STARK-based Signatures from the RPO Permutation

Shahla Atapoor[1], Cyprien Delpech de Saint Guilhem[2], Al Kindi[3]

[1]COSIC, KU Leuven, shahla.atapoor@kuleuven.be
[2]3MI Labs, Leuven, cyprien@3milabs.tech
[3]Polygon Labs, al.kindi@polygon.technology

3rd October 2024

Abstract

This work describes a digital signature scheme constructed from a zero-knowledge proof of knowledge of a pre-image of the Rescue Prime Optimized (RPO) permutation. The proof of knowledge is constructed with the DEEP-ALI interactive oracle proof combined with the Ben-Sasson–Chiesa–Spooner (BCS) transformation in the random oracle model. The EUF-CMA security of the resulting signature scheme is established from the UC-friendly security properties of the BCS transformation and the pre-image hardness of the RPO permutation.

The implementation of the scheme computes signatures in 13 ms and verifies them in 1 ms on a single core when the BCS transform is implemented with the Blake3 hash function. (The multi-threaded implementation signs in 9.2 ms and also verifies in 1 ms.) These speeds are obtained with parameters achieving 122 bits of average-case security for 2^{122}-bounded adversaries with access to at most 2^{64} signatures.

Contents

1 Introduction

Zero-knowledge scalable transparent arguments of knowledge (ZK-STARKs) are cryptographic proof systems [Ben+18] that have gained significant attention for their scalability and transparency, distinguishing themselves from other zero-knowledge systems. ZK-STARKs operate within the interactive oracle proof (IOP) framework in the random oracle model (ROM), eliminating the need for trusted setups and making them highly practical and efficient, especially in large-scale computational contexts.

The transparency of ZK-STARKs is due to their use of public-coin IOP systems and keyless hash functions which do not require trusted setups. These hash functions are also instrumental in converting interactive proofs into non-interactive ones. Techniques like the Fiat–Shamir transformation for sigma protocols [FS86], the Micali transformation for probabilistically checkable proofs (PCPs) [Mic00] and the BCS transformation for IOPs [BCS16] are based on this approach. In these scenarios, hash functions effectively link theoretical models to practical implementations, with the ROM providing a clear and effective framework for analyzing security.

The transparency of ZK-STARKs simplifies deployment by avoiding the need for complex multi-party cryptographic ceremonies to generate setup parameters. The hash-based transformations in ZK-STARKs further contribute to their efficient verification processes.

Beyond their transparency, ZK-STARKs also offer significant advantages in scalability. They enable faster proof generation and verification, particularly as the witness size increases, making them highly suitable for applications that involve large-scale computations. This combination of scalability and transparency makes ZK-STARKs a practical and effective solution for modern cryptographic requirements.

1.1 Recursive Zero-Knowledge Proofs

Recursive zero-knowledge proofs (ZKPs) represent an advanced class of proof systems wherein the validation of a proof is proven within another proof system [Bit+13]. To illustrate, consider a proof system $\Pi_1 = (\mathcal{P}_1, \mathcal{V}_1)$ where $\mathcal{P}_1$ is the prover and $\mathcal{V}_1$ is the verifier. In this setup, $\mathcal{P}_1$ generates a proof π_1, which is subsequently validated by $\mathcal{V}_1$.

In the context of recursive proofs, the role of $\mathcal{V}_1$ extends beyond mere validation. While verifying the initial proof π_1, $\mathcal{V}_1$ also produces a new proof π_2, within a second proof system Π_2, that attests to the correctness of the verification. Consequently, $\mathcal{V}_1$ operates both as the verifier of the initial proof and as the prover for the subsequent verifier.

The subsequent verifier, $\mathcal{V}_2$, receives and validates this new proof generated by $\mathcal{V}_1$. A notable advantage of this recursive approach is its enhanced efficiency: the time required to verify the second proof is substantially reduced compared to the verification of the initial proof, and the size of the second proof is smaller.

Additionally, recursive ZKPs are particularly noteworthy for their capability to handle multiple proofs simultaneously. In such cases, $\mathcal{V}_1$ can validate several different proofs and produce a single proof of validity for the next set of verifiers. These subsequent verifiers then only need to verify this single proof from $\mathcal{V}_1$, thereby simplifying the validation process and ensuring the correctness of all proofs without having to check each one individually. In this case, the potential of recursive ZKPs is further highlighted, showing significantly greater efficiency.

The power of recursive ZKPs becomes even more evident when we link them with cryptographic signature schemes. A signature scheme can be seen as a cryptographic primitive that verifies the authenticity and integrity of messages or transactions. In scenarios where multiple transactions or signatures need to be validated, recursive ZKPs offer a more efficient method. Instead of verifying each signature individually, which can be computationally expensive, recursive ZKPs allow for the aggregation of these signature verifications into a single, compact proof, if the signature scheme's verification algorithm is based on an underlying proof system. This "proof of signature verification" can then be passed to subsequent verifiers, reducing both the size of the data they need to process and the time required for verification.

In this sense, the recursive ZKP mechanism effectively functions as a signature system itself. It generates a single proof that guarantees the correctness of multiple verifications—similar to how a cryptographic signature validates the authenticity of a message. This idea highlights why designing a signature scheme built on recursive ZKPs is crucial. Not only does it improve efficiency, but it also streamlines the verification process for large-scale systems where multiple proofs or signatures need to be verified rapidly. The ability to generate a single proof from multiple verifications makes this approach particularly appealing for applications such as blockchains, financial transactions, and other distributed systems where efficiency and security are paramount.

1.2 Our Contribution

In this paper, we first describe a *zero-knowledge* and *round-by-round knowledge sound* IOP system. We then apply the BCS transformation to convert this IOP into a *UC-friendly zero-knowledge* and *UC-friendly knowledge sound* non-interactive argument (NArg) system using the results of Chiesa and Fenzi [CF24]. Our next step involves demonstrating that this NArg system, endowed with these UC-friendly properties, is sufficient to achieve *zero-knowledge* and *true-simulation knowledge soundness* (also known as *true-simulation extractability*). These properties are crucial as they form the prerequisites for constructing an EUF-CMA STARK-based signature scheme, using the approach described by Do Dinh [Do24].

2 Technical Overview

This paper describes an instance of the "signature scheme from zero-knowledge proof of a hard relation" paradigm. In reverse order, its various components are instantiated by:

1. Knowledge of a Rescue-Prime Optimised pre-image as the hard relation,
2. The DEEP-ALI interactive oracle proof of knowledge as the zero-knowledge proof system for the RPO pre-image hard relation,
3. The BCS transform of the DEEP-ALI IOP as the non-interactive proof system,
4. The construction of a signature scheme from a non-interactive zero-knowledge proof of knowledge.

2.1 The Rescue-Prime Optimised Hash Function

An instance of the Marvellous design strategy, Rescue-Prime Optimised (RPO) is an arithmetic hash function specifically desgined for the "Goldilocks field" $\mathbb{F}_p$, where $p = 2^{64} - 2^{32} + 1$ [Ash+22]. In this work only the parameter set targeting 128 bits of security will be used.

At this security level, the RPO hash function operates on a state $\vec{m} \in \mathbb{F}_p^{12}$ of 12 field elements, which is composed of a capacity $\vec{c} \in \mathbb{F}_p^4$ of length 4 and a rate $\vec{r} \in \mathbb{F}_p^8$ of length 8, such that $\vec{m} = \vec{c} \| \vec{r}$. In one invocation of the permutation, this state is acted upon sequentially by 7 round functions, which are identical in structure and differ only in the round constants that they add onto the state. We refer the reader to the RPO specification document for the description of the round constants; below we only describe the generic round functions.

Each of the round functions is composed of two sequential half-rounds whose operations are:

1. multiplication of the state vector by a 12×12 MDS circulant matrix,
2. addition of the half-round constant vector to the state vector,
3. either: (a) raising each state vector element to the power of $\alpha = 7 \in \mathbb{F}_p$, in the first half-round, or (b) raising each state vector element to the power of $\alpha^{-1} \in \mathbb{F}_p$, in the second half-round.

2.2 The DEEP-ALI Interactive Oracle Proof

The DEEP-ALI IOP protocol enables a Prover to convince a Verifier that a computation was performed correctly. In our case, the computation is the evaluation of the RPO hash function on a one-block message $\mathsf{sk} = (\mathsf{sk}_1, \cdots, \mathsf{sk}_4, 0, \cdots, 0) \in \mathbb{F}_p^8$ to produce a four-element hash value $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_4) = \mathsf{RPO}(\mathsf{sk})$.

Algebraic intermediate representation of the RPO computation. To be compatible with the DEEP-ALI IOP protocol, this computation is represented as an 8×12 table, where the i-th row is the state vector of the RPO hash function after the computation of the i-th round. To be proven as a valid representation of the computation $\mathsf{pk} \leftarrow \mathsf{RPO}(\mathsf{sk})$, while not revealing the value of sk, this table should fulfill the following public conditions:

1. its first row should have the form $(0, 0, 0, 0, *, *, *, *, 0, 0, 0, 0) \in \mathbb{F}_p^{12}$,
2. any two adjacent rows should be a valid input-output pair of the corresponding RPO round function,
3. its last row should have the form $(*, *, *, *, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3, \mathsf{pk}_4, *, *, *, *) \in \mathbb{F}_p^{12}$.

The DEEP-ALI IOP is then parameterised with an algebraic intermediate representation (AIR), which, together with some parameters, specifies multivariate polynomials that capture the public conditions that define "correctness" of a computation when represented as such a table, also known as an execution trace. In our case, the conditions described above can be expressed with 24 such constraint polynomials.

These constraint polynomials are defined in such a way that they evaluate to zero on the values of the execution trace if and only if the trace represents a correct computation of the RPO hash function.

DEEP-ALI proof of AIR satisfiability. Given an AIR as public input and a private execution trace that the Prover claims is correct, the DEEP-ALI protocol allows the Verifier to check that the trace does indeed satisfy the *constraint polynomials* defined by the AIR. This is achieved by the Prover first interpolating the columns of the trace table into *witness polynomials*, randomising them to preserve the secrecy of the initial and intermediary computation values, and committing to them as oracles.

Given these commitments, the Verifier provides combination randomness that the Prover uses to compute a *quotient polynomial*. This linearly combines the compositions of the constraint and witness polynomials in such a way that the satisfiability of the AIR is equated with the divisibility of these composed polynomials by public nullifier polynomials. The Prover then splits the quotient polynomial into *quotient segment polynomials* and commits to a randomisation of these as oracles. Finally, the Verifier performs two steps:

1. it requests the evaluations of the randomised witness and quotient segment polynomials at a random point to check the correctness of the quotient polynomial relative to the combination randomness and the constraint polynomials; and
2. it engages with the Prover in an instance of the FRI protocol using the oracle commitments to the randomised witness and quotient segment polynomials to verify that (a) the evaluations provided by the Prover are correct, and (b) the randomised witness and quotient segment polynomials are of appropriately low degree, which implies the divisibility of the combined constraint and witness polynomials by the nullifier polynomials.

If the check on the evaluations passes and the FRI protocol accepts, then the Verifier knows that, with high probability, the witness polynomials interpolated from the execution trace does indeed satisfy the AIR for the RPO permutation.

2.3 The BCS Transformation: from IOP to ZK-STARK

The DEEP-ALI IOP presented above and its FRI sub-protocol are useful abstractions but they rely on theoretical "oracles" that must be instantiated with concrete cryptographic primitives. Furthermore, the interaction that they require between the Prover and the Verifier must be done away with to construct a signature scheme where the signature can be publicly verified without participation from the signing party.

The Ben-Sasson–Chiesa–Spooner (BCS) transformation [BCS16] handles both of these requirements, constructing a succinct non-interactive argument of knowledge (SNARK) in the random oracle model (ROM) when given an arbitrary IOP. It does this with two main components: (a) a Merkle-tree commitment scheme in the ROM, together with opening proofs, to replace the Prover producing oracle commitments that the Verifier can query, and (b) a ROM-based replacement of the interactive Verifier, where each Prover message is queried to the ROM to generate the next Verifier random message, similarly to the Fiat–Shamir transformation.

Given an IOP with established zero-knowledge and knowledge soundness security for a given relation, the BCS transformation then produces a ZK-STARK for the same relation whose security properties can be related to those of the IOP and of the Merkle tree commitment scheme with some security loss owing to the use of the random oracle model and to the powers of the adversary in the corresponding security experiments

2.4 A Signature Scheme from a ZK-STARK

Finally, the ZK-STARK for proving knowledge of an RPO pre-image resulting from the BCS transformation of the DEEP-ALI IOP is used to construct a signature scheme where the secret signing key is the pre-image known by the Prover, the public verifying key is the image of the signing key under the RPO hash function, and the signature is an argument of knowledge where the message to be signed is appended to the instance of the RPO relation being proven. Including the message in the instance in this way does not change the RPO computation that is being proven, but it feeds the message into the chain of calls to the random oracle made during the BCS transformation, thus tying the proof to the individual message such that it cannot be re-purposed for a different message.

This framework for constructing signature schemes was first introduced by Katz and Vaikuntanathan [KV09], then refined and generalised by Dodis et al. [Dod+10], and then adapted to the random oracle model by Faust et al. [Fau+12]. In this work, we use the results recently obtained by Do Dinh [Do24] for the provable EUF-CMA security of these signature schemes.

3 An AIR for RPO

In this section, we first formally define algebraic intermediate representations (AIRs) and their satisfiability and then specify the specific AIR that we will use to verify the correctness of an RPO hash function execution trace.

3.1 Algebraic Intermediate Representation

As defined by Ben-Sasson et al. [Ben+18] and the StarkWare Team [Sta21], an AIR captures the step-by-step execution of a state machine by describing the set of constraints that the state of the registers of the machine must satisfy during correct execution. More precisely:

Definition 1. An *algebraic intermediate representation* (AIR) is a tuple $\mathsf{A} = (\mathbb{F}, \mathsf{w}, \tilde{\mathsf{w}}, \mathsf{h}, g, H, \mathsf{s}, \mathsf{Cset}, \mathsf{d})$ where:

- $\mathbb{F}$ is a finite field,
- $\mathsf{w} \in \mathbb{N}^{>0}$ specifies the execution trace width (i.e., the number of registers),
- $\tilde{\mathsf{w}} \in \mathbb{N}^{\geq 0}$ specifies the width of an optional additional "virtual trace" table which can contain public constants required in the execution of the state machine,[1]
- $\mathsf{h} \in \mathbb{N}^{>0}$ specifies the logarithm of the trace height (i.e., the number of steps in the trace),
- $g \in \mathbb{F}^*$ is a generator of the multiplicative group H, called the trace domain, such that $H \subset \mathbb{F}^*$ and $|H| = 2^{\mathsf{h}}$,
- $\mathsf{Cset} := \{\mathsf{C}_1, \ldots, \mathsf{C}_{\mathsf{s}}\}$ is a finite set of size s of constraints, where each constraint is a tuple of the form (Q_i, H_i) with:
  - $Q_i \in \mathbb{F}[\vec{X} \| \vec{Y} \| \vec{Z}]^{\leq \mathsf{d}_i}$ is a multivariate constraint polynomial of total degree at most d_i in $2\mathsf{w} + \tilde{\mathsf{w}}$ variables, where $\vec{X}$ and $\vec{Y}$, both of length w, are variables over adjacent rows of the execution trace, and $\vec{Z}$ of length $\tilde{\mathsf{w}}$, are optional variables over the additional public "virtual trace".
  - $H_i \subseteq H$ is a subset of the trace domain, called the i-th enforcement domain,
- $\mathsf{d} = \max_{i \in [\mathsf{s}]}\{\mathsf{d}_i\}$.

Remark 1. This definition restricts the transition constraints Q_i to apply only to two adjacent rows of the execution trace. This simplifies the description of the IOP and is enough for our purposes.

Definition 2. Given a width $\mathsf{w} \in \mathbb{N}^{>0}$, an *AIR assignment $\vec{p}$ of length* w is a tuple of polynomials:

$$\vec{p} = (p_1, \ldots, p_{\mathsf{w}}) \in (\mathbb{F}[X])^{\mathsf{w}}.$$

Given an AIR constraint $Q \in \mathbb{F}[\vec{X} \| \vec{Y} \| \vec{Z}]^{\leq \mathsf{d}_i}$ and virtual trace polynomials $\vec{z} \in (\mathbb{F}[X])^{\tilde{\mathsf{w}}}$, the *composition of Q, $\vec{z}$ and the assignment $\vec{p}$*, is the univariate polynomial denoted by $Q^{\vec{z}} \circ \vec{p} \in \mathbb{F}[X]$ that is the result of assigning $X_i \leftarrow p_i(X)$, $Y_i \leftarrow p_i(g \cdot X)$ and $Z_i \leftarrow z_i(X)$:

$$(Q^{\vec{z}} \circ \vec{p})(X) = Q(p_1(X), \ldots, p_{\mathsf{w}}(X), p_1(g \cdot X), \ldots, p_{\mathsf{w}}(g \cdot X), z_1(X), \ldots, z_{\tilde{\mathsf{w}}}(X)).$$

Via interpolation, the columns of the execution trace of a state machine for a given input can be encoded as polynomials in an AIR assignment, and the columns of the public virtual trace can be encoded as the virtual trace polynomials. We can then verify that this execution trace satisfies the constraints of a given AIR and suitable public virtual trace, by checking, for every i-th constraint, whether the i-th univariate polynomial resulting from the composition of the i-th constraint polynomial with the AIR assignment vanishes on the i-th enforcement domain. This is formally defined as follows:

[1] This is an instantiation of the concept of transparent oracles from IOPs where the values of the virtual trace table are part of the instance being represented by the AIR.

Definition 3. Given an AIR A, an AIR assignment $\vec{p} \in (\mathbb{F}[X])^{\mathsf{w}}$ *satisfies* A for virtual trace polynomials $\vec{z} \in (\mathbb{F}[X])^{\tilde{\mathsf{w}}}$ if and only if

$$\forall i \in [\mathsf{s}],\, x \in H_i \implies \left(Q_i^{\vec{z}} \circ \vec{p}\right)(x) = 0,$$

or equivalently,

$$\forall i \in [\mathsf{s}],\, \frac{Q_i^{\vec{z}} \circ \vec{p}}{v_{H_i}} \in \mathbb{F}[X],$$

where $v_{H_i} \in \mathbb{F}[X]$ is the unique monic polynomial of degree $|H_i|$ which vanishes on H_i, i.e., $v_{H_i}(X) := \prod_{h \in H_i} (X - h)$.

3.2 AIR Constraints for the RPO Permutation

Given a vector $\mathsf{pk} \in \mathbb{F}^4$, the following AIR, denoted $\mathsf{A}_{\mathsf{RPO}}(\mathsf{pk})$ describes one invocation of the RPO permutation, as described by Ashur et al. [Ash+22], which outputs pk as the hash value.

- $\mathbb{F}$ is the "Goldilocks field" $\mathbb{F}_p$ where $p = 2^{64} - 2^{32} + 1$,
- $\mathsf{w} = 12$, for the size 12 of the RPO state vector,
- $\tilde{\mathsf{w}} = 24$, for the two vectors of half-round constants used in the RPO permutation,
- $\mathsf{h} = 3$, for the $2^3 = 8$ rows encoding 7 executions of the round function,
- $g = 2^{24} = 16\,777\,216$, an 8-th root of unity in $\mathbb{F}$,
- $H = \langle g \rangle$,
- $\mathsf{Cset} := \mathsf{Cset}^{\mathsf{T}} \cup \mathsf{Cset}^{\mathsf{B}_f} \cup \mathsf{Cset}^{\mathsf{B}_l}$ consisting of
  - $|\mathsf{Cset}^{\mathsf{T}}| = 12$ transition constraints for the 7 round functions,
  - $|\mathsf{Cset}^{\mathsf{B}_f}| = 8$ boundary constraints on the first row, and
  - $|\mathsf{Cset}^{\mathsf{B}_l}| = 4$ boundary constraints on the last row,
- $\mathsf{s} = |\mathsf{Cset}| = 12 + 8 + 4 = 24$.
- $\mathsf{d} = 7$.

Given these parameters and the structure of the RPO permutation, we then formalise the different constraints as follows:

Transition constraints $\mathsf{Cset}^{\mathsf{T}}$. The RPO permutation $f_{\mathsf{RPO}} : \mathbb{F}^{\mathsf{w}} \to \mathbb{F}^{\mathsf{w}}$ can be factored into the following composition of maps

$$f_{\mathsf{RPO}}(\vec{m}) = f_{\mathrm{R}_7} \circ \cdots \circ f_{\mathrm{R}_1}(\vec{m}),$$

where $f_{\mathrm{R}_i} : \mathbb{F}^{\mathsf{w}} \to \mathbb{F}^{\mathsf{w}}$ expresses the i-th round function. Each of these round functions can itself be factored into the following composition of maps

$$f_{\mathrm{R}_i}(\vec{x}) = f_{\alpha^{-1}} \circ f_{\mathrm{ARC2}_i} \circ f_{\mathrm{MDS}} \circ f_{\alpha} \circ f_{\mathrm{ARC1}_i} \circ f_{\mathrm{MDS}}(\vec{x}),$$

with:

- $f_{\mathrm{MDS}}(\vec{x}) = M \cdot \vec{x}$ where M is the MDS circulant matrix of the RPO primitive,

- $f_{\mathrm{ARC1}_i}(\vec{x}) = \vec{x} + \vec{k}_{1,i}$ where $\vec{k}_{1,i}$ is the i-th vector of first half-round constants,
- $f_{\mathrm{ARC2}_i}(\vec{x}) = \vec{x} + \vec{k}_{2,i}$ where $\vec{k}_{2,i}$ is the i-th vector of second half-round constants,
- $f_\alpha(\vec{x}) : (x_i) \mapsto (x_i^\alpha)$ where $\alpha = 7$,
- $f_{\alpha^{-1}}(\vec{x}) : (x_i) \mapsto (x_i^{\alpha^{-1}})$ where $\alpha^{-1} = \frac{4(p-1)}{7} = 10\,540\,996\,611\,094\,048\,183 \in \mathbb{F}$.

The execution trace should start with the input state vector to the RPO permutation as its first (0-th) row and end with the output state vector as the last (7-th) row. For rows $i = 1, \ldots, 7$, each of these should contain the output of the i-th round function f_{R_i} applied to the previous row.

Let $\vec{x}$ and $\vec{y}$ be two consecutive rows of the execution trace, then it should hold that $\vec{y} = f_{\mathrm{R}_i}(\vec{x})$ for some $i = 1, \ldots, 7$. This is equivalent to requiring that

$$f_\alpha(\vec{y}) = f_{\mathrm{ARC2}_i} \circ f_{\mathrm{MDS}} \circ f_\alpha \circ f_{\mathrm{ARC1}_i} \circ f_{\mathrm{MDS}}(\vec{x}). \tag{1}$$

This can be expressed component-wise as requiring that $y_j^\alpha = p_j(\vec{x} \| \vec{z}_i)$ where, for $j \in [\mathsf{w}]$, $p_j \in \mathbb{F}[\vec{X} \| \vec{Z}]^7$ expresses the j-th component of the right hand side of Eq. (1) when $\vec{z}_i \in \mathbb{F}^{\tilde{\mathsf{w}}}$ takes the values of the two 12-element vectors of half-round constants for the i-th round from the virtual trace table,[2] i.e.,

$$p_j(\vec{x} \| \vec{z}_i) = (f_{\mathrm{ARC2}_i} \circ f_{\mathrm{MDS}} \circ f_\alpha \circ f_{\mathrm{ARC1}_i} \circ f_{\mathrm{MDS}}(\vec{x}))_j.$$

This means that $\mathsf{C}_j = (Q_j, H_j) \in \mathsf{Cset}^{\mathsf{T}}$, for $j = [\mathsf{w}]$, with

$$Q_j(\vec{X} \| \vec{Y} \| \vec{Z}) = p_j(\vec{X} \| \vec{Z}) - Y_j^\alpha,$$

and $H_j = \{g^i : i = 0, \ldots, 6\}$; this choice of H_j selects all rows, except for the last one, on which to enforce the constraints.

First row boundary constraints $\mathsf{Cset}^{\mathsf{B}_f}$. We enforce that the first four elements of the state, containing the capacity, as well as the last four elements, containing the second half of the rate, are all set to zero in the first row of the execution trace. This constrains the Prover to only include a four-element secret key as input to the RPO hash function. To this end, we define $\mathsf{C}_j = (Q_j, H_j) \in \mathsf{Cset}^{\mathsf{B}_f}$, for $j \in [8]$, with

$$\begin{aligned} Q_j(\vec{X} \| \vec{Y} \| \vec{Z}) &= X_j, && \text{for } j = 1, \ldots, 4, \\ Q_j(\vec{X} \| \vec{Y} \| \vec{Z}) &= X_{j+4}, && \text{for } j = 5, \ldots, 8, \end{aligned}$$

and $H_j = \{g^0\}$, for $j \in [8]$, to select only the first row to which apply these constraints.

Last row boundary constraints $\mathsf{Cset}^{\mathsf{B}_l}$ To enforce that the final state of the RPO permutation computation includes pk in the first half of its rate, we define the following constraints $\mathsf{C}_j = (Q_j, H_j) \in \mathsf{Cset}^{\mathsf{B}_l}$, for $j \in [4]$:

$$Q_j(\vec{X} \| \vec{Y} \| \vec{Z}) = X_{j+4} - \mathsf{pk}_j,$$

with $H_j = \{g^7\}$ to select only the last row to which to apply these constraints.

[2] That is, for the i-th round, elements $(z_i)_1, \ldots, (z_i)_{12}$ take the values of the first half-round constant vector, and elements $(z_i)_{13}, \ldots, (z_i)_{24}$ takes the values of the second half-round constant vector.

4 DEEP-ALI: an IOP for AIRs

This section describes the DEEP-ALI interactive oracle proof (IOP) system for AIRs as defined in Section 3.1.

Reed–Solomon code notation. For a finite field $\mathbb{F}$, a domain $D \subset \mathbb{F}^*$ of size $n \in \mathbb{N}^{>0}$ and a degree $k \in [n]$, we denote the Reed–Solomon code of length n and rate $\rho = \frac{k}{n}$ by

$$\mathsf{RS}[\mathbb{F}, D, k] = \left\{ (p(x))_{x \in D} : p \in \mathbb{F}[X]^{<k} \right\}.$$

4.1 Relations and Interactive Oracle Proofs

We refine the definition of relations to the case of AIRs for RPO computations which output a specific four-element digest.

Definition 4 (RPO Relation)**.** A *relation* $\mathcal{R}$ is a set of instance-witness pairs $(\mathbb{x}, \mathbb{w})$. Letting $\mathsf{A}_{\mathsf{RPO}}(\mathsf{pk})$ denote an AIR for an RPO computation which outputs $\mathsf{pk} \in \mathbb{F}^4$ as its digest, we define the *RPO-satisfiability relation*

$$\mathcal{R}_{\mathsf{RPO}} = \{(\mathsf{A}_{\mathsf{RPO}}(\mathsf{pk}), \vec{w} \in (\mathbb{F}[X])^{\mathsf{w}}) : \vec{w} \text{ satisfies } \mathsf{A}_{\mathsf{RPO}}(\mathsf{pk})\}_{\mathsf{pk} \in \mathbb{F}^4}.$$

We also formally define a k-round interactive protocol to prove satisfiability of an AIR.

Definition 5 (adapted from [CY24, Section 23.1])**.** An IOP for AIR-satisfiability is a tuple of algorithms $\mathsf{IOP} = (\mathcal{P}_{\mathsf{IOP}}, \mathcal{V}_{\mathsf{IOP}})$ that works as follows. The IOP Prover $\mathcal{P}_{\mathsf{IOP}}$ receives as input an instance $\mathbb{x} = \mathsf{A}$ and a witness $\mathbb{w} = \vec{w} \in (\mathbb{F}[X])^{\mathsf{w}}$ and the IOP Verifier $\mathcal{V}_{\mathsf{IOP}}$ receives as input the same instance $\mathbb{x}$.

Over $\mathsf{k} \in \mathbb{N}^{>0}$ rounds, they interact by, at each round $i \in [\mathsf{k}]$, $\mathcal{P}_{\mathsf{IOP}}$ sending an oracle proof string Π_i and $\mathcal{V}_{\mathsf{IOP}}$ optionally querying any of the previous oracles at any location and sending a message ρ_i.

After the interaction, the Verifier $\mathcal{V}_{\mathsf{IOP}}$ outputs a bit denoting whether it accepts ($\top$) or rejects ($\bot$) the Prover's claim that $\vec{w}$ satisfies A.

4.2 The DEEP-ALI IOP for AIR Satisfiability

We now detail the DEEP-ALI IOP protocol for AIR satisfiability [Sta21; Hab22] and discuss its parameters and its soundness error.

4.2.1 The protocol

Protocol 1 (DEEP-ALI IOP for AIR). Given an AIR instance $\mathbb{x} = \mathsf{A}$, let $\mathbb{G}$ be a degree-e extension field of $\mathbb{F}$, i.e., $[\mathbb{G} : \mathbb{F}] = e$.

Prover setup. Given an execution trace $T \in \mathbb{F}^{2^{\mathsf{h}} \times \mathsf{w}}$, let $w_1, \ldots, w_{\mathsf{w}} \in \mathbb{F}[X]^{<|H|}$ be the polynomials interpolating each of the (execution) trace columns over the trace domain H, i.e., it holds that $w_j(g^i) = T_{i,j}$. These w *witness polynomials* form the AIR assignment $\vec{w}$ that the Prover claims satisfies A.

Prover round 1. To protect its knowledge of the witness polynomial, the Prover will instead prove that the AIR A is satisfied by a randomised version $\hat{\vec{w}}$ of the witness assignment $\vec{w}$. We denote by $\hat{w}_1, \ldots, \hat{w}_{\mathsf{w}} \in \mathbb{F}[X]^{<\beta\cdot|H|}$ the *randomised witness polynomials* constructed by the Prover as

$$\hat{w}_i = w_i + v_H \cdot r_i,$$

for randomly sampled *witness randomising polynomials* $r_i \leftarrow\!\!\$\ \mathbb{F}[X]^{<(\beta-1)\cdot|H|}$, where the zero-knowledge blow-up parameter β is specified in Section 4.2.2.

The Prover next computes the Reed–Solomon encodings of the $\hat{w}_i$ polynomials, for $i \in [\mathsf{w}]$, over a domain D of size $n = k/\rho$ that is a coset of a smooth subgroup of $(\mathbb{F}^*, \times)$ such that $D \cap H = \emptyset$. The first oracle proof string Π_1 sent by $\mathcal{P}_{\mathsf{IOP}}$ is the w Reed–Solomon codewords:

$$\Pi_1 = \{(\hat{w}_i(x))_{x\in D}\}_{i\in[\mathsf{w}]}.$$

Verifier round 1. In response to the first proof string, the Verifier challenges the prover by sending random values $\lambda_i \leftarrow\!\!\$\ \mathbb{G}$, for $i \in [\mathsf{s}]$, one for each constraint in the AIR A:

$$\rho_1 = (\lambda_1, \ldots, \lambda_{\mathsf{s}}).$$

Prover round 2. With these challenges, the Prover linearly combines the compositions of the constraint polynomials with the randomised witness polynomials to compute the *quotient polynomial* as follows:

First, the Prover interpolates the $\tilde{\mathsf{w}}$ columns of the public virtual trace to obtain $\vec{z} \in (\mathbb{F}[X])^{\tilde{\mathsf{w}}}$. Then, it computes the quotient polynomial $q \in \mathbb{G}[X]^{\leq d_q}$ as:

$$q = \sum_{i=1}^{\mathsf{s}} \lambda_i \cdot \frac{Q_i^{\vec{z}} \circ \hat{w}}{v_{H_i}}, \tag{2}$$

where the degree $d_q = \max_{i\in\mathsf{s}}\{\mathsf{d}_i\,(\beta|H| - 1) - |H_i|\}$.

The Prover then splits q into the unique *quotient segment polynomials* $q_j \in \mathbb{G}[X]^{<\ell}$, $j \in [f]$, with degree parameter ℓ and number parameter f specified in Section 4.2.3, such that

$$q = \sum_{j=1}^{f} X^{\ell\cdot(j-1)} \cdot q_j. \tag{3}$$

Finally, the Prover independently and uniformly samples *quotient segment randomizing polynomials* $t_i \leftarrow\!\!\$\ \mathbb{G}[X]^{<d_t}$, $i \in [f-1]$, for some degree d_t specified in Section 4.2.3, and sets

$$\begin{aligned}
\hat{q}_1 &= q_1 + X^{\ell} \cdot t_1, & &\vdots \\
\hat{q}_2 &= q_2 + X^{\ell} \cdot t_2 - t_1, & \hat{q}_{f-1} &= q_{f-1} + X^{\ell} \cdot t_{f-1} - t_{f-2}, \\
&\vdots & \hat{q}_f &= q_f - t_{f-1}.
\end{aligned}$$

Note that this construction of the *randomised quotient segment polynomials* $\hat{q}_j$ implies

$$q = \sum_{j=1}^{f} X^{\ell\cdot(j-1)} \cdot \hat{q}_j, \tag{4}$$

and $\hat{q}_j \in \mathbb{G}[X]^{<\beta\cdot|H|}$ for $j \in [f]$. The second oracle proof string Π_2 sent by $\mathcal{P}_{\mathsf{IOP}}$ is then the set of f Reed–Solomon codewords:

$$\Pi_2 = \{(\hat{q}_j(x))_{x\in D}\}_{j\in[f]}.$$

Verifier round 2. In response, the Verifier sends to the Prover a random DEEP query $\rho_2 = r \leftarrow\!\!\$\ \mathbb{G} \setminus (D \cup H)$.

Prover round 3. The Prover responds with a third oracle proof string Π_3 which contains the evaluation claims:

$$\begin{aligned} (v_{i,1}, v_{i,2}) &= (\hat{w}_i(r), \hat{w}_i(g\cdot r)), && \text{for } i \in [\mathsf{w}], \\ v_j &= \hat{q}_j(r), && \text{for } j \in [f]. \end{aligned}$$

This proof string is read in full by the Verifier.

Batched-FRI for low-degree and evaluation checks. Finally, the Prover and Verifier engage in an instance of the batched-FRI protocol (Protocol 2) with the three proof strings sent by $\mathcal{P}_{\mathsf{IOP}}$, low degree parameter $d = \beta \cdot |H|$, and for n_{FRI} queries.

Output. The Verifier accepts if Protocol 2 accepts and if the evaluation claims contained in Π_3 satisfy the overall identity Eq. (2) evaluated at $X = r$, where the Verifier can reconstruct $q(r)$ from Eq. (4) and interpolate $\vec{z}$ for Eq. (2) from the public virtual trace.

Notation. From now on, we denote by $\mathsf{IOP}_{\mathsf{RPO}}$ the specialisation of the DEEP-ALI protocol above to instances of the RPO relation $\mathcal{R}_{\mathsf{RPO}}$.

4.2.2 Setting the zero-knowledge blow-up parameter β

The degree $(\beta-1)\cdot|H|$ of the witness randomizing polynomials r_i is set using Eq. (13) in [HK24], i.e., it is set such that

$$(\beta - 1) \cdot |H| \geq 2\,(e + n_{FRI})\,. \tag{5}$$

Note that, for a fixed rate ρ of the Reed–Solomon code, n_{FRI} also circularly depends on β through D. This means that one might need to increase the initial choice of n_{FRI} or of e after the first β is chosen using Eq. (5) because this initial choice of β might require a larger Reed–Solomon domain D and hence might lead to a decrease in the soundness of the IOP when all other parameters are fixed.

4.2.3 Setting the degree of the quotient segment polynomials

The degree d_t of the quotient segment randomizing polynomials t_i is determined using Eq. (12) in [HK24], that is

$$1 + n_{FRI} \leq d_t.$$

Let $n_q := d_q + 1$ be the number of coefficients in the quotient polynomial, then we can set the number of quotient segment polynomials to be

$$f = \left\lceil \frac{n_q}{\beta \cdot |H| - d_t} \right\rceil,$$

and using f we can define the number of coefficients in each segment polynomial to be

$$\ell = \left\lceil \frac{n_q}{f} \right\rceil.$$

This choice guarantees that each randomised quotient segment polynomial $\hat{q}_j$ has degree less than $\beta \cdot |H|$, for $j \in [f]$.

4.3 The FRI IOP for Low-Degree Testing

Protocol 2 (Zero-knowledge FRI batch evaluation proof [Ben+19]). Given oracle access to the $\mathsf{w} + f$ randomised polynomials

$$\hat{w}_1(X), \ldots, \hat{w}_{\mathsf{w}}(X) \in \mathbb{F}[X]^{<\beta\cdot|H|},$$
$$\hat{q}_1(X), \ldots, \hat{q}_f(X) \in \mathbb{G}[X]^{<\beta\cdot|H|},$$

and to their $2\mathsf{w} + f$ evaluation claims

$$(v_{i,1}, v_{i,2}) = (\hat{w}_i(z), \hat{w}_i(g \cdot z)), \qquad i = 1, \ldots, \mathsf{w},$$
$$v_j = \hat{q}_j(z), \qquad j = 1, \ldots, f,$$

the batched-FRI protocol proceeds as follows:

Prover round 1. The Prover samples a *mask polynomial* $R(X) \leftarrow\!\!\$\ \mathbb{G}[X]^{<\beta\cdot|H|-1}$ and sends its Reed–Solomon encoding to the Verifier as the first proof oracle Π_1^{FRI}.

Verifier round 1. The Verifier responds with *trace batching random coefficients* $\gamma_i \leftarrow\!\!\$\ \mathbb{G}$, for $i \in [\mathsf{w}]$, and *quotient segment batching random coefficients* $\delta_j \leftarrow\!\!\$\ \mathbb{G}$, for $j \in [f]$.

FRI low-degree and evaluation check. The Prover and Verifier then engage in an instance of the FRI protocol for the *batched polynomial*

$$R(X) + \sum_{i=1}^{\mathsf{w}} \gamma_i \cdot \left(\frac{\hat{w}_i(X) - v_{i,1}}{X - z} + \frac{\hat{w}_i(X) - v_{i,2}}{X - g \cdot z} \right) + \sum_{j=1}^{f} \delta_j \frac{\hat{q}_j(X) - v_j}{X - z}, \tag{6}$$

for the Reed–Solomon code $\mathsf{RS}[\mathbb{G}, D, \beta \cdot |H|]$ with n_{FRI} query rounds and agreement parameter $\alpha = \left(1 + \frac{1}{2m}\right) \cdot \sqrt{\rho}$, for some proximity parameter $m \geq 3$ and rate $\rho = \frac{\beta\cdot|H|}{|D|}$.

4.4 Security Errors of the DEEP-ALI IOP

We analyse the zero-knowledge and knowledge soundness errors of the DEEP-ALI protocol as presented above.

4.4.1 Zero-knowledge error

Theorem 7 of Haböck and Kindi [HK24] shows that, with the constraints on parameter choices described in Section 4.2.2, Protocol 1 is perfect honest-verifier zero-knowledge and hence the zero-knowledge error $\epsilon_{\mathsf{IOP}}^{\mathrm{ZK}}$ of the DEEP-ALI IOP is equal to zero.

4.4.2 Round-by-round knowledge soundness error

As we will see in Section 5, the stronger notion of state-restoration soundness is needed to use the BCS transformation to obtain a secure STARK. The following theorem allows us to reason instead about the DEEP-ALI IOP's round-by-round soundness error.

Theorem 1 ([CY24, Theorem 31.3.1]). *Let* $\mathsf{IOP} = (\mathcal{P}_{\mathsf{IOP}}, \mathcal{V}_{\mathsf{IOP}})$ *be a public-coin IOP with round complexity* k*. If IOP has round-by-round knowledge soundness error* $\epsilon^{\mathrm{RbR}}_{\mathsf{IOP}}$ *then IOP has straight-line state restoration knowledge soundness error* $\epsilon^{\mathrm{SR}}_{\mathsf{IOP}}$ *against a* t*-move malicious prover with*

$$\epsilon^{\mathrm{SR}}_{\mathsf{IOP}}(n,t) \leq (t+\mathsf{k}) \cdot \epsilon^{\mathrm{RbR}}_{\mathsf{IOP}}(n).$$

Then, in the list decoding regime, Theorem 5 in the updated analysis of [Sta21] can be adapted to give the following theorem in our setting:

Theorem 2 (DEEP-ALI RbR Soundness (list decoding)). *For any multiplicity parameter* $3 \leq m < \frac{k}{2}$ *where* $k = \beta \cdot |H|$*, the DEEP-ALI IOP (Protocol 1) has* $\mathsf{k} = 3 + |\vec{t}|$ *rounds, where* $\vec{t} = (t_i)$ *and* t_i *is the degree of the reduction map in the* i*-th round of the FRI commit phase, see Section 5.4 in [Sta21], and is round-by-round sound with error*

$$\epsilon^{\mathrm{RbR}}_{\text{DEEP-ALI}} = \max_i\{\epsilon_1, \cdots, \epsilon_{\mathsf{k}}\},$$

where:

1. $\epsilon_1 = \frac{L}{|\mathbb{G}|}$, *with* $L := \frac{m}{\rho - \frac{2\cdot m}{|D|}}$,
2. $\epsilon_2 = \frac{L^2 \cdot \max\{\mathsf{d}\cdot(k^+-1)+(k-1), k+(f-1)\cdot\ell+(k^+-1)\}}{|\mathbb{G}|-|D\cup H|}$, *with* $k^+ = \beta\cdot|H|+2$,
3. $\epsilon_3 = \frac{\left(m+\frac{1}{2}\right)^7}{3\cdot\sqrt{\rho}^3} \cdot \frac{|D|^2}{|\mathbb{G}|}$,
4. *For any* $4 \leq i \leq \mathsf{k}-1$: $\quad \epsilon_i = (t_{i-4}-1)\cdot\left(\epsilon_3 \prod_{j=0}^{i-4}\frac{1}{t_j^2} + \frac{2m+1}{\sqrt{\rho}}\cdot\frac{|D|+1}{|\mathbb{G}|}\right)$,
5. $\epsilon_{\mathsf{k}} = \left(\sqrt{\rho}\cdot\left(1+\frac{1}{2m}\right)\right)^{n_{\mathrm{FRI}}}$.

Moreover, the IOP is RbR knowledge sound with the same error.

Proof. The analysis of RbR soundness in the proof of Theorem 5 in [Sta21] applies as is, with the individual bounds on each ϵ_i updated, when needed, to take into account the changes induced by our decomposition of the quotient polynomial into quotient segment polynomials. Note that here, and also in the original analysis in [Sta21], the parameter m should be constrained so that $\eta := \sqrt{\rho}\cdot\left(1+\frac{1}{2m}\right) - \sqrt{\rho^+}$, where $\sqrt{\rho^+} := \frac{\beta\cdot|H|+2}{|D|}$, satisfies $\eta > 0$ in order to be able to apply Theorem 1 in [Sta21]. The upper bound in the statement of the Theorem guarantees exactly this.

As for RbR knowledge soundness, the analysis in Section 2.5 in [Blo+23] goes through, with the appropriate adaptation to the error bounds. □

Under the unique decoding regime, a different result can be stated.

Theorem 3 (DEEP-ALI RbR soundness (unique decoding)). *The DEEP-ALI IOP (Protocol 1) has* $\mathsf{k} = 3 + |\vec{t}|$ *rounds, where* $\vec{t} = (t_i)$ *and* t_i *is the degree of the reduction map in the* i*-th round of the FRI commit phase and is round-by-round sound with error*

$$\epsilon^{\mathrm{RbR}}_{\text{DEEP-ALI}} = \max_i\{\epsilon_1, \cdots, \epsilon_{\mathsf{k}}\}$$

where:

1. $\epsilon_1 = \frac{1}{|\mathbb{G}|}$,
2. $\epsilon_2 = \frac{\max\{\mathsf{d}\cdot(k^+-1)+(k-1), k+(f-1)\cdot\ell+(k^+-1)\}}{|\mathbb{G}|-|D\cup H|}$, *with* $k = \beta\cdot|H|$ *and* $k^+ = k+2$,
3. $\epsilon_3 = \frac{|D|}{|\mathbb{G}|}$,
4. *For any* $4 \le i \le \mathsf{k}-1$: $\epsilon_i = (t_{i-4}-1)\cdot\frac{|D|+1}{|\mathbb{G}|}$,
5. $\epsilon_{\mathsf{k}} = \left(\frac{1+\rho^+}{2}\right)^{n_{\mathrm{FRI}}}$, *where* $\rho^+ := \frac{\beta\cdot|H|+2}{|D|}$.

Moreover, the IOP is RbR knowledge sound with the same error.

Proof. The proof proceeds as in the proof of Theorem 2 with the main difference being that in the unique decoding regime we instead have the following bound

$$\max\{\mathcal{P}, \mathcal{P}^{\vec{r}}\} \le 1.$$

Moreover, we can take the agreement parameter to be the smallest possible for the Reed–Solomon code with rate ρ, respectively ρ^+, namely $1-\frac{1-\rho}{2}$, respectively $1-\frac{1-\rho^+}{2}$. This, together with the adaptation to the degree due to our segment polynomial decomposition, yields the expressions for ϵ_1 and ϵ_2.

Now, for ϵ_3, we invoke Theorem 9 instead of Theorem 7.4 from [Ben+20]. The rest of the argument goes through from then on unaltered.

For ϵ_i, $4 \le i \le \mathsf{k}-1$, we can use Theorem 10 in the analysis of Lemma 8.2 in [Ben+20] in order to derive the following bound

$$\Pr_{z^{(l)}}[E^{(l+1)}] \le (t_l - 1)\cdot\frac{|D|+1}{|\mathbb{G}|},$$

which is exactly the expression above. Finally, for ϵ_{k}, we note that the agreement parameter needs to be greater than or equal to one minus the unique decoding radius of the Reed–Solomon code with rate ρ^+, namely $1-\frac{1-\rho^+}{2}$. Hence, setting $\alpha = 1-\frac{1-\rho^+}{2} = \frac{1+\rho^+}{2}$ in the analysis of Theorem 8.3 in [Ben+20] yields the above expression on ϵ_{k}. □

5 The BCS Transformation: a ZK-STARK for AIRs

This section gathers definitions of the random oracle model and non-interactive arguments and results on the UC-security of ZK-STARKs produced with the BCS transformation. It also reduces the non-UC zero-knowledge and knowledge soundness security of such ZK-STARKs to their UC versions.

5.1 Definitions of Schemes and Security Notions

In this paper, we consider non-interactive argument systems in the ROM. This model is a framework where algorithms can make arbitrary queries to an oracle f sampled at random; such algorithms are called *oracle algorithms*.

Definition 6 (Random oracle [CY24])**.** For $\lambda \in \mathbb{N}$, let $\mathcal{U}(\lambda)$ denote the uniform distribution over all functions of the form $f : \{0,1\}^* \to \{0,1\}^\lambda$. A function f sampled from $\mathcal{U}(\lambda)$ at random, is called *a random oracle (with output size λ).*

To realize certain protocols (such as provably zero-knowledge argument systems), it is necessary to enable certain oracle algorithms to program the random oracle on chosen inputs.

Definition 7 (Explicitly programmable ROM)**.** The *explicitly programmable ROM*, given a random oracle f, allows an oracle algorithm $\mathcal{S}^f$ to output a tuple μ of input-output pairs which can be used to program f which is then denoted $f[\mu]$; that is, when queried on an input that appears in μ, $f[\mu]$ will return the corresponding output instead of the value initially determined by f.

If a random oracle f is used in multiple places, we denote by $f \leftarrow f[\mu]$ the action of programming f with μ such that future calls to f are answered by $f[\mu]$, instead. This action can be repeated for several programming tuples.

Definition 8 (Non-interactive argument in the ROM [CY24])**.** Let $f \leftarrow \mathcal{U}(\lambda)$ be a random oracle. A *non-interactive argument (NARG) in the ROM* is a tuple $\mathsf{NArg} = (\mathcal{P}_{\mathsf{NArg}}, \mathcal{V}_{\mathsf{NArg}})$ where $\mathcal{P}^f_{\mathsf{NArg}}$ is an oracle algorithm that receives as inputs an instance $\mathbb{x}$ and a witness $\mathbb{w}$ and outputs an argument string π, and where $\mathcal{V}^f_{\mathsf{NArg}}$ is an oracle algorithm that receives as inputs the instance $\mathbb{x}$ and the argument string π and outputs a bit denoting whether to accept ($\top$) or reject ($\bot$) the argument.

5.2 The BCS Transformation and its UC Security

The BCS transformation converts a public-coin IOP into a transparent and succinct non-interactive argument system [BCS16, Construction 25.1.1, pp. 248–9].

In Chiesa and Fenzi's work [CF24], the security of the BCS transformation, with random oracle output length $\lambda \in \mathbb{N}$ and privacy parameter $s \in \mathbb{N}$, is proven in the UC framework against $(t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}})$-budget environments which can make (1) t_{q} *sampling* queries to the random oracle, (2) t_{p} *programming* queries to the random oracle, (3) $t_{\mathcal{P}}$ queries to the prover, and (4) $t_{\mathcal{V}}$ queries to the verifier. Their results can then be summarised as follows.

Theorem 4 ([CF24, Lemma 9.9, 9.12])**.** *Let* IOP *be a public-coin IOP with proof length* l*, query complexity* q *and round complexity* k*.*

- *If* IOP *is honest-verifier zero-knowledge with error* $\epsilon^{\mathrm{ZK}}_{\mathsf{IOP}}$*, then* $\mathsf{BCS}[\mathsf{IOP}, \lambda, s]$ *is UC-friendly zero-knowledge with error*

$$\epsilon^{\text{UC-ZK}}_{\mathsf{BCS}}(\lambda, s, t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}}) = t_{\mathcal{P}} \cdot \left(\frac{t_{\mathsf{q}} + t_{\mathsf{p}}}{2^s} + \epsilon^{\text{UC-Hid}}_{\mathsf{MT}}(\lambda, s, \mathsf{l}, \mathsf{q}, t_{\mathsf{q}}, t_{\mathsf{p}}, \mathsf{k}) + \epsilon^{\mathrm{ZK}}_{\mathsf{IOP}} \right),$$

  where $\epsilon^{\text{UC-Hid}}_{\mathsf{MT}}$ *denotes the UC-friendly hiding error of* MT*, the Merkle tree commitment scheme [CF24, Lemma 7.12].*

- *If* IOP *is state-restoration knowledge sound with error* $\epsilon^{\mathrm{SR}}_{\mathsf{IOP}}$*, then* $\mathsf{BCS}[\mathsf{IOP}, \lambda, s]$ *is UC-friendly knowledge sound with error*

$$\epsilon^{\text{UC-KS}}_{\mathsf{BCS}}(\lambda, s, t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}}) = t_{\mathcal{V}} \cdot \left(\epsilon^{\mathrm{SR}}_{\mathsf{IOP}} + \epsilon^{\text{UC-Ext}}_{\mathsf{MT}} \right),$$

  where $\epsilon^{\text{UC-Ext}}_{\mathsf{MT}}$ *denotes the UC-friendly extraction error of* MT*, the Merkle tree commitment scheme [CF24, Lemma 7.15].*

5.3 Zero-Knowledge Security

To use the results of Do Dinh [Do24] for the security of our signature scheme built with a non-interactive argument scheme produced by the BCS transformation, we must first show that it satisfies the following definition of zero-knowledge.

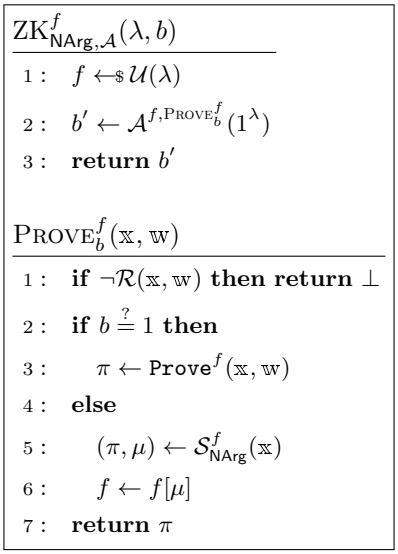

Figure 1: The argument zero-knowledge game ZK^f in the ROM.

Definition 9 (adapted from [Do24, Definition 2.8])**.** Let $f \leftarrow \mathcal{U}(\lambda)$ be a random oracle. A non-interactive argument NArg in the ROM is *zero-knowledge in the ROM* with error $\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}}$ if there exists a PPT argument simulator $\mathcal{S}^f_{\mathsf{NArg}}$ such that, for every PPT algorithm $\mathcal{A}$ making t_{q} queries to f and $t_{\mathcal{P}}$ queries to $\textsc{Prove}$,

$$|\Pr[\mathrm{ZK}_{\mathsf{NArg},\mathcal{A}}(\lambda, 1) = 1] - \Pr[\mathrm{ZK}_{\mathsf{NArg},\mathcal{A}}(\lambda, 0) = 1]| \leq \epsilon^{\mathrm{ZK}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}})$$

and game ZK is defined in Fig. 1.

However, the results of Chiesa and Fenzi [CF24] (restated in Theorem 4) prove the (in appearance stronger) UC-friendly zero-knowledge property, defined as follows.

Definition 10 (adapted from [CF24, Definition 5.10])**.** Let $f \leftarrow \mathcal{U}(\lambda)$ be a random oracle. A non-interactive argument NArg in the ROM is *UC-friendly zero-knowledge in the ROM* with error $\epsilon^{\text{UC-ZK}}_{\mathsf{NArg}}$, for the UC-ZK game defined in Fig. 2, if there exists a PPT argument simulator $\mathcal{S}^f_{\mathsf{NArg}}$ such that

$$|\Pr[\text{UC-ZK}_{\mathsf{NArg},\mathcal{A}}(\lambda, 1) = 1] - \Pr[\text{UC-ZK}_{\mathsf{NArg},\mathcal{A}}(\lambda, 0) = 1]| \leq \epsilon^{\text{UC-ZK}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}}).$$

We show that the UC-friendly notion of zero-knowledge is indeed stronger than the one used by Do Dinh [Do24] by proving the following theorem:

Theorem 5. *If* NArg *is UC-friendly zero-knowledge with error* $\epsilon^{\text{UC-ZK}}_{\mathsf{NArg}}$ *in the ROM (Definition 10), then* NArg *is zero-knowledge in the ROM (Definition 9) with*

$$\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}}) = \epsilon^{\text{UC-ZK}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, 0, t_{\mathcal{P}}, 0).$$

Proof. We define the simulator of the ZK game, $\mathcal{S}^f$, to be the argument simulator $\mathcal{S}^f$ from the UC-ZK game, which exists and is PPT by the assumption that NArg is UC-friendly zero-knowledge in the ROM. Let $\mathcal{B}$ be a reduction adversary against the *UC-friendly zero-knowledge* game, using an arbitrary PPT adversary $\mathcal{A}$ against the argument simulation distinguishing game ZK.

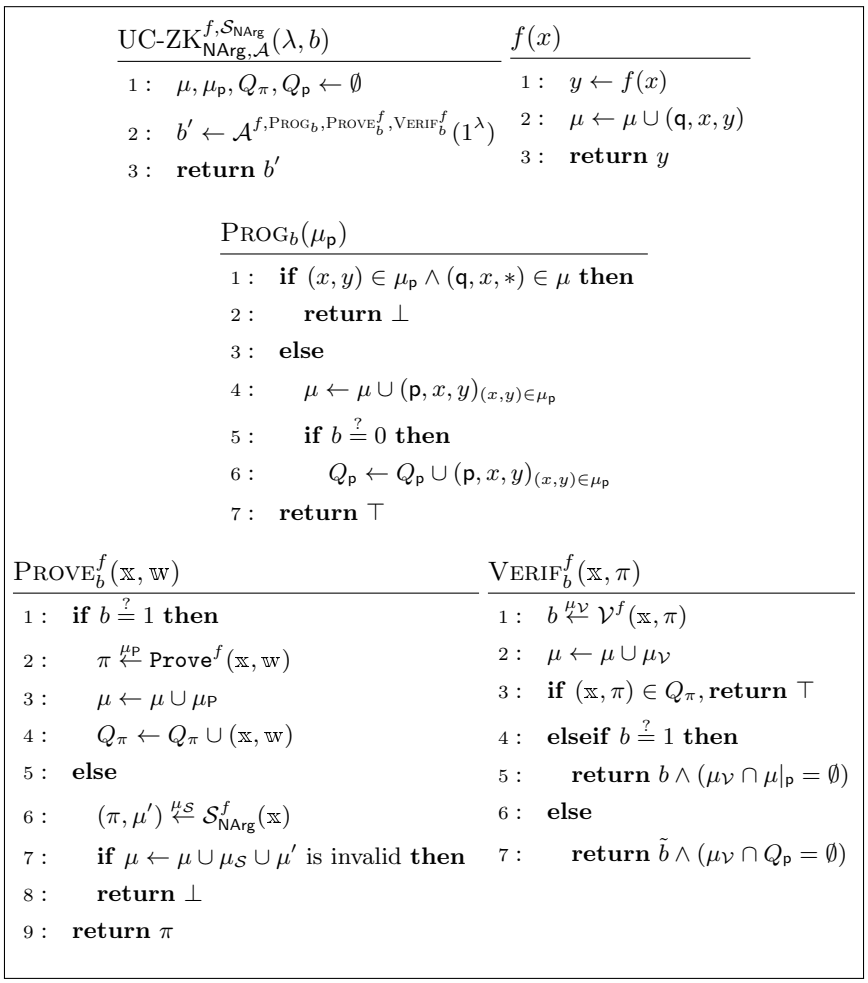

Figure 2: The UC-friendly zero-knowledge game UC-ZK^f in the ROM.

The reduction $\mathcal{B}$ initializes adversary $\mathcal{A}$ with input λ. Then, $\mathcal{A}$ receives the well-formed input from $\mathcal{B}$ and a well-formed output y from its oracle f simulated by $\mathcal{B}$ using its own f oracle; this implies that $t_{\mathsf{q}}^{\text{UC-ZK}} = t_{\mathsf{q}}^{\text{ZK}}$. Since $\mathcal{A}$ is not allowed to make programming queries to the random oracle in the ZK game, it holds that $t_{\mathsf{p}}^{\text{UC-ZK}} = 0$ for $\mathcal{B}$.

Next, in the ZK game, $\mathcal{A}$'s oracle $\textsc{Prove}_b^f$ is answered with $\mathcal{B}$'s prover query with the given $(\mathbb{x}, \mathbb{w})$. This implies that $t_{\mathcal{P}}^{\text{UC-ZK}} = t_{\mathcal{P}}^{\text{ZK}}$. In the case where $b = 1$ in the ZK game, $\mathcal{B}$ runs its real prover $\mathtt{Prove}_{\text{UC-ZK}}^f$ and returns the proof π as the well-formed output of the oracle $\textsc{Prove}_b^f$ to $\mathcal{A}$. In the case where $b = 0$ in the ZK game, $\mathcal{B}$ runs its simulator $\mathcal{S}_{\text{UC-ZK}}^f(\mathbb{x})$ and returns the proof π as the well-formed output of the oracle $\textsc{Prove}_b^f$ to $\mathcal{A}$.

When $\mathcal{A}$ eventually outputs b', the output of $\mathcal{B}$ is also set to b'; given that, $\mathcal{A}$'s queries are answered indistinguishably from the ZK we can conclude that the reduction $\mathcal{B}$ has the same probability to win the UC-friendly zero-knowledge game as $\mathcal{A}$ does of winning the ZK game, and we, therefore, have $\epsilon_{\mathsf{NArg}}^{\text{ZK}} = \epsilon_{\mathsf{NArg}}^{\text{UC-ZK}}$. □

Corollary 1. If IOP is public-coin and honest-verifier zero-knowledge with error $\epsilon_{\mathsf{IOP}}^{\text{ZK}}$, then the non-interactive argument $\mathsf{BCS}[\mathsf{IOP}, \lambda, s]$ is zero-knowledge in the ROM against $(t_{\mathsf{q}}, t_{\mathcal{P}})$-adversaries with error

$$\epsilon_{\mathsf{BCS}}^{\text{ZK}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}}) = t_{\mathcal{P}} \cdot \left(\frac{t_{\mathsf{q}}}{2^s} + \epsilon_{\mathsf{MT}}^{\text{UC-Hid}} + \epsilon_{\mathsf{IOP}}^{\text{ZK}} \right).$$

Proof. The corollary follows from Theorems 4 and 5. □

5.4 True-Simulation Knowledge Soundness

Definition 11 (adapted from [Do24, Definition 3.3])**.** Let $f \leftarrow \mathcal{U}(\lambda)$ be a random oracle. A non-interactive argument NArg in the ROM has *true-simulation knowledge soundness in the ROM* with error $\epsilon_{\mathsf{NArg}}^{\text{tSKS}}$ against $(t_{\mathsf{q}}, t_{\mathcal{P}})$-adversaries relative to the PPT simulator $\mathcal{S}_{\mathsf{NArg}}^f$ if there exists a deterministic polynomial-time extractor algorithm $\mathcal{E}_{\mathsf{NArg}}^f$ such that

$$\Pr[\text{tSKS}_{\mathsf{NArg},\mathcal{R},\mathcal{A}}^f(\lambda)] \leq \epsilon_{\mathsf{NArg}}^{\text{tSKS}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}}),$$

where the game tSKS is defined in Fig. 3.

Definition 12 ([CF24])**.** Let $f \leftarrow \mathcal{U}(\lambda)$ be a random oracle. A non-interactive argument NArg has *UC-friendly knowledge soundness in the ROM* with respect to a simulator $\mathcal{S}_{\mathsf{NArg}}^f$ with error $\epsilon_{\mathsf{NArg}}^{\text{UC-KS}}$ if there exists a PPT extractor $\mathcal{E}_{\mathsf{NArg}}^f$ such that, for every $(t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}})$-budget adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{advWin} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{advWin} \leftarrow \mathsf{UC\text{-}KS}_{\mathcal{S},\mathcal{E}}^f(n, \mathcal{A}) \end{array} \right] \leq \epsilon_{\mathsf{NArg}}^{\text{UC-KS}}(\lambda, s, t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}}),$$

where the $\mathsf{UC\text{-}KS}$ game is defined in Fig. 4.

Theorem 6. *If* NArg *is UC-friendly knowledge sound in the ROM with error* $\epsilon_{\mathsf{NArg}}^{\text{UC-KS}}$*, then* NArg *is true-simulation knowledge sound in the ROM with error*

$$\epsilon_{\mathsf{NArg}}^{\text{tSKS}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}}) = \epsilon_{\mathsf{NArg}}^{\text{UC-KS}}(\lambda, s, t_{\mathsf{q}}, 0, t_{\mathcal{P}}, 1).$$

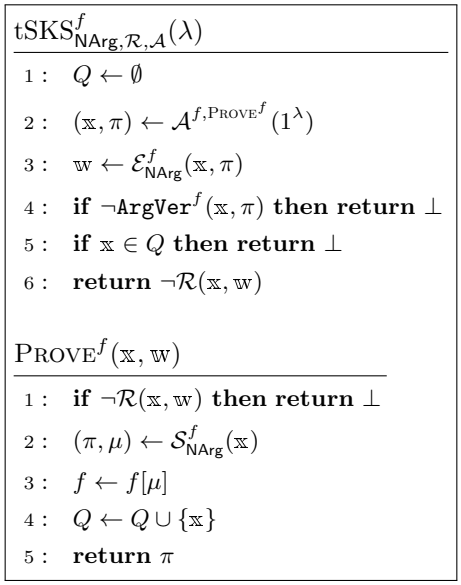

Figure 3: The true-simulation knowledge soundness game tSKS^f in the ROM.

Proof. We define the extractor of game tSKS as $\mathcal{E}^f_{\mathrm{tSKS}}$ to directly call the extractor $\mathcal{E}^f_{\mathsf{UC\text{-}KS}}$ from game $\mathsf{UC\text{-}KS}$, which exists and is PPT by the assumption that NArg is UC-friendly knowledge sound in the ROM. We also define the simulator of game tSKS as $\mathcal{S}^f_{\mathrm{tSKS}}$ to directly call the simulator $\mathcal{S}^f_{\mathsf{UC\text{-}KS}}$ from game $\mathsf{UC\text{-}KS}$, which exists and is PPT by the assumption that NArg is UC-friendly knowledge sound in the ROM.

Let $\mathcal{B}$ denote a reduction adversary against the UC-friendly knowledge soundness game, using an arbitrary PPT adversary $\mathcal{A}$ against the argument true-simulation knowledge soundness game.

The reduction $\mathcal{B}$ passes the input λ directly to the adversary $\mathcal{A}$ as an input. When it makes queries to the random oracle f, $\mathcal{A}$ receives the well-formed output from $\mathcal{B}$ which it obtains from its own oracle f; this implies that $t^{\text{UC-KS}}_{\mathsf{q}} = t^{\text{KS}}_{\mathsf{q}}$. Also, since $\mathcal{A}$ is not allowed to make programming queries in the tSKS, it holds that $t^{\text{UC-KS}}_{\mathsf{p}} = 0$ for $\mathcal{B}$.

Next, in the tSKS game, $\mathcal{A}$'s proving oracle $\textsc{Prove}^f$ uses $\mathcal{B}$'s own prover query with the given instance-witness pair $(\mathbb{x}, \mathbb{w})$. To answer such a query, the $\mathsf{UC\text{-}KS}$ game runs $\mathcal{S}^f_{\text{UC-KS}}(\mathbb{x})$ and returns to $\mathcal{B}$ the proof π which in turn it returns to $\mathcal{A}$ as the output of the oracle $\textsc{Prove}^f$. Because the simulator is the same in both game, $\mathcal{B}$'s answer to $\mathcal{A}$'s query is distributed identically to the tSKS.

Finally, when $\mathcal{A}$ outputs $(\mathbb{x}, \pi)$ and halts in the tSKS game, the reduction $\mathcal{B}$ submits a verifier query to the $\mathsf{UC\text{-}KS}$ game with input $(\mathbb{x}, \pi)$; this implies that $t^{\text{UC-KS}}_{\mathcal{V}} = 1$.

Assuming that $\mathcal{A}$ wins the tSKS game relative to the extractor $\mathcal{E}^f_{\text{UC-KS}}$ implies that the pair $(\mathbb{x}, \pi)$ it returned (1) passes verification, (2) does not contain an instance $\mathbb{x}$ that was previously queried and (3) causes the extractor to output a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. These three properties of the pair $(\mathbb{x}, \pi)$ imply that $\mathcal{B}$'s query to the verification oracle causes $\mathsf{advWin} \leftarrow \top$ (in particular since $Q_{\mathsf{p}} = \emptyset$ due to the absence of programming queries), and therefore imply that

$$\epsilon^{\text{tSKS}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}}) = \epsilon^{\text{UC-KS}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, 0, t_{\mathcal{P}}, 1).$$

□

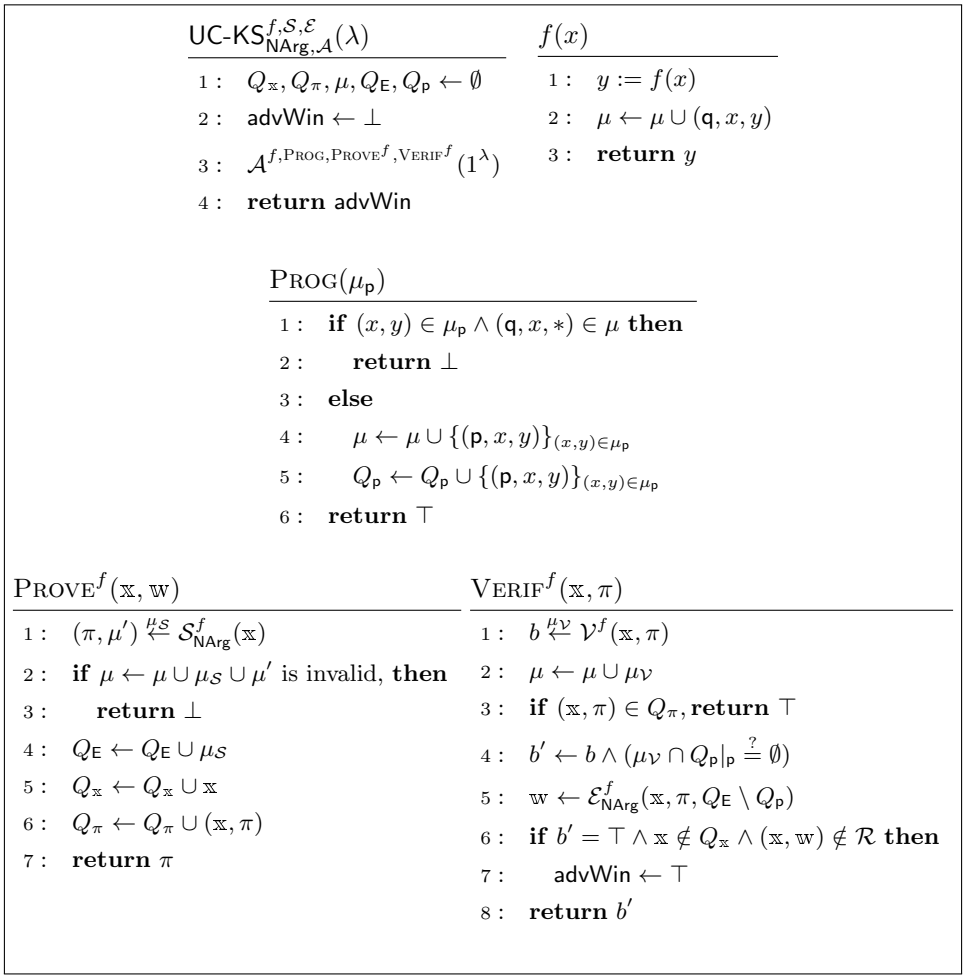

Figure 4: The UC-friendly knowledge soundness game UC-KS^f in the ROM.

$\mathrm{EUF\text{-}CMA}^{f}_{\mathsf{Sig},\mathcal{A}}(\lambda)$	$\mathrm{SIGN}^{f}_{\mathsf{sk}}(\mathsf{m})$
1 : $Q \leftarrow \emptyset$	1 : $\sigma \leftarrow \mathsf{Sig.Sign}^{f}(1^{\lambda}, \mathsf{sk}, \mathsf{m})$
2 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Sig.KeyGen}(1^{\lambda})$	2 : $Q \leftarrow Q \cup \{\mathsf{m}\}$
3 : $(\mathsf{m}, \sigma) \leftarrow \mathcal{A}^{f, \mathrm{SIGN}^{f}_{\mathsf{sk}}}(1^{\lambda}, \mathsf{vk})$	3 : **return** σ
4 : **return** $\mathsf{Sig.SigVer}(1^{\lambda}, \mathsf{vk}, \mathsf{m}, \sigma) \wedge \mathsf{m} \overset{?}{\notin} Q$	

Figure 5: The unforgeability game $\mathrm{EUF\text{-}CMA}^{f}_{\mathsf{Sig},\mathcal{A}}$ in the random oracle model.

Corollary 2. If IOP is straightline state restoration knowledge sound, then the non-interactive argument system $\mathsf{BCS}[\mathsf{IOP}, \lambda, s]$ has true-simulation knowledge soundness in the ROM with error

$$\epsilon^{\mathrm{tSKS}}_{\mathsf{BCS}}(\lambda, s, t_{\mathsf{q}}, t_{\mathcal{P}}) = \epsilon^{\mathrm{SR}}_{\mathsf{IOP}} + \epsilon^{\mathrm{UC\text{-}Ext}}_{\mathsf{MT}}.$$

Proof. The corollary follows from Theorems 4 and 6. □

6 A Signature Scheme from RPO Pre-Images

6.1 Hard Relations and RPO Pre-Image Resistance

Definition 13 ([Do24, Definition 4.4])**.** Given a relation $\mathcal{R}$, a probabilistic algorithm $G_{\mathcal{R}}$ in an *instance-witness* sampler for $\mathcal{R}$ if, for every instance size bound $n \in \mathbb{N}$, $G_{\mathcal{R}}(1^n)$ outputs a valid instance-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with $|\mathbb{x}| \leq n$.

Definition 14. An instance-witness sampler $G_{\mathcal{R}}$ for a relation $\mathcal{R}$ *has hardness error* $\epsilon_{G_{\mathcal{R}}}$ if for every security parameter $\lambda \in \mathbb{N}$ and PPT algorithm $\mathcal{A}$,

$$\Pr\left[(\mathbb{x}, \mathbb{w}') \in \mathcal{R} \;\middle|\; \begin{array}{c}(\mathbb{x}, \mathbb{w}) \leftarrow G_{\mathcal{R}}(1^{\lambda}) \\ \mathbb{w}' \leftarrow \mathcal{A}(1^{\lambda}, \mathbb{x})\end{array}\right] \leq \epsilon_{G_{\mathcal{R}}}(\lambda).$$

6.2 Scheme and Security Definitions

Definition 15 (Signature scheme)**.** Let $f \leftarrow \mathcal{U}(\kappa)$ be a random oracle. For security parameter $\lambda \in \mathbb{N}$ and message length $m \in \mathbb{N}$, a *signature scheme in the ROM* is a tuple of algorithms $\mathsf{Sig}[\lambda, m] = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SigVer})$ where:

- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}[\lambda]()$ generates signing-verifying key pairs,
- $\sigma \leftarrow \mathsf{Sign}^{f}[\lambda, m](\mathsf{sk}, \mathsf{m})$ generates signatures for messages $\mathsf{m} \in \{0,1\}^m$,
- $\{\top, \bot\} \ni \mathsf{b} \leftarrow \mathsf{SigVer}^{f}[\lambda, m](\mathsf{vk}, \mathsf{m}, \sigma)$ accepts or rejects a given signature for the given message and verifying key.

Definition 16 (Unforgeable signature scheme)**.** Let $f \leftarrow \mathcal{U}(\kappa)$ be a random oracle. A signature scheme $\mathsf{Sig}[\lambda, m]$ in the ROM is *unforgeable under chosen message attack* (EUF-CMA) with error $\epsilon^{\mathrm{EUF}}_{\mathsf{Sig}}$ in the ROM if, for every PPT algorithm $\mathcal{A}$ with RO query complexity $t_{\mathsf{q}} \in \mathbb{N}$ and signing oracle query complexity $t_{\mathsf{S}} \in \mathbb{N}$,

$$\Pr[\mathrm{EUF\text{-}CMA}^{f}_{\mathsf{Sig},\mathcal{A}}] \leq \epsilon^{\mathrm{EUF}}_{\mathsf{Sig}}(\lambda, m, t_{\mathsf{q}}, t_{\mathsf{S}})$$

where the EUF-CMA game is defined in Fig. 5

Let $\mathsf{NArg}_{\mathsf{RPO}}$ be the non-interactive argument system for $\mathcal{R}_{\mathsf{RPO},m}$ in the ROM of Section 5. Given a random oracle $f \leftarrow \mathcal{U}(\lambda)$, the RPO-based signature scheme $\mathsf{Sig}_{\mathsf{RPO}}[\lambda, m]$ is defined by:

- $\mathsf{Sig}_{\mathsf{RPO}}.\mathsf{KeyGen}[\lambda]() \to (\mathsf{sk} \leftarrow\!\!\$\, \mathbb{F}^4, \mathsf{pk} \leftarrow \mathsf{RPO}(\mathsf{sk}\|0^4))$,
- $\mathsf{Sig}_{\mathsf{RPO}}.\mathsf{Sign}^f[\lambda, m](\mathsf{sk}, \mathsf{m}) \to \mathsf{NArg}_{\mathsf{RPO}}.\mathcal{P}^f(1^\lambda, (\mathsf{RPO}(\mathsf{sk}\|0^4), \mathsf{m}), \mathsf{sk})$,
- $\mathsf{Sig}_{\mathsf{RPO}}.\mathsf{SigVer}^f[\lambda, m](\mathsf{pk}, \mathsf{m}, \sigma) \to \mathsf{NArg}_{\mathsf{RPO}}.\mathcal{V}^f(1^\lambda, (\mathsf{pk}, \mathsf{m}), \sigma)$.

Figure 6: The RPO-based signature scheme

6.3 The Signature Construction

We specialise the signature scheme construction presented by Do Dinh [Do24, Section 4.3] to the RPO relation from Definition 4,

$$\mathcal{R}_{\mathsf{RPO},m} = \{((\mathbb{x}, \mathsf{m}), \mathbb{w}) : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{RPO}} \wedge \mathsf{m} \in \{0,1\}^m\},$$

together with the non-interactive argument system from Section 5. The resulting scheme $\mathsf{Sig}_{\mathsf{RPO}}$ is presented in Fig. 6.

Theorem 7 ([Do24, Theorem 4.7])**.** *If $\mathcal{R}_{\mathsf{RPO}}$ has hardness error $\epsilon^{\mathrm{Rel}}_{\mathsf{RPO}}$ and $\mathsf{NArg}_{\mathsf{RPO}}$ has computational true-simulation knowledge soundness error $\epsilon^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}$ and computational zero-knowledge error $\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}$, then $\mathsf{Sig}_{\mathsf{RPO}}[\lambda, m]$ has unforgeability error $\epsilon^{\mathrm{EUF}}_{\mathsf{Sig}_{\mathsf{RPO}}}$ such that*

$$\begin{aligned}\epsilon^{\mathrm{EUF}}_{\mathsf{Sig}_{\mathsf{RPO}}}(\lambda, m, t_{\mathsf{q}}, t_{\mathsf{S}}) \leq\ & \epsilon^{\mathrm{Rel}}_{\mathsf{RPO}}(\lambda) \\ & + \epsilon^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}(\lambda, t_{\mathsf{q}}, t_{\mathsf{S}}, \lambda + m) \\ & + \epsilon^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}(\lambda, t_{\mathsf{q}}, t_{\mathsf{S}}, \lambda + m).\end{aligned}$$

7 Concrete Parameters and Performance

This section analyses the concrete security of the STARK-based signature scheme, establishes parameters and presents performance results.

7.1 Concrete Security

When discussing the concrete parameters and performance of cryptographic systems, one critical aspect to consider is the security level associated with these parameters. Security levels are often defined in terms of either worst-case or average-case security, providing a measure of how resistant a system is to attacks. In practice, determining the average-case security of a cryptographic primitive can be more insightful when focusing on real-world performance, as it represents the expected resistance an adversary would encounter under typical conditions rather than the most extreme scenarios.

The average-case security level is determined by evaluating the resources an attacker would need on average to succeed in an attack. Specifically, if t represents the computational effort and $\epsilon(t)$ the probability of a successful attack, then the average-case security level is given by $\frac{t}{\epsilon(t)}$ [CY24]. This value expresses the expected resources

required for a successful attack, making it a useful metric when selecting security parameters that balance efficiency and protection.

By opting for average-case security, cryptographic systems can avoid the higher costs and performance overhead associated with worst-case security. This allows for more practical deployment, particularly when the objective is to maintain adequate security while optimizing performance.

7.1.1 Signature scheme security

We first analyze the requirements to obtain κ_{Sig} bits of average-case security for $\mathsf{Sig}_{\mathsf{RPO}}$ against t-queries adversaries. Let $\kappa_{\mathsf{RPO}}, \kappa^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}, \kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}$ denote the average-case relation hardness security level for RPO and true-simulation knowledge soundness and zero-knowledge security levels for $\mathsf{NArg}_{\mathsf{RPO}}$ against t-bounded adversaries and let

$$\kappa' = \min\{\kappa_{\mathsf{RPO}}, \kappa^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}, \kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}\}.$$

Following from Theorem 7, we have

$$\begin{aligned}
\frac{\epsilon^{\mathrm{EUF}}_{\mathsf{Sig}_{\mathsf{RPO}}}}{t} &\leq \frac{\epsilon^{\mathrm{Rel}}_{\mathsf{RPO}}}{t} + \frac{\epsilon^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}}{t} + \frac{\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}}{t} \leq 2^{-\kappa_{\mathsf{RPO}}} + 2^{-\kappa^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}} + 2^{-\kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}}, \\
&\leq 3 \cdot 2^{-\kappa'}, \\
\implies \frac{t}{\epsilon^{\mathrm{EUF}}_{\mathsf{Sig}_{\mathsf{RPO}}}} &\geq \frac{2^{\kappa'}}{3} = 2^{\kappa' - \log 3},
\end{aligned}$$

which implies that $\kappa_{\mathsf{Sig}} = \kappa' - \log 3$.

7.1.2 RPO relation security

For the RPO permutation, the hardness error of the relation $\mathcal{R}_{\mathsf{RPO}}$ is exactly equal to the pre-image error. RPO was designed to operate in overwrite mode [Ash+22, Sections 2.6 and 4.4.2] the security of which was analysed by Bertoni et al. [Ber+11, Section 4.3]. From this we can conclude

$$\epsilon^{\mathrm{Rel}}_{\mathsf{RPO}}(\lambda) \leq 2^{-2 \cdot \log p} \qquad \text{and} \qquad \frac{t}{\epsilon^{\mathrm{Rel}}_{\mathsf{RPO}}} \geq 2^{-4 \cdot \log p},$$

for query bound $t \leq 2^{-4 \log p}$, and Goldilocks prime $p = 2^{64} - 2^{32} + 1$.

7.1.3 NArg true-simulation knowledge soundness security

As demonstrated in Section 5, the BCS transformation of an IOP system provides true-simulation knowledge soundness to the resulting NArg with some security loss compared to the state-restoration soundness of the IOP owing to the use of the Merkle tree commitment scheme. Our Corollary 2 and Chiesa and Yegev's Theorem 26.1.1 [CY24] (for the bound on the extraction error of the Merkle tree commitment scheme) give the following error bound on the transformed IOP's knowledge soundness.

$$\epsilon^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}(\lambda, t) \leq \epsilon^{\mathrm{SR}}_{\mathsf{IOP}_{\mathsf{RPO}}}(\lambda, t) + 3 \cdot \frac{t^2}{2^\lambda},$$

for every security parameter $\lambda \in \mathbb{N}$ and query bound $t \in \mathbb{N}$. Using the fact that state-restoration knowledge soundness is implied by round-by-round knowledge soundness (Theorem 1), we then have

$$\epsilon^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}(\lambda, t) \leq t \cdot \epsilon^{\mathrm{RbR}}_{\mathsf{IOP}_{\mathsf{RPO}}} + 3 \cdot \frac{t^2}{2^\lambda}.$$

Assuming that $\lambda \geq 2\kappa + 3$, and that $\mathsf{IOP}_{\mathsf{RPO}}$ has κ' bits of worse-case security, i.e., $\epsilon^{\mathrm{RbR}}_{\mathsf{IOP}_{\mathsf{RPO}}} \leq 2^{-\kappa'}$, we can bound the average-case true-simulation knowledge soundness security of $\mathsf{NArg}_{\mathsf{RPO}}$ as follows.

$$\begin{aligned}
\frac{t}{\epsilon^{\mathrm{tSKS}}_{\mathsf{NArg}_{\mathsf{RPO}}}} \geq \frac{t}{t \cdot \epsilon^{\mathrm{RbR}}_{\mathsf{IOP}_{\mathsf{RPO}}} + 3 \cdot \frac{t^2}{2^\lambda}} &= \frac{1}{\epsilon^{\mathrm{RbR}}_{\mathsf{IOP}_{\mathsf{RPO}}} + 3 \cdot \frac{t}{2^\lambda}} \geq \frac{1}{2^{-\kappa'} + 3 \cdot \frac{t}{2^{2\kappa+3}}} \\
&\geq \frac{1}{2^{-\kappa'} + \frac{t}{2^{2\kappa+1}}} \geq \frac{1}{2^{-\kappa'} + 2^{-\kappa-1}} \\
&= 2^\kappa,
\end{aligned}$$

where we used the bound $t \leq 2^\kappa$ on the number of queries, and assumed that $\kappa' = \kappa + 1$.

7.1.4 NArg zero-knowledge security

Given Theorem 4, and assuming $s \leq 2 \cdot \lambda$, by Remark 18.6.9 in [CY24] which applies to the UC-Hiding version of the MT commitment scheme, we have:

$$\begin{aligned}
\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}}(\lambda, t_{\mathsf{q}}, t_{\mathcal{P}}) &= t_{\mathcal{P}} \cdot \left(\frac{t_{\mathsf{q}}}{2^s} + \epsilon^{\text{UC-Hid}}_{\mathsf{MT}} + \epsilon^{\mathrm{ZK}}_{\mathsf{IOP}} \right) \\
&\leq t_{\mathcal{P}} \cdot \left(\frac{t_{\mathsf{q}}}{2^s} + \frac{2 \cdot \mathsf{q} \cdot \mathsf{l} \cdot t_{\mathsf{q}}}{2^s} + \epsilon^{\mathrm{ZK}}_{\mathsf{IOP}} \right).
\end{aligned}$$

We conservatively assume that an adversary can obtain at most $t_{\mathcal{P}} \leq 2^{64}$ arguments from its PROVE oracle and consider that $t = t_{\mathsf{q}}$ for the average-case security analysis. We also assume $\mathsf{q} \leq 2^{10}$ and $\mathsf{l} \leq 2^{18}$ (which will hold for $\mathsf{IOP}_{\mathsf{RPO}}$). Furthermore, we also know that, for suitable parameters, $\epsilon^{\mathrm{ZK}}_{\mathsf{IOP}} = 0$ [HK24] and we therefore have

$$\begin{aligned}
\frac{\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}}}{t} &\leq \frac{t_{\mathcal{P}}}{t} \cdot \left(\frac{t_{\mathsf{q}}}{2^s} + \frac{2 \cdot \mathsf{q} \cdot \mathsf{l} \cdot t_{\mathsf{q}}}{2^s} + \epsilon^{\mathrm{ZK}}_{\mathsf{IOP}} \right) \\
&\leq \frac{2^{64}}{2^s} + \frac{2^{64} \cdot 2 \cdot 2^{10} \cdot 2^{18}}{2^s} + 0 = \frac{2^{64}(1 + 2^{29})}{2^s} \\
\implies \frac{t}{\epsilon^{\mathrm{ZK}}_{\mathsf{NArg}}} &\geq 2^{s-94}.
\end{aligned}$$

Therefore, for a given value of $s \geq 94$, $\mathsf{NArg}_{\mathsf{RPO}}$ constructed from the BCS transformation achieves $\kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}} = s - 94$ bits of average-case zero-knowledge security against $2^{\kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}}$-bounded adversaries.

7.1.5 List decoding regime analysis of IOP knowledge soundness

We choose a Reed–Solomon rate $\rho = \frac{1}{8}$ and a domain $D \subseteq \mathbb{F} \setminus \{0\}$, a coset of a subgroup of the multiplicative group such that $D \cap H = \varnothing$, a number of FRI queries $n_{FRI} = 85$ and a field extension degree $e = 3$.

Given this, we compute the zero-knowledge blow-up parameter $\beta = 32$, the number of quotient segments polynomials $f = 11$ each with $\ell = 162$ coefficients.

We pick the FRI folding factor to be 8 and let the degree of the final polynomial sent by the FRI Prover to be at most 31. This means that there is only one fold during the FRI commit phase. We choose the proximity parameter $m = 58$ as the best parameter which reduces the FRI query phase error while still keeping the FRI commit phase error smaller than 2^{-126}.

The vector of RbR soundness errors is then equal to

1. $\log_2(\epsilon_1) = -182$,
2. $\log_2(\epsilon_2) = -161$,
3. $\log_2(\epsilon_3) = -126$,
4. $\log_2(\epsilon_4) = -129$,
5. $\log_2(\epsilon_5) = -126$.

7.1.6 Unique decoding regime analysis of IOP knowledge soundness

We choose a Reed–Solomon rate $\rho = \frac{1}{8}$ and a domain $D \subseteq \mathbb{F}\setminus\{0\}$, a coset of a subgroup of the multiplicative group such that $D \cap H = \varnothing$, a number of FRI queries $n_{FRI} = 126$ and a field extension degree $e = 2$. Choosing a lower extension degree implies that verification operations are easier to perform recursively within a subsequent proof system.

Given this, we compute the zero-knowledge blow-up parameter $\beta = 32$, the number of quotient segment polynomials $f = 11$ each with $\ell = 162$ coefficients.

We start with $\epsilon^{\mathrm{S}}_{\mathrm{FRI}}$ and pick the folding factor to be 2 and choose the degree of the final polynomial sent by the FRI prover to be at most 255. This means that there is only one fold during the FRI commit phase.

The vector of RbR soundness errors is then equal to

1. $\log_2(\epsilon_1) = -120$,
2. $\log_2(\epsilon_2) = -116$,
3. $\log_2(\epsilon_3) = -117$,
4. $\log_2(\epsilon_4) = -117$,
5. $\log_2(\epsilon_5) = -104$.

As noted in Section 6.3 in [Sta21], we can use proof-of-work, also referred to as grinding, to reduce ϵ_5. Using the notation in the aforementioned work, we choose $z_5 = 12$ which gives $\log_2(\epsilon'_5) = -116$.

7.1.7 Concrete parameters

Soundness in the list decoding regime. Under the list decoding regime analysis of Section 7.1.5, we see that $\mathsf{IOP}_{\mathsf{RPO}}$ has $\kappa' = 126$ bits of worse-case round-by-round knowledge soundness security. For the analysis of the true-simulation knowledge soundness of $\mathsf{NArg}_{\mathsf{RPO}}$ of Section 7.1.3, this gives $\kappa = \kappa' - 1 = 125$, which implies the requirement that $\lambda \geq 2\kappa + 3 = 255$. We therefore set $\lambda = 256$ which then implies that $\mathsf{NArg}_{\mathsf{RPO}}$ has 125 bits of average-case true-simulation knowledge soundness security against 2^{125}-bounded adversaries.

Soundness in the unique decoding regime. Under the unique decoding regime analysis of Section 7.1.6, we see that $\mathsf{IOP}_{\mathsf{RPO}}$ has $\kappa' = 116$ bits of worse-case round-by-round knowledge soundness security. For the analysis of the true-simulation knowledge soundness of $\mathsf{NArg}_{\mathsf{RPO}}$ of Section 7.1.3, this gives $\kappa = \kappa' - 1 = 115$, which implies the requirement that $\lambda \geq 2\kappa + 3 = 233$. We therefore also set $\lambda = 256$ in this regime which then implies that $\mathsf{NArg}_{\mathsf{RPO}}$ has 115 bits of average-case true-simulation knowledge soundness security against 2^{115}-bounded adversaries.

Zero-knowledge parameters. The condition set by Section 7.1.4 is that $s \leq 2 \cdot \lambda = 512$. Given the lower bound set by the knowledge soundness security, it is sufficient to choose a convenient value of s such that $\kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}} = s - 94 \geq \{115, 125\}$ (since only the minimum of these values matters for the security of the signature scheme). We therefore set $s = 256$ which achieves 162-bits of average-case zero-knowledge security against 2^{162}-bounded adversaries.

Given these two security levels for $\mathsf{NArg}_{\mathsf{RPO}}$ and the security level of $\mathcal{R}_{\mathsf{RPO}}$ discussed in Section 7.1.2, the analysis of Section 7.1.1 implies that the signature scheme $\mathsf{Sig}_{\mathsf{RPO}}$ has

$$\kappa_{\mathsf{Sig}} = 115 - \log 3 > 113 \qquad \text{or} \qquad \kappa_{\mathsf{Sig}} = 125 - \log 3 > 122$$

bits of average-case existential unforgeability security, in the unique decoding regime or list decoding regime respectively, against 2^{113}-bounded, or 2^{122}-bounded, adversaries that can obtain up to 2^{64} signatures under the same public key.

Parameters. These parameter choices, as well as those of the DEEP-ALI IOP for RPO discussed in Sections 7.1.5 and 7.1.6, are summarised in Table 1.

Parameter	Symbol	UDR	LDR
Random oracle output length	λ	256	256
Commitment randomness length	s	256	256
Zero-knowledge blow-up factor	β	32	32
Trace domain size	$\lvert H \rvert$	8	8
RS code rate	ρ	$\frac{1}{8}$	$\frac{1}{8}$
LDE domain size	$\lvert D \rvert$	2048	2048
Multiplicity parameter	m	-	58
Extension degree	e	2	3
Agreement parameter	α	$\frac{1}{2} + \frac{\beta \cdot \lvert H \rvert + 2}{2 \lvert D \rvert}$	$\sqrt{\rho} \cdot \left(1 + \frac{1}{2m}\right)$
Number of FRI queries	n_{FRI}	126	85
FRI folding factor	$t = (t_0)$	(2)	(8)
Max degree of final FRI polynomial	-	255	31

Table 1: Concrete parameters in the unique and list decoding regimes used in the DEEP-ALI IOP for RPO.

7.2 Performance

We benchmarked our scheme on an Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz; the results are provided in Table 2 for the unique decoding regime parameter set, providing 113 bits of average-case unforgeability security, and in Table 3 for the list decoding regime parameter set, providing 122 bits of average-case unforgeability security. The benchmarks were run in two modes, single threaded and multi-threaded using all available 8 cores.

These benchmarks use either RPO or Blake3 as the hash function for the BCS transformation. The first would be required for recursive verification of signatures within a proof system since the Blake3 hash function is costly to prove. However, using Blake3 within the BCS transformation provides much better performance when executed on a CPU.

Hash function for BCS	Signing (ms)	Verifying (ms)	Size (Kb)
RPO (single)	370	27	100
RPO (multi)	175	27	100
Blake3 (single)	23	1	100
Blake3 (multi)	17	1	100

Table 2: Performance of the signature scheme in the unique decoding regime.

Hash function for BCS	Signing (ms)	Verifying (ms)	Size (Kb)
RPO (single)	210	22	80
RPO (multi)	94	21	80
Blake3 (single)	13	1	80
Blake3 (multi)	9.2	1.1	80

Table 3: Performance of the signature scheme in the list decoding regime.

References

[Ash+22] Tomer Ashur et al. *Rescue-Prime Optimized*. Cryptology ePrint Archive, Paper 2022/1577. 2022. URL: https://eprint.iacr.org/2022/1577.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa and Nicholas Spooner. 'Interactive Oracle Proofs'. In: *Theory of Cryptography - 14th International Conference, TCC 2016-B, Proceedings, Part II*. Vol. 9986. Lecture Notes in Computer Science. 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2.

[Ben+18] Eli Ben-Sasson et al. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Paper 2018/046. 2018. URL: https://eprint.iacr.org/2018/046.

[Ben+19] Eli Ben-Sasson et al. 'Aurora: Transparent succinct arguments for R1CS'. In: *Advances in Cryptology–EUROCRYPT 2019*. Springer. 2019, pp. 103–128. DOI: 10.1007/978-3-030-17653-2_4.

[Ben+20] Eli Ben-Sasson et al. 'Proximity gaps for Reed–Solomon codes'. In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 900–909. DOI: 10.1109/FOCS46700.2020.00088.

[Ber+11] Guido Bertoni et al. *Cryptographic sponge functions*. Keccak Team Website. 2011. URL: https://keccak.team/files/CSF-0.1.pdf.

[Bit+13] Nir Bitansky et al. 'Recursive composition and bootstrapping for SNARKS and proof-carrying data'. In: *Symposium on Theory of Computing Conference, STOC'13*. Ed. by Dan Boneh, Tim Roughgarden and Joan Feigenbaum. ACM, 2013, pp. 111–120. DOI: 10.1145/2488608.2488623.

[Blo+23] Alexander R. Block et al. 'Fiat-Shamir Security of FRI and Related SNARKs'. In: *Advances in Cryptology - ASIACRYPT 2023*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14439. Lecture Notes in Computer Science. Springer, 2023, pp. 3–40. DOI: 10.1007/978-981-99-8724-5_1.

[CF24] Alessandro Chiesa and Giacomo Fenzi. *zkSNARKs in the ROM with Unconditional UC-Security*. Cryptology ePrint Archive, Paper 2024/724. 2024. URL: https://eprint.iacr.org/2024/724.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. Self published, 2024. URL: https://github.com/hash-based-snargs-book.

[Do24] Jérémi Do Dinh. 'Simulation Security in the Random Oracle Model'. https://jdodinh.io/assets/files/m-thesis.pdf. MA thesis. School of Computer and Communication Science, EPFL, Aug. 2024.

[Dod+10] Yevgeniy Dodis et al. 'Efficient Public-Key Cryptography in the Presence of Key Leakage'. In: *Advances in Cryptology - ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 613–631. DOI: 10.1007/978-3-642-17373-8_35.

[Fau+12] Sebastian Faust et al. 'On the Non-malleability of the Fiat-Shamir Transform'. In: *Progress in Cryptology - INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. Lecture Notes in Computer Science. Springer, 2012, pp. 60–79. DOI: 10.1007/978-3-642-34931-7_5.

[FS86] Amos Fiat and Adi Shamir. 'How to Prove Yourself: Practical Solutions to Identification and Signature Problems'. In: *Advances in Cryptology - CRYPTO '86*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194. DOI: 10.1007/3-540-47721-7_12.

[Hab22] Ulrich Haböck. *A summary on the FRI low degree test*. Cryptology ePrint Archive, Paper 2022/1216. 2022. URL: https://eprint.iacr.org/2022/1216.

[HK24] Ulrich Haboeck and Al Kindi. *A note on adding zero-knowledge to STARKs*. Cryptology ePrint Archive, Paper 2024/1037. 2024. URL: https://eprint.iacr.org/2024/1037.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. 'Signature Schemes with Bounded Leakage Resilience'. In: *Advances in Cryptology - ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 703–720. DOI: 10.1007/978-3-642-10366-7_41.

[Mic00] Silvio Micali. 'Computationally Sound Proofs'. In: *SIAM J. Comput.* 30.4 (2000), pp. 1253–1298. DOI: 10.1137/S0097539795284959.

[Sta21] StarkWare. *ethSTARK Documentation*. Tech. rep. 2021. URL: https://eprint.iacr.org/2021/582.

A Appendix

Let $\mathbb{F}$ be a field and let $V := \mathsf{RS}[\mathbb{F}, \mathcal{D}, k]$ be the Reed-Solomon code with rate $\rho := \frac{k}{n}$ and $n = |\mathcal{D}|$.

For $u, v \in \mathbb{F}^{\mathcal{D}}$, we define $\mathsf{agree}(u, v) := 1 - \Delta(u, v)$ where Δ is the relative Hamming distance. We define $\mathsf{agree}(u, V) := 1 - \Delta(u, V)$ where $\Delta(u, V) = \min_{w \in V} \Delta(u, w)$.

We restate the main results from [Ben+20] in the unique decoding regime i.e., when the proximity parameter $\delta \in (0, \frac{1-\rho}{2}]$ or equivalently when the agreement parameter $\alpha = 1 - \theta \in [\frac{1+\rho}{2}, 1)$. We also give the specialization of Theorem 7.1 in [Ben+20] to the unique decoding regime. Our version gives tighter bounds when $\alpha \in [\frac{1+\rho}{2}, 1)$.

A.1 Correlated Agreement

Theorem 8 ([Ben+20, Theorem 6.1]). *Fix $\delta \in (0, \frac{1-\rho}{2}]$. Let $\mathcal{U} := \{u_0, u_1, \ldots, u_l\}$ be a subset of $\mathbb{F}_q^{\mathcal{D}}$ and let $S \subset \mathbb{F}_q$ be defined as*

$$S := \left\{ z \in \mathbb{F}_q : \mathsf{agree}\left(\sum_{i=0}^{l} z^i u_i, V \right) \geq 1 - \delta \right\}.$$

If

$$P_{z \in \mathbb{F}_q}\left[|S| \right] > \frac{l \cdot n}{q},$$

then, for all $z \in \mathbb{F}_q$, we have

$$\mathsf{agree}\left(\sum_{i=0}^{l} z^i u_i, V \right) \geq 1 - \delta.$$

Moreover, there exists a subset of code words $\mathcal{V} := \{v_0, \ldots, v_l\}$ such that the subset $\mathcal{D}' \subset \mathcal{D}$ defined by

$$\mathcal{D}' := \left\{ x \in \mathcal{D} : (u_0(x), \ldots, u_l(x)) = (v_0(x), \ldots, v_l(x)) \right\},$$

satisfies

$$|\mathcal{D}'| \geq (1 - \delta)|\mathcal{D}|.$$

The last point implies that, for all $z \in \mathbb{F}_q$, we have

$$\mathsf{agree}\left(\sum_{i=0}^{l} z^i u_i, \sum_{i=0}^{l} z^i v_i \right) \geq 1 - \delta.$$

Theorem 9 ([Ben+20, Theorem 1.6]). *Fix $\delta \in (0, \frac{1-\rho}{2}]$. Let $\mathcal{U} := \{u_0, u_1, \ldots, u_l\}$ be a subset of $\mathbb{F}_q^{\mathcal{D}}$ and define $\mathsf{U} := \left\{ u_0 + \sum_{i=1}^{l} z_i u_i : (z_1, \ldots, z_l) \in \mathbb{F}_q^l \right\}$.*

If

$$P_{u \in \mathsf{U}}\left[\mathsf{agree}\left(u, V\right) \geq (1 - \delta) \right] > \frac{n}{q},$$

then

$$P_{u \in \mathsf{U}}\left[\mathsf{agree}\left(u, V\right) \geq (1 - \delta) \right] = 1.$$

Moreover, there exists a subset of code words $\mathcal{V} := \{v_0, \ldots, v_l\}$ such that the subset $\mathcal{D}' \subset \mathcal{D}$ defined by

$$\mathcal{D}' := \left\{ x \in \mathcal{D} : (u_0(x), \ldots, u_l(x)) = (v_0(x), \ldots, v_l(x)) \right\},$$

satisfies

$$|\mathcal{D}'| \geq (1 - \delta)|\mathcal{D}|.$$

The last point implies that, for all $z \in \mathbb{F}_q$, we have

$$\mathsf{agree}\left(u_0 + \sum_{i=1}^{l} z_i u_i, v_0 + \sum_{i=1}^{l} z_i v_i \right) \geq 1 - \delta.$$

A.2 Correlated Weighted Agreement

Let $\mu : \mathcal{D} \to [0,1]$, the (relative) μ-agreement is defined as

$$\mathsf{agree}_{\mu}(u,v) := \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mu(x) \cdot 1_{\{u(x)=v(x)\}}.$$

Note that when $\mu = 1$, μ-agreement coincides with the standard notion of relative agreement. We can also generalize the notion of agreement between a function $u \in \mathbb{F}_q^{\mathcal{D}}$ and V and define

$$\mathsf{agree}_{\mu}(u,V) := \max_{v \in V} \mathsf{agree}_{\mu}(u,v).$$

For a $\mathcal{D}^{'} \subset \mathcal{D}$, we can define its μ-weighted size as

$$\mu\left(\mathcal{D}^{'}\right) := \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}^{'}} \mu(x).$$

This implies that

$$\mathsf{agree}_{\mu}(u,v) = \mu\left(\{x \in \mathcal{D} : u(x) = v(x)\}\right).$$

We can also generalize the notion of correlated agreement and define the μ-weighted correlated agreement between a subset $\mathcal{U} := \{u_0, u_1, \ldots, u_l\} \subset \mathbb{F}_q^{\mathcal{D}}$ and V as the maximal μ-weighted subset $\mathcal{D}^{'} \subset \mathcal{D}$ defined as

$$\mathcal{D}^{'} := \{x \in \mathcal{D} : (u_0(x), \ldots, u_l(x)) = (v_0(x), \ldots, v_l(x))\},$$

for a subset of code words $\mathcal{V} := \{v_0, \ldots, v_l\} \subset V$.

We restrict our attention to the special case when the weight function μ has the form $\mu(x) = \frac{m_x}{M}$ for a fixed integer $M \geq 1$ and $m_x \in [0, M]$ for all $x \in \mathcal{D}$. We are now ready to state the weighted version of the correlated agreement results stated in 8.

Theorem 10. *Fix $\delta \in (0, \frac{1-\rho}{2}]$. Let $\mathcal{U} := \{u_0, u_1, \ldots, u_l\}$ be a subset of $\mathbb{F}_q^{\mathcal{D}}$ and let $S \subset \mathbb{F}_q$ be defined as*

$$S := \left\{ z \in \mathbb{F}_q : \mathsf{agree}_{\mu}\left(\sum_{i=0}^{l} z^i u_i, V\right) \geq 1 - \delta \right\}.$$

for $\mu : \mathcal{D} \to [0,1]$ a weighting function with common denominator $M \geq 1$. Assume that:

$$|S| > l \cdot (M \cdot n + 1).$$

Then, there exists a subset of code words $\mathcal{V} := \{v_0, \ldots, v_l\}$ such that the subset $\mathcal{D}^{'} \subset \mathcal{D}$ defined by

$$\mathcal{D}^{'} := \{x \in \mathcal{D} : (u_0(x), \ldots, u_l(x)) = (v_0(x), \ldots, v_l(x))\},$$

satisfies

$$\mu\left(\mathcal{D}^{'}\right) \geq (1 - \delta).$$

Proof. By definition of S, we have that for each $z \in S$ there is $P_z \in V$ such that $\mathsf{agree}_{\mu}\left(\sum_{i=0}^{l} z^i u_i, P_z\right) \geq (1-\delta)$. By definition of μ, we get that

$$\mathsf{agree}\left(\sum_{i=0}^{l} z^i u_i, P_z\right) \geq (1 - \delta).$$

Since $|S| > l \cdot (M \cdot n + 1)$ implies $|S| > l \cdot n$, we get by Theorem 8, that there exists a subset of code words $\mathcal{V} := \{v_0, \ldots, v_l\}$ such that

$$\mathcal{D}^{'} := \{x \in \mathcal{D} : (u_0(x), \ldots, u_l(x)) = (v_0(x), \ldots, v_l(x))\},$$

satisfies

$$\frac{|D^{'}|}{|\mathcal{D}|} \geq (1 - \delta).$$

On the other hand, by unique decoding and the analysis in the proof of Theorem 6.1 in [Ben+20], we have that for each $z \in S$, $P_z = \sum_{i=0}^{l} z^i v_i$. Thus we have that, for each $z \in S$,

$$\mathsf{agree}_{\mu}\left(\sum_{i=0}^{l} z^i u_i, \sum_{i=0}^{l} z^i v_i\right) \geq (1 - \delta).$$

We are now in the setting of Lemma 7.6 in [Ben+20] and we can conclude that

$$\mu\left(\mathcal{D}^{'}\right) \geq (1 - \delta).$$

□