STARK-Based Signatures from the RPO Permutation

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

[1] COSIC, KU Leuven, Leuven, Belgium
[2] 3MI Labs, Leuven, Belgium
[3] Miden, UAE

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 instantiated using the DEEP-ALI interactive oracle proof and made non-interactive via the Ben-Sasson–Chiesa–Spooner (BCS) transformation in the random oracle model. The resulting construction yields a signature scheme with transparent setup. Our design is motivated by recursive zero-knowledge applications, in which signature verification must itself be efficiently provable inside larger proof systems. To this end, the choice of the RPO permutation, the use of a simple algebraic intermediate representation (AIR), and working over the Goldilocks field are made with the goal of enabling efficient recursive verification and aggregation. The implementation of the scheme computes signatures in 4.6–7.2 ms and verifies them in 0.46–0.52 ms when the BCS transform is implemented with Blake3. When the BCS transform is instead instantiated with the RPO permutation itself, the configuration required when signature verification is to be proven recursively inside a proof system, signing takes 20.9–30.4 ms with Metal acceleration and 59.2–229.2 ms on CPU, while verification takes 5.09–5.79 ms. We validate the recursion-friendliness claim end to end by proving one signature verification inside the Miden zkVM and reporting the recursive prover time and proof size. These speeds are obtained with parameters achieving 113 or 122 bits of average-case security, depending on the chosen preset, against adversaries that can obtain up to 2^{64} signatures.

Keywords: STARK · RPO · Digital Signatures · Post-Quantum Cryptography

1 Introduction

Zero-knowledge scalable transparent arguments of knowledge (ZK-STARKs) are cryptographic proof systems [BSBHR18] 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 has made them particularly attractive for the development of zero-knowledge virtual machines (zkVMs) and for applications in blockchain technology, where trust minimization and scalability are crucial. A wide variety of such

This is the IACR ePrint version of this work which was published with the same title, authors and content in *IACR Communications in Cryptology*, vol. 3, no. 2 on Aug. 03, 2026 with DOI 10.62056/absgbn-3y. When citing this work, please cite the published version.

E-mail: shahla.atapoor@kuleuven.be (Shahla Atapoor), cyprien.desaintguilhem@3milabs.tech (Cyprien Delpech de Saint Guilhem), al.kindi@miden.team (Al Kindi)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-08-20.

zkVMs have been developed in recent years[1] with a particular focus on proving execution of blockchain transactions, a computation where verification of user-generated signatures is a critical component.

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 [BCCT13]. 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 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 the context of zkVMs and wider blockchain applications, this recursive approach has two particular advantages. First, it makes user-generated signatures efficiently verifiable within the zkVM, which is crucial for proving validation of transactions. Second, it allows for the aggregation of multiple signature verifications into a single proof, which can be particularly useful in the consensus layer of blockchains where many nodes express their agreement on the validity of a block by signing it. By using recursive ZKPs, nodes can aggregate other nodes' approval signatures into a single proof of approval, which can be more efficiently verified by the network.

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 for 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 especially appealing for applications like blockchains, financial transactions, and other distributed systems

[1] See https://ethproofs.org/zkvms for zkVMs competing to prove Ethereum blocks in real-time.

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 [Can01], 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 [Do 24].

A second thread running through our construction is that each of its four components is chosen with recursive verification in mind. The Goldilocks field $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$ is the native field of several current STARK-based zkVMs, so that a verifier operating over $\mathbb{F}_p$ can be arithmetised without field arithmetic emulation. The Rescue-Prime Optimized permutation has a low-degree AIR, which makes it inexpensive to re-prove inside the same IOP framework when RPO is used as the BCS hash. The DEEP-ALI IOP and its BCS compilation yield a verifier whose algebraic and hashing steps are both native to this setting. We make this quantitative in Section 7.1, where we give operation counts for one execution of our verifier and contrast them with the multi-limb or non-arithmetisation-friendly cost of verifying large-field or hash-based post-quantum signatures inside the same zkVM. We complement this analysis with an end-to-end experiment (Section 7.2) in which one signature verification is proven inside the Miden zkVM.

1.3 Related works

In recent years, numerous post-quantum signature schemes have emerged, each offering different trade-offs between efficiency, scalability, and security. Among these, we have selected the relevant and similar schemes from two major categories: those from the first NIST post-quantum cryptography competition, and those introduced in the second "additional" NIST competition for digital signatures. Our comparisons focus on these schemes due to their structural similarities and relevance to our STARK-based approach.

First NIST post-quantum cryptography competition. Notable contenders from the first competition include *Picnic3* [KZ20], *SPHINCS+* [BHK+19], and *Dilithium* [DKL+18], which were all finalists in the competition. While *Picnic* was an alternate finalist, *SPHINCS+*, *Dilithium*, and *Falcon* all reached the final round and have since been standardized by NIST as *SLH-DSA* [Nat24] (FIPS 205), *ML-DSA* [DKL+18] (FIPS 204), and *FN-DSA* (upcoming FIPS 206), respectively. While these schemes provide strong security guarantees, they encounter challenges related to performance in practical applications.

Picnic [KZ20] is based on ZKPs and avoids number-theoretic assumptions, but it suffers from larger signature sizes and slower verification processes, limiting its applicability in performance-sensitive environments. The stateless hash-based scheme SPHINCS+ [BHK+19] offers robust security but also faces issues with larger signatures and computational inefficiencies. Meanwhile, Dilithium [DKL+18], part of the CRYSTALS family, relies on lattice-based primitives, resulting in increased computational overhead and larger key sizes.

In contrast, our STARK-based signature scheme offers good efficiency with post-quantum security and is the first designed with *recursive verification* in mind, making it practical for environments where proof of signature verification is desirable. Due to this, *Ziggy* [BSBSofS20], *Preon* [Pre23], and *Loquat* [ZSE+24] are the closest related works.

Comparison with Ziggy. Ziggy [BSBSofS20] is the closest prior work in spirit: it also constructs a post-quantum signature scheme by proving knowledge of a pre-image of an arithmetisation-friendly hash via a STARK. Our scheme differs from Ziggy on four axes: the underlying IOP and its analysis, the choice of hash primitive and the resulting AIR, the security parameters and the assumptions they rely on, and the recursion friendliness of the verifier.

IOP and analysis. Ziggy is built on top of the EthSTARK proof system, which predates several refinements to the analysis of FRI-based IOPs. Our scheme uses the DEEP-ALI IOP [BGKS20] together with more recent soundness analyses in both the unique decoding regime and the list decoding regime (Theorems 2 and 3), and in particular does not rely on conjectured soundness bounds for FRI or for Reed–Solomon list decoding beyond the Johnson bound.

Hash primitive and AIR. Ziggy instantiates its hard relation with the original Rescue hash function, whereas we use the Rescue-Prime Optimized permutation [AKM+22], which was designed specifically as an arithmetic primitive for the Goldilocks field. The resulting AIR (Section 3) is compact: an 8×12 trace with 24 constraints of maximum degree 7. This compactness directly translates into a smaller execution trace for the IOP prover and into a simpler recursive verifier.

Parameters and security. Ziggy targets 128 bits of security but its parameters predate the analyses we rely on. Our scheme achieves 113 bits (unique decoding regime) or 122 bits (list decoding regime) of average-case existential-unforgeability security, respectively, against adversaries with access to at most 2^{64} signing queries, with no reliance on conjectures.

Recursion friendliness. This is the axis on which the two schemes are most different. Ziggy was neither analysed nor benchmarked with recursive verification in mind: its reported signing time of over 100 ms and its use of an older proof pipeline make it unattractive as a target for recursive proving. Our scheme, by contrast, signs in 4.6–7.2 ms and verifies in 0.46–0.52 ms in its fastest conventional configuration (Blake3-instantiated BCS transform), and signs in 20.9–30.4 ms with Metal acceleration, or 59.2–229.2 ms on CPU, in its recursion-friendly configuration (RPO-instantiated BCS transform); it moreover admits the explicit verification-cost analysis of Section 7.1, which quantifies the cost of proving one signature verification inside a Goldilocks-native zkVM. To the best of our knowledge, ours is the first STARK-based signature scheme whose design and parameter choices are explicitly justified in terms of recursive provability.

Comparison with Preon. Preon [Pre23], much like our own scheme, relies on STARKs to ensure post-quantum security. However, it is constructed on top of the general-purpose Aurora proving system to demonstrate adaptability across a variety of cryptographic tasks, including group signatures. To compare the performance of our signature scheme with Preon, we use the reported Preon measurements at the 128-bit security level. Their construction is based on conjectures of varying strength, yielding aggressive (Preon128A), balanced (Preon128B), and conservative (Preon128C) parameter sets. We emphasize that our scheme does not depend on conjectures, whether they relate to the soundness of the FRI protocol beyond the Johnson bound, the list decodability of Reed-Solomon codes beyond the Johnson bound, or conjectures related to properties of the underlying IOP or its BCS compilation. Preon's adaptability introduces complexity that negatively impacts its performance: its reported signing times range between 76 s and 81 s, orders of magnitude slower than our scheme's conventional configuration. Preon's verification times, which range from 397 ms to 2,303 ms, are likewise far slower than ours. Furthermore, Preon's signature sizes, which vary from 139 KB to 1.73 MB, are substantially larger than our scheme's (Section 7).

NIST additional signatures competition. The NIST competition for "Additional Signatures - Post-Quantum Cryptography" is distinct from the previous rounds. This competition has had only one round so far, allowing us to reference Round 1 metrics for the candidates. Notable contenders include *Preon*, as discussed earlier, along with *FAEST* and *Ascon-Sign*. Below, we summarize their constructions and compare them to our approach.

Comparison with FAEST. FAEST [BBdSG+23] is a post-quantum signature scheme that leverages a VOLE-based Zero-Knowledge Proof of Knowledge (ZKPoK) for AES (Advanced Encryption Standard). FAEST achieves a signing time of approximately 0.9 ms for 6.5 KB signatures and 8.1 ms for 5 KB signatures. Our scheme's Blake3-configuration signing time of 4.6–7.2 ms is in the same millisecond range, while our design offers benefits in contexts requiring recursive verification. Both schemes provide robust post-quantum security and resilience; however, FAEST's verification algorithm is not designed to be compatible with recursive verification within a zero-knowledge proof system due to its heavy use of AES- and SHA3-based primitives. In contrast, our scheme can be instantiated with the RPO hash function which is much better suited for recursive verification scenarios. While FAEST's signature sizes range from 5 KB to 6.5 KB, our scheme's sizes range from 71.1 KiB to 80.2 KiB. This size difference arises from the distinct underlying proof systems used by the two schemes.

Comparison with Ascon-Sign. Ascon-Sign [SGJ+24] is a lightweight signature scheme designed for resource-constrained environments; it is a post-quantum scheme that builds on the SPHINCS+ framework but replaces the internal hash functions with Ascon-Hash and Ascon-XOF. Ascon-Sign offers two versions: Ascon-Sign-128s, which prioritizes smaller signatures, and Ascon-Sign-128f, optimized for faster signing times. The signing time for Ascon-Sign-128s is approximately 14,225 ms, while Ascon-Sign-128f achieves a faster time of 198 ms. In comparison, our scheme signs in 4.6–7.2 ms in its Blake3 configuration. In terms of verification time, Ascon-Sign-128s verifies in 5.5 ms, and Ascon-Sign-128f in 12.47 ms, whereas our Blake3 configuration verifies in 0.46–0.52 ms. Regarding signature size, Ascon-Sign-128s produces smaller signatures of 7.856 KB, and Ascon-Sign-128f results in larger ones of 17.088 KB. Our scheme's signature size, on the other hand, is between 71.1 KiB and 80.2 KiB. While Ascon-Sign offers compact signatures and is tailored for constrained devices, our STARK-based scheme delivers faster signing and verification times, making it more suitable for performance-critical applications.

Comparison with Loquat. Loquat [ZSE+24] is a scheme which was also designed as proof-system friendly. It is based on the Legendre PRF assumption over fields of large prime order ($p = 2^{127} - 1$). Already such a large order field makes it less suitable for recursive verification inside a Goldilocks-native zkVM, or another zkVM that does not natively support large-field arithmetic. Its signature size at the 128-bit security level (46 KB) is smaller than the one produced by our scheme, however the signing and verification times of 5.04 s and 0.21 s, respectively, are significantly slower than ours when using standard hash functions. Using the arithmetic hash function Griffin [GHR+23], signing takes 105 s and verifying takes 11 s compared with our 20.9–30.4 ms with Metal, 59.2–229.2 ms on CPU, and 5.09–5.79 ms verification when using the RPO hash function which is also well arithmetized over the Goldilocks field. Our scheme is therefore more performant for signature computation with either standard or arithmetized hash functions, and is more suitable for recursive verification inside a Goldilocks-native zkVM.

Comparison with CAPSS. In follow-up work posted after the original submission of this paper, Feneuil and Rivain introduced CAPSS [FR25], a framework sharing our central

design idea: a post-quantum signature scheme constructed as a proof of knowledge of a pre-image of an arithmetization-oriented permutation, with instantiations over Rescue–Prime, Poseidon, Griffin, and Anemoi. The two works differ in the underlying proof system and in the optimisation target. CAPSS instantiates its argument with SmallWood, a hash-based polynomial commitment and zero-knowledge argument in the MPC-in-the-head/Threshold-Computation-in-the-Head lineage, and optimises for R1CS constraint count, obtaining notably compact signatures (9–13.3 KB for its Anemoi instance at the 128-bit security level) and reporting improvements over Loquat. Our construction instead uses the DEEP-ALI IOP compiled with the BCS transformation: the resulting signatures are larger (71–80 KiB), but the verifier consists exclusively of native Goldilocks field operations, operations in a small-degree extension, and calls to the RPO permutation, so that it can be arithmetised directly inside a Goldilocks-native STARK zkVM without a general-purpose SNARK layer or field emulation. Moreover, our soundness analysis relies only on Reed–Solomon decoding up to the Johnson bound and requires no conjectures on FRI or on list decodability. The two approaches occupy complementary points in the design space: CAPSS targets small R1CS instances for a general-purpose SNARK, whereas our scheme targets recursion inside the STARK-based zkVMs over the Goldilocks field.

Comparison with hash-based multi-signatures for Ethereum. In further follow-up work, Drake, Khovratovich, Kudinov, and Wagner [DKKW25] address the same consensus-layer aggregation setting that motivates our construction (Section 1.1), but with a complementary architecture. Their scheme retains an XMSS-style hash-based signature as the base primitive and aggregates many such signatures by proving their verification inside an external succinct (non-interactive) argument. In our scheme, by contrast, the signature is itself a STARK proof, so that aggregation reduces to recursive verification of proofs within the same proof-system framework, without a second signature layer. We also note that their security analysis deliberately avoids the random oracle model for the base signature, formulating explicit standard-model requirements on the hash function; our analysis instead proves the BCS compilation in the ROM, with recursive verification instantiating the oracle by the concrete arithmetised RPO permutation, as is standard for STARK recursion.

Summary. We note that while many post-quantum signature schemes, including FAEST, target 128-bit security, the 122-bit level in our scheme remains sufficient for most practical applications. This adjustment arises from the parameters of the DEEP-ALI IOP, an efficient interactive oracle-proof system designed for AIR.

Additionally, the compatibility of our scheme with recursive verification enhances its practical utility, especially in blockchain networks where efficient verification is crucial. The comparisons of our scheme with other related works are summarized in Table 1.

2 Technical Overview

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

1. Knowledge of a Rescue-Prime Optimized pre-image as the hard relation,

[3]Performance figures for schemes other than ours are taken from the respective specification documents and reference implementations. A precise like-for-like comparison is not possible due to differences in benchmarking hardware, compiler toolchains, and implementation maturity; the table is intended as a broad-strokes positioning rather than a head-to-head benchmark. Our own figures range over the two parameter sets of Table 2; these reach 113 or 122 bits of average-case unforgeability security rather than the nominal 128 bits of the other rows.

Table 1: Comparison of post-quantum signature schemes at the 128-bit security level.[3] Our measurements were obtained on an Apple M4 Pro; the RPO rows separate Metal acceleration from CPU-only signing.

Signature Scheme	Signing Time	Verification Time	Signature Size	Public Key Size
Picnic3 [KZ20]	5.24 - 41.16 ms	2.95 - 18.21 ms	12.3 - 15.9 KB	32 B
SPHINCS+ [BHK+19]	200 ms	50 - 200 ms	20- 80 KB	32 KB
Dilithium [DKL+18]	$<$ 100 ms	$<$ 100 ms	2 - 5 KB	2 KB
Ziggy [BSBSofS20]	$>$ 100 ms	$>$ 100 ms	12.1 - 17.1 KB	32 B
Preon128A	76 s	397 ms	139 KB	32 B
Preon128B	81 s	561 ms	372 KB	32 B
Preon128C [Pre23]	77 s	2,303 ms	1,725 KB	32 B
FAEST [BBdSG+23]	0.9 - 8.1 ms	0.9 ms	5 - 6.5 KB	32 B
AsconSign-128s	14,225 ms	5.5 ms	7.856 KB	32 B
AsconSign-128f [SGJ+24]	198 ms	12.47 ms	17.088 KB	32 B
Loquat (SHA)	5,040 ms	210 ms	57 KB	4 KB
Loquat (Griffin) [ZSE+24]	105 s	11 s	57 KB	4 KB
Ours (Blake3)	4.6–7.2 ms	0.46–0.52 ms	71.1–80.2 KiB	32 B
Ours (RPO, Metal)	20.9–30.4 ms	5.09–5.69 ms	71.3–80.1 KiB	32 B
Ours (RPO, CPU)	59.2–229.2 ms	5.19–5.79 ms	71.3–80.1 KiB	32 B

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 ZKPoK.

2.1 The Rescue-Prime Optimized Hash Function

An instance of the Marvellous design strategy, Rescue-Prime Optimized (RPO) is an arithmetic hash function specifically designed for the "Goldilocks field" $\mathbb{F}_p$, where $p = 2^{64} - 2^{32} + 1$ [AKM+22]. In this work only the parameter set targeting 128 bits of *RPO-internal* security will be used; the resulting signature scheme achieves 122 bits of average-case existential-unforgeability in the list decoding regime (Section D).

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 [BGKS20]. 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 do 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 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 ROM 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 (RO) 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. [DHLW10], and then adapted to the ROM by Faust et al. [FKMV12]. In this work, we use the results recently obtained by Do Dinh [Do 24] for the provable EUF-CMA security of these signature schemes.

Each of the four components above is, in addition, compatible with proving the signature verification itself inside a STARK-based zkVM over the same Goldilocks field: the field is native to such VMs, the RPO permutation admits a low-degree AIR, the DEEP-ALI IOP verifier consists of a small number of algebraic checks plus a FRI query phase, and the BCS transformation can be instantiated with an arithmetisation-friendly hash (again RPO) so that no emulation of a foreign field or of a non-arithmetic hash is required. A detailed cost breakdown is given in Section 7.1, and an end-to-end recursive verification benchmark inside the Miden zkVM is reported in Section 7.2.

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. [BSBHR18] 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.

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,[4]
- $\mathsf{h} \in \mathbb{N}^{>0}$ specifies the logarithm of the trace height (i.e., the number of trace steps),
- $g \in \mathbb{F}^*$ 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. Formally defined as follows:

[4] 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. [AKM+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

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

- $f_{\mathrm{MDS}}(\vec{x}) = M \cdot \vec{x}$ 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 equation 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,[5] 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 $j \in [\mathsf{w}]$, 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.

4 DEEP-ALI: an IOP for AIRs

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

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

[5] 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.

4.1 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 formally define an k-round interactive protocol for proving AIR satisfiability.

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 Sec. 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 Sec. 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 Sec. 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 the evaluation claims in Π_3 satisfy Equation 2 at $X = r$, reconstructing $q(r)$ via Equation 4 and interpolating $\vec{z}$ 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 [BCR+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*

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

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 Sec. 4.2.2, Protocol 1 is perfect honest-verifier ZK and hence the ZK error $\epsilon^{\mathrm{ZK}}_{\mathsf{IOP}}$ of the DEEP-ALI IOP is equal to zero.

4.4.2 Round-by-round knowledge soundness error

As we will see in Sec. 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 proof of Theorem 2 is given in App. B.1. □

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* $4 \leq i \leq \mathsf{k} - 1$: $\quad \epsilon_i = (t_{i-4} - 1) \cdot \frac{|D|+1}{|\mathbb{G}|}$,
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 of Theorem 3 is given in App. B.2. □

5 The BCS Transformation: a ZK-STARK for AIRs

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

Overview of the reduction strategy. Before giving the formal definitions, we outline how the security of the final signature scheme is assembled from the properties established so far, since the argument passes through several intermediate notions. The chain of reductions is:

$$\underbrace{\epsilon^{\mathsf{RbR}}_{\mathsf{IOP}}}_{\text{Thms. 2, 3}} \implies \underbrace{\epsilon^{\mathsf{SR}}_{\mathsf{IOP}}}_{\text{Thm. 1}} \implies \underbrace{\epsilon^{\mathsf{UC\text{-}KS}}_{\mathsf{BCS}}, \epsilon^{\mathsf{UC\text{-}ZK}}_{\mathsf{BCS}}}_{\text{Thm. 4}} \implies \underbrace{\epsilon^{\mathsf{tSKS}}_{\mathsf{NArg}}, \epsilon^{\mathsf{ZK}}_{\mathsf{NArg}}}_{\text{Thms. 5, 6}} \implies \underbrace{\epsilon^{\mathsf{EUF}}_{\mathsf{Sig}}}_{\text{Thm. 7}} .$$

In words: Section 4 established that the DEEP-ALI IOP is perfectly honest-verifier zero-knowledge and round-by-round knowledge sound (Theorems 2 and 3). Round-by-round knowledge soundness implies state-restoration knowledge soundness (Theorem 1), which is the notion required by the BCS transformation. The results of Chiesa and Fenzi [CF24] (restated as Theorem 4) then show that the BCS compilation of such an IOP is UC-friendly zero-knowledge and UC-friendly knowledge sound. The signature framework of Do Dinh [Do 24], however, is stated in terms of the weaker notions of plain zero-knowledge and true-simulation knowledge soundness in the ROM; the new technical contributions of this section, Theorems 5 and 6, prove that the Finally, Section 6 combines the hardness of the RPO relation with the two properties of the resulting non-interactive argument to conclude EUF-CMA security of the the signature scheme (Theorem 7), and Appendix D instantiates every error term with concrete parameters.

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 (RO [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 RO (with output size λ).*

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

Definition 7 (Explicitly programmable ROM)**.** The *explicitly programmable ROM*, given a RO 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 RO 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 RO. 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.

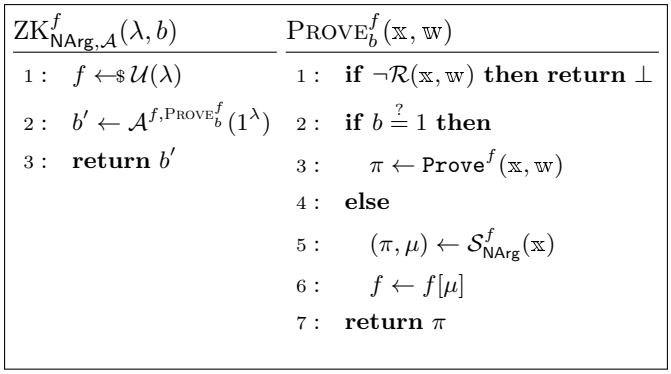

Figure 1: The argument zero-knowledge game ZK^f in the ROM.

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 ZK with error* $\epsilon^{\mathrm{ZK}}_{\mathsf{IOP}}$*, then* $\mathsf{BCS}[\mathsf{IOP}, \lambda, s]$ *is UC-friendly ZK 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 [Do 24] 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 ZK.

Definition 9 (adapted from [Do 24, Definition 2.8])**.** Let $f \leftarrow \mathcal{U}(\lambda)$ be a random oracle. A non-interactive argument NArg in the ROM is *ZK 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 PROVE,

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

$\mathrm{tSKS}^{f}_{\mathsf{NArg},\mathcal{R},\mathcal{A}}(\lambda)$

1 : $Q \leftarrow \emptyset$
2 : $(\mathbb{x}, \pi) \leftarrow \mathcal{A}^{f, \textsc{Prove}^f}(1^\lambda)$
3 : $\mathbb{w} \leftarrow \mathcal{E}^{f}_{\mathsf{NArg}}(\mathbb{x}, \pi)$
4 : **if** $\neg \mathtt{ArgVer}^f(\mathbb{x}, \pi)$ **then return** $\bot$
5 : **if** $\mathbb{x} \in Q$ **then return** $\bot$
6 : **return** $\neg\mathcal{R}(\mathbb{x}, \mathbb{w})$

$\textsc{Prove}^f(\mathbb{x}, \mathbb{w})$

1 : **if** $\neg\mathcal{R}(\mathbb{x}, \mathbb{w})$ **then return** $\bot$
2 : $(\pi, \mu) \leftarrow \mathcal{S}^{f}_{\mathsf{NArg}}(\mathbb{x})$
3 : $f \leftarrow f[\mu]$
4 : $Q \leftarrow Q \cup \{\mathbb{x}\}$
5 : **return** π

Figure 2: The true-simulation knowledge soundness game tSKS^f in the ROM.

and game ZK is defined in Fig. 1.

However, the results of Chiesa and Fenzi [CF24] (restated in Theorem 4) prove (in appearance stronger) the UC-friendly ZK property which is defined in App. A.1. We show that the UC-friendly notion of ZK is indeed stronger than the one used by Do Dinh [Do 24] by proving the following theorem:

Theorem 5. *If* NArg *is UC-friendly ZK with error* $\epsilon^{\text{UC-ZK}}_{\mathsf{NArg}}$ *in the ROM (Definition 15), then* NArg *is ZK in the ROM (Definition 9) with*

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

Proof. The proof of Theorem 5 is given in App. B.3. □

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

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

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

5.4 True-Simulation Knowledge Soundness

Definition 10 (adapted from [Do 24, 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 $\epsilon^{\text{tSKS}}_{\mathsf{NArg}}$ against $(t_{\mathsf{q}}, t_{\mathcal{P}})$-adversaries relative to the PPT simulator $\mathcal{S}^{f}_{\mathsf{NArg}}$ if there exists a deterministic polynomial-time extractor algorithm $\mathcal{E}^{f}_{\mathsf{NArg}}$ such that

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

where the game tSKS is defined in Fig. 2.

However, the results of Chiesa and Fenzi [CF24] (restated in Theorem 4) prove (in appearance stronger) the UC-friendly knowledge sound property which is defined in App. A.2. We show that the notion of UC-friendly knowledge sound is indeed stronger than the one used by Do Dinh [Do 24] by proving the following theorem:

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

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

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

Figure 3: The unforgeability game $\text{EUF-CMA}^{f}_{\mathsf{Sig},\mathcal{A}}$ in the random oracle model.

Proof. The proof of Theorem 6 is given in App. B.4. □

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*

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

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 11 ([Do 24, 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 12. 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 $\mathcal{A}$,

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

6.2 Scheme and Security Definitions

Definition 13 (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 14 (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 $\epsilon^{\text{EUF}}_{\mathsf{Sig}}$ 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 $t_{\mathsf{S}} \in \mathbb{N}$,

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

where the EUF-CMA game is defined in Fig. 3

Let $\mathsf{NArg}_{\mathsf{RPO}}$ be the non-interactive argument system for $\mathcal{R}_{\mathsf{RPO},m}$ in the ROM of Sec. 5. Given a random oracle $f \leftarrow \mathcal{U}(\lambda)$, the RPO-based signature $\mathsf{Sig}_{\mathsf{RPO}}[\lambda, m]$ 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 4: The RPO-based signature scheme

6.3 The Signature Construction

We specialise the signature scheme construction presented by Do Dinh [Do 24, 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 Sec. 5. The resulting scheme $\mathsf{Sig}_{\mathsf{RPO}}$ is presented in Fig. 4.

Theorem 7 ([Do 24, 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 ZK 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*

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

7 Concrete Parameters and Performance

This section presents our parameter choices and performance results. Table 2 summarizes the two implementation presets used in our DEEP-ALI IOP for RPO, which are established and motivated in Sec. D. Relative to the originally submitted implementation, we keep the UDR and LDR parameter sets but send the batched polynomial directly, i.e. with no FRI folding round. For the UDR preset this also reduces the number of queries from 126 to 121 so that the zero-fold masking polynomial fits the same Reed–Solomon dimension.

We benchmarked our signature scheme on an Apple M4 Pro. The results are detailed in Table 3 for the unique decoding regime parameter set, which achieves 113 bits of average-case unforgeability security, and in Table 4 for the list decoding regime parameter set, which provides an enhanced level of security at 122 bits. The reported timings are averages over the benchmark driver in the implementation artifact[6].

These benchmarks instantiate the BCS transformation with either RPO or Blake3. Blake3 is the fastest conventional configuration and is appropriate when signatures are generated and verified directly on a CPU. RPO is the arithmetisation-friendly configuration used for recursive verification, since RPO admits the compact AIR of Section 3 and can therefore be re-proven natively over the Goldilocks field. For RPO we report both the Metal backend, which accelerates the grinding and commitment work, and a CPU-only run. Verification is CPU time in both cases. All recursion-related claims in this paper, in particular the verification-cost analysis of Section 7.1, refer to the RPO instantiation.

[6]The benchmark driver is `scripts/paper-benchmarks.sh` in https://github.com/0xMiden/miden-signature at commit `99304c22e3ed384d0e227b670d1d7465ed7c4e98`.

Table 2: Concrete parameters in the unique and list decoding regimes used in the DEEP-ALI IOP for RPO.

Parameter	Symbol	UDR	LDR
Random oracle output length	λ	256	256
Commitment randomness length	s	256	256
ZK 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}	121	85
FRI folding factor	$t = (t_0, \ldots)$	()	()
Max degree of remainder polynomial	-	255	255

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

Hash function for BCS	Signing (ms)	Verifying (ms)	Size (KiB)
RPO (Metal)	30.4	5.69	80.1
RPO (CPU)	229.2	5.79	80.1
Blake3	7.2	0.52	80.2

Table 4: Performance of the signature scheme in the list decoding regime.

Hash function for BCS	Signing (ms)	Verifying (ms)	Size (KiB)
RPO (Metal)	20.9	5.09	71.3
RPO (CPU)	59.2	5.19	71.3
Blake3	4.6	0.46	71.1

Table 5: Performance of the Preon128 signature scheme for its 3 parameter sets, as reported in its specification (single core).

Parameter set	Signing (s)	Verifying (ms)	Size (KB)
Preon128A	76	397	139
Preon128B	81	561	372
Preon128C	77	2,303	1,725

For context, Table 5 recalls the published Preon128 measurements. Its construction relies on conjectures of varying strength, yielding aggressive (Preon128A), balanced (Preon128B), and conservative (Preon128C) parameter sets. Our approach does not rely on conjectures concerning FRI soundness beyond the Johnson bound, list decodability of Reed–Solomon codes beyond the Johnson bound, or properties of the underlying IOP or its BCS-compilation.

7.1 Verification cost and recursive proving

A central motivation for the design choices of our scheme is that the verifier $\mathsf{Sig}_{\mathsf{RPO}}.\mathsf{SigVer}$ can itself be proven inside a STARK-based zero-knowledge virtual machine (zkVM) operating natively over the Goldilocks field. We analyse here the cost of one execution of the verifier, which directly upper-bounds the number of trace rows required to arithmetise it inside such a zkVM. It decomposes into three logical blocks.

Table 6: Verifier operation counts for the zero-fold UDR and LDR instances. F-mul: base-field multiplications; G-mul: extension-field multiplications; G-inv: batched extension-field inversions; c_e·G-mul: the base-field cost induced by G-muls under Karatsuba ($c_e = 3$ for $e = 2$, $c_e = 6$ for $e = 3$); Perm: calls to the width-12, rate-8 RPO permutation.

Regime	e	n_{FRI}	$\mathbb{F}$-mul	$\mathbb{G}$-mul	$c_e \cdot \mathbb{G}$-mul	$\mathbb{G}$-inv	Perm
UDR	2	121	978	35 498	106 494	3	3 475
LDR	3	85	1 467	19 071	114 426	3	2 683

Transcript hashing (Fiat–Shamir). The BCS transformation replaces every verifier-to-prover message of the underlying IOP by a query to the random oracle. In our setting, this amounts to a constant number of hash calls to derive, in order: the constraint combination randomness $(\lambda_1, \ldots, \lambda_{24})$, the DEEP query $r \in \mathbb{G}$, the FRI batching coefficients $(\gamma_i)_{i\in[w]}$ and $(\delta_j)_{j\in[f]}$, the grinding nonces, and the n_{FRI} query indices. Since the instances of Table 2 use a zero-fold FRI schedule, there is no folding challenge.

DEEP algebraic check. The verifier reconstructs $q(r)$ from the f claimed quotient segment evaluations via Equation (4), and checks Equation (2) at $X = r$. This requires evaluating the 24 RPO AIR constraints at the claimed witness values $(v_{i,1}, v_{i,2})_{i\in[w]}$ and at the interpolated virtual-trace values, forming the λ-linear combination in $\mathbb{G}$, and evaluating the nullifier polynomials at r. The dominant cost comes from the w degree-7 transition constraints: each one composes two applications of the 12×12 MDS matrix, two half-round constant additions, and the $\alpha = 7$ power map.

Query opening phase. For each of the n_{FRI} queries, the verifier opens two Merkle paths of depth $\log_2 |D|$: one into the witness commitment and one into the auxiliary commitment. The auxiliary commitment contains the quotient segments together with the DEEP mask polynomial. The verifier then checks the DEEP quotient at each query point, using a batched inversion for the DEEP denominators and a Horner evaluation of the sent DEEP polynomial.

Operation counts. Table 6 reports operation counts for one execution of the verifier in each parameter regime. The counts are produced by the benchmark driver in the implementation artifact. The $\mathbb{F}$-mul column counts Goldilocks multiplications outside the hash permutation, the $\mathbb{G}$-mul and $\mathbb{G}$-inv columns count extension-field operations, and the Perm column counts calls to the width-12, rate-8 RPO permutation. Thus the arithmetic columns and the permutation column are disjoint; the table records how many hash permutations a recursive verifier must realize, but does not expand their internal AIR cost. In both regimes the query opening phase dominates the extension-field work and the number of hash calls scales linearly with n_{FRI}.

Implications for recursive proving. Because every operation in the UDR verifier is either a native Goldilocks operation, an operation in the quadratic extension of Goldilocks, or a call to RPO, arithmetising it inside an RPO-native Goldilocks zkVM incurs no field-emulation overhead. The LDR verifier is included for comparison with the higher-security non-recursive instance, but its cubic extension arithmetic is not native to Miden VM. The recursive benchmark below therefore uses the UDR instance.

Table 7: End-to-end recursive verification of 2^k UDR signatures inside an RPO-native Miden VM. Proof size is the serialized recursive proof.

k	Signatures	Trace length	Prove	Proof size	Verify
0	1	2^{16}	1.69 s	108 KiB	10.0 ms
1	2	2^{17}	3.22 s	116 KiB	10.2 ms

7.2 An end-to-end recursive verification benchmark

To validate the operation-count analysis of Section 7.1 experimentally, we implemented and benchmarked end-to-end recursive verification of UDR signatures: a STARK proof, produced by the Miden virtual machine, attesting to the correct execution of $\mathsf{Sig}_{\mathsf{RPO}}.\mathsf{SigVer}$ on valid message–signature pairs. The benchmark uses an RPO-native Miden VM branch and RPO for the signature's BCS transform.[7]

Table 7 reports five-run averages on an Apple M4 Pro, using Metal acceleration and a 96-bit target security level for the outer VM proof.

The measured trace lengths are consistent with the operation counts of Table 6: one UDR verification fits in a 2^{16}-row VM trace, and aggregating two signatures fits in a 2^{17}-row trace. This confirms that the verifier arithmetises without field-emulation overhead.

This experiment substantiates the central claim of this work: because every operation of the verifier is native to the Goldilocks field or to the RPO permutation, one signature verification can be proven inside a production STARK-based zkVM at practical cost, enabling the aggregation applications described in Section 1.1.

Acknowledgements

Shahla Atapoor was supported by the Flemish Government through the Cybersecurity Research Program with grant number VOEWICS02.

References

[AKM+22] Tomer Ashur, Al Kindi, Willi Meier, Alan Szepieniec, and Bobbin Threadbare. Rescue-prime optimized. Cryptology ePrint Archive, Paper 2022/1577, 2022. URL: https://eprint.iacr.org/2022/1577.

[BBdSG+23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 581–615. Springer, 2023. doi:10.1007/978-3-031-38554-4_19.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13*, pages 111–120. ACM, 2013. doi:10.1145/2488608.2488623.

[7] The benchmark is reproduced by running `./scripts/paper-benchmarks.sh` in https://github.com/0xMiden/miden-signature at commit `99304c22e3ed384d0e227b670d1d7465ed7c4e98`. The recursive Miden VM benchmark is https://github.com/0xMiden/miden-vm at commit `d23f310c93e2f2d31a106d0d6a0f8a3c8f777358`.

[BCI+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 900–909. IEEE, 2020. doi:10.1109/FOCS46700.2020.00088.

[BCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019. doi:10.1007/978-3-030-17653-2_4.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016. doi:10.1007/978-3-662-53644-5_2.

[BDPVA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions. Keccak Team Website, 2011. URL: https://keccak.team/files/CSF-0.1.pdf.

[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. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023*, volume 14439 of *Lecture Notes in Computer Science*, pages 3–40. Springer, 2023. doi:10.1007/978-981-99-8724-5_1.

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: sampling outside the box improves soundness. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, volume 151 of *LIPIcs*, pages 5:1–5:32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.ITCS.2020.5, doi:10.4230/LIPICS.ITCS.2020.5.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs$^+$ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*, pages 2129–2146. ACM, 2019. doi:10.1145/3319535.3363229.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. URL: https://eprint.iacr.org/2018/046.

[BSBSofS20] Michal Ben-Sasson, Eli Ben-Sasson, and others from StarkWare. Ziggy - a zk-stark based post-quantum secure signature scheme. StarkWare Git, 2020. URL: https://github.com/starkware-libs/ethSTARK/tree/ziggy#11-ziggy.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 136–145. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959888.

[CF24] Alessandro Chiesa and Giacomo Fenzi. zkSNARKs in the ROM with unconditional UC-security. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography - 22nd International Conference, TCC 2024 Part I*, volume 15364 of *Lecture Notes in Computer Science*, pages 67–89. Springer, 2024. doi:10.1007/978-3-031-78011-0_3.

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

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010. doi:10.1007/978-3-642-17373-8_35.

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum ethereum. *IACR Commun. Cryptol.*, 2(1):13, 2025. URL: https://doi.org/10.62056/aey7qjp10, doi:10.62056/AEY7QJP10.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018. URL: https://doi.org/10.13154/tches.v2018.i1.238-268, doi:10.13154/TCHES.V2018.I1.238-268.

[Do 24] Jérémi Do Dinh. Simulation Security in the Random Oracle Model. Master's thesis, School of Computer and Communication Science, EPFL, August 2024. https://jdodinh.io/assets/files/m-thesis.pdf.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the Non-malleability of the Fiat-Shamir Transform. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2012. doi:10.1007/978-3-642-34931-7_5.

[FR25] Thibauld Feneuil and Matthieu Rivain. CAPSS: A framework for snark-friendly post-quantum signatures. *IACR Cryptol. ePrint Arch.*, 2025:61, 2025. URL: https://eprint.iacr.org/2025/061.

[GHR+23] Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. Horst meets fluid-spn: Griffin for zero-knowledge applications. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023, Part III*, Lecture Notes in Computer Science, pages 573–606. Springer, 2023. doi:10.1007/978-3-031-38548-3_19.

[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 Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009. doi:10.1007/978-3-642-10366-7_41.

[KZ20] Daniel Kales and Greg Zaverucha. Improving the performance of the picnic signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):154–188, 2020. URL: https://doi.org/10.13154/tches.v2020.i4.154-188, doi:10.13154/TCHES.V2020.I4.154-188.

[Nat24] National Institute of Standards and Technology. Stateless Hash-Based Digital Signature Standard (SLH-DSA, FIPS 205). Technical Report FIPS 205, U.S. Department of Commerce, 2024. https://doi.org/10.6028/NIST.FIPS.205.

[Pre23] Team Preon. Preon, a zk-snark based signature scheme. Preon Team Website, 2023. URL: https://preon.btq.com/#resources.

[SGJ+24] Vikas Srivastava, Naina Gupta, Arpan Jati, Anubhab Baksi, Jakub Breier, Anupam Chattopadhyay, Sumit Kumar Debnath, and Xiaolu Hou. Ascon-sign submission to the nist post-quantum project, 2024. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/Ascon-sign-spec-web.pdf.

[Sta21] StarkWare Team. ethSTARK documentation. Technical report, StarkWare, 2021. URL: https://eprint.iacr.org/2021/582.

[ZSE+24] Xinyu Zhang, Ron Steinfeld, Muhammed F. Esgin, Joseph K. Liu, Dongxi Liu, and Sushmita Ruj. Loquat: A snark-friendly post-quantum signature based on the legendre PRF with applications in ring and aggregate signatures. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024, Part I*, Lecture Notes in Computer Science, pages 3–38. Springer, 2024. doi:10.1007/978-3-031-68376-3_1.

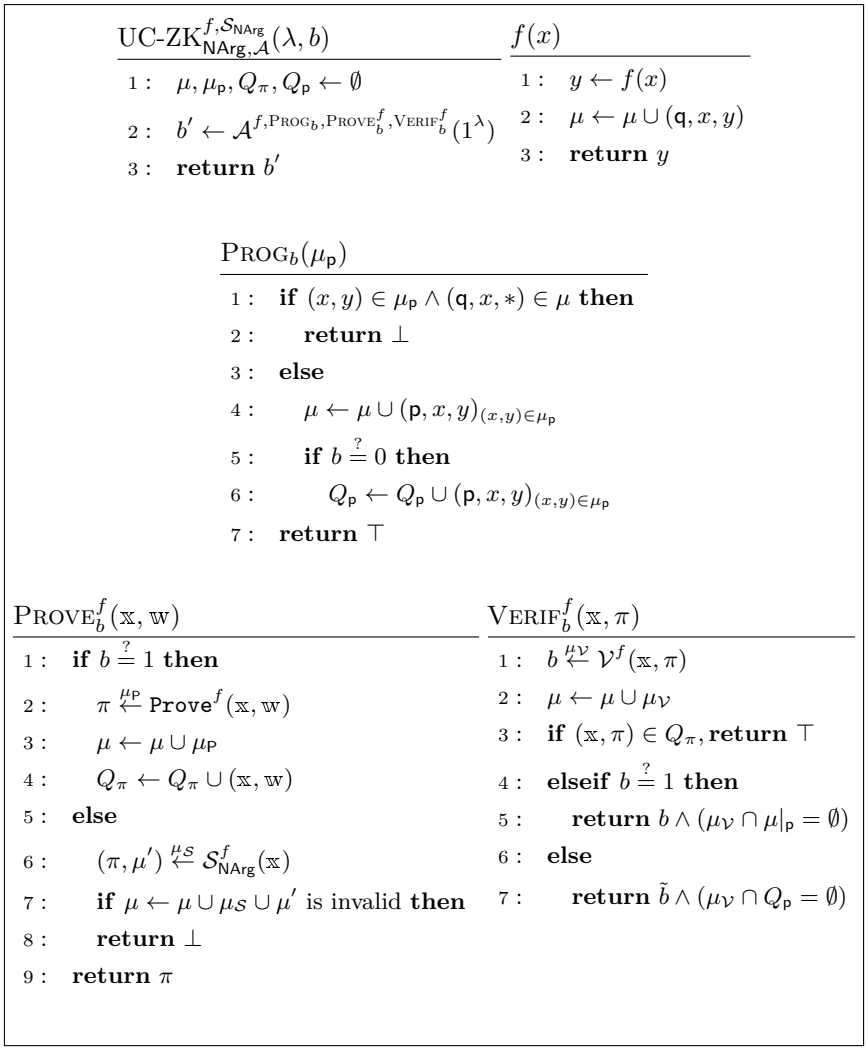

Figure 5: The UC-friendly zero-knowledge game UC-ZK^f in the ROM.

Supplementary Materials

A Definitions and Security Games

A.1 UC-friendly Zero-Knowledge

Definition 15 (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. 5, if there exists a PPT argument simulator $\mathcal{S}^f_{\mathsf{NArg}}$ s. t.

$$|\Pr[\text{UC-ZK}_{\mathsf{NArg},\mathcal{A}}(\lambda, 1) = 1] - \Pr[\text{UC-ZK}_{\mathsf{NArg},\mathcal{A}}(\lambda, 0) = 1]| \leq$$

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

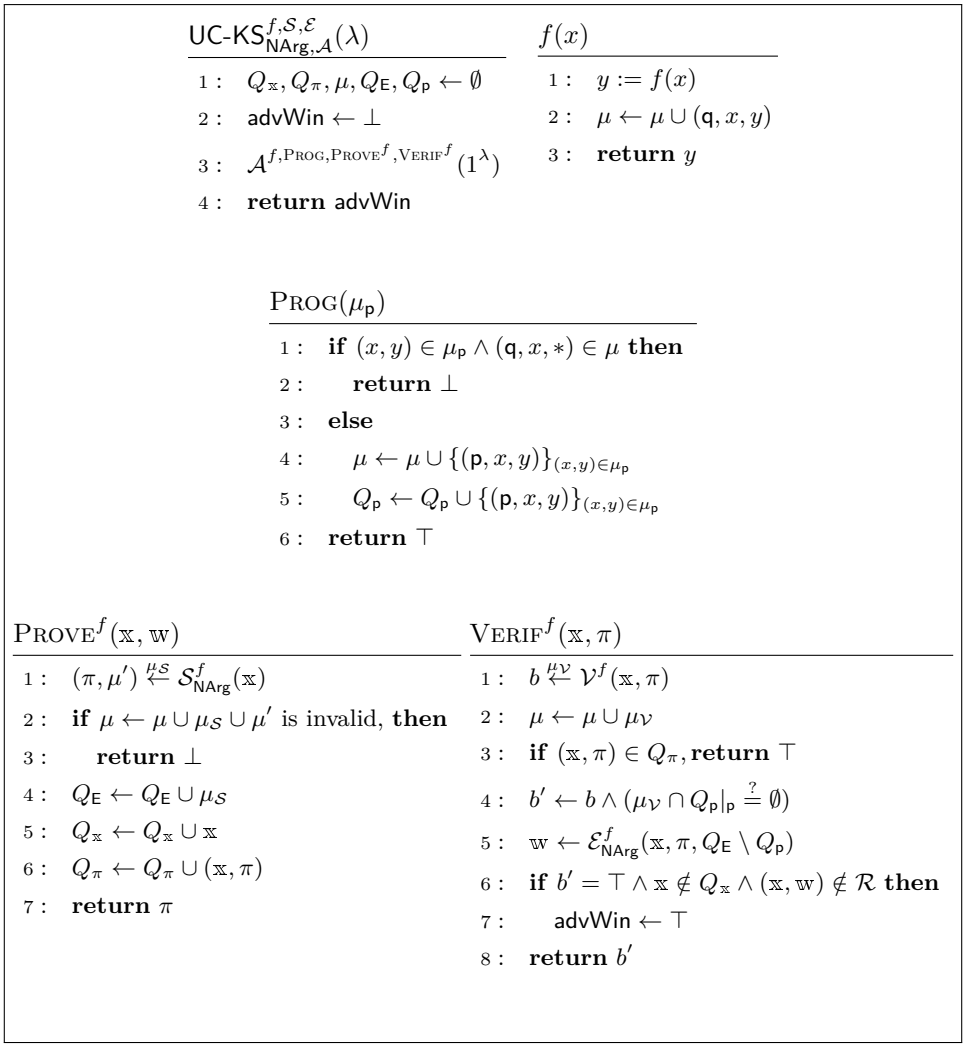

Figure 6: The UC-friendly knowledge soundness game UC-KS^f in the ROM.

A.2 UC-friendly Knowledge Soundness

Definition 16 ([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 $\mathcal{S}^f_{\mathsf{NArg}}$ with error $\epsilon^{\text{UC-KS}}_{\mathsf{NArg}}$ if there exists a PPT extractor $\mathcal{E}^f_{\mathsf{NArg}}$ such that, for every $(t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}})$-budget adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{advWin} \;\middle|\; \begin{array}{l} f \leftarrow \mathcal{U}(\lambda) \\ \mathsf{advWin} \leftarrow \text{UC-KS}^f_{\mathcal{S},\mathcal{E}}(n, \mathcal{A}) \end{array}\right] \leq$$

$$\epsilon^{\text{UC-KS}}_{\mathsf{NArg}}(\lambda, s, t_{\mathsf{q}}, t_{\mathsf{p}}, t_{\mathcal{P}}, t_{\mathcal{V}})$$

,

where the UC-KS game is defined in Fig. 6.

B Security Proofs

B.1 Proof of Theorem 2

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 [BGK+23] goes through, with the appropriate adaptation to the error bounds. □

B.2 Proof of Theorem 3

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

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

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 [BCI+20]. The rest of the argument goes through from then on unaltered.

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

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

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 [BCI+20] yields the above expression on ϵ_{k}. □

B.3 Proof of Theorem 5

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.

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 $\texttt{Prove}_{\text{UC-ZK}}^f$ 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 $\epsilon^{\text{ZK}}_{\mathsf{NArg}} = \epsilon^{\text{UC-ZK}}_{\mathsf{NArg}}$. □

B.4 Proof of Theorem 6

Proof. We define the extractor of game tSKS as $\mathcal{E}^f_{\text{tSKS}}$ to directly call the extractor $\mathcal{E}^f_{\mathsf{UC\text{-}KS}}$ from game UC-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 $\mathcal{S}^f_{\text{tSKS}}$ to directly call the simulator $\mathcal{S}^f_{\mathsf{UC\text{-}KS}}$ from game UC-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 UC-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 UC-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

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

□

C On Correlated Agreements

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 [BCI+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 [BCI+20] to the unique decoding regime. Our version gives tighter bounds when $\alpha \in [\frac{1+\rho}{2}, 1)$.

C.1 Correlated Agreement

Theorem 8 ([BCI+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 ([BCI+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*

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

C.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 [BCI+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 [BCI+20] and we can conclude that

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

□

D 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.

D.0.1 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 κ_{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 ZK 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

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

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

D.0.2 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 [AKM+22, Sections 2.6 and 4.4.2] the security of which was analysed by Bertoni et al. [BDPVA11, Section 4.3]. From this we can conclude

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

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

D.0.3 NArg true-simulation knowledge soundness security

As demonstrated in Sec. 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 correlation 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 $\kappa' = \kappa + 1$.

D.0.4 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:

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

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 ZK security against $2^{\kappa^{\mathrm{ZK}}_{\mathsf{NArg}_{\mathsf{RPO}}}}$-bounded adversaries

D.0.5 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 ZK blow-up parameter $\beta = 32$. The zero-fold implementation sends the batched polynomial directly, so the FRI folding schedule is $t = ()$ and the maximum degree of the sent polynomial is 255. The quotient polynomial is split into 15 segment polynomials, each with 85 coefficients. We choose the proximity parameter $m = 58$ as in the submitted parameter set; it keeps the query-phase error below the target with only two bits of grinding in the query round.

The vector of active RbR soundness errors, in the order of constraint batching, OOD sampling, DEEP proximity, and query sampling, is then bounded by

$$\log_2(\epsilon_{\mathrm{ALI}}) \leq -182, \quad \log_2(\epsilon_{\mathrm{OOD}}) \leq -161, \quad \log_2(\epsilon_{\mathrm{prox}}) \leq -126, \quad \log_2(\epsilon_{\mathrm{qry}}) \leq -126.$$

The implementation uses grinding schedule $(0, 0, 0, 2)$. Since the implementation uses no FRI folding round, there is no separate FRI commit-phase error term, and the query term is evaluated against the base-rate code. This is why the query-phase error meets the target with less grinding than in the submitted one-fold instantiation. Thus the round-by-round soundness remains at least 126 bits.

D.0.6 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 $n_{FRI} = 121$ 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 ZK blow-up parameter $\beta = 32$. The quotient polynomial is split into 15 segment polynomials, each with 118 coefficients.

We send the batched polynomial directly, so the FRI folding schedule is $t = ()$ and the maximum degree of the sent polynomial is 255. Thus there is no FRI commit phase in this implementation.

The vector of active RbR soundness errors, in the order of constraint batching, OOD sampling, DEEP proximity, and query sampling, is then bounded by

$$\log_2(\epsilon_{\mathrm{ALI}}) \leq -120, \quad \log_2(\epsilon_{\mathrm{OOD}}) \leq -116, \quad \log_2(\epsilon_{\mathrm{prox}}) \leq -117, \quad \log_2(\epsilon_{\mathrm{qry}}) \leq -100.$$

As noted in Section 6.3 in [Sta21], we can use proof-of-work, also referred to as grinding, to reduce round soundness errors. The implementation uses grinding schedule $(0, 0, 4, 16)$, which improves the proximity and query-round exponents to at least 121 and 116, respectively. Thus the round-by-round soundness remains at least 116 bits.

D.0.7 Concrete parameters

Soundness in the list decoding regime. Under the list decoding regime analysis, 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 $\mathsf{NArg}_{\mathsf{RPO}}$, 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, 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 $\mathsf{NArg}_{\mathsf{RPO}}$, 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 the zero-knowledge analysis 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 ZK security against 2^{162}-bounded adversaries.

Given these two security levels for $\mathsf{NArg}_{\mathsf{RPO}}$ and the security level of $\mathcal{R}_{\mathsf{RPO}}$, the analysis 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.