# Independent Security Analysis of Longfellow ZK

Ignacio Cascudo[*] Carmit Hazay[†] Muthu Venkitasubramaniam[‡] Eylon Yogev[§]

December 2025

**Abstract**

This report presents an independent cryptographic analysis of the Longfellow zero-knowledge protocol [FS24]. This review complements the implementation-oriented audit by Trail of Bits [oB25] and focuses exclusively on the protocol's theoretical foundations.

## 1 Introduction

**Scope of the report.** This document presents a focused theoretical review of the Longfellow zero-knowledge protocol as specified in [FS24], complementing the formal specification and security proofs given in the original paper. The scope of this report is limited to the theoretical security analysis of the construction rather than a code-level audit. In particular, we verify that the system achieves the claimed security guarantees of zero-knowledge and soundness. We carry out the analysis in the Interactive Oracle Proof (IOP) model, where we show that Longfellow constitutes a ZKIOP with well-defined round-by-round soundness and then analyze the properties of the protocol obtained by compiling the ZKIOP via the BCS-transformation [BCS16]. We further provide a refined concrete analysis that yields different parameter sets for the target security level, summarized in Section 6. This report does not include any claims on the Longfellow ZK implementation (`https://github.com/google/longfellow-zk/`), which are covered separately in the audit conducted by Trail of Bits [oB25].

We note that several recent (unpublished) results on so-called proximity gap theorems for error-correcting codes have appeared online [DG25, CHK25, CS25, BCGM25, GG25, BCH$^+$25, FS25], including some positive and some negative results on previously stated conjectures. Importantly, the Longfellow ZK system does not rely on any conjectures related to proximity gap theorems. Furthermore, the concrete parameters chosen by the Longfellow ZK system (and analyzed in this report) are not based on any recent positive results and instead rely on prior peer-reviewed and published work.

**Conclusions.** Based on our analysis, the Longfellow ZK system satisfies (up to standard theoretical analysis) all properties required for a zero-knowledge proof system, and the selected

[*]Ligero Inc.
[†]Ligero Inc.
[‡]Ligero Inc.
[§]Bar-Ilan University

parameters target more than 115 bits of security. The Longfellow ZK system takes adequate measures to mitigate the recent attacks on the Fiat-Shamir heuristic. The Longfellow ZK system relies on so-called proximity gap theorems, which have been analyzed in the fully provable regime. At the end of the report, we provide a mechanism to boost security beyond 100 bits through grinding and suggest additional defenses to further strengthen the protocol.

**About the review panel.** Ignacio Cascudo is an associate research professor at the IMDEA Software Institute in Madrid, Spain, and a researcher at Ligero Inc. He received his PhD at University of Oviedo, Spain, and was a postdoc at CWI Amsterdam, the Netherlands, and at Aarhus University, Denmark, and later professor at Aalborg University, also in Denmark. His expertise is on cryptography (in areas such as multiparty computation and secret sharing) and error-correcting codes. He has been general co-chair of Eurocrypt 2025.

Carmit Hazay is a professor in the Faculty of Engineering at Bar-Ilan University, Israel, and a co-founder of Ligero Inc. She received her Ph.D. from Bar-Ilan University in 2009 and was a postdoctoral researcher at the Weizmann Institute of Science and Reichman University, Israel, and Aarhus University, Denmark. Carmit's research focuses on the foundations of cryptography, with a particular emphasis on secure computation and zero-knowledge proof systems, both in theory and practice. She co-authored the book 'Efficient Secure Two-Party Protocols – Techniques and Constructions. She served as program co-chair of Eurocrypt 2023 and received the distinguished paper award at CCS'23.

Muthu Venkitasubramaniam is a professor of Computer Science at Georgetown University, DC and co-founder of Ligero Inc. An expert on zero-knowledge proofs, secure multiparty computation and concurrent security, Muthu earned his PhD in Computer Science from Cornell University. Muthu is one of the steering committee members of zero-knowledge proof standardization effort. Muthu is a recipient of a Google Faculty Research Award, Distinguished paper award at CCS'23 and received the ICDE 2017 Influential Paper Award for his work on introducing new privacy techniques.

Eylon Yogev is a professor in the Department of Computer Science at Bar-Ilan University, Israel, where he is also a member of the Bar-Ilan Center for Research in Applied Cryptography and Cyber Security. His research focuses on probabilistic proof systems, with a particular emphasis on efficient post-quantum zero-knowledge proofs (zkSNARKs). He co-authored the book Building Cryptographic Proofs from Hash Functions and is a recipient of the Krill Prize for Excellence, the ACM SIGMOD Research Award, and Best Paper Awards at CRYPTO 2024 and PODS 2020. He received his Ph.D. from the Weizmann Institute of Science and was a postdoctoral researcher at Tel Aviv University, the Technion, Boston University, and the Simons Institute for the Theory of Computing.

**Overview of the Longfellow protocol.** Longfellow is a zero-knowledge proof system that demonstrates the correctness of an arithmetic circuit's evaluation with high efficiency and minimal prover overhead. It builds upon two foundational components: the Ligero proof system [AHIV17] and the Goldwasser–Kalai–Rothblum (GKR) interactive proof [GKR08]. The GKR component provides a public-coin layered sumcheck protocol that verifies circuit satisfiability by checking polynomial relations between successive layers of gates. Ligero, in turn, serves as both a lightweight commitment scheme and a zero-knowledge proof system for enforcing linear and quadratic constraints on committed data. The Longfellow construction integrates these primitives so that the prover first commits to its witness and to a masked GKR transcript, then uses Ligero to prove that the committed transcript would cause the GKR verifier to accept—achieving soundness, succinctness, and zero-knowledge simultaneously.

The remainder of this report is organized as follows. Section 2 introduces notations and formal definitions. Section 3 summarizes the Longfellow protocol and its composition of Ligero and layered sumcheck. Section 4 presents the main security analysis, covering soundness, zero-knowledge, and the hybrid argument. Section 5 discusses the Fiat–Shamir transformation and evaluates its impact on the protocol's security guarantees. Finally, Section 6 details concrete parameter selection and numerical soundness bounds.

## 2 Preliminaries

**Basic notations.** We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large $\tau$'s it holds that $\mu(\tau) < \frac{1}{p(\tau)}$. We use the abbreviation PPT to denote probabilistic polynomial-time and denote by $[n]$ the set of elements $\{1, \dots, n\}$ for some $n \in \mathbb{N}$. We will generally represent vectors over some sets with bold font, e.g. $\mathbf{v}$, and matrices with upper case fonts $A$. We denote by $\mathbf{v}_i$ the $i^{th}$ coordinate of $\mathbf{v}$. Moreover, we denote by $A[j]$ the $j$-th column of matrix $A$. For an NP relation $\mathcal{R}$, we denote by $\mathcal{R}_x$ the set of witnesses of $x$ and by $\mathcal{L}_\mathcal{R}$ its associated language. That is, $\mathcal{R}_x = \{w \mid (x, w) \in \mathcal{R}\}$ and $\mathcal{L}_\mathcal{R} = \{x \mid \exists\, w \text{ s.t. } (x, w) \in \mathcal{R}\}$. We assume functions to be represented by a Boolean/arithmetic circuit C (with AND/MULTIPLY, XOR/ADD gates of fan-in 2 and NOT gates), and denote the size of C by $|\mathrm{C}|$. By default, we define the size as the total number of gates, excluding NOT gates but including input gates.

### 2.1 Coding Notation

A code of length $n$ over an alphabet $\Sigma$ is simply a subset $C \subseteq \Sigma^n$. For vectors $\mathbf{v}$, $\mathbf{w} \in \Sigma^n$ the Hamming distance $d(\mathbf{v}, \mathbf{w})$ is the number of coordinates in which they differ, i.e. $|\{i \in [n] : \mathbf{v}_i \neq \mathbf{w}_i\}|$. Moreover, denote by $d(\mathbf{v}, C)$ the minimum distance of $\mathbf{v}$ from $C$, i.e. $d(\mathbf{v}, C) = \min\{d(\mathbf{v}, \mathbf{c}) : \mathbf{c} \in C\}$. Finally denote by $d(V, C)$ the minimal distance between a vector set $V$ and a code $C$, namely $d(V, C) = \min\{d(\mathbf{v}, C) : \mathbf{v} \in V\}$. For a code $C \neq \{0\}$, its minimum distance is $d_{\min}(C) := \min\{d(\mathbf{c}, \mathbf{c}') : \mathbf{c}, \mathbf{c}' \in C, \mathbf{c} \neq \mathbf{c}'\}$. An elementary fact from coding theory is that given an arbitrary vector $\mathbf{v} \in \Sigma^n$, there exists at most one codeword $\mathbf{c}$ in $C$ with $d(\mathbf{c}, \mathbf{v}) \leq d_{\min}(C)/2$.

A linear code $L$ of length n over a finite field $\mathbb{F}$ is a code $L \subseteq \mathbb{F}^n$ which is also a vector space over $\mathbb{F}$. Consequently, it has a dimension $k$ as a vector space. We then say that $L$ is an $[n, k, d]$ linear code, where $d = d_{\min}(L)$ is its minimum distance.

The Ligero protocol uses Reed-Solomon (RS) codes, defined next.

**Definition 2.1** (Reed-Solomon Code). *For positive integers $n, k$, finite field $\mathbb{F}$, and a vector $\boldsymbol{\eta} = (\eta_1, \dots, \eta_n) \in \mathbb{F}^n$ of pairwise distinct field elements, the code $\mathsf{RS}_{\mathbb{F},n,k,\boldsymbol{\eta}}$ is the $[n, k, n-k+1]$ linear code over $\mathbb{F}$ that consists of all $n$-tuples $(p(\eta_1), \dots, p(\eta_n))$ where $p$ is a polynomial of degree $< k$ over $\mathbb{F}$.*

**Interleaved codes.** Given a linear code $L \subseteq \mathbb{F}^n$ and an integer $m \geq 1$, we denote $L^m$ the set whose elements are all $m \times n$ matrices $U$ over $\mathbb{F}$ such that every row $U_i$ of $U$ is a codeword of $L$. By seeing matrices in $\mathbb{F}^{m \times n}$ as vectors in $(\mathbb{F}^m)^n$ (i.e., regarding each column as a sole coordinate in the space $\mathbb{F}^m$), $L^m$ can be regarded as a code over the alphabet $\mathbb{F}^m$. Moreover, under such interpretation, the Hamming distance between matrices $A, B \in \mathbb{F}^{m \times n} = (\mathbb{F}^m)^n$ (not necessarily in $L^m$) becomes the number of columns $j$ such that $A[j] \neq B[j]$. Then the distance between $A$ and $L^m$ is $d(A, L^m) = \min\{d(A, U) : U \in L^m\}$, and the minimum distance of $L^m$ is $d_{\min}(L^m) = \min_{\substack{U,V \in L^m \\ U \neq V}} d(U, V)$.

It is easy to verify that $d_{\min}(L^m) = d_{\min}(L)$.

## 2.2 Zero-Knowledge Arguments

We denote by $\langle A(w), B(z)\rangle(x)$ the random variable representing the (local) output of machine $B$ when interacting with machine $A$ on common input $x$, when the random-input to each machine is uniformly and independently chosen, and $A$ (resp., $B$) has auxiliary input $w$ (resp., $z$).

**Definition 2.2** (Interactive argument system). *A pair of* PPT *interactive machines* $\langle \mathcal{P}, \mathcal{V}\rangle$ *is called an* interactive proof system for a language $\mathcal{L}$ *if there exists a negligible function* $\mathsf{negl}$ *such that the following two conditions hold:*

1. Completeness: *For every* $x \in \mathcal{L}$ *there exists a string* $w$ *such that for every* $z \in \{0,1\}^*$,
$$\Pr[\langle \mathcal{P}(w), \mathcal{V}(z)\rangle(x) = 1] \geq 1 - \mathsf{negl}(|x|).$$

2. Soundness: *For every* $x \notin \mathcal{L}$, *every interactive* PPT *machine* $\mathcal{P}^*$, *and every* $w, z \in \{0,1\}^*$
$$\Pr[\langle \mathcal{P}^*(w), \mathcal{V}(z)\rangle(x) = 1] \leq \mathsf{negl}(|x|).$$

**Definition 2.3** (Zero-knowledge). *Let* $\langle \mathcal{P}, \mathcal{V}\rangle$ *be an interactive proof system for some language* $\mathcal{L}$. *We say that* $\langle \mathcal{P}, \mathcal{V}\rangle$ *is* computational zero-knowledge with respect to an auxiliary input *if for every* PPT *interactive machine* $\mathcal{V}^*$ *there exists a* PPT *algorithm* $\mathcal{S}$, *running in time polynomial in the length of its first input, such that*

$$\{\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x)\}_{x \in \mathcal{L}, w \in \mathcal{R}_x, z \in \{0,1\}^*} \overset{c}{\approx} \{\langle \mathcal{S}\rangle(x, z)\}_{x \in \mathcal{L}, z \in \{0,1\}^*}$$

*(when the distinguishing gap is considered as a function of* $|x|$*). Specifically, the left term denotes the output of* $\mathcal{V}^*$ *after it interacts with* $\mathcal{P}$ *on common input* $x$, *whereas the right term denotes the output of* $\mathcal{S}$ *on* $x$.

## 2.3 Interactive Oracle Proofs

Interactive Oracle Proofs (IOP) [BCS16, RRR16] are a type of proof system that combines the aspects of Interactive Proofs (IP) [Bab85, GMR85] along with Probabilistic Checkable Proofs (PCP) [BFLS91, AS98, ALM+98], as well as generalizes Interactive PCPs (IPCP) [KR08]. In this model, as in the PCP model, the verifier does not need to read the entire proof and can instead query it at random locations. Similarly to the IP model, the prover and verifier interact over several rounds. A k-round IOP has k rounds of interaction. In the $i^{th}$ round of interaction, the verifier sends a uniform public message $m_i$ to the prover, and the prover generates $\pi_i$. After running k rounds of interaction, the verifier makes some queries to the proofs via oracle access and either accepts or rejects them.

**Definition 2.4.** *Let $\mathcal{R}(x, \omega)$ be an NP relation corresponding to an NP language $\mathcal{L}$. An IOP system for a relation $\mathcal{R}$ with round complexity $k$ and soundness $\epsilon$ is a pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ if it satisfies the following properties:*

- Syntax: *On common input $x$ and prover input $\omega$, $\mathcal{P}$ and $\mathcal{V}$ run an interactive protocol of $k$ rounds. In each round $i$, $\mathcal{V}$ sends a message $m_i$ and $\mathcal{P}$ generates $\pi_i$. Here the verifier has oracle access to $\{\pi_1, \pi_2, \ldots, \pi_k\}$. We can express $\pi = (\pi_1, \pi_2, \ldots, \pi_k)$. Based on the queries from these oracles, $\mathcal{V}$ accepts or rejects.*

- Completeness: *If $(x, \omega) \in \mathcal{R}$ then,*

$$\Pr[(\mathcal{P}(x, \omega), \mathcal{V}^{\pi}(x)) = 1] = 1$$

- Soundness: *For every $x \notin \mathcal{L}$, every unbounded algorithm $\mathcal{P}^*$ and proof $\tilde{\pi}$*

$$\Pr[(\mathcal{P}^*, \mathcal{V}^{\tilde{\pi}}) = 1] \leq \mathsf{negl}(|x|)$$

The notion of IOP can be extended to provide a zero-knowledge property as well. Next, we define the definition of zero-knowledge IOP.

**Definition 2.5.** *Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be an IOP for $\mathcal{R}$. We say that $\langle \mathcal{P}, \mathcal{V} \rangle$ is a (honest verifier) zero-knowledge IOP (or ZKIOP for short) if there exists a PPT simulator $\mathcal{S}$, such that for any $(x, \omega) \in \mathcal{R}$, the output of $\mathcal{S}(x)$ is distributed identically to the view of $\mathcal{V}$ in the interaction $(\mathcal{P}(x, \omega), \mathcal{V}^{\pi}(x))$.*

Honest-Verifier Zero-Knowledge (HVZK) is a relaxed notion of zero-knowledge that assumes the verifier follows the protocol honestly, that is, it samples and sends its messages exactly as prescribed. Under this assumption, a proof system is honest-verifier zero-knowledge if there exists an efficient simulator that can generate a transcript indistinguishable from a real interaction between the prover and an honest verifier, without knowing the witness. Our ZK proof follows in two steps. We first prove that the Longfellow protocol is an honest-verifier perfect ZKIOP, and then compile it into a ZK non-interactive argument using the Fiat-Shamir heuristic.

## 2.4 The Layered Sumcheck Protocol

The sumcheck protocol is an interactive proof that allows a prover to convince a verifier of the value of a large sum of a low-degree polynomial over the Boolean hypercube without the verifier having to compute the sum directly. The prover sends univariate polynomials that gradually reduce the high-dimensional sum into a single evaluation, while the verifier checks consistency at each step by issuing random challenges. In the context of verifiable computation, initiated with the GKR [GKR08] proof system, sumcheck is used to certify that the values assigned to the gates of one circuit layer are consistent with those of the next layer. Thus, verifying a full circuit reduces to checking a sequence of polynomial sums, each enforced by the sumcheck protocol, rather than recomputing the entire circuit. In particular, the GKR protocol is a public-coin interactive proof system for verifying the correct evaluation of an arithmetic circuit. It allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a given circuit C of size $S$ and depth $d$ evaluates to a claimed output on a public input $x$ and a private witness $w$. The proof system consists of the following phases:

- Layered circuit representation. The circuit C is assumed to be layered, with each layer consisting of additions and multiplications over a finite field $\mathbb{F}$. The prover begins with the claim that the output layer evaluates to a certain value, and the verifier checks this claim recursively layer by layer until reaching the input layer.
- Sumcheck subprotocol. At each layer, the prover must demonstrate that the values of the gates are consistent with the wiring of the circuit and with the claimed values of the next layer. This is accomplished using the sumcheck protocol: the prover and verifier engage in a sequence of rounds where the prover sends a polynomial, the verifier samples a random challenge, and the prover responds with evaluations consistent with that challenge. Correctness of the sumcheck ensures that, with high probability, the prover cannot cheat about the values computed by the circuit.
- Efficiency. The prover's complexity is quasi-linear in the size of the circuit. The verifier's algorithm $\mathcal{V}_{\text{GKR}}$ can naturally be divided into two parts $(\mathcal{V}^1_{\text{GKR}}, \mathcal{V}^2_{\text{GKR}})$, both (interactive) PPT algorithms. The first is the *interactive phase*, in which the verifier samples random challenges and exchanges messages with the prover during the execution of the sumcheck protocol. The second is a *check phase* which bundles together all of the verifier's consistency checks. The GKR protocol additionally implies verifier efficiency (where the verifier's complexity is sublinear in the circuit size) for structured circuits. However, this feature is neither required nor satisfied by the GKR protocols used in the Longfellow ZK system.

The GKR protocol has served as the foundation for numerous subsequent works that aimed to optimize its efficiency, reduce communication complexity, and improve practical performance. For the purpose of this review, we refer to the specific instantiation of the sumcheck protocol used within the Longfellow construction as the *layered sumcheck*.

## 2.5 The Ligero Proof System [AHIV17]

At a high level, the Ligero proof system, denoted by $(\mathcal{P}_{\text{LIG}}, \mathcal{V}_{\text{LIG}})$, works as follows: the prover encodes the witness into a structured matrix and convinces the verifier of correctness through three checks. It has five main steps:

1. Matrix Preparation - The prover first encodes the witness (the satisfying assignment to the circuit), together with the internal values of the circuit, into a two-dimensional matrix of codewords, denoted by $U$, so that each row is an evaluation of a low-degree polynomial. This encoding enables both redundancy (for error-checking) and algebraic structure (for enforcing circuit relations).
2. Commitment Phase - The prover commits to the entire matrix by hashing its columns into a Merkle tree and sending the root to the verifier. This ensures binding: once the root is fixed, the prover cannot change individual entries without being caught.
3. Challenge Response Phase - The verifier then sends random challenges, and the prover responds to the verifier's challenge with three responses: the degree test, linear test, and quadratic test, as used in Ligero.

4. Opening Phase - As a final challenge, the verifier queries the committed matrix. Whereas the prover responds by opening the requested columns, along with Merkle authentication paths, proving that they are consistent with the committed root.

5. Verification Tests - Finally, on the opened columns, the verifier spot checks the degree test, linear test, and quadratic test. These tests check that the encodings correspond to low-degree polynomials, satisfy the linear constraints, and enforce multiplication correctness. Passing these tests with high probability implies that the entire hidden matrix encodes a valid witness.

In the Longfellow protocol, Ligero plays two distinct roles. First, it is employed as a commitment scheme, denoted by $\mathsf{LigeroCommit}$, that allows the prover to commit to the witness $w$ along with random padding. Second, it is invoked as a proof system $(\mathcal{P}_{\text{LIG}}, \mathcal{V}_{\text{LIG}})$ to demonstrate that a committed transcript of the sumcheck protocol is valid. More concretely, the statement proven in the Ligero scheme can be constructed as follows:

**(1) Representation of the witness in two fields.** For implementation reasons, the witness is represented in two distinct finite fields: the primary field $\mathbb{F}_p$ and the binary extension field $\mathbb{GF}(2^{128})$. We denote by $w_1$ and $w_2$ the portions of the witness encoded in these two fields, respectively, and by $w$ the original witness that they both represent. The two representations must remain consistent across fields so that the proof system guarantees the correctness of the same underlying computation.

**(2) Cross-field authentication.** To ensure this consistency, the protocol introduces a lightweight authentication mechanism based on a message authentication code (MAC). The prover samples a private key $k_p$ and the verifier samples an independent key $k_v$, defining a joint key $k = k_p \oplus k_v$. The value $m = \mathsf{MAC}_k(w)$ serves as an authentication tag binding the two field representations of $w$. The MAC guarantees that both $w_1$ and $w_2$ correspond to the same underlying witness $w$, even though they are handled over different fields.

**(3) Augmented circuits and proven statement.** For each field instance $i \in \{1, 2\}$, the protocol defines an augmented circuit $\widetilde{C}_i$ that checks both circuit correctness and MAC consistency. The prover commits to $w'_i = (w_i, w, k_p, \mathsf{pad}_i)$ using the Ligero commitment scheme $\mathsf{com}_i = \mathsf{LigeroCommit}(w'_i; r)$, where $\mathsf{pad}_i$ is a random pad masking the layered sumcheck transcript. The layered GKR protocol is then executed over $\widetilde{C}_i$ with masked prover messages $T'_i = T_i + \mathsf{pad}_i$. Finally, the Ligero verifier is invoked on a circuit $\widetilde{C}'_i$, which decrypts $T'_i$ and runs the layered sumcheck verifier for $\widetilde{C}_i$. The resulting statement verified by $\mathcal{V}_{\text{LIG}}$ is that there exist $(w_i, w, k_p, \mathsf{pad}_i, r)$ such that

$$\mathsf{com}_i = \mathsf{LigeroCommit}(w'_i; r) \quad \text{and} \quad \mathcal{V}_{\text{GKR}}\big(\widetilde{C}'_i, (x, m, k_v), T'_i, (w_i, w, k_p)\big) = 1.$$

**The Ligero IOP.** It is possible to cast Ligero in the IOP model because its interaction naturally fits the structure of an interactive oracle proof. In Ligero, the proof oracle consists of an initial commitment that encodes the witness within a matrix and oracle responses to the verifier's queries, while the verifier's role is limited to sampling random challenges and reading a small number of columns from the prover's oracle. These operations exactly match the query–answer interface defined by the IOP model, in which the prover provides oracle access to its messages rather than sending them in full. As a result, Ligero can be viewed as an IOP that implements the consistency tests through oracle queries.

**Perfect simulation in the IOP model.** In the Ligero IOP, the verifier's view consists of the commitments produced by the prover and the partial openings revealed during the consistency tests. Each committed matrix encodes the witness using a Reed–Solomon (RS) code, and before committing, the prover adds a fresh random mask to every encoded row. This guarantees that the committed codewords are statistically independent of the witness. Equivalently, the RS encoding can be interpreted as a form of *packed secret sharing*, where each codeword symbol shares a linear combination of several witness values while preserving perfect privacy against any party that observes fewer than the reconstruction threshold number of positions. Hence, the privacy of the codewords, and thus of the commitment, is implied by the privacy of the underlying packed secret sharing scheme. During verification, all challenges are sampled uniformly at random, and the prover's responses depend only on these public coins and the random mask applied at commitment time. Consequently, every symbol or polynomial evaluation observed by the verifier is distributed uniformly over the field, ensuring that the entire transcript is statistically independent of the witness. This establishes that Ligero achieves perfect honest-verifier zero knowledge in the IOP model.

**Instantiating a commitment scheme via the Ligero proof system.** Recall that the Ligero proof system proceeds in three stages. In the first stage, the prover commits to the "extended" witness that includes the inputs and all intermediate computations. Namely, it arranges the extended witness values in a matrix, adds blinding rows at the bottom of the matrix, pads each row with sufficient randomness, encodes each row as a low-degree polynomial, and then computes a Merkle tree over all row encodings. Then the root of that Merkle tree is published as a commitment. In the second stage, the prover receives a challenge and responds with a degree, linear, and quadratic test. In the third stage, the prover receives columns to be opened from the encoded matrix committed via the Merkle tree in the first stage.

To realize the commitment scheme, the prover first receives a witness $w$ and employs the same strategy as in the first stage of the Ligero scheme to commit to $w$. Next, it receives a statement to be proven based on the completed sumcheck transcript. It then constructs the extended witness based on the statement and $w$ and commits to it once again using the procedure employed in the first stage to build a second Merkle tree. It then executes Stages 2 and 3 of the Ligero proof system, treating the two commitments to $w$ and to the extended witness as a single matrix. This is achieved by stacking the encoded matrices on top of one another. When the columns have to be opened, the corresponding columns are opened in both commitments separately.

In the Longfellow construction, they do not have to employ the second commitment during the prove phase. The key insight is that if the constraints to be proven in the prove phase are linear, no additional values need to be committed, i.e., there are no intermediate computations. However, the GKR verification constraint indeed has multiplication constraints, but it introduces a novel mechanism that requires proving only linear constraints in the prove phase. The idea is to generate random Beaver-style multiplication triples during the commit phase, then execute the Beaver reduction on these triples for the multiplication gates in the constraints, which requires revealing the masked wire values in the clear. Then, all constraints can be expressed as linear constraints on the values committed in the commit phase.

## 3 The Longfellow Protocol

The Longfellow construction combines two main components. The first is the Ligero scheme [AHIV17], which functions as a cryptographic commitment scheme and enables efficient verification of algebraic relations on committed data. In Longfellow, Ligero serves to verify the integrity of the transcript generated by a variant of the Goldwasser–Kalai–Rothblum (GKR) protocol [GKR08], a public-coin interactive proof system that builds on sumcheck arguments. The second component is this interactive protocol itself, which establishes that an arithmetic circuit $C$ with public input $x$ and private witness $w$ is satisfied, i.e., that $C(x, w) = 0$. Longfellow integrates these two ingredients: the prover begins by committing to a string $w'$, via Ligero, that includes the witness $w$ along with auxiliary padding. Next, the prover and verifier execute the GKR layered sumcheck protocol, in which each prover message is committed and appropriately padded. Finally, the prover uses the Ligero proof system to show that the committed value $w'$ is consistent with the transcript and that this transcript would indeed cause the GKR verifier to accept. By combining the succinctness and structure of GKR with the lightweight commitment and zero-knowledge properties of Ligero, Longfellow transforms GKR into a privacy-preserving argument system. The protocol is described in Figure 1. For convenience, we provide a list of our parameters in Table 1.

| Parameter | Description |
|---|---|
| $x$ | Public statement |
| $n$ | $\lvert w \rvert$ |
| $w_1$ | Witness over $\mathbb{F}_p$ |
| $n'_1$ | $\lvert w_1 \rvert$ |
| $w_2$ | Witness over $\mathbb{GF}(2^{128})$ |
| $n'_2$ | $\lvert w_2 \rvert$ |
| $N$ | $n/128$ |
| $\kappa$ | Random Oracle output length |

Table 1: Description of the parameters.

## 4 The Security Analysis of the Interactive Variant

The completeness of the protocol is evident from its description, as each step aligns with the presence of a valid witness that meets the set constraints. This report emphasizes the analysis of soundness (both standard and round-by-round) and zero-knowledge properties, offering a high-level discussion of their validity. We examine these properties within the Interactive Oracle Proof (IOP) framework (see Definition 2.4) and deduce that these properties are preserved in the resulting non-interactive protocol, achieved through the BCS transformation [BCS16]. In the IOP model, the interactive version of the Longfellow ZK protocol involves the prover using the proof

The ZKP for an NP-relation $\mathcal{R} = \mathcal{R}(x, w)$ is executed between a prover $\mathcal{P}$ that has input $(x, w) \in \mathcal{R}$ and a verifier $\mathcal{V}$ that has input $x$. Let $\mathcal{R}$ be represented by the verification circuits $\mathsf{C}_1(\cdot,\cdot,\cdot)$ defined over a field $\mathbb{F}_p$ and $\mathsf{C}_2(\cdot,\cdot,\cdot)$ defined over a field $\mathbb{GF}(2^{128})$. Namely, for every $x$, $\exists w \in \{0,1\}^{128N}$ such that $\mathcal{R}(x, w) = 1$ iff $\exists w_1 \in \mathbb{F}_p^{n'_1}$ and $w_2 \in \mathbb{GF}(2^{128})^{n'_2}$ such that $\mathsf{C}_1(x, w_1, w) = 0$ and $\mathsf{C}_2(x, w_2, w) = 0$. Let $(\mathcal{P}_{\text{GKR}}, \mathcal{V}_{\text{GKR}})$ denote the GKR layered sumcheck interactive system as discussed in Section 2.4, let $(\mathcal{P}_{\text{LIG}}, \mathcal{V}_{\text{LIG}})$ denote the Ligero ZKIOP, and let $\mathsf{LigeroCommit}$ denote the Ligero commitment algorithm as defined in Section 2.5. Let $MAC_{a,b}(x) = ax + b$ denote a MAC on the message $x$ where $a, b, x \in \mathbb{GF}(2^{128})$; for vectors $a, b, x \in \mathbb{GF}(2^{128})^N$, we define the componentwise extension $\overline{MAC}_{a,b}(x) = (MAC_{a_1,b_1}(x_1), MAC_{a_2,b_2}(x_2), \ldots, MAC_{a_N,b_N}(x_N))$.

- For $i \in \{1, 2\}$, define circuits $\widetilde{\mathsf{C}}_i((x, m, k_v), (w_i, w, k_p))$ where $k_p, k_v \in \{0,1\}^{256N}$ as follows:
  - Assert $\mathsf{C}_i(x, w_i, w) = 0$
  - Assert $\overline{MAC}_k(w) = m$ where $k = k_p \oplus k_v$ with all computation performed in $\mathbb{GF}(2^{128})$ (note that $\mathbb{GF}(2^{128})$ arithmetic is emulated in $\mathbb{F}_p$ for $\mathsf{C}_1$).

1. $\mathcal{P}$ and $\mathcal{V}$ compute the sizes $\ell_1 \leftarrow \mathsf{sz}_{\mathbb{F}_p}(\widetilde{\mathsf{C}}_1)$ and $\ell_2 \leftarrow \mathsf{sz}_{\mathbb{GF}(2^{128})}(\widetilde{\mathsf{C}}_2)$ of the sumcheck transcripts in the respective fields.
2. $\mathcal{P}$ samples $k_p \leftarrow \{0,1\}^{256N}$ and computes the augmented witnesses $w'_i \leftarrow (w_i, w, k_p, \mathsf{pad}_i)$ where $\mathsf{pad}_1 \leftarrow \mathbb{F}_p^{\ell_1}$ and $\mathsf{pad}_2 \leftarrow \mathbb{GF}(2^{128})^{\ell_2}$.
3. $\mathcal{P}$ computes $\mathsf{com}_1 \leftarrow \mathsf{LigeroCommit}(w'_1)$ and $\mathsf{com}_2 \leftarrow \mathsf{LigeroCommit}(w'_2)$ over fields $\mathbb{F}_p$ and $\mathbb{GF}(2^{128})$ respectively and sends $\mathsf{com}_1, \mathsf{com}_2$ to $\mathcal{V}$.
   (a) $\mathcal{V}$ samples $k_v \leftarrow \{0,1\}^{256N}$ and sends to $\mathcal{P}$.
   (b) $\mathcal{P}$ computes $m \leftarrow \overline{MAC}_{k_v \oplus k_p}(w)$ and sends to $\mathcal{V}$.
4. $\mathcal{P}$ and $\mathcal{V}$ sequentially execute two instances of the protocol $(\mathcal{P}_{\text{GKR}}, \mathcal{V}_{\text{GKR}})$ with the corresponding inputs $\widetilde{\mathsf{C}}_1, ((x, m, k_v), (w_1, w, k_p))$ and $\widetilde{\mathsf{C}}_2, ((x, m, k_v), (w_2, w, k_p))$ over their respective fields, with the following modification:
   - Whenever $\mathcal{P}_{\text{GKR}}$ instructs the prover to send the $j$th message $m_j \in \mathbb{F}_p$ to $\mathcal{V}_{\text{GKR}}$, $\mathcal{P}$ sends $t'_j \leftarrow m_j + \mathsf{pad}_i[j]$ where $\mathsf{pad}_i[j]$ is the $j^{th}$ block of $\mathsf{pad}_j$.
   - When $\mathcal{V}_{\text{GKR}}$ instructs the verifier to send the $j$th random challenge $r_j$, $\mathcal{V}$ returns $r_j$.

   At the end, $\mathcal{P}$ and $\mathcal{V}$ hold encrypted sumcheck transcripts $T'_1$ and $T'_2$ respectively for the instances $\widetilde{\mathsf{C}}_1, ((x, m, k_v), (w_1, w, k_p))$ and $\widetilde{\mathsf{C}}_2, ((x, m, k_v), (w_2, w, k_p))$.
5. Define the circuit $\mathsf{C}'_i((x, m, k_v), T'_i, (w_i, w, k_p))$ to first decrypt the transcript $T_i$ by subtracting out the pads from $\mathsf{pad}_i$, and then run $\mathcal{V}_{\text{GKR}}(\mathsf{C}'_i((x, m, k_v), T'_i, (w_i, w, k_p)))$
6. Sequentially for $i \in \{1, 2\}$, $\mathcal{P}$ and $\mathcal{V}$ run the protocol methods $\mathcal{P}_{\text{LIG}}((\mathsf{C}'_i, (x, m, k_v), T'_i, \mathsf{com}_i), (w_i, w, k_p))$ and $\mathcal{V}_{\text{LIG}}(\mathsf{C}'_i, (x, m, k_v), T'_i, \mathsf{com}_i)$ respectively, concluding with $\mathcal{V}$ output the result.

Figure 1: The Longfellow Protocol

oracle for commitment rather than a hash-based Merkle tree, as the proof oracle serves as an ideal vector commitment mechanism. We will identify the soundness and round-by-round soundness error in the interactive variant and conclude the soundness error of the final protocol by applying the analysis of the BCS transformation from [CY24]. We will demonstrate that the Longfellow ZK achieves perfect simulation in the IOP model and once again rely on the BCS-transformation to conclude that the final protocol satisfies computational zero-knowledge.

**Theorem 4.1.** *The proof system from Figure 1 is a ZKIOP (Definitions 2.4-2.5) with perfect zero-knowledge and round-by-round knowledge soundness error*

$$\begin{aligned}\varepsilon_K = \max\Big\{ & \frac{n_1+2}{|\mathbb{F}_p|}, \frac{n_2+2}{2^{128}}, \\ & \binom{(\frac{1+\rho_1}{2})n_1}{t} \Big/ \binom{n_1}{t} + \binom{2\rho_1 n_1 - 2}{t} \Big/ \binom{n_1}{t}, \\ & \binom{(\frac{1+\rho_2}{2})n_2}{t} \Big/ \binom{n_2}{t} + \binom{2\rho_2 n_2 - 2}{t} \Big/ \binom{n_2}{t}, \\ & \frac{N}{2^{128}}, \frac{2}{|\mathbb{F}_p|}, \frac{2}{2^{128}} \Big\}\end{aligned}$$

*where $n_i, \rho_i$ are the length and rate of the Reed Solomon code used in the Ligero system for $C'_i$, $i = 1, 2$, $t$ is the number of opened columns, and $D_j := \sum_{i=1}^{\mathsf{dep}_j} \lceil \log \mathsf{wid}_{j,i} \rceil$ for $j = 1, 2$, where $\mathsf{dep}_j$ denotes the depth (number of layers) of circuit $C'_j$ and $\mathsf{wid}_{j,i}$ denotes the width of the i-th layer of $C'_j$.*

*Proof.* Completeness is easily followed from the completeness of both proof systems and the correctness of the commitment scheme. We will next analyze plain soundness, followed by round-by-round soundness and then ZK.

**Soundness.** We argue the soundness of the protocol in the IOP model and analyze the non-interactive separately in Section 5. The Ligero commitments to $w'_i$ and proofs involving $w'_i$ are performed with Reed-Solomon codes of lengths $n_i$ and rates $\rho_i$ over fields $\mathbb{F}_i$, for $i = 1, 2$, where $\mathbb{F}_1 := \mathbb{F}_p$ and $\mathbb{F}_2 := GF(2^{128})$

Here are the high-level steps in arguing the soundness of the Longfellow ZK protocol:

1. Except with probability

$$\max\left\{ \frac{n_1+1}{|\mathbb{F}_1|} + \binom{(\frac{1+\rho_1}{2})n_1}{t} \Big/ \binom{n_1}{t}, \frac{n_2+1}{|\mathbb{F}_2|} + \binom{(\frac{1+\rho_2}{2})n_2}{t} \Big/ \binom{n_2}{t} \right\},$$

witnesses $w'_1$ and $w'_2$ can be extracted from the $\mathsf{com}_1$ and $\mathsf{com}_2$. Indeed, for each $i$, the extractor can obtain the matrix $U_i$ (the matrix produced in the matrix preparation part). By Theorem 4.3, if the prover passes the proof with probability more than $\frac{n_i+1}{|\mathbb{F}_1|} + (\frac{1+\rho_i}{2})^t$ then this matrix must be at distance at most $(d_i - 1)/2$ from the "interleaved code", meaning that there exist a matrix $V_i$, all of whose rows are codewords, which differs from $U$ in at most $(d_i - 1)/2$ columns. Here, $d_i$ is the minimum distance of the code. By the properties of error-correcting codes, this matrix $V_i$ is unique. Moreover, Reed-Solomon codes have efficient correction in the presence of $(d_i - 1)/2$ errors, by e.g., using the Berlekamp-Welch decoding algorithm. So the extractor can obtain $V_i$ and define the unique $w'_i$ as the messages encoded by this matrix $V_i$.

Note that in an honest execution of the protocol, $w'_1$ and $w'_2$ should be of the form $w'_1 = (w_1, w, k_p, \mathsf{pad}_1)$ and $w'_2 = (w_2, w, k_p, \mathsf{pad}_2)$ respectively. However, at this point, we have not yet guaranteed that $w$ and $k_p$ are the same in both $w'_1$ and $w'_2$. Therefore, we will, for the moment, denote them $w'_1 = (w_1, w^{(1)}, k_p^{(1)}, \mathsf{pad}_1)$ and $w'_2 = (w_2, w^{(2)}, k_p^{(2)}, \mathsf{pad}_2)$.

2. Except with probability

$$\max\left\{\frac{1}{|\mathbb{F}_1|} + \binom{2\rho_1 n_1 - 2}{t} \Big/ \binom{n_1}{t}, \frac{1}{|\mathbb{F}_2|} + \binom{2\rho_2 n_2 - 2}{t} \Big/ \binom{n_2}{t}\right\},$$

the inputs and extracted witnesses, $(x, T, \mathsf{com}_1)$, $w'_1 := (w_1, w^{(1)}, k_p^{(1)}, \mathsf{pad}_1)$ and $(x, T, \mathsf{com}_2)$, $w'_2 := (w_2, w^{(2)}, k_p^{(2)}, \mathsf{pad}_2)$ satisfy $\widetilde{C}_1$ and $\widetilde{C}_2$, respectively. This comes from the soundness analysis in Theorem 4.4. Note that this implies that:

(a) $\overline{MAC}_{(k_p^{(1)}) \oplus (k_v)}(w^{(1)}) = \overline{MAC}_{(k_p^{(2)}) \oplus (k_v)}(w^{(2)})$, and

(b) The sumcheck transcripts cause the sumcheck verifier to accept in both sumcheck interactions.

3. Except with probability $N \cdot 2^{-128}$, $w^{(1)} = w^{(2)}$.

Indeed, in order to find $w^{(1)} \neq w^{(2)}$, $k_p^{(1)}$ and $k_p^{(2)}$ with $\overline{MAC}_{(k_p^{(1)}) \oplus (k_v)}(w^{(1)}) = \overline{MAC}_{(k_p^{(2)}) \oplus (k_v)}(w^{(2)})$ a malicious prover would need to find some $i \in [N]$ where the $i$-th coordinates of $w^{(1)}$, $w^{(2)}$ are different, i.e. $w_i^{(1)} \neq w_i^{(2)}$, and where $MAC_{(k_p^{(1)})_i \oplus (k_v)_i}(w_i^{(1)}) = MAC_{(k_p^{(2)})_i \oplus (k_v)_i}(w_i^{(2)})$. Note $(k_p^{(1)})_i$, $(k_p^{(2)})_i$ and $w_i^{(1)}$, $w_i^{(2)}$ are all chosen before knowing $(k_v)_i$. Here $(k_p^{(1)})_i$, $(k_p^{(2)})_i$ and $(k_v)_i$ are the i-th coordinates of $k_p^{(1)}$, $k_p^{(2)}$, $k_v$, respectively.

Moreover, the MAC is key-homomorphic, meaning

$$MAC_{(k_p^{(1)})_i \oplus (k_v)_i}(w_i^{(1)}) = MAC_{(k_p^{(1)})_i}(w_i^{(1)}) \oplus MAC_{(k_v)_i}(w_i^{(1)})$$

and

$$MAC_{(k_p^{(2)})_i \oplus (k_v)_i}(w_i^{(2)}) = MAC_{(k_p^{(2)})_i}(w_i^{(2)}) \oplus MAC_{(k_v)_i}(w_i^{(2)})$$

Therefore we can rewrite $MAC_{(k_p^{(1)})_i \oplus (k_v)_i}(w_i^{(1)}) = MAC_{(k_p^{(2)})_i \oplus (k_v)_i}(w_i^{(2)})$ as

$$MAC_{(k_v)_i}(w_i^{(1)}) \oplus MAC_{(k_v)_i}(w_i^{(2)}) = MAC_{(k_p^{(1)})_i}(w_i^{(1)}) \oplus MAC_{(k_p^{(2)})_i}(w_i^{(2)})$$

Write $M_i := MAC_{(k_p^{(1)})_i}(w_i^{(1)}) \oplus MAC_{(k_p^{(2)})_i}(w_i^{(2)})$, which can be fully chosen by the adversary. The probability that the adversary finds $w_i^{(1)} \neq w_i^{(2)}$ and $M_i$ and with $MAC_{(k_v)_i}(w_i^{(1)}) \oplus MAC_{(k_v)_i}(w_i^{(2)}) = M_i$, without knowing $(k_v)_i$, is $2^{-128}$. Applying a union bound, we get the aforementioned bound $N \cdot 2^{-128}$.[1] Therefore, from now on we assume $w^{(1)} = w^{(2)}$ and denote it $w$.

4. Let $D_j := \sum_{i=1}^{\mathsf{dep}_j} \lceil \log \mathsf{wid}_{j,i} \rceil$ for $j = 1, 2$, where $\mathsf{dep}_j$ denotes the depth (number of layers) of circuit $C'_j$ and $\mathsf{wid}_{j,i}$ denotes the width of the $i$-th layer of $C'_j$. Then, except with probability
$$\max\left\{\frac{4D_1}{|\mathbb{F}_1|}, \frac{4D_2}{|\mathbb{F}_2|}\right\},$$
$w, w_1$ and $w, w_2$ satisfy the respective circuits. This comes from the soundness error of a sumcheck protocol, which by [Tha22, Proposition 4.1] is upper bounded by $vd/|\mathbb{F}|$ where $v$ is the number of variables of the polynomial and $d$ is the maximum degree of a variable. Concretely in this protocol, for the $i$-th layer of the circuit $C'_j$, sumcheck is used to verify a claim on a polynomial of maximum degree 2 in at most $2\lceil \log \mathsf{wid}_{j,i} \rceil$ variables. Therefore the soundness error of the sumcheck at each layer is $4\lceil \log \mathsf{wid}_{j,i} \rceil$. By applying the union bound to all layers, we get the expression above.

Putting everything together, we have shown that except with the probability
$$\begin{aligned}\varepsilon_K^{snd} = \max\Bigg\{&\frac{n_1+1}{|\mathbb{F}_p|} + \binom{(\frac{1+\rho_1}{2})n_1}{t}\Big/\binom{n_1}{t}, \frac{n_2+1}{|\mathbb{F}_p|} + \binom{(\frac{1+\rho_2}{2})n_2}{t}\Big/\binom{n_2}{t},\\ &\frac{1}{|\mathbb{F}_p|} + \binom{2\rho_1 n_1 - 2}{t}\Big/\binom{n_1}{t}, \frac{1}{2^{128}} + \binom{2\rho_2 n_2 - 2}{t}\Big/\binom{n_2}{t}\Bigg\}\\ &+ \frac{N}{2^{128}} + \max\left\{\frac{4D_1}{|\mathbb{F}_p|}, \frac{4D_2}{2^{128}}\right\} \qquad (1)\end{aligned}$$
if the proof passes the prover knows $w$ with $(x, w) \in \mathcal{R}$.

**Remark 1.** *Note that the soundness analysis above holds even if the prover is allowed to partially choose the statement $C'_i$ after having committed to the matrix $U_i$. Indeed, $V_i$ is uniquely determined by $U_i$ (if it exists), and otherwise the prover will be caught with high probability in the code test, which is independent of the statement. If the prover chooses a circuit such that it is not satisfied by the input encoded by $V_i$, then either the quadratic or linear test rejects except with probability at most $\binom{2\rho_i n_i - 2}{t}\Big/\binom{n_i}{t} + \frac{1}{|\mathbb{F}|}$ (see proof of Theorem 4.4 below).*

**Round-by-round (RBR) knowledge soundness.** We will now analyze the round-by-round soundness error of the protocol. The concept of round-by-round soundness was introduced in [CCH+19] as a method for evaluating the soundness of a proof system after it is transformed into a non-interactive protocol using the Fiat-Shamir heuristic. The final Longfellow ZK proof system indeed utilizes the BCS-transformation, which, in turn, relies on the Fiat-Shamir heuristic, to convert the ZKIOP discussed in this section into a non-interactive protocol. As shown in [BCS16], the soundness of the protocol resulting from the BCS-transformation can be analyzed by identifying the round-by-round knowledge soundness of the IOP. We repeat the definition of round-by-round knowledge from [BGK+23] (which in turn is a variant of [CMS19]).

[1] Indeed if the prover could find these with probability more than $2^{-128}$, the adversary would break one-time security of the MAC, which promises that even if the adversary can query $MAC_{(k_v)_i}(w_i^{(1)})$ for a chosen $w_i^{(1)}$, the probability of guessing $MAC_{(k_v)_i}(w_i^{(2)})$ for a different chosen $w_i^{(2)}$ is still $2^{-128}$.

**Definition 4.2.** *A $2r$-round protocol $\Pi$ for a language $L$ has round-by-round knowledge error $\varepsilon_K(\cdot)$ if there exists a polynomial time extractor $E$ and "doomed set" $\mathcal{D}$ or partial transcripts of interaction using $\Pi$ such that the following hold:*

1. *For all $x$, $(x, \emptyset) \in \mathcal{D}$, where $\emptyset$ denotes the empty transcript.*
2. *For any input $x$ and complete transcript $\tau$, if $(x, \tau) \in \mathcal{D}$ then $V(x, \tau) = \mathsf{reject}$.*
3. *If for $i \in [r]$, $(x, \tau)$ is a $2(i-1)$-round partial transcript such that $(x, \tau) \in \mathcal{D}$, then for every possible prover message $\alpha$ if*

$$\Pr[\beta \leftarrow V(x, (\tau, \alpha)) : (x, (\tau, \alpha, \beta)) \notin \mathcal{D}] > \varepsilon_K(x),$$

*then $E(x, \tau, \alpha)$ outputs a valid witness for $x$.*

We give a high-level argument of why the Longfellow ZK system is round-by-round knowledge sound by identifying the "doomed set", and the soundness analysis presented above will directly give the probability of transcripts going out of the doomed set.

- **Before the MAC-computation round:** There are two commitments made using LigeroCommit, one for each field. The verifier returns its share of the MAC key. All transcripts will be included in the doomed set, except the ones where valid witnesses $w'_1 := (w_1, w^{(1)}, k_p^{(1)}, \mathsf{pad}_1)$ and $w'_2 := (w_2, w^{(2)}, k_p^{(2)}, \mathsf{pad}_2)$ can be extracted and where $\overline{MAC}_{k_v \oplus k_p}(w^{(1)}) = \overline{MAC}_{k_v \oplus k_p}(w^{(2)})$ and $w^{(1)} \neq w^{(2)}$. From the analysis, this probability is at most $N \cdot 2^{-128}$.
- **Round by round soundness of GKR:** The GKR protocol is a leveled sumcheck protocol. The RBR soundness of the GKR sub-protocol will be depend on the RBR soundness of the sumcheck protocol. This is $d/|\mathbb{F}|$ where $d$ is the maximum degree of any variable in the multivariate polynomial. In the Longfellow ZK protocols $d = 2$. The actual protocol uses an "encrypted" version of the GKR to identify the doomed transcripts. We compute the actual transcripts by subtracting the pads extracted from the Ligero commitment. If the Ligero commitments are not extractable, all transcripts are included in the doomed transcripts.
- **Round by round soundness in the Ligero proof:** There are two rounds. In the first round, the prover responds with the code and linear/quadratic tests; in the second round, the prover reveals columns of the commitment. We analyze the RBR soundness for each of these rounds:

  **Prior to columns being revealed:** As detailed in [AHIV17], w.r.t code test, the doomed transcripts exclude all transcripts for which the code test result computed according to the committed matrix is $(d_i - 1)/2$-close to a codeword, yet the matrix is more than $(d_i - 1)/2$-far from any codeword. From Theorem 4.3, the probability over the randomness for the code test that such a transcript occurs is at most $(n_i + 1)/|\mathbb{F}|$. For the remaining transcripts, where the committed matrix is $(d_i - 1)/2$-close, doomed transcripts exclude the ones for which the extracted witness does not satisfy either the linear or quadratic constraints, yet the response to the linear and quadratic tests computed according to the committed matrix passes the tests. Such transcripts occur with probability at most $1/|\mathbb{F}|$. So overall, the probability of a transcript that is not doomed occurring is at most $(n_i + 2)/|\mathbb{F}|$.

**After the columns are revealed:** The set of doomed transcripts excludes two kinds of transcripts:

- Transcripts for which the committed matrix is $(d_i - 1)/2$-far and the code test result computed according to the committed matrix is also $(d_i - 1)/2$-far, yet the columns chosen to be revealed do not fail the column checks for the code test. In this case, any response from the prover will disagree columnwise with the committed matrix in at least $(d_i - 1)/2$ locations. This occurs with probability at most $\binom{(\frac{1+\rho_i}{2})n_i}{t} \Big/ \binom{n_i}{t}$.
- For the remaining transcripts, the doomed transcripts exclude the ones for which the column checks for the linear and quadratic test that passes, even though the witness encoded in the matrix does not satisfy the constraints. When this happens, it must be the case that the result of either the linear or quadratic test must be at least $(2\rho_i)$-far from the response computed according to the committed matrix. Such transcripts can occur with probability at most $\binom{2\rho_i n_i - 2}{t} \Big/ \binom{n_i}{t}$.

Putting it together, the RBR soundness of the Longfellow ZK protocol is given by:

$$
\begin{aligned}
\varepsilon_K = \max \Bigg\{ & \frac{n_1 + 2}{|\mathbb{F}_p|}, \frac{n_2 + 2}{2^{128}}, \\
& \binom{(\frac{1+\rho_1}{2})n_1}{t} \Big/ \binom{n_1}{t} + \binom{2\rho_1 n_1 - 2}{t} \Big/ \binom{n_1}{t}, \\
& \binom{(\frac{1+\rho_2}{2})n_2}{t} \Big/ \binom{n_2}{t} + \binom{2\rho_2 n_2 - 2}{t} \Big/ \binom{n_2}{t}, \\
& \frac{N}{2^{128}}, \frac{2}{|\mathbb{F}_p|}, \frac{2}{2^{128}} \Bigg\}
\end{aligned} \tag{2}
$$

**Remark 2** (Statistical security of ZKIOP vs Computational security of the final protocol.)**.** *The soundness error $\epsilon_K^{sound}$ from equation 1 is statistical (i.e. information-theoretic), namely, no unbounded adversary can break the soundness in the IOP model with probability better than $\epsilon_K^{snd}$. Recall that the final protocol applies the BCS-transformation to the IOP protocol, hence, it will only achieve a computational soundness guarantee. In the next section, we will rely on the round-by-round knowledge soundness error $\epsilon_K$ and analyze and identify $\lambda$ with the following soundness guarantee: for every $T$ and $\nu$ such that there exists a $T$-time adversary that succeeds in breaking soundness with probability $\nu$, referred to as "Advantage", it must hold that $\frac{T}{\nu} \geq 2^{\lambda}$.*

**Zero-knowledge.** For proving ZK, we define an efficient simulator $\mathcal{S}$ that produces an identically distributed view in the IOP model. Intuitively, this holds because the simulator defines the Merkle hashes (which serve as commitments to the witnesses) as the IOP oracles for which the verifier can only query a limited number of matrix columns. Consequently, the verifier learns no information about the committed strings in the Merkle tree. $\mathcal{S}$ operates as follows:

1. Upon receiving the public statement $x$, the verification circuits $\mathsf{C}_1(\cdot,\cdot,\cdot)$ and $\mathsf{C}_2(\cdot,\cdot,\cdot)$, and black-box access to a malicious verifier's algorithm $\mathcal{V}^*$ and the Ligero simulator $\mathcal{S}_{\text{LIG}}$, $\mathcal{S}$ defines the circuits $\widetilde{\mathsf{C}}_1$ and $\widetilde{\mathsf{C}}_2$, and computes the sizes $\ell_1 \leftarrow \mathsf{sz}(\widetilde{\mathsf{C}}_1)$ and $\ell_2 \leftarrow \mathsf{sz}(\widetilde{\mathsf{C}}_2)$ and further honestly chooses $k_p$.

2. $\mathcal{S}$ computes two fake augmented witnesses $w'_i \leftarrow (0^{|w|}, \mathsf{pad}_i)$ where where $\mathsf{pad}_1 \leftarrow \mathbb{F}_p^{\ell_1}$ and $\mathsf{pad}_2 \leftarrow \mathbb{GF}(2^{128})^{\ell_2}$.

3. $\mathcal{S}$ computes $\mathsf{com}_1 \leftarrow \mathsf{LigeroCommit}(w'_1)$ and $\mathsf{com}_2 \leftarrow \mathsf{LigeroCommit}(w'_2)$ over fields $\mathbb{F}_p$ and $\mathbb{GF}(2^{128})$, respectively, and defines $\mathsf{com}_1, \mathsf{com}_2$ as oracle $\pi_1$.

   (a) $\mathcal{S}$ receives from $\mathcal{V}$ the value $k_v$.

   (b) $\mathcal{S}$ computes $m \leftarrow \overline{MAC}_{k_v \oplus k_p}(w)$ for a fake witness $w$, and sends to $\mathcal{V}$.

4. $\mathcal{S}$ and $\mathcal{V}$ sequentially execute two instances of the protocol $(\mathcal{P}_{\text{GKR}}, \mathcal{V}_{\text{GKR}})$ with the corresponding inputs $\widetilde{\mathsf{C}}_1, ((x, m, k_v), (0^{|w_1|}, 0^{|w|}, k_p))$ and $\widetilde{\mathsf{C}}_2, ((x, m, k_v), (0^{|w_2|}, 0^{|w|}, k_p))$ as follows:

   - Whenever $\mathcal{P}_{\text{GKR}}$ instructs the prover to send the $i$th message of length $m_i$ to $\mathcal{V}_{\text{GKR}}$, $\mathcal{S}$ sends a random string $t'_i$ of that length.
   - Whenever $\mathcal{V}_{\text{GKR}}$ instructs the verifier to send the $i$th random challenge $r_i$, $\mathcal{S}$ receives from $\mathcal{V}^*$ a challenge $r_i$. If $\mathcal{V}^*$ fails to send a string of the corresponding length, $\mathcal{S}$ aborts the simulation.

   At the end, $\mathcal{S}$ and $\mathcal{V}^*$ hold a simulated sumcheck transcripts $T'_1$ and $T'_2$ for the instances $\widetilde{\mathsf{C}}_1, ((x, m, k_v), (0^{|w_1|}, 0^{|w|}, k_p))$ and $\widetilde{\mathsf{C}}_2, ((x, m, k_v), (0^{|w_2|}, 0^{|w|}, k_p))$.

5. Define the circuit $\mathsf{C}'_i(x, w'_i, T'_i)$ to first decrypt the transcript $T'_i$ by subtracting out the pads from $\mathsf{pad}_i$, and then run $\mathcal{V}_{\text{GKR}}(\mathsf{C}'_i((x, m, k_v), T'_i, (0^{|w_i|}, 0^{|w|}, k_p)))$.

6. Sequentially for $i \in \{1, 2\}$, $\mathcal{S}$ and $\mathcal{V}$ run in the IOP model, the protocol methods defined by $\mathcal{S}_{\text{LIG}}((\mathsf{C}'_i, (x, m, k_v), T'_i, \mathsf{com}_i), (0^{|w_i|}, 0^{|w|}, k_p))$ and $\mathcal{V}_{\text{LIG}}(\mathsf{C}'_i, (x, m, k_v), T'_i, \mathsf{com}_i)$ respectively, concluding with $\mathcal{V}$ output the result.

We next prove the following claim,

**Claim 4.1.** *The proof system from Figure 1 maintains a perfect honest verifier zero-knowledge guarantee in the IOP model as per definition 2.4.*

Note that the real and simulated views differ in two aspects. First, in the simulated view, the commitment is generated with respect to an arbitrary string rather than to the actual padding used in the real execution. Second, instead of running the real Ligero prover to demonstrate the validity of the sumcheck protocol, the simulator invokes a simulated Ligero proof. We will prove indistinguishability by reducing it to the security of these two objects.

*Proof.* Our proof proceeds via a hybrid argument denoted by $\mathcal{H}$, defined by a simulator $\mathcal{S}_{\mathcal{H}}$ that behaves identically to the real execution, except that it invokes the Ligero simulator in Step 6 instead of the real Ligero prover. Establishing that the distributed view produced in $\mathcal{H}$ is indistinguishable from the real execution follows directly from the privacy of the Ligero proof system. In contrast, proving that the view generated by $\mathcal{S}_{\mathcal{H}}$ is indistinguishable from the final simulated view relies on the hiding property of the Ligero commitment scheme. In both cases, the statement is straightforward in the IOP model.

More specifically, we first claim that the hybrid execution is identically distributed to the real execution. The only difference between the two executions lies in the invocation of the Ligero subprotocol: either the real prover or the simulator is used. By the argument that the Ligero simulation in the IOP model induces a perfect honest-verifier simulation, we conclude that the view generated within the Ligero execution is identical in both cases. Furthermore, since all other components of the Longfellow protocol, including the first two commitments, verifier challenges, and the padded sumcheck transcript, can be viewed as sampled independently of the witness and of the Ligero transcript, the joint distribution of the entire hybrid execution remains identical to that of the real execution.

It remains to show that the hybrid and simulated executions are identical. The only difference between these two executions concerns the committed message: in the simulated execution, the simulator commits to an arbitrary string, whereas in the hybrid execution, it commits to the correct padding (the correct witness $w$ and the string that masks a correct sumcheck transcript). We prove indistinguishability by relying on the hiding property of the Ligero commitment scheme. Recall that the hiding property of the Ligero commitment follows directly from the combination of the random masking polynomial and the privacy of the underlying packed secret sharing scheme. In Ligero, the prover encodes the witness using a Reed–Solomon code and adds a uniformly random masking polynomial before committing. This ensures that the encoded codewords, and any evaluations revealed during verification, are uniformly distributed and independent of the witness. Interpreting the Reed–Solomon encoding as a form of packed secret sharing, its privacy property further guarantees that revealing a bounded number of columns discloses no information about the shared secrets. Together, these two mechanisms imply that the Ligero commitment is perfectly hiding: any two commitments to witnesses of the same length are identically distributed. Consequently, the hybrid and simulated views are identically distributed.

■

This completes the proof. ■

## 4.1 Auxiliary Proofs

**Theorem 4.3.** *Suppose Ligero uses a Reed-Solomon code $L$ of length $n$, rate $\rho < 1/2$, and minimum distance $d$ over the field $\mathbb{F}$, and $t$ columns of the matrix $U$ are queried. If the code test is accepted, except with probability at most*

$$\binom{(\frac{1+\rho}{2})n}{t} \Big/ \binom{n}{t} + \frac{n+1}{|\mathbb{F}|}$$

*there exists a unique matrix $V$ such that each of the rows of $V$ are valid Reed-Solomon codewords, $U$ and $V$ differ in at most $(d-1)/2$ coordinates and $U$ and $V$ coincide in the queried columns.*

*Proof.* Recall that in the Ligero code test, the prover is challenged to send a random linear combination $\mathbf{r}^\top U$ of the rows of the committed matrix $U$. The prover responds to this challenge with a vector $\mathbf{v}$ (purportedly $\mathbf{r}^\top U$). We assume $\mathbf{v} \in L$ (otherwise the proof is rejected). Then, there are two possibilities, depending on how close $\mathbf{v}$ is (in Hamming distance) to the actual linear combination $\mathbf{r}^\top U$: either 1. $\mathbf{r}^\top U$ and $\mathbf{v}$ coincide in at most a fraction $\frac{1+\rho}{2}$ of the coordinates (i.e. $d(\mathbf{r}^\top U, \mathbf{v}) \geq (1 - \frac{1+\rho}{2})n$ ); or 2. they coincide in more, i.e. $d(\mathbf{r}^\top U, \mathbf{v}) \leq (1 - \frac{1+\rho}{2})n$. We analyze these cases separately.

1. If $d(\mathbf{r}^\top U, \mathbf{v}) \geq (1 - \frac{1+\rho}{2})n$: in this case, note that in Ligero, the verifier queries $t$ columns of $U$, compares $\mathbf{r}^\top U$ and $\mathbf{v}$ on those coordinates, and rejects if they are different from each

other. Hence, the only possibility that cheating is not detected is that the verifier queries $t$ coordinates that coincide. But $\mathbf{r}^\top U$ and $\mathbf{v}$ coincide in at most $(\frac{1+\rho}{2})n$ coordinates, So the probability of this happening (and hence cheating not being detected) is at most

$$\binom{(\frac{1+\rho}{2})n}{t} \Big/ \binom{n}{t}$$

2. If $d(\mathbf{r}^\top U, \mathbf{v}) \leq (1 - \frac{1+\rho}{2})n$, then, since $\mathbf{v} \in L$

$$d(\mathbf{r}^\top U, L) \leq \Big(1 - \frac{1+\rho}{2}\Big)n = \frac{1-\rho}{2}n = \frac{d-1}{2}$$

where $d := d_{\min}(L)$ is the minimum distance of $L$. In these conditions, the results on proximity testing [BCI+20] guarantee that, except with probability $\frac{n}{|\mathbb{F}|}$ there exists a matrix $V \in L^m$ with $d(U, V) \leq \frac{d-1}{2}$, i.e. there exists a matrix $V$ all of whose rows are codewords and which differs in at most $\frac{d-1}{2}$ columns from $U$. Since $d_{\min}(L^m) = d_{\min}(L) = d$, coding theoretic properties ensure this $V$ is unique. Moreover, [ACFY25, Lemma 4.10] ensures that the stronger property of *correlated mutual agreement* always holds in the case where $d(\mathbf{r}^\top U, L) \leq \frac{d-1}{2}$; in a nutshell, this property states that if $V$ exists, then $U$ and $V$ differ in the same set of positions as $\mathbf{r}^\top U$ differs from $\mathbf{v}$. Moreover, [ACFY25, Lemma 4.13] shows that mutual correlated agreement implies that $V$ is such that $\mathbf{r}^\top V = \mathbf{v}$.

Therefore applying the union bound, we have that except with probability at most

$$\binom{(\frac{1+\rho}{2})n}{t} \Big/ \binom{n}{t} + \frac{n}{|\mathbb{F}|},$$

if the verifier accepts the proof then, restricted to the $t$ coordinates that the verifier queries, $\mathbf{r}^\top V$ and $\mathbf{r}^\top U$ coincide. In turn, this implies $U$ and $V$ coincide in all those coordinates except with probability at most $1/|\mathbb{F}|$. Applying the union bound again yields the result. ∎

**Theorem 4.4.** *In the conditions of Theorem 4.3, suppose the matrix $V$ promised by the theorem exists. Then, except with probability*

$$\binom{2\rho n - 2}{t} \Big/ \binom{n}{t} + \frac{1}{|\mathbb{F}|}$$

*if the linear and quadratic tests of Ligero pass, the inputs encoded by $V$ satisfy the tested circuit.*

*Proof.* Let us assume that the inputs encoded by $V$ do not satisfy (at least) one of the tested circuits, either the linear or the quadratic. We will bound the probability of the prover passing this test (if $V$ does not satisfy both tests, we just fix one of them).

Both tests (linear and quadratic) of Ligero have in common that the prover is requested to reveal a certain polynomial (different for each of the tests) of degree at most $2k - 2$, where $k$ is the dimension of the code, i.e., $k = \rho n$, and which depends on the polynomials associated with the rows of $V$, the statement, and on randomness chosen by the verifier.

Moreover, in both of the tests, once the prover announces the polynomial $p_V$, the verifier performs two checks:

1. A check that $p_V$ is constructed correctly from the encoding matrix $V$; this is done by using the $t$ columns of $V$ and checking these are consistent with the evaluations of $p_V$.

2. A test that the inputs encoded by $p_V$ satisfy the relation tested (namely, the linear relation or quadratic relation in either test).

For the test that we have fixed that $V$ does not satisfy, the prover can do two different things in order to attempt to pass the test:

- Reveal the polynomial $p_V$ constructed honestly from $V$. In this case, test 2. above fails except with probability $1/|\mathbb{F}|$. See [AHIV17] for the details.

- Reveal a polynomial $q$ which is different from the $p_V$ constructed honestly from $V$. Then $q$ and $p_V$ can coincide in at most $2k-2$ positions. The verifier only accepts test 1. above if the $t$ queried positions are among those $2k-2$. The probability of this is

$$\binom{2k-2}{t} \Big/ \binom{n}{t}.$$

Therefore, the test will reject except with probability

$$\binom{2k-2}{t} \Big/ \binom{n}{t} + \frac{1}{|\mathbb{F}|},$$

and recall $k = \rho n$. Note that this holds as soon as the inputs encoded by $V$ do not satisfy one of the tested circuits. If it does not satisfy *both*, then the probability that the prover gets caught is at least the probability that it gets caught in one of them. Hence the bound

$$\binom{2\rho n-2}{t} \Big/ \binom{n}{t} + \frac{1}{|\mathbb{F}|}$$

for the soundness error still holds in that case.

∎

## 5 The Fiat-Shamir Heuristic: From ZKIOP to ZKSNARK

In the previous section, we analyzed the protocol in the interactive oracle proof (IOP) model. To make it succinct and non-interactive, Longfellow ZK applies the BCS [BCS16] transformation, which uses Merkle trees to implement the vector commitment (or proof oracle), and the well-known Fiat–Shamir transformation. While this transformation is straightforward for three-message protocols, it is more nuanced for multi-round protocols such as the Longfellow ZK. For the transformation to preserve soundness in the multi-round setting, the underlying interactive proof must satisfy a stronger property known as round-by-round soundness, which was analyzed in the previous section.

The formal definition of this transformation and a rigorous security analysis are given in [CY24]. Formally, we get the following theorem

**Theorem 5.1.** *Let $\Pi$ be the Longfellow Protocol, and let $\varepsilon_K$ be its round-by-round knowledge error. Let $\Pi'$ be the compiled scheme after applying the BSC+Fiat-Shamir transformation with a random oracle that outputs $\kappa$ bits. Then, $\Pi'$ has adaptive knowledge error $\varepsilon_{\mathsf{ARG}}$ against $T$-size adversaries where*

$$\varepsilon_{\mathsf{ARG}}(T) \leq T \cdot \varepsilon_K + \frac{9}{2} \cdot \frac{T^2}{2^\kappa}.$$

We remark that the theorem above differs from the corresponding result in [CY24]. The theorem in [CY24] is stated in terms of the state-restoration error of the scheme, where the first term in the expression is replaced with the state-restoration soundness error. State-restoration soundness is strictly weaker than (and implied by) round-by-round knowledge soundness (see [CY24, Chapter 31]). In particular, the theorem presented in [CY24] demonstrates that the state restoration (knowledge) soundness error is bounded by $(T + \ell) \cdot \varepsilon_K$, where $\ell$ is the round complexity of the protocol and $\epsilon_K$ is the round-by-round soundness error.

In the theorem stated above, we are able to eliminate the dependency on the round complexity $\ell$ and use $T \cdot \varepsilon_K$ instead of $(T + \ell) \cdot \varepsilon_K$. This additional factor arises because any Fiat–Shamir query to the random oracle can correspond to a query for any of the $\ell$ rounds. In contrast, in the Longfellow protocol, each query to the random oracle can be uniquely associated with a specific protocol transcript and a specific round. This uniqueness is ensured by embedding the message type and length into the query string. Consequently, by closely examining the proof of the Fiat–Shamir theorem in [CY24], we observe that the $\ell$ factor in the soundness error can be eliminated.

## 5.1 White-Box Attacks on Fiat-Shamir

The theorem above guarantees unconditional security in the random oracle model. In this model, the Fiat–Shamir hash function is treated as a truly random function, with both the malicious prover and the verifier having oracle access to it. To deploy the scheme in practice, the random oracle is instantiated using a concrete hash function.

However, a line of work highlights the risks of replacing a random oracle with a concrete hash function, regardless of the implementation [CGH04, Bar01, GK03, BBH+]. These attacks, often called "white-box" attacks, as they exploit the circuit representation of the hash function, a scenario that cannot occur in the idealized random oracle model. More recently, [KRS25] presented a white-box attack on the Fiat–Shamir transformation targeting schemes based on the GKR protocol.

The Longfellow protocol mitigates these attacks in several ways:

1. **Specialized computation:** The attacks exploit the fact that general-purpose proof systems can encode circuits that compute the Fiat–Shamir hash itself. In contrast, the Longfellow protocol is designed for a specific computation, which naturally avoids such self-referential circuits.

2. **High-complexity hash functions:** Following the suggestion in [KRS25], the Fiat–Shamir hash function in Longfellow is chosen to have greater circuit depth and a higher gate count than the circuit being proven. This prevents the circuit from computing the hash, thwarting the main mechanism of the white-box attacks.

3. **Integrated proof-of-work:** In addition, the Longfellow protocol incorporates the defense mechanism from the XFS protocol [AY25], which attaches a strong *proof-of-work* to the Fiat-Shamir hash. This further increases the circuit's computational complexity, making attacks significantly harder. Several constructions of this proof-of-work are possible. In the case of the Longfellow protocol, since the verifier's runtime is not the bottleneck, one can implement it by appending a long suffix of 0-bytes to the hash, proportional to the transcript size. By appending the 0-bytes as a *suffix*, precomputation or compression is prevented, making it infeasible for the circuit to compute their hash.

## 5.2 SNARKs with Everlasting Privacy

Everlasting privacy is a notion of privacy for the prover that, like zero knowledge, ensures that the verifier learns nothing beyond the validity of the statement being proven. However, everlasting privacy strengthens this guarantee by protecting the prover's privacy even against computationally unbounded adversaries. In this setting, the adversary may have unlimited computational power and may also make an unbounded number of queries to the random oracle.

It is known that standard zero-knowledge proofs for SNARKs cannot withstand unbounded adversaries. The reason is that any proof for zero-knowledge must rely on a *programmable* random oracle. In such proofs, the simulator programs the oracle at a certain point. While this programming is undetectable to any efficient adversary, an unbounded adversary can exhaustively query the entire oracle domain and thereby detect the points at which the simulator performed this programming. Consequently, the zero-knowledge property cannot be maintained in the presence of unbounded adversaries.

To address this limitation, one can consider a weaker privacy notion known as witness indistinguishability (WI). Witness indistinguishability requires that proofs generated using any two valid witnesses for the same statement are indistinguishable. Unlike zero knowledge, the proof of witness indistinguishability does not rely on programmable random oracles and can therefore tolerate unbounded adversaries. This leads to the notion of everlasting witness indistinguishability (everlasting WI), which guarantees that the witness's privacy is preserved even if the adversary later gains unbounded computational power.

The proof of everlasting WI is similar to the proof of zero-knowledge. The main difference is that in the zero-knowledge proof, the simulator programs the random oracle to output a specific random string (provided by the simulator of the underlying IP). In WI, we move to an unbounded simulator that inverts the random oracle on a given output, instead of programming it. Since the simulator is inefficient, the end result is not a simulation-based definition, but rather an indistinguishability one. The proof of everlasting witness indistinguishability (WI) follows a structure similar to that of the standard zero-knowledge proof. The key difference lies in how the random oracle is handled. In the zero-knowledge setting, the simulator must program the random oracle to output a specific value specified by the simulator of the underlying interactive proof. In contrast, in the everlasting WI setting, an unbounded simulator is used, which can efficiently invert the random oracle on a given output, rather than programming it. As this simulator is computationally unbounded, the resulting privacy notion that is derived is not via a simulation, but rather through an indistinguishability-based one: namely, that proofs generated using any two witnesses are indistinguishable, even to an unbounded adversary.

## 5.3 Further Security Layers

While the construction already includes all standard security mechanisms required for SNARKs, it is possible to incorporate additional defenses that further strengthen the protocol.

**Domain separation.** One such layer is the use of domain separation in all oracle queries. In this approach, each oracle query is prefixed (or suffixed) with structured metadata identifying its specific role within the protocol. For example, in Merkle tree computations, the input to the hash function can be prefixed with both the layer number and the node index within that layer. This ensures that hash evaluations across distinct contexts are derived from disjoint input domains. As a result, any collision found for a particular query, such as a single leaf in the tree, cannot be reused in another layer, node, or any other context. This reduces the potential impact of finding collisions, where the overall effect of any single collision on the protocol's security is negligible.

**Grinding.** A second technique, known as grinding, allows adding a small number of security bits to the round-by-round soundness error. When we apply grinding, we require the honest prover to perform a mildly hard proof-of-work after each message of the protocol. The difficulty of this proof-of-work is governed by a grinding parameter $z_i$ for round $i$. For example, if the prover wants to send message $m_i$ at round $i$ of the IOP, then it will send $m_i, s_i$ for some nonce $s_i$ such that the Fiat-Shamir hash begins with $z_i$ 0's. The remaining bits of the hash are used to derive randomness for the next round. This mechanism effectively reduces the computational advantage of a cheating prover while only slightly increasing the honest prover's running time. The reason is that the honest prover must solve the proof-of-work only once (per round), whereas a cheating prover, who must attempt many potential proofs to find a false one, needs to solve multiple independent proof-of-work instances.

Suppose we introduce grinding parameters $z_1, \ldots, z_\ell$ for a $\ell$-round IOP. In each round $i$, the prover must, on average, spend $2^{z_i}$ time to find a suitable nonce $s_i$. Hence, the total prover runtime increases by an additive factor of $\sum_{i \in [\ell]} 2^{z_i}$. From the security perspective, grinding directly reduces the round-by-round soundness error. Specifically, if the original error in round $i$ is $\varepsilon_i$, then after adding grinding with parameter $z_i$, the prover must also ensure that the first $z_i$ bits of the randomness are zero. This effectively reduces the success probability by a factor of $2^{-z_i}$, yielding a new error of $\varepsilon_i \cdot 2^{-z_i}$.

Formally, we have the following theorem:

**Theorem 5.2** ([Sta21, Theorem 6]). *Fix any $\ell$-round IOP with round-by-round soundness $\varepsilon_1, \ldots, \varepsilon_\ell$, Then, for any grinding parameters $z_1, \ldots, z_\ell$, after grinding the resulting IOP has round-by-round soundness error $\varepsilon_1 \cdot 2^{-z_1}, \ldots, \varepsilon_\ell \cdot 2^{-z_\ell}$.*

The most effective point to apply the grinding technique is during the query phase. Suppose we aim for a security level of 128 bits; this means that the underlying IOP must have a soundness error of at most $2^{-128}$. In the query phase, each query contributes a certain number of bits of security, depending on the code rate $\rho$ and the best-known proximity gap theorems. This optimization reduces both the size of the SNARK (since there are fewer authentication paths) and the verifier's computational complexity.

Another effective point to apply grinding is in the sumcheck rounds. In each round of the sumcheck protocol, the soundness error is roughly $d/q$, where $d$ is the polynomial degree and $q$

is the field size. If we work over a field of size $q = 2^{128}$ and aim for 128-bit security, this per-round error falls short of the target. How can we reconcile this gap?

One approach is to increase the field size by using a higher-degree field extension, thereby increasing $q$. While this achieves the desired soundness, it significantly increases the prover's runtime, the proof size, and the verifier's cost. A more efficient alternative is to introduce a grinding parameter $z_i = \log d$ for each round $i$ of the sumcheck protocol. This reduces the error to $(d/q) \cdot 2^{-z_i} = 1/q$, achieving full 128-bit security without enlarging the field. This approach preserves the same extension field and prover runtime, adding only a small *additive* cost for computing the proof of work—typically negligible compared to the overall proving time.

## 6 Parameter Selection

As discussed in the previous section, if the round-by-round knowledge error of the ZKIOP is $\varepsilon_K$, then the knowledge error of the protocol under the BCS-transformation is at most

$$T \cdot \varepsilon_K + \frac{9}{2} \cdot \frac{T^2}{2^{256}}$$

For the Longfellow ZK, the round-by-round knowledge error $\varepsilon_K$ of the corresponding ZKIOP is given in Theorem 4.1.

We analyze concrete security based on the parameters in the open source implementation:

- $p \simeq 2^{256}$
- $n_1 = 2945$ and $n_2 = 4096$
- We assume $N \leq 2^{13}$

In these conditions, for the target security of $2^{-115}$, we observe that for $T < 2^{128}$ we have $T/2^{256} < 2^{-128}$, so it suffices to ensure the RBR error $\varepsilon_K < 2^{-115}$. The round with the maximum RBR soundness error is after the columns are revealed, namely.

$$\varepsilon_K = \max \left\{ \binom{(\frac{1+\rho_1}{2})n_1}{t} \Big/ \binom{n_1}{t} + \binom{2\rho_1 n_1 - 2}{t} \Big/ \binom{n_1}{t}, \binom{(\frac{1+\rho_2}{2})n_2}{t} \Big/ \binom{n_2}{t} + \binom{2\rho_2 n_2 - 2}{t} \Big/ \binom{n_2}{t} \right\}$$

More precisely, by opening $t$ columns in the Ligero proof system, we target 115-bit security for the Longfellow ZK system, where the definition of $\lambda$-bit security says that for every $T$ and $\nu$ such that there exists a $T$-time adversary that succeeds in breaking soundness with probability $\nu$, referred to as "Advantage", it must hold that $\frac{T}{\nu} \geq 2^{\lambda}$.

In the following table, we collect values of the minimum $t$ (the number of opened columns) needed in Ligero to achieve $\varepsilon_K \leq 2^{-115}$ for several $\rho = \rho_1 = \rho_2$, which in turn will guarantee $\lambda > 115$.

The actual parameters chosen by Longfellow ZK are $\rho = 1/7$ and 140 queries, and thus, we are able to conclude that these parameters guarantee 115 bits of security.

|  | $\varepsilon_K \leq 2^{-115}$ |
|---|---|
| $\rho = 1/4$ | 166 |
| $\rho = 1/6$ | 145 |
| $\rho = 1/7$ | 140* |
| $\rho = 1/8$ | 136 |

Figure 2: Minimum $t$ to achieve soundness error $\varepsilon_K$ when Reed-Solomon codes of rate $\rho$ are used.

## References

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-Solomon proximity testing with super-fast verification. In *EUROCRYPT*, pages 214–243, 2025.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS*, pages 2087–2104, 2017.

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

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

[AY25] Gal Arnon and Eylon Yogev. Towards a white-box secure fiat-shamir transformation. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VI*, pages 27–56. Springer, 2025.

[Bab85] László Babai. Trading group theory for randomness. In *STOC*, pages 421–429, 1985.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '11, pages 106–115, 2001.

[BBH$^+$] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. On the (in)security of kilian-based snargs. In Dennis Hofheinz and Alon Rosen, editors, *TCC*.

[BCGM25] Sarah Bordage, Alessandro Chiesa, Ziyi Guan, and Ignacio Manzur. All polynomial generators preserve distance with mutual correlated agreement. *IACR Cryptol. ePrint Arch.*, page 2051, 2025.

[BCH$^+$25] Eli Ben-Sasson, Dan Carmon, Ulrich Haböck, Swastik Kopparty, and Shubhangi Saraf. On proximity gaps for reed–solomon codes. *IACR Cryptol. ePrint Arch.*, page 2055, 2025.

[BCI$^+$20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. In *FOCS*, 2020.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC*, pages 31–60, 2016.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BGK$^+$23] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michal Zajac. Fiat-shamir security of FRI and related snarks. *IACR Cryptol. ePrint Arch.*, page 1071, 2023.

[CCH$^+$19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *STOC*, pages 1082–1090, 2019.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[CHK25] Soham Chatterjee, Prahladh Harsha, and Mrinal Kumar. Deterministic list decoding of reed-solomon codes. *CoRR*, abs/2511.05176, 2025.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In *TCC*, pages 1–29, 2019.

[CS25] Elizabeth Crites and Alistair Stewart. On reed–solomon proximity gaps conjectures. *IACR Cryptol. ePrint Arch.*, page 2046, 2025.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024.

[DG25] Benjamin E. Diamond and Angus Gruen. On the distribution of the distances of random words. *IACR Cryptol. ePrint Arch.*, page 2010, 2025.

[FS24] Matteo Frigo and Abhi Shelat. Anonymous credentials from ECDSA. *IACR Cryptol. ePrint Arch.*, page 2010, 2024.

[FS25] Giacomo Fenzi and Antonio Sanso. Small-field hash-based snargs are less sound than conjectured. *IACR Cryptol. ePrint Arch.*, page 2197, 2025.

[GG25] Rohan Goyal and Venkatesan Guruswami. Optimal proximity gaps for subspace-design codes and (random) reed-solomon codes. *IACR Cryptol. ePrint Arch.*, page 2054, 2025.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, 2003.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

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

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In *ICALP*, pages 536–547, 2008.

[KRS25] Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. How to prove false statements: Practical attacks on fiat-shamir. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO*, pages 3–26, 2025.

[oB25] Trail of Bits. Security assessments of google longfellow library. 2025. https://github.com/trailofbits/publications/blob/master/reviews%2F2025-08-googlelongfellow-securityreview.pdf.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *STOC*, pages 49–62, 2016.

[Sta21] StarkWare. ethstark documentation. Technical Report 2021/582, Cryptology ePrint Archive, 2021. Preprint.

[Tha22] Justin Thaler. *Proofs, arguments, and zero-knowledge*. 2022. https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html.