1)}[X^{\nu_p}]$, hence their evaluation is always the same: $p(\mathbf{X}) = \omega(0^{\mu+\nu_g-\nu_p}, \mathbf{X})$. Thus, the verifier passes the consistency check.

In the wiring check phase, the permutation-check is performed on $(\sigma, ([[\omega]], [[\omega]]), \omega)$, eventually leading to the product-check described in Section 4.3. In the product-check, the following must hold:

$$\Big((1, [[f_1]], [f_2]]); f_1, f_2\Big) \in R_{prod}$$

where $f_1 = r + \iota_{id} + r_2 \cdot \omega$ and $f_2 = r + \iota_\sigma + r_2 \cdot \omega$. By the definition of the product-check, for any $\mathbf{b} \in B_\mu$, the evaluation of the denominator $f_2(\mathbf{b})$ is required to be non-zero. We consider the following polynomial.

$$F_2(X) = X + \iota_\sigma(\mathbf{b}) + r_2 \cdot \omega(\mathbf{b})$$

for some $\mathbf{b} \in B_\mu$. The CRT trasformation of F_2 results into ϕ slots of a univariate polynomial of total degree 1 over a finite field $\mathbb{F}_{q^e}$. The probability that there exists $\mathbf{b} \in B_\mu$ such that there is at least one slot i where $\mathsf{CRT}(F_2)_i(\mathbf{b}) = 0$ is:

$$\begin{aligned}
\Pr_{r' \leftarrow \$ \mathbb{F}_{q^e}}[\exists i \in [\phi] : \mathsf{CRT}(F_2)_i(r') = 0] &\leq \bigcup_{i \in [\phi]} \Pr_{r' \leftarrow \$ \mathbb{F}_{q^e}}[\mathsf{CRT}(F_2)_i(r') = 0] \\
&\leq \sum_{i \in [\phi]} \frac{1}{q^e} \\
&= \frac{\phi}{q^e}.
\end{aligned}$$

Consequently, the completeness error is at most $\frac{\phi}{q^e}$.

Knowledge Soundness

For any $\Big((\varsigma, s, \sigma, \psi^{(1)}, \ldots, \psi^{(\varphi(\mathfrak{f})-1)}); (p, [[\omega]]); \omega\Big) \in R_{oni}$, at least one of the following holds:

- $([[\hat{f} \cdot \bar{s}]], \hat{f} \cdot \bar{s}) \notin R_{zero}$,
- $\Big(\psi^{(i)}, ([[\omega_{in}]], [[\omega_{out}]])(\omega_{in}, \omega_{out})\Big) \notin R_{auto}$ for some $i \in A$,
- $\big(\sigma, ([[\omega]], [[\omega]]), (\omega, \omega)\big) \notin R_{perm}$,
- $p(\mathbf{X}) \neq \omega(0^{\mu+\nu_g-\nu_p}, \mathbf{X})$

In the first case, the probability that the verifier still accepts in the zero-check is at most $\epsilon_{zero}^{d+1,\mu}$. In the second case, the probability that the verifier still accepts in all the automorphism-checks is, at most $\epsilon_{auto}^{d,\mu}$. In the third case, the probability that the verifier still accepts in the permutation-check is, at most $\epsilon_{perm}^{1,\mu+\nu_g}$. In the last case, the probability that the verifier still accepts is at most $\frac{\nu_p}{q^e}$ by Lemma 3.11. Therefore, the probability that the verifier accepts for any $(\sigma; (p, [[\omega]]); \omega) \notin R_{hplonk}$ is at most $\max\left\{\epsilon_{zero}^{d+1,\mu}, \epsilon_{auto}^{d,\mu}, \epsilon_{perm}^{1,\mu+\nu_g}, \frac{\nu_p}{q^e}\right\}$. □

Complexity Analysis

Let $\ell_A := |A|$, where $0 \leq |A| \leq \varphi(\mathfrak{f}) - 1$. The complexity of the Onigoroshi is presented below;

- The prover time is $\mathsf{pt}_{oni} = O(\ell d^2) + O(\ell_A \cdot \ell d^2) + O(2^{\mu+\mu_g} d^2) = O(\ell_A \ell d^2)$ R_q-ops.
- The verifier time is $\mathsf{vt}_{oni} = \ell_p + O(\mu) + O(\ell_A \cdot \mu) + O(\mu + \mu_g) = O(\ell_A \mu + \ell_p)$. The term ℓ_p is the number of public inputs that the verifier has to read at the beginning of the protocol.
- The round complexity and the number of oracles sent by the prover is $\mathsf{rc}_{oni} = \mu + \ell_A \cdot (2\mu + 1) + (\mu + \mu_g + 1) = 2(\ell_A + 1)\mu + \ell_A + \mu_g + 1$
- The query complexity is $\mathsf{qc}_{oni} = (3\mu + 1) + \ell_A \cdot (6\mu + 3) + (3\mu + 3\mu_g + 2) = (6\ell_A + 6)\mu + 3\ell_A + 3\mu_g + 3$.
- The size of the proof (the size of the oracles sent by the prover) is $\mathsf{pl}_{oni} = O(\mu d) + O(\ell_A \mu d) + O(2^{\mu+\mu_g}) = O(2^{\mu})$.
- The size of the witness is $O(2^{\mu})$.

6 Application

As explained in the previous section, Onigoroshi supports relations over a cyclotomic ring R_q that involve automorphisms in $\mathsf{Aut}(R_q)$. In this section, we provide more detailed descriptions of some of the applications listed in Section 1.1.

6.1 Verifiable FHE (with Bootstrapping).

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that enables computations on encrypted data. FHE can be used to add privacy-preserving properties to cloud systems where the server can homomorphically perform computation on clients' encrypted data. Some FHE schemes have bootstrapping operations that refresh cyphertexts after involving a certain number of homomorphic operations, enabling further homomorphic operations on them [MS18, LMK+23b, GPV23]. In FHE schemes, computation on encrypted data requires malleable cyphertexts. Typically, FHE schemes assume that the server correctly performs computation. However, this assumption brings issues of integrity in FHE. There are key-recovery attacks exploiting the malleability of FHE [CT15, CGG16]. To address the issues, verifiable FHE (vFHE) was introduced [VKH23], which is FHE with integrity property. A common approach for constructing vFHEs is to combine an FHE scheme with a verifiable computation (VC) scheme [GGP10] or SNARK.

However, combining an FHE scheme and a VC scheme in an efficient manner is not trivial work since many VC schemes or SNARKs operate over a field, while FHE schemes work over polynomial rings. Consequently, a VC scheme and SNARK over rings have been proposed [BCS23, GNS23, BCFK21]. None of the aforementioned VC and SNARKs support relations involving automorphisms, which are utilised for bootstrapping operations.

Onigoroshi can be used as a building block of a vFHE with bootstrapping, enabling the verification of non-arithmetic operations involving automorphisms in addition to ring addition and multiplication operations.

6.2 SIMD-satisfiability of Arithmetic Circuits over Finite Fields.

In this section, we describe how to prove multiple statements over a finite field $\mathbb{F}_{q^e}$ in a single instruction multiple data (SIMD) manner using Onogoroshi. More precisely, Onigoroshi allows the prover to prove ϕ many statements defined over the same algebraic circuit C over $\mathbb{F}_{q^e}$. We denote the Onigoroshi relation over a finite field by R^*_{oni}, which is similar to the plonk relation in HyperPlonk [CBBZ23].
Let

$$\Big(((p_1, \ldots, p_\phi), ([[\omega_1]], \ldots, [[\omega_\phi]]));\ (\omega_1, \ldots, \omega_\phi)\Big)$$

be ϕ many instance-witness pairs for the relation R^*_{oni} such that

$$\Big((\varsigma, \sigma); (p_i, [[\omega_i]]); \omega_i\Big) \in R^*_{oni} \quad \text{for each } i \in [\phi].$$

where $\sigma : B_{\mu+\nu_g} \to B_{\mu+\nu_g}$ is a permutation, $\varsigma \in \mathbb{F}_{q^e}^{(\leq 1)}[X^{\mu+\nu_q}]$ is a selector polynomial, $p_i \in \mathbb{F}_{q^e}^{(\leq 1)}[X^{\nu_p}]$ is a public input polynomial, and $\omega_i \in \mathbb{F}_{q^e}^{(\leq 1)}[X^{\mu+\nu_g}]$ is a witness polynomial.
We first convert ϕ pairs of an instance and a witness over $\mathbb{F}_{q^e}$ into a single pair of an instance and witness over a cyclotomic ring R_q using the inverse of CRT transformation described in Lemma 3.9, denoted as CRT^{-1}. We obtain the following by applying CRT^{-1}:

$$CRT^{-1}\Big((p_1, \ldots, p_\phi)\Big) \to p \in R_q^{(\leq 1)}[X^{\nu_p}],$$
$$CRT^{-1}\Big((\omega_1, \ldots, \omega_\phi)\Big) \to \omega \in R_q^{(\leq 1)}[X^{\mu+\nu_g}].$$

Then, it holds that:

$$\big((q, \sigma); (p, [[\omega]]); \omega\big) \in R_{oni}$$

for the relation R_{oni}. Subsequently, we perform Onigoroshi presented in Section 5 for $\big((q, \sigma); (p, [[\omega]]); \omega\big)$, which is exactly equivalent to performing Onigoroshi over a finite field $\mathbb{F}_{q^e}$ for ϕ many inputs.

7 Conclusions

We presented the first polynomial interactive proof (PIOP) over cyclotomic rings that can prove the satisfiability of algebraic circuits over cyclotomic rings involving automorphism gates. Our PIOP allows to natively prove relations over cyclotomic rings involving automorphisms (even without automorphisms). These types of relations appear in the constructions of lattice-based cryptographic primitives.

We first introduced automorphism gates to circuits over a cyclotomic ring that takes an element from the ring and outputs the evaluation of an automorphism on the input. We presented Automorphism-Check, a novel polynomial interactive oracle proof (PIOP) that can verify the consistency between the input and output of all the automorphism gates in a circuit comprising the same automorphism.

Furthermore, we discussed that Verifiable Fully Homomorphic Encryption (vFHE) is a prominent example of such constructions, which ensures that computation on encrypted data is performed correctly by the server (integrity property). Since Onigoroshi supports relations with automorphisms, it can be used to verify computation involving automorphisms in addition to arithmetic operations, which appear in bootstrapping operations performed in some FHE schemes. However, we leave the concrete instantiation of vFHE based on Onigoroshi as future work.

Finally, we showed that Onigoroshi can be employed to verify multiple statements over a field in a SIMD manner. This is achieved by packing multiple statements into a single statement over a cyclotomic ring via the CRT inverse transformation and then applying Onigoroshi to this statement.

References

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019.

[ABPS24] Shahla Atapoor, Karim Baghery, Hilder V. L. Pereira, and Jannik Spiessens. Verifiable FHE via lattice-based SNARKs. Cryptology ePrint Archive, Paper 2024/032, 2024. https://eprint.iacr.org/2024/032.

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 2021. Springer, Cham.

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 102–132. Springer, Cham, August 2022.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[AL21] Martin R. Albrecht and Russell W. F. Lai. Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 519–548, Virtual Event, August 2021. Springer, Cham.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[BBCCL21] Olivier Bégassat, Alexandre Belling, Théodore Chapuis-Chkaiban, and Nicolas Liochon. A specification for a zk-evm. 2021.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Berlin, Heidelberg, May 2016.

[BCFK21] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Cham, May 2021.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BCG+18] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 595–626. Springer, Cham, December 2018.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 19–46. Springer, Cham, November 2020.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457. Springer, Cham, May / June 2022.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

[BCS23] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Lattice-based succinct arguments for NP with polylogarithmic-time verification.

In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 227–251. Springer, Cham, August 2023.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1:3–40, 1991.

[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, 1991.

[BHM20] Xavier Boyen, Thomas Haines, and Johannes Müller. A verifiable and practical lattice-based decryption mix net with external auditing. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 336–356. Springer, Cham, September 2020.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Cham, April / May 2017.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255. Springer, Cham, April / May 2018.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 441–469. Springer, Cham, August 2020.

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530. Springer, Cham, April 2023.

[CGG16] Ilaria Chillotti, Nicolas Gama, and Louis Goubin. Attacking FHE-based applications by software fault injections. Cryptology ePrint Archive, Report 2016/1164, 2016. https://eprint.iacr.org/2016/1164.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020.

[CLM23] Valerio Cini, Russell W. F. Lai, and Giulio Malavolta. Lattice-based succinct arguments from vanishing polynomials - (extended abstract). In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 72–105. Springer, Cham, August 2023.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. Cryptology ePrint Archive, Paper 2024/281, 2024. https://eprint.iacr.org/2024/281.

[Con15] Keith Conrad. Cyclotomic extensions, 2015.

[CT15] Massimo Chenal and Qiang Tang. On key recovery attacks against existing somewhat homomorphic encryption schemes. In Diego F. Aranha and Alfred Menezes, editors, *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 239–258. Springer, Cham, September 2015.

[DAFS24] Thomas Debris-Alazard, Pouria Fallahpour, and Damien Stehlé. Quantum oblivious lwe sampling and insecurity of standard model lattice-based snarks. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 423–434, 2024.

[EFG22] Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Report 2022/1763, 2022. https://eprint.iacr.org/2022/1763.

[FMN23] Giacomo Fenzi, Hossein Moghaddas, and Ngoc Khanh Nguyen. Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. Cryptology ePrint Archive, Paper 2023/846, 2023. https://eprint.iacr.org/2023/846.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss,

Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Cham, May 2020.

[FQZ+21] Boyuan Feng, Lianke Qin, Zhenfei Zhang, Yufei Ding, and Shumo Chu. ZEN: An optimizing compiler for verifiable, zero-knowledge neural network inferences. Cryptology ePrint Archive, Report 2021/087, 2021. https://eprint.iacr.org/2021/087.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

[GGP09] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. Cryptology ePrint Archive, Report 2009/547, 2009. https://eprint.iacr.org/2009/547.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Berlin, Heidelberg, August 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Berlin, Heidelberg, May 2013.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GNS23] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.

[GPV23] Antonio Guimarães, Hilder V. L. Pereira, and Barry Van Leeuwen. Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VI*, volume 14443 of *LNCS*, pages 3–35. Springer, Singapore, December 2023.

[GPvL23] Antonio Guimarães, Hilder V. L. Pereira, and Barry van Leeuwen. Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. Cryptology ePrint Archive, Paper 2023/014, 2023. https://eprint.iacr.org/2023/014.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Berlin, Heidelberg, December 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315, 2020. https://eprint.iacr.org/2020/315.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[IBM21] Ibm quantum delivers 120x speedup of quantum workloads with qiskit runtime. 2021.

[IBM23] The hardware and software for the era of quantum utility is here. 2023.

[ISW21] Yuval Ishai, Hang Su, and David J. Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 212–234. ACM Press, November 2021.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

[KPS18] Ahmed E. Kosba, Charalampos Papamanthou, and Elaine Shi. xJsnark: A framework for efficient verifiable computation. In *2018 IEEE Symposium on Security and Privacy*, pages 944–961. IEEE Computer Society Press, May 2018.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st FOCS*, pages 2–10. IEEE Computer Society Press, October 1990.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Berlin, Heidelberg, March 2012.

[LLC22] Thomas Lavaur, Jérôme Lacan, and Caroline PC Chanel. Enabling blockchain services for ioe with zk-rollups. *Sensors*, 22(17):6493, 2022.

[LMK+23a] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient fhew bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 227–256. Springer, 2023.

[LMK+23b] Yongwoo Lee, Daniele Micciancio, Andrey Kim, Rakyong Choi, Maxim Deryabin, Jieun Eom, and Donghoon Yoo. Efficient FHEW bootstrapping with small evaluation keys, and applications to threshold homomorphic encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 227–256. Springer, Cham, April 2023.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Berlin, Heidelberg, May 2013.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.

[LS18] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 204–224. Springer, Cham, April / May 2018.

[LXZ21] Tianyi Liu, Xiang Xie, and Yupeng Zhang. Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[Mic02] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *43rd FOCS*, pages 356–365. IEEE Computer Society Press, November 2002.

[Moo96] E Hastings Moore. A doubly-infinite system of simple groups. *Mathematical Papers Read at the International Mathematics Congress Held in Connection with the World's Columbian Exposition, Macmillan Co.*, pages 208–242, 1896.

[MS18] Daniele Micciancio and Jessica Sorrell. Ring packing and amortized FHEW bootstrapping. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl, July 2018.

[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 Computer Society Press, May 2013.

[PK22] Jim Posen and Assimakis A. Kattis. Caulk+: Table-independent lookup arguments. Cryptology ePrint Archive, Report 2022/957, 2022. https://eprint.iacr.org/2022/957.

[PSS19] Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution version 1.4. *Tornado cash privacy solution version*, 1:7, 2019.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Cham, August 2020.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275, 2020. https://eprint.iacr.org/2020/1275.

[STW24] Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 180–209. Springer, Cham, May 2024.

[T+22] Justin Thaler et al. Proofs, arguments, and zero-knowledge. *Foundations and Trends® in Privacy and Security*, 4(2–4):117–660, 2022.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 71–89. Springer, Berlin, Heidelberg, August 2013.

[TW24] Louis Tremblay Thibault and Michael Walter. Towards verifiable FHE in practice: Proving correct execution of TFHE's bootstrapping using plonky2. Cryptology ePrint Archive, Paper 2024/451, 2024. https://eprint.iacr.org/2024/451.

[VKH23] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption. *arXiv preprint arXiv:2301.07041*, 2023.

[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*, pages 926–943. IEEE Computer Society Press, May 2018.

[WW23] Hoeteck Wee and David J Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 201–235. Springer, 2023.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764. Springer, Cham, August 2019.

[ZBK+22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 3121–3134. ACM Press, November 2022.

[ZGK+22] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. Baloo: Nearly optimal lookup arguments. Cryptology ePrint Archive, Report 2022/1565, 2022. https://eprint.iacr.org/2022/1565.