Lynx: Symmetric Primitive for Shorter and Faster VOLE-in-the-Head Signatures

Lin Jiao[1], Hongsen Yang[2], Hongrui Cui[3], Yituo He[3], Yonglin Hao[1], Xiaojie Guo[4], Qunxiong Zheng[2], Jiang Zhang[1], Yu Yu[2,5], and Kang Yang[1]

[1] State Key Laboratory of Cryptology, Beijing, China
[2] Information Engineering University, Zhengzhou, China
[3] Shanghai Jiao Tong University, Shanghai, China
[4] Shanghai Qi Zhi Institute, Shanghai, China
[5] East China Normal University, Shanghai, China jiaolin_jl@126.com,moonlight0833@outlook.com, rickfreeman@sjtu.edu.cn,yituo_he@sjtu.edu.cn,haoyonglin@yeah.net,guoxiaojie@sqz.ac.cn,qunxiong_zheng@163.com, zhangj@sklc.org,yuyuathk@gmail.com,yangk@sklc.org

Abstract. VOLE-in-the-Head (VOLEitH) is one of the most promising frameworks to design post-quantum digital signatures based on symmetric primitives. However, all existing symmetric primitives do not capture the specialized characteristics of the VOLEitH framework and are not VOLEitH-friendly, leaving room for improving the efficiency of VOLEitH-based signatures. In this paper, we propose a VOLEitH-friendly symmetric primitive called Lynx, which is optimal in terms of the number of required VOLE correlations that directly determines the efficiency of VOLEitH-based signature schemes. In particular, Lynx adopts a multi-branch structure featuring a new truncation function: (a) nonlinear components are set to minimize the witness length and polynomial degree, as well as the number of finite-field multiplications; (b) linear layers are strategically interleaved to strengthen security. The security of Lynx is rigorously validated by covering all current attacks in the presence of both classical and quantum adversaries. Built upon Lynx, we design a post-quantum signature scheme, Lynxer, in the VOLEitH framework, which is shorter and faster than all known post-quantum signature schemes from symmetric primitives. According to our experimental results, compared to the state-of-the-art symmetric-based signature schemes in the same setting, i.e., Rainier (CCS'22), AIMer (CCS'23) and FAESTv2 (Crypto'25), our signature scheme Lynxer reduces the "public-key size + signature size" by 25%∼51%, and improves the signing (resp., verification) time up to 90.5% (resp., 89.9%).

Keywords: Post-quantum cryptography · Digital signatures · VOLE-in-the-Head

1 Introduction

Digital signatures are a cryptographic cornerstone, essential across the digital economy. Post-quantum signature schemes are being rapidly deployed in critical infrastructure—to secure software/firmware update pipelines against supply chain attacks that could compromise billions of devices; authenticate IoT sensors and industrial control systems in smart grids and manufacturing facilities; and power quantum-resistant blockchains and digital identity systems. It is well-known that the widely deployed RSA-based and discrete-logarithm-based signature schemes are insecure when large-scale quantum computers become a reality [59]. To address this security threat, the National Institute of Standards and Technology (NIST) called for submissions of Post-Quantum Cryptography (PQC) algorithms in 2016. For now, NIST has standardized two post-quantum signature schemes (i.e., ML-DSA [53] and SLH-DSA [54]), and is standardizing a third (i.e., Falcon [29]). To broaden the design diversity and hardness assumptions, NIST has issued a new call for additional post-quantum signature candidates [52], currently in the third round.

Symmetric-based signature schemes (i.e., those based only on symmetric primitives) are among the most promising post-quantum signature schemes. The MPC-in-the-Head (MPCitH) framework [37] is a key paradigm for designing symmetric-based signature schemes. In such signature schemes, a pair of public and secret keys consists of $y = f(x)$ and x, where f is a One-Way Function (OWF); a signature is a non-interactive zero-knowledge (NIZK) proof of knowledge of secret key x, generated by locally emulating an MPC protocol execution in the head and applying the cut-and-choose technique for soundness. Picnic [64], built on [31,16,41], is the first post-quantum signature scheme based on symmetric primitives and MPCitH in NIST's PQC standardization process and a third-round alternate candidate of the NIST PQC competition. It uses an MPC-friendly block cipher LowMC [4] to reduce the number of AND gates and thus the size of signatures. Subsequent works such as Banquet [12] and Limbo [21] improved the efficiency of Picnic by replacing LowMC with AES.

While LowMC is a block cipher, it is overkill for MPCitH-based signature schemes that only require an OWF. Initially, Dobraunig et al. [23] proposed a tailor-made MPCitH-friendly OWF (named RAIN), and used it to design a symmetric-based post-quantum signature scheme Rainier in the Picnic framework. Later, Kim et al. [44] designed an alternative MPCitH-friendly OWF (named AIM) and combined it with an improved Picnic framework BN++ [40] to

obtain the symmetric-based signature scheme AIMer. Specifically, AIMer has been selected as a finalist in Korea's PQC competition [45]. Due to the cryptanalysis [66,48] that weakened its security, AIM was upgraded to AIM2 to resist those attacks [43]. Design-wise, RAIN uses full-state large-field inversion S-boxes, whereas AIM2 adopts Mersenne-exponent power-map S-boxes.

Recently, the VOLE-in-the-Head (VOLEitH) framework [11] was introduced to design symmetric-based post-quantum signature schemes. VOLEitH can be viewed as a fundamental improvement over MPCitH [8,27]; it locally emulates a Vector Oblivious Linear Evaluation (VOLE) protocol [56] execution in the head, and then uses a VOLE-based zero-knowledge proof (e.g., QuickSilver [63]) to prove knowledge of the secret key. Using AES to instantiate an OWF, Baum et al. [11] designed a VOLEitH-based post-quantum signature scheme FAEST, and FAEST achieves shorter signature size and faster signing/verification time than MPCitH-based signature schemes from AES. The first version of FAEST (i.e., FAESTv1) was submitted to the NIST PQC competition for additional signature schemes [10] ; FAESTv1 was improved in [8,7], resulting in FAESTv2 [9], advanced to the third round of the competition. However, compared to the symmetric-based signature schemes from MPCitH-friendly OWFs [23,44], the most optimized version of FAEST (i.e., FAESTv2), based on AES, still requires a larger communication cost (i.e., "public-key size + signature size"[6]). This motivates us to design VOLEitH-friendly symmetric primitives.

However, no tailored symmetric primitive has been designed to capture the specialized characteristics of the VOLEitH framework. Specifically, the primary design goal of existing MPCitH-friendly symmetric primitives [23,44] is to reduce the number of multiplication gates, since MPCitH compiles an OWF into a circuit, and fewer multiplication gates enable shorter signature sizes and faster computation times. Nevertheless, the VOLEitH framework encodes an OWF as a polynomial over a finite field; there, the polynomial degree and witness length dominate the signature size and computation time. Therefore, RAIN [23] and AIM2 [43] are *not* VOLE-friendly. Furthermore, to prove security in the Quantum Random Oracle Model (QROM), the VOLEitH-based signature scheme FAESTv2 assumes that the underlying function f behind public and private keys is not only one-way but also satisfies a special pseudorandomness property (i.e., $f(x)$ is computationally indistinguishable from a uniformly random value when x is sampled at random) [7]. Such pseudorandomness has not been considered by existing MPCitH-friendly symmetric primitives [23,44]. Although MPC-friendly block ciphers such as LowMC suffice to provide the required pseudorandomness, they are unnecessarily heavy for VOLEitH.

In the pre-quantum era, symmetric primitives often support 128-bit, 192-bit, and 256-bit security against classical attacks. Accordingly, in the post-quantum era, they are expected to support 80-bit, 128-bit, 192-bit, and 256-bit security in the presence of quantum attacks, corresponding to 160-bit, 256-bit, 384-bit, and 512-bit classical security, respectively, due to the Grover's quantum algorithm [32][7]. Cryptographic algorithms with classical security beyond 256 bits may be desirable in high-assurance applications (e.g., national infrastructure and long-term archives). In particular, both NIST and ISO/IEC have reserved space for security levels beyond 256 bits [5,38]. Furthermore, cryptographic hash functions with 384- or 512-bit security levels, such as SHAKE-512 [51] with 512-bit security, have already been developed. In the recent PQC competition [36], for both public-key and private-key cryptographic algorithms, supporting a 512-bit security level is required and providing a 384-bit security level is encouraged. However, to date, there are no symmetric-based signature schemes that support 384-bit or 512-bit security levels. This motivates us to design a post-quantum signature scheme from a VOLEitH-friendly symmetric primitive, which supports all six security levels (i.e., 128, 160, 192, 256, 384 and 512 bits) solidly.

1.1 Our Contributions

In this paper, we propose Lynx, a tailored VOLEitH-friendly symmetric primitive supporting six security levels (128, 160, 192, 256, 384, and 512 bits). It features a multi-branch structure centered on a finite-field-multiplication-based truncation function, as well as inverse-function S-boxes and interleaved linear layers. This design simultaneously minimizes the crucial metrics that determine the "pk + sig" size, i.e., the witness length and the degree of constraint polynomials (i.e., those proven in zero-knowledge). Moreover, while prioritizing optimal communication cost, Lynx also minimizes the number of finite field multiplications within the constraint polynomials at each security level, which is a factor affecting signing and verification latency (see Section 1.2 for technical details). Since the attack model limits adversaries to accessing only a single input-output pair, our analysis comprehensively evaluates Lynx against advanced algebraic attacks targeting key recovery, as cited in [66,48,15,62,61,58,19,22,46,60]. Furthermore, it extends to an assessment of pseudorandomness, incorporating standardized randomness tests as empirical evidence for the output-pseudorandomness assumption on which the QROM security proof (as in FAESTv2 [9]) relies. We also assess the resilience of Lynx against quantum attacks, accounting for the quadratic speedup of Grover's algorithm. Built on Lynx, we propose Lynxer, a new post-quantum signature scheme in the VOLEitH framework, and offer both "small" and "fast" variants for minimal communication cost and low latency, respectively, at each security level.

[6] "public-key size + signature size" is abbreviated as "pk + sig" size.
[7] When clear from context, x-bit security means classical x-bit security.

We evaluate the performance of Lynxer using both reference and optimized implementations. The results show that Lynxer outperforms state-of-the-art symmetric-based post-quantum signature schemes in both "pk + sig" size and signing/verification runtime. Lynxer's minimum "pk + sig" sizes are 2970/6700/12064 bytes at 128/192/256-bit common security levels. Compared to the optimized VOLEitH-based signature schemes from standard AES and its EM variant (i.e., FAESTv2 [9]), Lynxer reduces the "pk + sig" size by 34.6%∼44.2% for FAEST and by 24.6%∼37.0% for FAEST-EM. Compared to the symmetric-based signature schemes from MPCitH-friendly OWFs, the reduction is 41.8%∼50.8% for Rainier-3/4 and 26.9%∼41.1% for the updated AIMer (i.e., AIMer2 [42]). In reference implementations, Lynxer reduces the signing and verification runtime by 72.6%∼80.6% for FAEST, 52.3%∼64.5% for FAEST-EM, and 50.4%∼74.1% for AIMer in small variants; the reduction is 83.8%∼90.5% for FAEST, 79.4%∼86.0% for FAEST-EM, and 9.5%∼48.4% for AIMer[8] in fast variants. In optimized implementations, Lynxer reduces the runtime by 18.9%∼35.8% for FAEST and 9.7%∼19.6%for FAEST-EM in small variants; the reduction is 30.1%∼42.0% for FAEST, 10.1%∼31.8% for FAEST-EM, and 65.1%∼85.1% for Rainier-3/4 in fast variants. These performance gains highlight the necessity of developing a VOLEitH-friendly symmetric primitive, and our work precisely fills this crucial gap in post-quantum signature design.

1.2 Technical Overview

This section addresses three core design challenges in Lynx; further details are in Section 3.

Minimizing communication cost. The core focus in QuickSilver-style VOLEitH protocols is the "pk + sig" size, which is dominated by (i) the witness length ℓ and (ii) the degree bound of constraint polynomials d. Our primitive is defined over $\mathbb{F}_{2^n}$, where the parameter n matches the desired security level. It is engineered to jointly minimize these two parameters, achieving the highly efficient configuration of $d = 2$ and $\ell = 3n$. This configuration is minimal under the constraint of resisting the attack described in [66], thereby yielding a signature size that depends optimally on $5n+B$ across all security levels in the VOLEitH framework, where B is a small constant.

Ensuring robust security. Achieving better communication cost must not compromise security. Lynx's structure, especially the nonlinear truncation function, generates cross-terms among pairwise witness variables in the constraint polynomials. Concurrently, the linear layers are carefully designed to ensure high-degree terms for every witness variable. These techniques strengthen resilience against advanced large-field algebraic attacks (e.g., Magic Pot, resultant and linearized polynomial decomposition attacks [15,62,61,60]). Moreover, the linear layers are interleaved with nonlinear components, effectively preventing the elimination of variables from spawning additional Boolean equations [48,46]. This isolation enhances resistance against $\mathbb{F}_2$-based algebraic attacks (such as Gröbner Basis Attacks [58]). Finally, the use of random linear layers and constant additions ensures that attacks requiring multiple input-output pairs become fundamentally inapplicable.

Minimum Runtime. We adopt the inverse function x^{-1} as the S-boxes across all six security levels ranging from 128-bit to 512-bit within a uniform structure. Under the minimal communication cost, Lynx's constraint polynomials require the fewest number of finite-field multiplications and no Frobenius automorphisms ($x \mapsto x^{2^i}$), namely exactly 3 field multiplications at each security level. Furthermore, the inverse function achieves high algebraic degrees in both the forward and inverse evaluation directions. Moreover, the number of linear layers is also minimal under the security requirement. Consequently, Lynxer delivers scalable security across all target levels with optimized performance.

2 Preliminaries

2.1 Notation

Let $\mathbb{Z}$ denote the set of integers and $\mathbb{Z}^+$ be the set of positive integers. Let $n \in \mathbb{Z}^+$ be the security parameter. Let $\mathbb{F}_2$ and $\mathbb{F}_{2^n}$ denote the Galois field of order 2 and 2^n. For $i, j \in \mathbb{Z}$ with $i < j$, define $[i, j) := \{i, i+1, \ldots, j-1\}$, and let $[i] := [0, i)$. Vectors are denoted by bold lowercase letters (e.g. $\mathbf{v}$), and their components are accessed through square brackets (e.g. $\mathbf{v}[i]$). We use $r \leftarrow \mathcal{S}$ to denote sampling a uniformly random element from the set $\mathcal{S}$.

2.2 IT-MAC

We adopt the generalized Information-Theoretic Message Authentication Code (IT-MAC) definition in [1], adapted to our setting.

Definition 1. *For $d \in \mathbb{Z}$, a degree-d IT-MAC for $a \in \mathbb{F}_{2^n}$ is defined as $[\![a]\!]^d := (p_a(x), k_a, \Delta)$, where a prover ($\mathsf{P}$) holds a polynomial $p_a(x) \in \mathbb{F}_{2^n}[x]$ with $\deg(p) = d$ and leading coefficient a; while a verifier (V) holds the evaluation $k_a = p_a(\Delta)$, for a global key $\Delta \in \mathbb{F}_{2^n}$ unknown to P. The degree superscript d is omitted when clear.*

[8] The range is slightly broader, with a low point only at AIMer-192f—attributed to its balanced parameter configuration.

Let $[\![a]\!]^{d_a}, [\![b]\!]^{d_b}$ be two IT-MACs, with corresponding polynomials $p_a(x), p_b(x)$ and evaluations k_a, k_b, assuming w.l.o.g. $d_a \geq d_b$. IT-MAC supports the local homomorphic operations:

- **Addition**. To compute $[\![a+b]\!]^{d_a} = [\![a]\!]^{d_a} + [\![b]\!]^{d_b}$, P computes $p_{a+b}(x) = p_a(x) + p_b(x) \cdot x^{d_a - d_b}$, while V computes $k_{a+b} = k_a + k_b \cdot \Delta^{d_a - d_b}$.
- **Multiplication**. To compute $[\![ab]\!]^{d_a \cdot d_b} = [\![a]\!]^{d_a} \cdot [\![b]\!]^{d_b}$, P computes $p_{ab}(x) = p_a(x) \cdot p_b(x)$, while V computes $k_{ab} = k_a \cdot k_b$.
- **Constants**. For any $c \in \mathbb{F}_{2^n}$ and $d \in \mathbb{Z}$, define $[\![c]\!]^d$ by $p_c(x) = cx^d$ and $k_c = c\Delta^d$. Combining this with the above operations, constant addition and scalar multiplication preserve degree. Then, subtraction follows as $[\![a]\!] - [\![b]\!] := [\![a]\!] + [\![-1]\!]^1 \cdot [\![b]\!]$. Moreover, to de-randomize a random IT-MAC $[\![r]\!]^1$ (where $r \leftarrow \mathbb{F}_2$) into a target value $x \in \mathbb{F}_2$, P publishes $\delta := x - r$, and both parties compute $[\![x]\!]^1 := [\![r]\!]^1 + \delta$.

IT-MAC satisfies the hiding and binding properties:

- **Hiding**. Random degree-1 IT-MACs are typically generated from VOLE protocols (e.g., the one in Section 2.4), which inherently ensure that the committed value a is independent of V's authentication material. Thus, hiding is guaranteed.
- **Binding**. To open a committed value a, P sends its (purported) associated polynomial $\tilde{p}_{a'}(x)$ to V, who checks whether $\tilde{p}_{a'}(\Delta) = k_a$. Any attempt by P to open a different value $a' \neq a$ while passing this check implies $\tilde{p}_{a'}(\Delta) = p_a(\Delta)$. Assuming $\Delta \leftarrow \mathbb{F}_{2^n}$, the probability of such a collision is at most $\frac{\deg(p_a)}{2^n}$.

2.3 QuickSilver

QuickSilver [63] is an efficient zero-knowledge protocol for proving relations over IT-MAC-authenticated values. In this paper, we only require its instantiation for degree-2 relations; generalization to higher-degree relations is feasible. Given $\boldsymbol{k} \in \mathbb{F}_2^\ell$, suppose P and V already hold $[\![\boldsymbol{k}]\!]^1$. Then P can prove $f(\boldsymbol{k}) = 0$, where f is a degree-2 ℓ-variate polynomial, as follows: Both parties compute $[\![e]\!]^2 := f([\![\boldsymbol{k}]\!]^1)$, thereby reducing the check of $f(\boldsymbol{k}) = 0$ to checking whether $e = 0$, which is accomplished via IT-MAC opening.

Utilizing the homomorphism of IT-MAC, multiple degree-2 relations can be verified simultaneously. Given the authenticated witness $[\![\boldsymbol{k}]\!]^1$, for $m \in \mathbb{Z}^+$, and each $i \in [m]$, suppose we need to check $f_i(\boldsymbol{k}) = 0$, where each f_i is a degree-2 ℓ-variate polynomial $f_i(x)$. The protocol proceeds:

- P and V compute $[\![e_i]\!]^2 := f_i([\![\boldsymbol{k}]\!]^1)$;
- V randomly samples and sends $\chi_1, \ldots, \chi_m \leftarrow \mathbb{F}_{2^n}$ to P;
- P and V compute $[\![\bar{e}]\!]^2 := \sum_{i=1}^m \chi_i [\![e_i]\!]^2$, and V checks that $\bar{e} = 0$ via IT-MAC opening.

Finally, since the coefficients of $p_{\bar{e}}(x)$ depend on the witness $\boldsymbol{k}$, to ensure zero-knowledge, P masks $[\![\bar{e}]\!]^2$ with a random IT-MAC of zero (i.e., $[\![0]\!]^2$, whose underlying polynomial $p_0(x)$ has a uniformly random degree-1 coefficient). This is achieved by aggregating n random degree-1 IT-MACs of elements from $\mathbb{F}_2$ into a single IT-MAC committing to a random element in $\mathbb{F}_{2^n}$. We summarize the relevant results:

Lemma 1 (QuickSilver [63]). *Suppose $\Delta \leftarrow \mathbb{F}_{2^n}$ is uniformly random. Then the batch checking procedure satisfies:*

- ***Soundness***. *If $\exists i$ s.t. $f_i(\boldsymbol{k}) \neq 0$, then* $\Pr[\mathsf{V}$ *accepts*$] \leq \frac{3}{2^n}$;
- ***HVZK***. *$\exists$ a simulator that samples V's view without having access to $\boldsymbol{k}$.*

2.4 VOLE-in-the-Head

VOLEitH is a protocol for generating degree-1 IT-MACs of random elements from $\mathbb{F}_2$ in a publicly-verifiable manner; it enables the transformation of a QuickSilver protocol into a publicly verifiable zero-knowledge proof system. We sketch its main steps as follows. Let $\mathsf{PRG} : \mathbb{F}_2^\eta \to \mathbb{F}_2^\ell$ be a pseudorandom generator (PRG).

- For $\tau, k \in \mathbb{Z}^+$, using a batch all-but-one vector commitment (BAVC) scheme, P commits to τ sets of 2^k-length seed vectors $\{\mathsf{sd}_j^i\}_{i \in [\tau],\ j \in [2^k]}$.
- For $i \in [\tau]$, P computes $\boldsymbol{u}^i := \sum_{j \in \mathbb{F}_{2^k}} \mathsf{PRG}(\mathsf{sd}_j^i)$, and $\boldsymbol{v}^i := \sum_{j \in \mathbb{F}_{2^k}} (-j) \cdot \mathsf{PRG}(\mathsf{sd}_j^i)$. [9]
- For $i \in [1, \tau)$, P computes $\boldsymbol{c}^i := \boldsymbol{u}^0 - \boldsymbol{u}^i$ and sends it to V. Then P defines $\boldsymbol{u} := \boldsymbol{u}^0$ and $\boldsymbol{v} := \boldsymbol{v}^0 \| \cdots \| \boldsymbol{v}^{\tau-1}$, i.e., every entry of $\boldsymbol{v}$ is the concatenation of the corresponding entries from $\boldsymbol{v}^0, \ldots, \boldsymbol{v}^{\tau-1}$. $\boldsymbol{u}$ and $\boldsymbol{v}$ define a vector of degree-1 IT-MACs, i.e., $[\![\boldsymbol{u}]\!]^1$ is defined by $p_{\boldsymbol{u}}(x) := \boldsymbol{u}x + \boldsymbol{v}$.
- To prevent P from sending inconsistent $\boldsymbol{c}^i$ values, P and V execute the SoftSpokenOT check [56].

[9] We slightly abuse $\mathbb{F}_{2^k}$ and $[2^k]$ by their implicit bijection.

- Using the resulting random IT-MACs, P and V run the QuickSilver protocol. Crucially, QuickSilver requires V to send only random coefficients $\{\chi_i\}$, so the protocol proceeds even if V has not yet got its IT-MAC values.
- After P sends all QuickSilver messages, V randomly samples τ indices $\Delta^i \leftarrow [2^k]$ and sends them to P. Using BAVC opening, for the i-th seed set, P opens all seeds except the one indexed by Δ^i. Then, V computes $\boldsymbol{w}^i := \sum_{j\in\mathbb{F}_{2^k}}(\Delta^i - j)\cdot \mathsf{PRG}(\mathsf{sd}^i_j)$. For $i \in [1,\tau)$, V additionally adds $\boldsymbol{c}^i \cdot \Delta^i$ to $\boldsymbol{w}^i$. Finally, V defines $\boldsymbol{w} := \boldsymbol{w}^0 \| \cdots \| \boldsymbol{w}^{\tau-1}$ (element-wise concatenation). $\boldsymbol{w}$ defines V's authentication material in $[\![\boldsymbol{u}]\!]^1$, i.e., $k_{\boldsymbol{u}} = \boldsymbol{w}$. Using $k_{\boldsymbol{u}}$, V performs the final consistency checks in the QuickSilver protocol and outputs the verdict.

We summarize the IT-MAC generation functionality by VOLEitH in Figure 1 and recall the results in Lemma 2.

Lemma 2 (VOLEitH [11]). *Under the Fiat-Shamir transform, the VOLEitH framework described above transforms a public-coin zero-knowledge protocol in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model into a non-interactive zero-knowledge proof system in the random oracle model.*

Functionality $\mathcal{F}_{\mathsf{sVOLE}}$

Parameters: Let $\ell \in \mathbb{Z}^+$ be a length parameter. Three parties: a prover P, a verifier V, and a simulator S.
Init: Upon receiving (Init) from P and V, proceed as:

1. Sample $\Delta \leftarrow \mathbb{F}_{2^n}, \boldsymbol{u} \leftarrow \mathbb{F}_2^\ell$, and $\boldsymbol{v} \leftarrow \mathbb{F}_{2^n}^\ell$. Define $\boldsymbol{w} := \boldsymbol{u}\Delta + \boldsymbol{v}$.
   - If P is corrupted, receive $\boldsymbol{u} \in \mathbb{F}_2^\ell, \boldsymbol{v} \in \mathbb{F}_{2^n}^\ell$ from S, then recompute $\boldsymbol{w}$;
   - If V is corrupted, receive $\Delta \in \mathbb{F}_{2^n}$ and $\boldsymbol{w} \in \mathbb{F}_{2^n}^\ell$ from S, then recompute $\boldsymbol{v} = \boldsymbol{w} - \boldsymbol{u}\Delta$.
2. Send $(\boldsymbol{u}, \boldsymbol{v})$ to P.
3. If P is corrupted, receive a leakage query $L \subseteq \mathbb{F}_{2^n}$ from S.

Get: Upon receiving (Get) from P and V, proceed as:

1. If $\Delta \notin L$, send (failed) to V and abort.
2. Otherwise, send $(\Delta, \boldsymbol{w})$ to V.

Fig. 1: (Subfield) VOLE functionality by VOLEitH.

Digital Signatures from VOLEitH. Using the Fiat-Shamir transform [28], we obtain a signature scheme from the VOLEitH protocol. The signing/verification keys correspond to the input/output values of an OWF, while each signature corresponds to a transcript proving knowledge of the OWF preimage. We recall the signature size formula in Eq.(1), as it critically informs the design of the underlying OWF. The meaning of each parameter is specified in Section 4.2. The BAVC leaf-node commitment size is fixed to $2n$ bits [9].

$$\begin{aligned}\text{Size} = & \underbrace{\ell}_{\text{Fixing witness}} + \underbrace{(\tau-1)(\ell+dn+B)}_{\boldsymbol{c}^1,\ldots,\boldsymbol{c}^{\tau-1}} + \underbrace{(T_{\mathsf{open}}+2\tau)n}_{\text{BAVC Opening}} \\ & + \underbrace{(d-1)n}_{\text{QuickSilver}} + \underbrace{n+B}_{\text{SoftSpoken}} + \underbrace{n}_{\text{Challenge}} + \underbrace{128}_{\text{Salt}} + \underbrace{32}_{\text{Counter}}\end{aligned} \tag{1}$$

3 Lynx: VOLEitH-Friendly Primitive

3.1 Design Rationale

Choice of Field. Lynx is a primitive defined over $\mathbb{F}_{2^n}$. Its security level matches the field extension degree n. Tailored for VOLEitH, it is built on nonlinear operations over $\mathbb{F}_{2^n}$, exactly the underlying field adopted in VOLE, which incurs no field conversion overhead and can fully leverage modern CPU instruction sets for acceleration. Linear layers are realized as matrix multiplications over $\mathbb{F}_2$, thereby preserving the proof size while achieving a high algebraic degree when viewed as linearized polynomials over $\mathbb{F}_{2^n}$.

Choice of Structure. The signature size Eq.(1) is linear in

$$\hat{\ell} = \ell + d \cdot n + B, \tag{2}$$

where ℓ is the length of the secret witness, comprising the secret key input and the intermediate variables needed to commit; d is the maximum degree of the constraint polynomials; and B is a constant associated with the VOLE consistency

check [63,9]. For a fixed $\hat{\ell}$, reducing the number of field multiplications (as well as Frobenius automorphisms) in the constraint polynomials improves both signing and verification efficiency. Our goal is to minimize the "pk + sig" size while reducing the runtime.

To minimize the signature size, we jointly reduce both d and ℓ. Nonlinear components are essential for cryptographic security; hence $d \geq 2$. As in most QuickSilver-style systems [63], Lynx adopts $d = 2$ to maintain a tight bound. Next, we turn to ℓ. Suppose the primitive contains ϑ nonlinear components; then generally, $\vartheta - 1$ intermediate variables must be committed—only the nonlinear component connected to the primitive's output can be directly checked and requires no intermediate commitment. Thus, $\ell = \vartheta n$, and the objective reduces to minimizing ϑ.

- Can ϑ be 2? No —the guess-and-determine S-box linearization technique from [66] enables key recovery by linearizing one nonlinear component and solving the lower-dimensional quadratic system induced by the remaining nonlinear component, thereby posing a serious threat to such primitives as two-round RAIN.
- Lynx adopts $\vartheta = 3$. These three nonlinear components require careful design and placement to resist all potential attacks. Specifically, $\mathbb{F}_{2^n}$-based algebraic attacks (such as Magic Pot, resultant and linearized polynomial decomposition attacks [15,62,61,60]) exploit structural properties of the primitive including S-box composition and linear layer design. In contrast, $\mathbb{F}_2$-based algebraic attacks (such as Gröbner Basis Attacks [58]) rely critically on the overdefined degree of the quadratic Boolean equation system derived from constraint polynomials. To achieve one-wayness, we depart from conventional truncation designs (especially those relying on linear operations) and introduce a multi-input, single-output nonlinear component. Finite-field multiplication is the natural choice. To realize "multi-input", our design adopts a multi-branch framework: each branch applies an inverse-function S-box to an affine transformation of the secret input in parallel; the resulting outputs are then multiplied collectively over $\mathbb{F}_{2^n}$. Combined with a feed-forward of all witnesses and a random and uniform constant, this produces the final output. The design maintains structural uniformity across six security levels ($n = 128, 160, 192, 256, 384, 512$). Moreover, the truncation function based on finite-field multiplication inherently introduces cross-terms among intermediate variables, thereby strengthening resilience against these large-field algebraic attacks. Lynx employs the inverse function in all S-box branches, thereby ensuring a high algebraic degree in both forward and backward evaluations.

Therefore, $\hat{\ell} = 5n + B$ remains optimal across all security levels within VOLEitH paradigm, as well as the number of finite-field operations in Lynx's constraint polynomials including 3 field multiplications and no Frobenius automorphisms that affect the signing and verification runtimes.

Configuration of Linear Layers. When setting the linear layers, we ensure that high-degree, dense linearized polynomials separate all nonlinear components (including those between input and output, across branches, and between S-boxes and the final truncation function). This prevents variable elimination from generating additional quadratic Boolean equations and keeps the overdefined degree of the Boolean equation system derived from constraint polynomials sufficiently low [46]. Moreover, the secret key and intermediate variables are all routed through these linear layers, ensuring their appearance with high-degree terms in Lynx's constraint polynomial system over $\mathbb{F}_{2^n}$, thereby providing resistance against large-field algebraic attacks. Consequently, Lynx employs exactly four linear layers.

Moreover, Lynx incorporates randomly generated linear layers derived from the user's IV via an extendable output function (XOF). Equipping each branch with a randomly sampled linear layer for every user suffices to endow Lynx with instance-specific security, against attacks requiring multiple input-output pairs. To achieve efficiency, we fix the linear layer in the feed-forward path. It satisfies strict cryptographic criteria, thereby eliminating the risk of random degradation.

Setting of Constants. Constants in each branch must be set to prevent attackers from using identical guesses to conduct attacks. They must also be IV-dependent to thwart strong key-reuse attacks (excluded under standard security model), where the same secret key is used with different IVs.

Furthermore, to support the QROM security proof in FAESTv2 [9], Lynx requires not only one-wayness but also output pseudorandomness. It provides resilience against both statistical distinguishers and standardized randomness tests.

3.2 Specification

Lynx is an IV-based symmetric primitive over $\mathbb{F}_{2^n}$. It takes a secret key k and an initial vector iv as inputs, and outputs $t \in \mathbb{F}_{2^n}$. Lynx provides n =128-, 160-, 192-, 256-, 384-, 512-bit security against preimage recovery on k and pseudorandomness distinguishing on t. The construction combines (i) nonlinear components of inverse-function S-boxes $S_i(x) = x^{-1}$ for $i = 1, 2$ and field multiplications as truncation function, with (ii) linear components of random constant additions c_i for $i = 1, 2$ and binary matrix multiplications L_i for $i = 0, 1, 2, 3$. The schematic of Lynx is shown in Figure 2. Let v_i be the outputs of S_i, for $i = 1, 2$. Lynx is defined as

$$\begin{cases} v_1 = (k + c_1)^{-1}, \\ v_2 = (L_0(k) + c_2)^{-1}, \\ t = L_1(v_1) \cdot L_2(v_2) + L_3(k + v_1 + v_2), \end{cases} \tag{3}$$

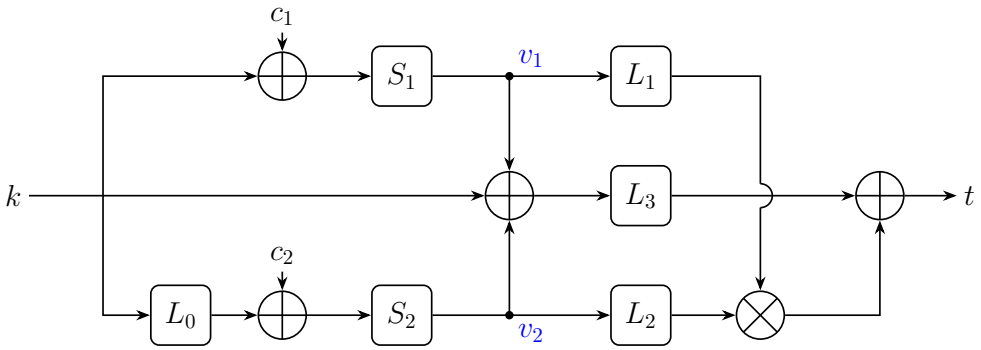


Fig. 2: The structure of Lynx (Input: key k and iv; Output: t; Intermediate variable: v_1 and v_2; Nonlinear and linear components: S_i and L_i; Constant: c_i). L_0, L_1, L_2 and c_1, c_2 are derived from XOF(iv).

where the inverse function $S(x) = x^{-1}$ over $\mathbb{F}_{2^n}$ is with the convention that

$$S(x) = x^{2^n-2} = \begin{cases} x^{-1}, & \text{if } x \neq 0, \\ 0, & \text{if } x = 0. \end{cases}$$

A requirement for Lynx is that given a k, the constants c_i for $i = 1, 2$ are randomly selected to ensure no S-box receives a zero input.

The cryptographic properties of the nonlinear components are as follows: the algebraic degree over $\mathbb{F}_2$, differential uniformity, and correlation are $n-1$ (resp. 2), 2^{2-n} (resp. 2^{-n}), and $2^{1-n/2}$ (resp. 2^{-n}) for the inverse function $S(x) = x^{-1}$ (resp. the finite-field multiplication $Mul(x, y) = x \cdot y$), respectively.

Linear Components. An $n \times n$ binary matrix multiplication L can be represented as a linearized polynomial over $\mathbb{F}_{2^n}$:

$$L(x) = \textstyle\sum_{j=0}^{n-1} a_j x^{2^j}, \quad a_j \in \mathbb{F}_{2^n}.$$

The invertible matrices L_i for $i = 0, 1, 2$ and constants c_i for $i = 1, 2$ are generated by an XOF seeded with iv. L_3 is a fixed invertible matrix with $a_j \neq 0$ for $j = 0, \ldots, n-1$[10].Given a key k, we re-sample the iv whenever $k + c_1 = 0$, $L_0(k) + c_2 = 0$, or $k + c_1 = L_0(k) + c_2$ is detected. This entails no key loss: for a fixed key K, the IV space shrinks by at most $n - \log_2(2^n - 3)$ bits, which is negligible.

Concrete Instances. The finite fields are defined as $\mathbb{F}_{2^n} = \mathbb{F}_2/\phi(x)$, where $\phi(x)$ denotes the irreducible polynomial:

- $\mathbb{F}_{2^{128}} : X^{128} + X^7 + X^2 + X + 1$
- $\mathbb{F}_{2^{160}} : X^{160} + X^5 + X^3 + X^2 + 1$
- $\mathbb{F}_{2^{192}} : X^{192} + X^7 + X^2 + X + 1$
- $\mathbb{F}_{2^{256}} : X^{256} + X^{10} + X^5 + X^2 + 1$
- $\mathbb{F}_{2^{384}} : X^{384} + X^{16} + X^{15} + X^6 + 1$
- $\mathbb{F}_{2^{512}} : X^{512} + X^8 + X^5 + X^2 + 1$

3.3 Cryptanalysis

In VOLEitH-based signatures, each randomly sampled secret k is uniquely bound to a randomly sampled initial vector iv, and together they produce exactly one output t. Given an (iv, t) pair, the adversary aims to find any k satisfying $\text{Lynx}(\text{iv}, k) = t$ for preimage-resistance; and aims to distinguish t from a uniformly random string for indistinguishability. This naturally extends to the multi-user setting (given multiple independent $\{(\text{iv}_i, t_i)\}_i$ pairs).

3.3.1 Algebraic Attacks.

Given the attack model outlined above, our analysis focuses on algebraic attacks. First, we derive the equation systems from Lynx. Then the key recovery attack on Lynx reduces to solving those systems.

[10] The specific L_3 is given in the implementation code. For a randomly sampled L, the invertibility is guaranteed by standard generation methods—e.g., rejection sampling or LU decomposition—and the density and maximum degree hold with overwhelming probability.

Table 1: Complexity against Gröbner Basis Attacks

Scheme	m_e	n_e	Gröbner Basis ($\omega = 2$)	
			D_{reg}	complex.
Lynx-128	$13n$	$3n$	15	177.9
Lynx-160	$13n$	$3n$	18	216.6
Lynx-192	$13n$	$3n$	21	255.3
Lynx-256	$13n$	$3n$	27	332.7
Lynx-384	$13n$	$3n$	37	467.7
Lynx-512	$13n$	$3n$	48	612.4

† All complexities are reported in $\log_2$ unless otherwise noted.

For $y := S(x) = x^{-1}$, the constraint polynomial is $x \cdot y = 1$. For $z := Mul(x, y) = x \cdot y$, the constraint polynomial is $x \cdot y = z$. Thus, the constraint polynomial system over $\mathbb{F}_{2^n}$ for Lynx in variables k, v_1 and v_2 are

$$\begin{cases} f_1 : v_1 \cdot (k + c_1) + 1 = 0, \\ f_2 : v_2 \cdot (L_0(k) + c_2) + 1 = 0, \\ f_3 : L_1(v_1) \cdot L_2(v_2) + L_3(k + v_1 + v_2) + t = 0. \end{cases} \tag{4}$$

A polynomial in $\mathbb{F}_{2^n}$ corresponds to a set of multivariate Boolean equations once a basis is fixed.

- For $x \cdot y = 1$ over $\mathbb{F}_{2^n}$, it yields $5n$ linearly independent quadratic equations over $\mathbb{F}_2$ [18], derived from

$$x \cdot y{=}1,\ x^2 \cdot y{=}x,\ x \cdot y^2{=}y,\ x^4 \cdot y{=}x^3,\ x \cdot y^4{=}y^3 \tag{5}$$

- For $x \cdot y = z$ over $\mathbb{F}_{2^n}$, it yields $3n$ linearly independent quadratic equations over $\mathbb{F}_2$, derived from

$$x \cdot y = z,\ x^2 \cdot y = x \cdot z,\ x \cdot y^2 = y \cdot z. \tag{6}$$

Moreover, the settings of L_0 and L_3 prevent variable k elimination via equivalent representation, ensuring no additional independent quadratic equations in v_1 and v_2 can be derived. Therefore, we derive the system of equations $\{\eta_i\}_i$ over $\mathbb{F}_2$ for Lynx. The numbers of equations m_e and variables n_e are given in Table 1.

Gröbner basis Attack. The Gröbner basis attack on the system of m_e equations $\{\eta_i\}_i$ in n_e variables defined over $\mathbb{F}_2$ proceeds as [58]: compute a grevlex Gröbner basis, then convert it to lex order; extract and solve a univariate polynomial from the basis; substitute the root back and repeat until all variables are recovered. The complexity is governed by the degree of regularity $D_{\texttt{reg}}$ as $\mathcal{O}\left(\binom{n_e + D_{\text{reg}}}{D_{\text{reg}}}^{\omega}\right)$ [50], where ω is the linear algebra constant for Gaussian elimination, and especially $\omega = 2$ in practice for sparse Macaulay matrices (e.g., in F4/F5) [14,3]. For semi-regular systems ($m_e > n_e$), $D_{\texttt{reg}}$ is the smallest d such that the coefficient of z^{d} in the Hilbert series [30]

$$\text{HS}(z) = (1 - z)^{-n_e} \prod_{i=1}^{m_e + n_e} (1 - z^{\text{d}_i})$$

is non-positive. Here, $\text{d}_i = \deg(\eta_i)$ for $i = 1, 2, \ldots, m_e$, and $\text{d}_{m_e+i} = 2$ for the field equations $x_i^2 = x_i$ ($i = 1, \ldots, n_e$), which are included to restrict solutions to $\mathbb{F}_2$.

In line with standard Gröbner basis attacks, we estimate the complexity of Lynx. The equation system is semi-regular, which is verified on concrete reduced Lynx instances. Corresponding $D_{\texttt{reg}}$ values and attack complexities appear in Table 1, which confirms that Gröbner basis attacks do not threaten Lynx.

Remark 1. The XL algorithm [19] extends linearization by multiplying each original equation with all monomials of degree at most $D - 2$, for some $D > 2$, yielding an expanded linear system with $E_D = m_e \sum_{i=0}^{D-2} \binom{n_e}{i}$ equations of degree at most D, involving at most $T_D = \sum_{i=0}^{D} \binom{n_e}{i}$ monomials. Solving becomes feasible via Gaussian elimination once $E_D \geq T_D$, under the heuristic assumption that the expanded equations are linearly independent.

Regarding the XL algorithm over $\mathbb{F}_2$ for Lynx, the linearization step of XL necessarily generates trivial syzygies of the forms $f_i \cdot f_j - f_j \cdot f_i = 0$ and $f_i \cdot f_i - f_i = 0$. Once these trivial syzygies are discarded, the working degree D at which XL succeeds coincides with the degree of regularity $d_{\texttt{reg}}$ of the underlying system—the same quantity that governs the complexity of a Gröbner basis computation. Hence XL's cost is already captured by the Gröbner basis analysis above. We therefore do not evaluate the XL algorithm separately.

Table 2: Complexity against Dinur's Method

Scheme	Case 1			Case 2				
	n	d	complex.	n	$\deg_2 S_1$	$\deg_2 S_2$	d	complex.
Lynx-128	$3n$	2	330.1	n	127	127	≥ 127	141.3
Lynx-160	$3n$	2	408.9	n	159	159	≥ 159	174.3
Lynx-192	$3n$	2	487.7	n	191	191	≥ 191	206.8
Lynx-256	$3n$	2	644.9	n	255	255	≥ 255	271.6
Lynx-384	$3n$	2	959.0	n	383	383	≥ 383	400.8
Lynx-512	$3n$	2	1272.7	n	511	511	≥ 511	529.6

Dinur's Method. Dinur proposed a generic polynomial-method-based algorithm for solving multivariate equation systems over $\mathbb{F}_2$ [22]. It runs in $n^2 \cdot 2^{(1-1/(2.7d))n}$ time complexity and $n^2 \cdot 2^{(1-1/(1.35d))n}$ memory complexity, where n is the number of variables and d is the degree of the system, assuming that the number of equations $m \geq n$ and $d \ll n$ for practical applicability.

- We can directly evaluate Lynx's resistance to Dinur's method by solving the quadratic equation system $\{\eta_i\}_i$ in n_e variables;
- Alternatively, bypassing intermediate variables v_1 and v_2, we derive a high-degree algebraic system in k alone as

$$t = L_3(k + (k + c_1)^{-1} + (L_0(k) + c_2)^{-1}) + L_1((k + c_1)^{-1}) \cdot L_2((L_0(k) + c_2)^{-1}). \tag{7}$$

As d increases and approaches n, Dinur's method rapidly loses effectiveness and collapses to exhaustive search.

Corresponding n, d values and attack complexities appear in Table 2. Lynx is secure against these attacks.

Linearization and G&D Attacks. We evaluate Lynx against the guess-and-determine (G&D) attack based on S-box linearization [66], which first broke two-round RAIN and AIM. Each S-box S_i satisfies $x^{d_i} = (x^u)^b \cdot x^{2^a}$. Guessing $x^u = \alpha$ reduces S_i to a linear map $\alpha^b \cdot Mx$ over $\mathbb{F}_2$. The guessing complexity equals the order of x^u, i.e., $\frac{2^n-1}{\gcd(2^n-1,u)}$. All d_i chosen in Lynx ensure that linearizing any S_i requires guessing complexity exceeding $2^{n/2}$. This attack is infeasible against constructions employing three nonlinear components with different inputs—such as Lynx.

- To derive solvable quadratic equation system in k over $\mathbb{F}_2$, an attacker must linearize both S_1 and S_2 simultaneously. Since their inputs are different, the total guessing complexity exceeds 2^n.
- Linearizing only S_1 or S_2 and deriving linear equations from $x^u = \alpha$ enables the attacker to express k as an affine function of r free variables. Substituting into Eq.(7) yields a system of high-degree equations in those r variables over $\mathbb{F}_2$, and solving it requires exhaustive search over all 2^r candidates. For each α, the equation $x^u = \alpha$ has exactly $\gcd(2^n - 1, u)$ solutions in $\mathbb{F}_{2^n}$. Since the linear system captures only partial information about $x^u = \alpha$, its solution space strictly contains those $\gcd(2^n - 1, u)$ values. Hence $2^r > \gcd(2^n - 1, u)$, and the total cost exceeds 2^n.

Resultant Attack. We now solve polynomial systems over $\mathbb{F}_{2^n}$. The resultant enables solving such systems by detecting common roots [62]. Let $f(\boldsymbol{x}, y) = \sum_{i=0}^{m} f_i'(\boldsymbol{x})y^i, g(\boldsymbol{x}, y) = \sum_{i=0}^{n} g_i'(\boldsymbol{x})y^i \in \mathbb{F}_q[\boldsymbol{x}, y]$, where $\boldsymbol{x} = (x_1, \ldots, x_r)$. Their resultant $R(f, g, y)$ is the determinant of Sylvester matrix of f and g as univariate polynomials in y. It vanishes iff f and g share a non-constant common factor. Crucially, $R(f, g, y) = h(\boldsymbol{x}) \in \mathbb{F}_q[\boldsymbol{x}]$ eliminates y. Iteratively applying resultants to the given polynomials in r variables yields a univariate polynomial; its roots, substituted back stepwise, give all common solutions. Computing $R(f, g, y)$ costs $\mathcal{O}(D \cdot (\log D)^2 \cdot (m + n)^\omega)$ field operations, where $D = \max\{\deg f_i'(\boldsymbol{x}), \deg g_i'(\boldsymbol{x})\}$ and ω is the matrix-multiplication exponent. Solving the univariate equation $G(x) = 0$ costs $\mathcal{O}(d \log d(\log d + \log q \log\log d))$, where $d = \deg(G)$ and $q = 2^n$.

For Lynx, the resultant elimination path is: $g_1 := R(f_1, f_3, v_1),\ g_2 := R(f_2, g_1, v_2)$. This yields a univariate polynomial $g_2(k)$; solving it recovers the key. Let $\deg_x g$ be the maximal degree of x in g. Computing g_1 costs $D_1 \cdot (\log D_1)^2(\deg_{v_1} f_1 + \deg_{v_1} f_3)^\omega$; computing g_2 costs $D_2 \cdot (\log D_2)^2(\deg_{v_2} f_2 + \deg_{v_2} g_1)^\omega$. Here, D_1 is the product of the maximal k- and v_2-degrees in f_1 and f_3; D_2 is the maximal k-degree in f_2 and g_1. Converting field operations into bit operations requires multiplying by an additional factor of $M(n) = n \log n \log\log n$. Since the degree reaches maximum in each of these resultant elimination steps, the total cost is shown in Table 3, rendering resultant elimination infeasible against Lynx.

Linearized Polynomial Decomposition. For a linearized polynomial $L(x)$, it is highly likely to find two linearized polynomials $L'(x)$ and $L''(x)$ of degree $2^{n/2}$ such that $L(x) = L''^{-1} \circ L'(x)$ [60,61]. This enables meet-in-the-middle modeling, yielding equations $L''(y) = L'(x)$ of degree $2^{n/2}$ given the output y of L, achieving substantial degree reduction. The method applies to schemes with a single (or effectively reduced) linear layer—e.g., AIM, 2-round RAIN, or 3-round RAIN under S-box linearization.

However, Lynx interleaves four linearized polynomials with nonlinear operations, blocking composite decomposition. Let w_1 and w_2 be the outputs of L_1 and L_2, and suppose $L_i(x) = L_i''^{-1} \circ L_i'(x)$ for $i = 1, 2, 3$. This yields the system

$$\begin{cases} L_1'(\frac{1}{k+c_1}) = L_1''(w_1), \\ L_2'(\frac{1}{L_0(k)+c_2}) = L_2''(w_2), \\ L_3'(\frac{1}{k+c_1} + \frac{1}{L_0(k)+c_2} + k) = L_3''(w_1 \cdot w_2 + t), \end{cases}$$

which contains the product of $w_1 \cdot w_2$. Thus, efficiently eliminating intermediate variables to derive a low-degree univariate equation is infeasible—even eliminating any intermediate variable via resultants. The complexity remain the same as directly using resultant method (shown in Table 3), which exceeds 2^n.

Magic Pot. Magic Pot is a new algebraic-elimination framework [15], *broken full AIM2*. The attack partitions the variables into a main variable y and r auxiliary variables $\boldsymbol{x} = (x_1, \ldots, x_r)$ that are partially linearized. The cipher's equations, together with their q-Frobenius multiples (up to degree D in the auxiliary variables), are stacked into a polynomial matrix $M \in \mathbb{F}[y]^{H \times W}$ with H rows and $W = \binom{\ell n}{\leq D+1}$ columns ($H \geq W$). To verify that the equation matrix has full rank over the polynomial ring $mathbbF[y]$, we evaluate it at a fixed constant. The attack then computes a left kernel vector $z \in \mathbb{F}[p]^H$ satisfying $z \cdot M = 0$ by approximate-basis iteration and HalfGCD-based degree minimization, which eliminates all auxiliary variables and yields a single univariate polynomial $g(y)$ whose roots are exactly the candidate values of the master key. The main cost is the kernel computation with a complexity of roughly

$$\xi \cdot (4W^2 + 4s) + M(\xi)(2n + 2W + 22 \log_2 \xi)$$

field operations, where ξ is the sum of the p-degrees of all rows, s is the total number of Frobenius equations used, and $M(\xi)$ is the cost of multiplying two polynomials of degree ξ (approximately $O(\xi \log \xi \log \log \xi)$ with FFT-based multiplication). The attack targets primitives with some variable missing high-degree terms.

We analyze Lynx using the Magic Pot method. Eq.(4) show that Lynx includes all pairwise cross-terms among k, v_1, and v_2, as well as high-degree monomials in each variable, guaranteed by the configuration of linearized polynomials. Consequently, no variable can be effectively isolated to form a sparse polynomial ring representation. For example, we set k as the main variable, and $D = 1$ is forced. Then we derive an $H \times W$ matrix in $\mathbb{F}[k]$ with $W = 2n^2 + n + 1$ and $H = 3(M + 1)(2n + 1)$, where M is the highest Frobenius exponent. Full-rank elimination requires $H \geq W$ and $M + 1 = n/2$. We verified full rank by evaluating the matrix at $p = 0$. L_3 full-support forces $\log_2 \xi \approx 1.5n$. Then, the total complexity is shown in Table 3.

n	Resultant ($\omega = 2$)	Linearized Polynomial Decomposition ($\omega = 2$)	Magic Pot
128	448	448	233.6
160	560	560	283.2
192	672	672	332.5
256	911	911	430.6
384	1344	1344	625.5
512	1792	1792	819.6

Table 3: Complexities of attacks over $\mathbb{F}_{2^n}$

Other Algebraic Attacks. (1) The Gröbner basis attack over $\mathbb{F}_{2^n}$ is infeasible against both the single-variable equation Eq.(7) and the multivariate system Eq.(4), as Lynx's polynomial systems almost surely achieve maximum degree. **(2)** [48] converts the equation into a single-variable polynomial via equivalent representation, then into a low-degree Boolean equation system for solving. However, in Lynx, multiple $L(\frac{1}{k+c})$-like terms in Eq.(7) block further expansion, thus making the attack impossible. **(3)** [46] observes that RAIN's special round structure—lacking linear layers at both input and output—introduces extra algebraic relations, enabling construction of more overdefined quadratic systems and thus drastically lowering the complexity of Gröbner basis attacks, thereby pointing that *3-round RAIN is risky* for 256-bit security. Lynx does not have such structure.

3.3.2 Statistical Attacks

For applications of VOLEitH-based post-quantum signatures, adversaries only obtain a single plaintext-ciphertext pair (i.e., the public key). In this case, known statistical attacks, including differential, linear and integral attacks, are inherently infeasible. For completeness, we briefly note them below.

Differential and Linear Attacks. For $f : \mathbb{F}_{2^n}^m \to \mathbb{F}_{2^n}$, the differential uniformity and correlation are defined as:

$$\mathrm{diff}_f = 2^{-nm} \cdot \max_{\Delta_x \neq 0, \Delta_f} \#\{x \in \mathbb{F}_{2^n}^m : f(x) \oplus f(x \oplus \Delta_x) = \Delta_f\},$$
$$\mathrm{cor}_f = 2^{-nm} \cdot \max_{\Gamma_f \neq 0, \Gamma_x} \left|\sum_{x \in \mathbb{F}_{2^n}^m} (-1)^{\mathrm{Tr}(\Gamma_f \cdot f(x) + \Gamma_x \cdot x)}\right|,$$

where $\mathrm{Tr} : \mathbb{F}_{2^n} \to \mathbb{F}_2$ is the trace function, i.e., $\mathrm{Tr}(x) = x + x^2 + x^{2^2} + \cdots + x^{2^{n-1}}$. In Lynx, activating any nonlinear component implies activating all nonlinear components. Hence, the lower bound of the differential (resp. linear) trail equals the product of their individual diff (resp. cor^2) values. These bounds are strictly less than 2^{-n}.

Interpolation and Integral Attacks. With inverse functions, the algebraic degree of t as a polynomial in k is on the order of 2^n. This immediately rules out the interpolation attack: recovering the univariate polynomial representation of $t(k)$ would require on the order of 2^n (plaintext, ciphertext) samples.

The same high degree defeats integral cryptanalysis. Viewing the primitive over $\mathbb{F}_2$ bitwise, every output bit of the inverse function has algebraic degree exactly $n-1$ in its ANF, while the multiplicative gate raises it to n. A vanishing integral requires summing over an affine subspace of dimension exceeding this degree, which calls for 2^n inputs.

3.3.3 One-Time Pseudorandom Function

The output pseudorandomness of Lynx is only required for the security proof in the QROM [7], and is unnecessary for the proof in the classical ROM.

Definition 2 (Pseudorandomness of Lynx). *Let $\mathbb{F}_{2^n} \times \mathcal{V}$ be the set of* valid pairs (k, iv) *for which both S-box inputs are nonzero and unequal*[11]*. We say that* $\mathsf{Lynx} : \mathbb{F}_{2^n} \times \mathcal{V} \to \mathbb{F}_{2^n}$ *is* pseudorandom *if for any probabilistic polynomial-time distinguisher $\mathcal{D}$ restricted to making at most one oracle query, the advantage*

$$\mathrm{Adv}^{\mathrm{pr}}_{\mathcal{D}}(n) = |\Pr[\mathcal{D}(iv, \mathsf{Lynx}(iv, k)) = 1] - \Pr[\mathcal{D}(iv, u \leftarrow_{\$} \mathbb{F}_{2^n}) = 1]|$$

is negligible in n given iv.

As only a single plaintext-ciphertext pair is available per user for applications of VOLEitH-based post-quantum signatures, known statistical attacks, including differential, linear and integral attacks, are inherently infeasible as analyzed in Section 3.3.2. The remaining concern is a distinguishing attack on the output distribution, which we address via the algebraic analysis in Section 3.3.1 and the randomness tests below.

Randomness Testing. We subject $\geq 10^{10}$-bit concatenated samples of $\{t_i\}_i$ to the NIST SP 800-22 test suite [6], including the frequency, block-frequency, runs, overlapping template, and random walk tests, among others. All variants of Lynx pass all required tests at all targeted security levels.

3.3.4 Quantum Attacks

Grover's algorithm [32] provides a quadratic speedup over classical brute-force key search. In this section, we analyze Lynx's resistance to Grover's preimage attack and examine whether any specialized quantum algorithm for Lynx could outperform it.

Grover's Preimage Attack. We adopt NIST's quantum resource metric (Grover's cost) = (circuit size) × (circuit depth) in Clifford/T gates [52]. For a one-way function $f : \{0,1\}^n \to \{0,1\}^m$, Grover's preimage search requires [39]

$$(\text{Grover's circuit size/depth}) = (\text{size/depth of } f) \times 2 \times \left\lfloor \frac{\pi}{4}\sqrt{2^n} \right\rfloor.$$

When applying Grover's algorithm to Lynx given t, it is more efficient to compute v_1 and v_2 via Eq.(4) per k candidate than to directly check $\mathsf{Lynx}(\mathsf{iv}, k) = t$. Lynx's quantum implementation breaks down into three basic operations: finite-field multiplication (FFM), finite-field squaring (FFS), and random matrix multiplication (MM), enabling direct cost citations for each.

Table 4 reports the count and depth of basic operations per Lynx's component. Table 4 also gives the costs of Grover's preimage attack for Lynx, where Lynx-128, Lynx-192 and Lynx-256 meet NIST's PQC security levels I, III, and V—defined as the quantum-resistant equivalents of AES-128, -192, -256, respectively. Grover's cost may be further reduced—for example, via improved circuit representation or more efficient S-box addition chains. However, such optimizations are unlikely to bring the cost below that of AES at the corresponding security level, as finite-field multiplications alone already exceed AES's total cost.

[11] Restricting the sampling to $\mathcal{V}$ discards at most 3 ivs, which is negligible.

Table 4: Total cost of Grover's algorithm on Lynx

n	#FFM $d_1, d_2, \times$	FFM depth	#FFS d_1, d_2	FFS depth	#MM	Grover's cost	Security level
128	$12, 12, 1$	13	$127, 127$	127	4	163.8	I($\geq$157)
160	$12, 12, 1$	13	$159, 159$	159	4	197.3	N/A
192	$13, 13, 1$	14	$191, 191$	191	4	230.5	III($\geq$221)
256	$14, 14, 1$	15	$255, 255$	255	4	295.8	V($\geq$285)
384	$15, 15, 1$	16	$383, 383$	383	4	426.2	N/A
512	$16, 16, 1$	17	$511, 511$	511	4	555.6	N/A

† d_1, d_2 and $\times$ correspond to the number and depth of basic operations needed to implement the S-box and the final multiplication.

Other Quantum Attacks. (1) GroverXL [13] and QuantumBooleanSolve [26] are hybrid quantum-classical attacks on MQ systems over $\mathbb{F}_2$. Both exploit Grover's quadratic speedup to reduce brute-force guessing cost, then solve the residual low-dimensional system classically via FXL or BooleanSolve, and optimize the fraction of guessed variables to balance quantum and classical effort. Their asymptotic complexities are nearly identical; only the classical solver differs. For Lynx, the complexities exceed Grover's generic bound of $2^{0.5n}$ across all security levels, e.g., $\mathcal{O}(2^{1.008n})$ at a 25% guess rate for $n = 128$. This stems from introducing $2n$ intermediate variables during algebraic solving, which nullifies Grover's quadratic advantage. (2) A quantum algorithm based on the HHL algorithm [17] solves sparse linear systems with exponential speedup under restrictive conditions when the input's (i.e., k's) Hamming weight is below $\mathcal{O}(\log n)$; otherwise, it is slower than Grover's algorithm. (3) Quantum generic attacks exploit no special property of the underlying primitives and treat them as public random permutations or functions. Most rely on Simon's period-finding algorithm (Q2) or Grover/offline Simon variants (Q1). Since Simon's algorithm requires multiple queries, these attacks do not apply to Lynx.

In summery, the complexities of attacks over $\mathbb{F}$ are given in Table 1, Table 2 and Table 4 for Gröbner basis attack, Dinur's method and Grover's Preimage Attack. Large-field attacks are inherently resisted by the structural design of Lynx, with all complexities far exceeding security bounds.

4 LYNXER – VOLEitH-Based Signature

In this section, we construct Lynxer, a signature scheme based on Lynx, using VOLEitH framework to prove its input-output relation. We then discuss its parameter choices and EUF-CMA security.

4.1 Lynxer

Lynxer follows the VOLEitH workflow of FAESTv2 [9]; its scheme level is shown in Figure 3 and pseudocode in Appendix B.

The Lynxer algorithm

- $\mathsf{KeyGen}(1^n)$. For $n \in \{128, 160, 192, 256, 384, 512\}$, sample a secret value $k \in \mathbb{F}_{2^n}$ and a public seed iv. Using iv, derive the public matrices L_0, L_1, L_2 and constants c_1, c_2 used by Lynx, compute $t := \mathsf{Lynx}_{\mathsf{iv}}(k)$, and re-sample ($\mathsf{iv}$) if any S-box input is zero or the S-box inputs are equal. Output $\mathsf{sk} := (\mathsf{iv}, k)$, $\mathsf{pk} := (\mathsf{iv}, t)$.
- $\mathsf{Sign}(\mathsf{sk}, m)$. Parse $\mathsf{sk} = (\mathsf{iv}, k)$, recompute the public values derived from iv, and define the witness $w := (k, v_1, v_2)$, $v_1 := S_1(k + c_1)$, $v_2 := S_2(L_0(k) + c_2)$. Use QuickSilver within VOLEitH to prove the degree-2 constraints. The signature σ is the Fiat-Shamir transcript of this proof process, with the message m included in the random oracle inputs.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$. Parse $\mathsf{pk} = (\mathsf{iv}, t)$, reconstruct the same public values from iv. Run verifier's action in QuickSilver using the proof transcript in σ. Accept iff the proof verifies successfully.

Fig. 3: The overall structure of Lynxer.

QuickSilver Constraints. The QuickSilver proof in Sign proves three relations over the witness (k, v_1, v_2) defined in Eq.(4).

Table 5: VOLEitH proof system parameters for Lynxer.

Scheme	Parameters				Sizes (B)			
	$\hat{\ell}$	τ	w_{grind}	T_{open}	sk	pk	sig	sig+pk
Lynxer-128s	656	11	7	103	32	32	2938	2970
Lynxer-160s	816	14	6	129	40	40	4608	4648
Lynxer-192s	976	16	8	162	48	48	6652	6700
Lynxer-256s	1296	22	6	218	64	64	12000	12064
Lynxer-384s	1936	32	6	336	96	96	27012	27108
Lynxer-512s	2576	42	8	456	128	128	48168	48296
Lynxer-128f	656	16	8	112	32	32	3652	3684
Lynxer-160f	816	21	8	139	40	40	5802	5842
Lynxer-192f	976	24	8	176	48	48	8348	8396
Lynxer-256f	1296	32	8	234	64	64	14772	14836
Lynxer-384f	1936	48	6	332	96	96	32228	32324
Lynxer-512f	2576	64	6	445	128	128	57364	57492

† For all Lynxer-n scheme, $d = 2$ and $\ell = 3n$.

4.2 Parameters

The signature size and running time of VOLEitH-based signatures are determined by five core parameters:

- The BAVC opening size threshold T_{open};
- The grinding parameter w_{grind};
- The number of small GGM trees τ;
- The witness length ℓ;
- The degree of relations proved in QuickSilver d.

For Lynx-n, $\ell = 3n$ and $d = 2$; thus, only the first three parameters require tuning. Following the conventions in [9,42,2], we select a "fast" variant for low latency and a "small" variant for minimal communication cost for each security level, fixing those three parameters to yield expected rejection counts of about 300 and 2000, respectively. The parameters of Lynxer are listed in Table 5.

4.3 EUF-CMA Security

We state the EUF-CMA security of Lynxer in Theorem 1, following the FAESTv2 template of [9, Theorems 9.24 and 9.36]. The $(1 - \frac{\mathrm{dom}(\mathsf{OWF})}{\mathrm{cod}(\mathsf{OWF})})^{-1}$ term[12] is an artifact of the proof technique in [9] and can be bounded by 4 by letting the first two bits of k to have at least one 0, which may be eliminated via a new proof strategy.

Theorem 1 ([9]). *Let H_0, H_1, $H_{2,0}$, $H_{2,1}$, $H_{2,2}$, $H_{2,3}$, H_3, and H_4 be modeled as random oracles and* AES *an ideal cipher. For simplicity, let $\mathcal{A}$ be an adversary that runs* Verify *of* Lynxer *on its forgery and returns $\perp$ if verification fails. Let Q_0, Q_1, $Q_{2,i}$, Q_3, Q_4, and Q_{AES} denote upper bounds on the number of queries made by $\mathcal{A}$ to H_0, H_1, $H_{2,i}$, H_3, H_4, and* AES*, respectively, and let $Q_2 = Q_{2,1} + Q_{2,2} + Q_{2,3}$. Then there exist adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

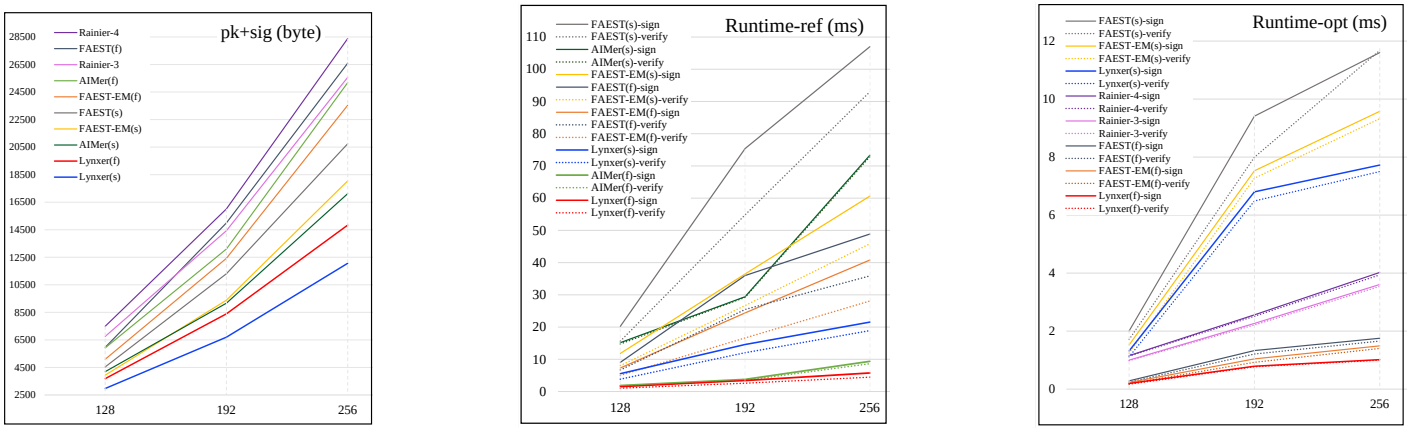
$$\mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{A}} \leq \left(1 - \frac{\mathrm{dom}(\mathsf{OWF})}{\mathrm{cod}(\mathsf{OWF})}\right)^{-1} \left(\mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{B}_2} + \mathsf{Adv}^{\mathsf{PRG\text{-}OWF}}_{\mathcal{B}_1}\right),$$

where

$$\mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{B}_2} \leq \frac{(Q_{2,0} + 2Q_{2,1} + Q_{2,3} + 4)^2}{2^{2n}} + \frac{(Q_1 + 1)^2}{2^{2n}} + \frac{(Q_{\mathsf{AES}} + 1)^2}{2^{2n}} + (2Q_{2,1} + Q_{2,3} + 9)\frac{3}{2^n}.$$

Moreover, let H_0, H_1, $H_{2,1}$, $H_{2,2}$, $H_{2,3}$, and H_4 be modeled as random oracles and let H_3 be a PRF. Let L be the number of leaves in the BAVC tree. Then, for any PPT adversary $\mathcal{A}$ making Q_{sig} signing queries and Q_0, Q_1, $Q_{2,1}$, $Q_{2,2}$, and

[12] dom(OWF) (resp. cod(OWF)) denotes the domain (resp. co-domain) size of Lynx-n.



(a) communication cost (b) Reference runtime (c) Optimized runtime

Fig. 4: Performance comparison for symmetric-based post-quantum signature schemes in MPCitH/VOLEitH framework. Figure (a) shows that Lynxer-fast and Lynxer-small have the smallest pk+sig size. Lynxer-fast is the fastest overall, and its advantage grows with the security level.

$Q_{2,3}$ queries to H_0, H_1, $H_{2,1}$, $H_{2,2}$, and $H_{2,3}$, respectively, there exists an EUF-KO adversary $\mathcal{B}$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}[Q_{\mathsf{sig}}] &\le 2\mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{B}} \\ &+ 2Q_{\mathsf{sig}}\Big(\mathsf{Adv}^{\mathsf{PRF}}_{H_3} + (\lceil \log L \rceil + 1)\mathsf{Adv}^{\mathsf{PRP}}_{\mathsf{PRG}_{2n}}[1]\Big) \\ &+ 2Q_{\mathsf{sig}}\Big(\frac{2Q_1 + Q_{2,1}}{2^{2n}} + \frac{Q_{\mathsf{sig}} + Q_{2,2}}{2^{5n+64}} + \frac{Q_{\mathsf{sig}} + Q_{2,3}}{2^{3n+64}}\Big).\end{aligned}$$

5 Performance Evaluation

5.1 Implementation

We provide both reference and optimized implementations of Lynxer, built upon the FAESTv2 source code [25,24]. The reference implementation is written in C with standard Karatsuba algorithm to accelerate Galois field multiplication, as in AIMer's reference implementation [57]; the optimized implementation is written in C++ with AVX2 instructions. For $n \in \{128, 160, 192, 256\}$, we follow FAESTv2 using an AES-based PRG for leaf commitments and SHAKE as the XOF. For $n \in \{384, 512\}$, we use SHAKE with $2n$-bit capacity as both PRG and XOF, which accounts for the observed performance drop at these security levels; should standardized, high-performance alternatives become available, substituting them would directly improve runtime efficiency. Our source code is available at

"https://anonymous.4open.science/r/Lynxer-Impl-B6E1"

We tested the performance of Lynxer and other schemes on a Debian 12 system equipped with an AMD Ryzen 7 7800X3D 8-core processor (@4.2GHz) and 128GB of RAM, using GCC/G++ 12.2.0 with the compilation flags -O3, -march=native, -std=c++20 (exclusively for G++).

5.2 Performance Comparison

All compared schemes are benchmarked under the same environment using publicly available implementations.

5.2.1 Performance comparison with state-of-the-art post-quantum signature schemes from symmetric primitives in the MPCitH/VOLEitH framework

For FAESTv2, instantiated as FAEST and FAEST-EM, we test the reference implementation [25] and the optimized version [24]. Only the reference implementation [57] for AIMer2.0[13] and the optimized implementation [35] for Rainier are available. Results are summarized in Table 6. We compare the "pk + sig" size, signing/verification runtime in reference and

[13] The GitHub repository (https://github.com/samsungsdsresearch-papers/AIMer) on AIMer's official website (https://aimer-signature.org) cited in [42] is no longer valid.

Table 6: Performance of state-of-the-art symmetric-based post-quantum signature schemes in MPCitH/VOLEitH framework.

Scheme	Sizes (Byte)				Runtime-ref (ms)		Runtime-opt (ms)		Primitive
	sk	pk	sig (ratio)	pk+sig (ratio)	Sign (ratio)	Verify (ratio)	Sign (ratio)	Verify (ratio)	
AIMer-128s	48	32	4160 (29.4%)	4192 (29.2%)	15.165 (63.5%)	14.750 (74.0%)	N/A	N/A	AIM
FAEST-128s	32	32	4506 (34.8%)	4538 (34.6%)	20.233 (72.6%)	15.630 (75.5%)	2.005 (34.3%)	1.712 (35.0%)	AES
FAEST-EM-128s	32	32	3906 (24.8%)	3938 (24.6%)	11.812 (53.1%)	8.039 (52.3%)	1.524 (13.6%)	1.240 (10.2%)	AES-EM
Lynxer-128s	32	32	**2938**	**2970**	5.539	3.837	1.317	1.113	**Lynx**
Rainier-3-128	16	32	6720 (45.7%)	6752 (45.4%)	N/A	N/A	1.000 (80.5%)	0.982 (82.7%)	Rain-3
Rainier-4-128	16	32	7456 (51.0%)	7488 (50.8%)	N/A	N/A	1.155 (83.1%)	1.137 (85.1%)	Rain-4
AIMer-128f	48	32	5888 (38.0%)	5920 (37.8%)	1.864 (21.1%)	1.713 (39.6%)	N/A	N/A	AIM
FAEST-128f	32	32	5924 (38.4%)	5956 (38.1%)	9.055 (83.8%)	6.686 (84.5%)	0.286 (32.0%)	0.242 (30.1%)	AES
FAEST-EM-128f	32	32	5060 (27.8%)	5092 (27.7%)	7.295 (79.8%)	5.016 (79.4%)	0.227 (14.2%)	0.188 (10.1%)	AES-EM
Lynxer-128f	32	32	**3652**	**3684**	1.470	1.035	0.195	0.169	**Lynx**
Lynxer-160s	40	40	**4608**	**4648**	10.780	7.651	5.056	4.358	**Lynx**
Lynxer-160f	40	40	**5802**	**5842**	2.533	1.827	0.653	0.603	**Lynx**
AIMer-192s	72	48	9120 (27.1%)	9168 (26.9%)	29.413 (50.4%)	29.225 (58.8%)	N/A	N/A	AIM
FAEST-192s	40	48	11260 (40.9%)	11308 (40.7%)	75.395 (80.6%)	54.820 (78.0%)	9.411 (27.7%)	7.996 (18.9%)	AES
FAEST-EM-192s	48	48	9340 (28.8%)	9388 (28.6%)	36.479 (60.0%)	26.673 (54.8%)	7.528 (9.7%)	7.272 (10.8%)	AES-EM
Lynxer-192s	48	48	**6652**	**6700**	14.589	12.043	6.802	6.487	**Lynx**
Rainier-3-192	24	48	14384 (42.0%)	14432 (41.8%)	N/A	N/A	2.265 (65.1%)	2.212 (65.2%)	Rain-3
Rainier-4-192	24	48	15968 (47.7%)	16016 (47.6%)	N/A	N/A	2.561 (69.2%)	2.513 (69.3%)	Rain-4
AIMer-192f	72	48	13056 (36.1%)	13104 (35.9%)	3.764 (9.5%)	3.458 (25.6%)	N/A	N/A	AIM
FAEST-192f	40	48	14948 (44.2%)	14996 (44.0%)	35.990 (90.5%)	25.464 (89.9%)	1.330 (40.6%)	1.210 (36.3%)	AES
FAEST-EM-192f	48	48	12380 (32.6%)	12428 (32.4%)	24.392 (86.0%)	16.628 (84.5%)	1.043 (24.3%)	0.925 (16.7%)	AES-EM
Lynxer-192f	48	48	**8348**	**8396**	3.406	2.573	0.790	0.771	**Lynx**
AIMer-256s	96	64	17056 (29.6%)	17120 (29.5%)	73.327 (70.6%)	72.886 (74.1%)	N/A	N/A	AIM
FAEST-256s	48	48	20696 (42.0%)	20744 (41.8%)	107.008 (79.9%)	93.019 (79.7%)	11.602 (33.4%)	11.678 (35.8%)	AES
FAEST-EM-256s	64	64	17984 (33.3%)	18048 (33.2%)	60.640 (64.5%)	45.849 (58.8%)	9.578 (19.3%)	9.325 (19.6%)	AES-EM
Lynxer-256s	64	64	**12000**	**12064**	21.542	18.903	7.725	7.501	**Lynx**
Rainier-3-256	32	64	25504 (42.1%)	25568 (42.0%)	N/A	N/A	3.608 (71.8%)	3.542 (71.8%)	Rain-3
Rainier-4-256	32	64	28320 (47.8%)	28384 (47.7%)	N/A	N/A	4.023 (74.7%)	3.938 (74.7%)	Rain-4
AIMer-256f	96	64	25120 (41.2%)	25184 (41.1%)	9.369 (38.5%)	8.631 (48.4%)	N/A	N/A	AIM
FAEST-256f	48	48	26548 (44.4%)	26596 (44.2%)	48.873 (88.2%)	35.880 (87.6%)	1.754 (42.0%)	1.657 (39.8%)	AES
FAEST-EM-256f	64	64	23476 (37.1%)	23540 (37.0%)	40.806 (85.9%)	28.092 (84.1%)	1.492 (31.8%)	1.403 (28.9%)	AES-EM
Lynxer-256f	64	64	**14772**	**14836**	5.759	4.454	1.017	0.998	**Lynx**
Lynxer-384s	96	96	27012	27108	272.782	267.612	56.395	56.335	Lynx
Lynxer-384f	96	96	32228	32324	35.739	32.353	6.207	6.182	Lynx
Lynxer-512s	128	128	48168	48296	881.203	868.327	142.552	146.354	Lynx
Lynxer-512f	128	128	57364	57492	99.022	90.462	14.724	14.663	Lynx

† The FAEST, FAEST-EM, and AIMer in this table refer to FAESTv2 [9] and AIMer2 [42], respectively.

Table 7: More performance comparisons of optimized implementations at the 128-bit security level.

Scheme	Sizes (Byte)				Runtime (ms)		Assumption
	sk	pk	sig	pk+sig	Sign	Verify	
ML-DSA-44 [49]	2560	1312	2420	3732	0.040	0.012	Lattice
FN-DSA-512 [55]	7553	897	666	1563	0.145	0.026	Lattice
SLH-DSA-SHA2-128s [33]	64	32	7856	7888	78.481	0.158	Hash
SLH-DSA-SHAKE-128s [33]	64	32	7856	7888	93.631	0.184	Hash
SLH-DSA-SHA2-128f [33]	64	32	17088	17120	3.742	0.364	Hash
SLH-DSA-SHAKE-128f [33]	64	32	17088	17120	4.687	0.428	Hash
Picnic-L1-full [65]	52	35	32065	32100	1.518	1.133	LowMC
Picnic3-L1 [65]	52	35	14612	14647	6.736	5.279	LowMC
Banquet [12]	16	32	13284	13316	22.008	18.985	AES
Limbo [21]	16	32	14512	14544	N/A	N/A	AES
BN++Rain3 [40]	16	32	5248	5280	0.904	0.849	Rain-3
BN++Rain4 [40]	16	32	5984	6016	1.020	0.955	Rain-4
Lynxer-128s	32	32	2938	2970	1.317	1.113	Lynx
Lynxer-128f	32	32	3652	3684	0.195	0.169	Lynx

optimized implementations in Figure 4, respectively. We observe that Lynxer significantly outperforms all these schemes, including FAEST, FAEST-EM, AIMer, and Rainier, in all settings across all common security levels. Lynxer is the only scheme supporting 384-bit and 512-bit security.

For VOLEitH-based signature schemes, FAEST employs AES as its underlying OWF and has advanced to the third round of NIST additional post-quantum signature competition. Compared to FAEST, Lynxer reduces the "pk + sig" size by 34.6%∼44.2%. In terms of signing and verification runtime, Lynxer's reference implementation achieves a reduction of 72.6%∼80.6% (small variants), and 83.8%∼90.5% (fast variants). In optimized implementations, the runtime is reduced by 18.9%∼35.8% (small variants) and 30.1%∼42.0% (fast variants)[14]. These substantial gains in size and runtime, relative to the use of tailored primitive instead of standard primitives like AES, conclusively demonstrate the importance and necessity of designing a VOLEitH-friendly symmetric primitive.

FAEST-EM is a FAEST variant that uses AES as an Even Mansour cipher, where the key schedule need not be executed as part of the zero-knowledge proof, thereby reducing signature size. Compared to FAEST-EM, Lynxer achieves a reduction of 24.6%∼37.0% in "pk + sig" size. For reference implementations, Lynxer reduces signing and verification time by 52.3%∼64.5% (small variants) and 79.4%∼86.0% (fast variants). In optimized implementations, the gains are 9.7%∼19.6% (small variants) and 10.1%∼31.8% (fast variants). These margins are narrower than those against FAEST, since FAEST-EM moves away from standard AES construction. However, Lynxer still delivers significantly larger performance improvements, underscoring the advantages of co-designing a VOLEitH-friendly underlying primitive with the scheme over even using modified symmetric primitives.

For MPCitH-based signatures from tailored OWFs, (1) Rainier-3 and Rainier-4[15]are signature schemes based on RAIN, a MPCitH-friendly OWF, in the Picnic framework. Compared to Rainier-3 and Rainier-4, Lynxer delivers marked performance gains. It reduces the "pk + sig" size by 41.8%∼50.8%. Moreover, in optimized implementations, Lynxer cuts signing and verification runtime by 65.1%∼85.1% in fast variants. (2) AIMer is a signature scheme based on AIM, an alternative MPCitH-friendly OWF, in the enhanced Picnic framework BN++, and a finalist in Korea's PQC competition. Compared to AIMer, Lynxer achieves a "pk + sig" size reduction of 26.9%∼41.1%. In terms of signing and verification runtime, the reference implementation yields speedups of 50.4%∼74.1% (small variants), and 9.5%∼48.4% (fast variants). These consistent results demonstrate that Lynxer, built on a friendly symmetric primitive in the VOLEitH framework, delivers superior efficiency and smaller communication cost than even those purpose-built MPCitH-based designs using tailored OWFs.

5.2.2 More performance comparisons

We also compare the optimized performance of Lynxer to a selection of standardized and established post-quantum signature schemes at the 128-bit security level. The compared schemes include lattice-based, hash-baseds, and other symmetric-primitive-based schemes. For the standardized schemes and submitted candidates (i.e., ML-DSA, FN-DSA and

[14] With the assistance of the AVX2 instructions, the reduction ratio of Lynxer's runtime in optimized implementations has decreased.
[15] A Gröbner basis attack on 3-round RAIN was proposed at ASIACRYPT 2024 [47], recommending four rounds instead.

SLH-DSA and Picnic), we test the published NIST implementations. For the symmetric-primitive-based schemes, we test the public implementations when available[16], e.g., Banquet's implementation [20] and BN++Rain3/4's implementation [34]. The comparison is summarized in Table 7. Overall, Lynxer achieves a highly competitive size-runtime tradeoff against this broader set of baselines, offering notably signature sizes while maintaining practical signing and verification runtime.

Compared to lattice-based schemes, Lynxer-128s reduces the "pk + sig" size by 20.4% over ML-DSA-44 [49], standardized in FIPS 204 and derived from CRYSTALS-Dilithium based on MSIS and MLWE problems over module lattices. While ML-DSA-44 and FN- DSA-512 remain fastest in runtime, they require much larger public and private keys. Furthermore, FN-DSA [55], standardizing in FIPS 206 and derived from FALCON based on SVP over NTRU lattices, requires complex operations like (discrete) Gaussian sampling or complex float/fix point arithmetic.

Compared to hash-based schemes SLH-DSA-SHA2/ SHAKE [33], standardized in FIPS 205 and derived from SPHINCS+ with security rooted in SHA-2 and SHAKE hash functions, Lynxer-128f cuts the "pk + sig" size by 53.3% relative to their small variants, while achieving a signing speedup of 403∼481 times at comparable verification runtime. When compared to their fast variants, Lynxer-128f reduces the "pk + sig" size by 78.5%, signing runtime by a factor of 19∼24 times, and verification runtime by a factor of 2.1∼2.5 times.

In addition to the comparison in Section 5.2.1, we extend our evaluation to the following symmetric-based signature schemes. (1) Picnic [65] is the first NIST PQC alternate candidate built on MPCitH and MPC-friendly block cipher LowMC. The "pk + sig" size of Picnic-L1-full is 11 times that of Lynx-128s at comparable runtime. The runtime of Picnic3-L1 is 31∼35 times that of Lynx-128f, and its "pk + sig" size is 4 times. (2) Banquet [12] and Limbo [21] built upon Picnic by replacing LowMC with AES. Compared to Banquet, Lynxer reduces the "pk + sig" size by over 70% in both variants, while accelerating signing and verification by about 17 times in small variants and over 110 times in fast variants. Compared to Limbo, Lynxer cuts the "pk + sig" size by over 70%. (3) BN++Rain3/4 [40] adopts an improved Picnic framework (BN++) that slightly reduces the "pk + sig" size and runtime compared to the earlier Rainier [23], but the overall reduction ratios compared with Lynxer are not significantly different from those of Rainier in Table 6.

Overall, across all hash-based and symmetric-key-based post-quantum signature schemes, only Lynxer achieve a "pk + sig" size below 3 KB at 128-bit security level.

[16] No public implementation of Limbo is available; therefore, we only reported sizes [21] and mark runtime entries as `N/A`

References

1. Agarwal, A., Baum, C., Braun, L., Scholl, P.: Low-bandwidth mixed arithmetic in VOLE-based ZK from low-degree PRGs. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part IV. LNCS, vol. 15604, pp. 396–426. Springer, Cham (May 2025). 10.1007/978-3-031-91134-7_14
2. Aguilar Melchor, C., Bettaieb, S., Bidoux, L., Feneuil, T., Gaborit, P., Gama, N., Gueron, S., Howe, J., Hülsing, A., Joseph, D., Joux, A., Kulkarni, M., Persichetti, E., Randrianarisoa, T.H., Rivain, M., Yue, D.: SDitH — Syndrome Decoding in the Head. Tech. rep., National Institute of Standards and Technology (2024), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures
3. Albrecht, M.R., Cid, C., Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M.: Algebraic cryptanalysis of STARK-friendly designs: Application to MARVELlous and MiMC. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 371–397. Springer, Cham (Dec 2019). 10.1007/978-3-030-34618-8_13
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 430–454. Springer, Berlin, Heidelberg (Apr 2015). 10.1007/978-3-662-46800-5_17
5. Barker, E.: Recommendation for key management: Part 1 – general. Tech. Rep. NIST Special Publication (SP) 800-57 Part 1 Revision 5, National Institute of Standards and Technology, Gaithersburg, MD, USA (May 2020). 10.6028/NIST.SP.800-57pt1r5, https://doi.org/10.6028/NIST.SP.800-57pt1r5
6. Bassham, L.E., Rukhin, A.L., Soto, J., Nechvatal, J., Smid, M.E., Barker, E.B., Leigh, S., Levenson, M.S., Vangel, M.G., Banks, D., Heckert, N.A., Dray, J.F., Vo, S.C.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. Rep. NIST Special Publication (SP) 800-22 Rev. 1a, National Institute of Standards and Technology, Gaithersburg, MD (2010). 10.6028/NIST.SP.800-22r1a
7. Baum, C., Beullens, W., Braun, L., Delpech de Saint Guilhem, C., Klooß, M., Majenz, C., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: Shorter, tighter, FAESTer: Optimizations and improved (QROM) analysis for VOLE-in-the-head signatures. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part VI. LNCS, vol. 16005, pp. 124–156. Springer, Cham (Aug 2025). 10.1007/978-3-032-01887-8_5
8. Baum, C., Beullens, W., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part I. LNCS, vol. 15484, pp. 463–493. Springer, Singapore (Dec 2024). 10.1007/978-981-96-0875-1_15
9. Baum, C., Braun, L., Beullens, W., Delpech de Saint Guilhem, C., Klooß, M., Majenz, C., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: FAEST. Tech. rep., National Institute of Standards and Technology (2024), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-3-additional-signatures
10. Baum, C., Braun, L., Delpech de Saint Guilhem, C., Klooß, M., Majenz, C., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: FAEST. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
11. Baum, C., Braun, L., Delpech de Saint Guilhem, C., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 581–615. Springer, Cham (Aug 2023). 10.1007/978-3-031-38554-4_19
12. Baum, C., Delpech de Saint Guilhem, C., Kales, D., Orsini, E., Scholl, P., Zaverucha, G.: Banquet: Short and fast signatures from AES. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 266–297. Springer, Cham (May 2021). 10.1007/978-3-030-75245-3_11
13. Bernstein, D.J., Yang, B.Y.: Asymptotically faster quantum algorithms to solve multivariate quadratic equations. In: PQCrypto. pp. 487–506. Springer (2018)
14. Bettale, L., Faugère, J.C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology **3**(3), 177–197 (2009). doi:10.1515/JMC.2009.009, https://doi.org/10.1515/JMC.2009.009
15. Biryukov, A., García Fernández, P., Udovenko, A.: Magic pot: Cryptanalysis of full aim2 in the standard and related-/reused-key settings using new elimination framework. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology – EUROCRYPT 2026. pp. 63–90. Springer Nature Switzerland, Cham (2026)
16. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1825–1842. ACM Press (Oct / Nov 2017). 10.1145/3133956.3133997
17. Chen, Y.A., Gao, X.S.: Quantum algorithm for boolean equation solving and quantum algebraic attack on cryptosystems. Journal of Systems Science and Complexity **35**(1), 373–412 (February 2022). 10.1007/s11424-021-1103-7
18. Cheon, J.H., Lee, D.H.: Resistance of s-boxes against algebraic attacks. In: Roy, B., Meier, W. (eds.) Fast Software Encryption. pp. 83–93. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
19. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Berlin, Heidelberg (May 2000). 10.1007/3-540-45539-6_27
20. Daniel Kales: Banquet implementation (2026), https://github.com/dkales/banquet, gitHub repository. Verified on 2026-04-19. Public repository; the README describes it as an implementation of the Banquet signature scheme
21. Delpech de Saint Guilhem, C., Orsini, E., Tanguy, T.: Limbo: Efficient zero-knowledge MPCitH-based arguments. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 3022–3036. ACM Press (Nov 2021). 10.1145/3460120.3484595

22. Dinur, I.: Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 374–403. Springer, Cham (Oct 2021). 10.1007/978-3-030-77870-5_14
23. Dobraunig, C., Kales, D., Rechberger, C., Schofnegger, M., Zaverucha, G.: Shorter signatures based on tailor-made minimalist symmetric-key crypto. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 843–857. ACM Press (Nov 2022). 10.1145/3548606.3559353
24. FAEST Team: FAEST avx2 implementation (2025), https://github.com/faest-sign/faest-avx, gitHub repository. Verified on 2026-04-08. Public repository; GitHub organization page describes it as "FAEST implementation for AVX2 and other extensions on the x86-64 platform" and reports last update on Feb. 27, 2025
25. FAEST Team: FAEST reference implementation (2025), https://github.com/faest-sign/faest-ref, gitHub repository. Verified on 2026-04-08. Public repository; GitHub organization page describes it as "FAEST reference implementation" and reports last update on Oct. 27, 2025
26. Faugère, J.C., Horan, S., Kahrobaei, D., Kaplan, M., Kashefi, E., Perret, L.: Fast quantum algorithm for solving multivariate quadratic equations. Cryptology ePrint Archive, Paper 2017/1236 (2017), https://eprint.iacr.org/2017/1236
27. Feneuil, T., Rivain, M.: Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. Journal of Cryptology **38**(3), 28 (Jul 2025). 10.1007/s00145-025-09543-8
28. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). 10.1007/3-540-47721-7_12
29. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON: Fast-Fourier Lattice-based Compact Signatures over NTRU. Official Website (2021), https://falcon-sign.info/, nIST Post-Quantum Cryptography Standardization Round 3 finalist; last updated November 2021
30. Fröberg, R.: An inequality for hilbert series of graded algebras. Mathematica Scandinavica **56**(2) (1985)
31. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for Boolean circuits. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 1069–1083. USENIX Association (Aug 2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/giacomelli
32. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96). pp. 212–219. Association for Computing Machinery, New York, NY, USA (1996)
33. Hülsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbaan, B., Beullens, W.: SPHINCS$^+$. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
34. Institute of Applied Information Processing and Communications, Graz University of Technology: BN++ and helium signatures implementation (2026), https://github.com/isec-tugraz/bnpp_helium_signatures, gitHub repository. Verified on 2026-04-19. Public repository; the README says it contains the code for the BN++ and Helium signature-scheme variants from the paper
35. Institute of Applied Information Processing and Communications, Graz University of Technology: Rainier signatures implementation (2026), https://github.com/isec-tugraz/rainier-signatures, gitHub repository. Verified on 2026-04-08. Public repository for the Rainier signature scheme implementation
36. Institute of Commercial Cryptography Standards (ICCS): Call for proposals for the next-generation public-key cryptographic algorithms. Official Announcement of the Next-generation Commercial Cryptographic Algorithms Program (NGCC) (Oct 2025), https://www.niccs.org.cn/niccs/Notice/pc/content/content_1975896137741635584.html, accessed: 2026-04-09
37. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007). 10.1145/1250790.1250794
38. ISO/IEC JTC 1/SC 27: Information security — digital signatures with appendix — part 4: Stateful hash-based mechanisms (Jun 2024), https://www.iso.org/standard/80492.html
39. Jang, K., Baksi, A., Kim, H., Seo, H., Chattopadhyay, A.: Improved quantum analysis of SPECK and LowMC. In: Isobe, T., Sarkar, S. (eds.) INDOCRYPT 2022. LNCS, vol. 13774, pp. 517–540. Springer, Cham (Dec 2022). 10.1007/978-3-031-22912-1_23
40. Kales, D., Zaverucha, G.: Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Report 2022/588 (2022), https://eprint.iacr.org/2022/588
41. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 525–537. ACM Press (Oct 2018). 10.1145/3243734.3243805
42. Kim, S., Cho, J., Ha, J., Kwon, J., Lee, B., Lee, J., Lee, J., Lee, S., Moon, D., Son, M., Yoon, H.: The AIMer signature scheme. Tech. rep., Samsung SDS, KAIST, CryptoLab, Sungshin Women's University (Jan 2026), https://aimer-signature.org/docs/AIMer-specification-v2.0.pdf, official specification document for the AIMer post-quantum digital signature scheme
43. Kim, S., Ha, J., Son, M., Lee, B.: Efficacy and mitigation of the cryptanalysis on AIM. Cryptology ePrint Archive, Paper 2023/1474 (2023), https://eprint.iacr.org/2023/1474
44. Kim, S., Ha, J., Son, M., Lee, B., Moon, D., Lee, J., Lee, S., Kwon, J., Cho, J., Yoon, H., Lee, J.: AIM: Symmetric primitive for shorter signatures with stronger security. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 401–415. ACM Press (Nov 2023). 10.1145/3576915.3616579
45. KpqC Project: AIMer signature algorithm final specification. KpqC Algorithms Exhibit (Jan 2026), https://kpqc.or.kr/contents/03_exhibit/sub_03.html

46. Liu, F., Mahzoun, M., Meier, W.: Modelling ciphers with overdefined systems of quadratic equations: Application to friday, vision, RAIN and biscuit. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part VII. LNCS, vol. 15490, pp. 424–456. Springer, Singapore (Dec 2024). 10.1007/978-981-96-0941-3_14
47. Liu, F., Mahzoun, M., Meier, W.: Modelling ciphers with overdefined systems of quadratic equations: Application to friday, vision, RAIN and biscuit. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024, Proceedings, Part VII. pp. 424–456. Lecture Notes in Computer Science, Springer (2024). 10.1007/978-981-96-0941-3_14, https://doi.org/10.1007/978-981-96-0941-3_14
48. Liu, F., Mahzoun, M., Øygarden, M., Meier, W.: Algebraic attacks on RAIN and AIM using equivalent representations. IACR Trans. Symmetric Cryptol. **2023**(4), 166–186 (2023). 10.46586/TOSC.V2023.I4.166-186, https://doi.org/10.46586/tosc.v2023.i4.166-186
49. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
50. Magali Bardet, J.C.F., Salvy, B.: On the complexity of gröbner basis computation of semi-regular overdetermined algebraic equations. In: International Conference on Polynomial System Solving. pp. 71–74 (2004)
51. Müller, S.: XDRBG-512 Algorithm. Technical report (September 2025), https://leancrypto.org/papers/xdrbg512.pdf, leancrypto technical paper
52. NIST: Call for additional digital signature schemes for the post-quantum cryptography standardization process. Technical report (2022), https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
53. NIST: Module-Lattice-Based Digital Signature Standard. Tech. Rep. FIPS 204 (Aug 2024). 10.6028/NIST.FIPS.204, https://csrc.nist.gov/pubs/fips/204/final, based on the CRYSTALS-Dilithium algorithm, final standard
54. NIST: Stateless Hash-Based Digital Signature Standard. Tech. Rep. FIPS 205 (Aug 2024). 10.6028/NIST.FIPS.205, https://csrc.nist.gov/pubs/fips/205/final, based on the SPHINCS$^+$ algorithm, final standard
55. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
56. Roy, L.: SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 657–687. Springer, Cham (Aug 2022). 10.1007/978-3-031-15802-5_23
57. Samsung SDS Open Source: AIMer reference implementation (2026), https://github.com/samsungsds-opensource/AIMer, gitHub repository. Verified on 2026-04-08. Public repository containing `AIMer-package.zip`; GitHub reports 1 commit, latest visible update Jan. 30, 2026
58. Sauer, J.F., Szepieniec, A.: Sok: Gröbner basis algorithms for arithmetization oriented ciphers. IACR Cryptol. ePrint Arch. p. 870 (2021), https://eprint.iacr.org/2021/870
59. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). 10.1109/SFCS.1994.365700
60. Sun, Y., Chen, S., Liu, G., Wang, M., Niu, C.: High exponents may not suffice to patch AIM (on attacks, weak parameters, and patches for AIM2). Cryptology ePrint Archive, Paper 2025/2272 (2025), https://eprint.iacr.org/2025/2272
61. Yang, H.S., Zheng, Q.X., Yang, J.: Algebraic cryptanalysis of AO primitives based on polynomial decomposition - applications to rain and full AIM-I/III/V. In: Hanaoka, G., Yang, B.Y. (eds.) ASIACRYPT 2025, Part I. LNCS, vol. 16245, pp. 383–410. Springer, Singapore (Dec 2025). 10.1007/978-981-95-5018-0_13
62. Yang, H.S., Zheng, Q.X., Yang, J., feng Liu, Q., Tang, D.: A new security evaluation method based on resultant for arithmetic-oriented algorithms. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part VII. LNCS, vol. 15490, pp. 457–489. Springer, Singapore (Dec 2024). 10.1007/978-981-96-0941-3_15
63. Yang, K., Sarkar, P., Weng, C., Wang, X.: QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2986–3001. ACM Press (Nov 2021). 10.1145/3460120.3484556
64. Zaverucha, G., Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D.: Picnic. Tech. rep., National Institute of Standards and Technology (2017), available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions
65. Zaverucha, G., Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Katz, J., Wang, X., Kolesnikov, V., Kales, D.: Picnic. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions
66. Zhang, K., Wang, Q., Yu, Y., Guo, C., Cui, H.: Algebraic attacks on round-reduced rain and full AIM-III. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part III. LNCS, vol. 14440, pp. 285–310. Springer, Singapore (Dec 2023). 10.1007/978-981-99-8727-6_10

A Missing Definitions

We make explicit the five security notions used in Theorem 1, adapting the corresponding games from the FAESTv2 specification [9, Secs. 9.3–9.6]. Let $\mathcal{O}$ denote the public oracles available in the model under consideration, including the stated random oracles and, where applicable, the ideal cipher. All probabilities are over key generation, the coins of the experiments and adversaries, and the sampled oracles.

Definition 3 (EUF-KO advantage). *Following [9, Definition 9.22], the EUF-KO experiment for a signature scheme* $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *and an adversary* $\mathcal{A}$ *is*

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{O}}(1^n),$$
$$(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}),$$
$$\mathsf{Exp}^{\mathsf{EUF\text{-}KO}}_{\mathsf{SIG},\mathcal{A}} := \mathsf{Verify}^{\mathcal{O}}(\mathsf{pk}, m, \sigma).$$

The adversary receives the public key but no signatures. Its advantage is its probability of producing an accepting message–signature pair:

$$\mathsf{Adv}^{\mathsf{EUF\text{-}KO}}_{\mathcal{A}} := \Pr\left[\mathsf{Exp}^{\mathsf{EUF\text{-}KO}}_{\mathsf{Lynxer},\mathcal{A}} = 1\right].$$

Definition 4 (EUF-CMA advantage). *Following [9, Definition 9.22], the EUF-CMA experiment gives* $\mathcal{A}$ *access to the signing oracle*

$$\mathcal{O}_{\mathsf{Sign}}(m): \quad \mathcal{M} \leftarrow \mathcal{M} \cup \{m\}; \quad \textit{return } \mathsf{Sign}^{\mathcal{O}}(\mathsf{sk}, m).$$

The experiment initializes $\mathcal{M} := \varnothing$*, samples* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{O}}(1^n)$*, and obtains* $(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}, \mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk})$*. It outputs one exactly when*

$$\mathsf{Verify}^{\mathcal{O}}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \quad \textit{and} \quad m^\star \notin \mathcal{M}.$$

For an adversary making at most Q_{sig} *signing queries, we define*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A}}[Q_{\mathsf{sig}}] := \Pr\left[\mathsf{Exp}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{Lynxer},\mathcal{A}} = 1\right].$$

Definition 5 (PRG-OWF advantage). *The term* $\mathsf{Adv}^{\mathsf{PRG\text{-}OWF}}_{\mathcal{B}}$ *measures pseudorandomness of the output of the underlying OWF, rather than one-wayness by inversion; this is the game used in the proof of FAESTv2's EUF-KO bound [9, Theorem 9.24 and its proof]. Sample* $b \leftarrow \{0,1\}$ *and sample* (iv, k) *according to the valid-key distribution of* $\mathsf{Lynxer.KeyGen}$*. Set*

$$t_0 := \mathsf{Lynx}_{\mathsf{iv}}(k), \qquad t_1 \leftarrow \mathbb{F}_{2^n},$$

give (iv, t_b) *to* $\mathcal{B}$*, and let* b' *be its output. Writing* $p := \Pr[b' = b]$*, define*

$$\mathsf{Adv}^{\mathsf{PRG\text{-}OWF}}_{\mathcal{B}} := |2p - 1|.$$

Thus, $\mathcal{B}$ *receives one public OWF instance and must distinguish an honestly generated* Lynx *output from an independent uniform field element.*

Definition 6 (H_3-PRF advantage). *We use the oracle-indistinguishability definition of a PRF from [9, Definition 9.8 and Sec. 9.3.3], instantiated as in the proof of [9, Theorem 9.36]. For* $\rho \leftarrow \{0,1\}^n$*, define the keyed function*

$$H_{3,\rho}(x) := H_3(x \parallel \rho), \qquad H_{3,\rho} : \mathcal{X} \longrightarrow \{0,1\}^{n+128},$$

where signing evaluates it at $x = \mathsf{sk} \parallel \mu$ *to derive* $(r, \mathsf{iv}^{\mathsf{pre}})$*. In the real experiment a distinguisher* $\mathcal{D}$ *receives oracle access to* $H_{3,\rho}$*; in the ideal experiment it receives oracle access to a uniformly sampled function* $R : \mathcal{X} \to \{0,1\}^{n+128}$*. Its advantage is*

$$\mathsf{Adv}^{\mathsf{PRF}}_{H_3,\mathcal{D}} := \left|\Pr\left[\mathcal{D}^{H_{3,\rho}}(1^n) = 1\right] - \Pr\left[\mathcal{D}^{R}(1^n) = 1\right]\right|.$$

The notation $\mathsf{Adv}^{\mathsf{PRF}}_{H_3}$ *in Theorem 1 suppresses the distinguisher produced by the reduction.*

Definition 7 (PRP-type advantage of PRG_{2n}). *We adapt the PRP-type game for an IV-based tweakable PRG from [9, Definition 9.10]. Let* $\alpha := 2n/128$ *and let*

$$\mathsf{PRG}_{2n} : \{0,1\}^n \times \{0,1\}^{128} \times \{0,1\}^{32} \longrightarrow (\{0,1\}^{128})^{\alpha}.$$

The experiment samples $b \leftarrow \{0,1\}$*, a seed* $\mathsf{sd} \leftarrow \{0,1\}^n$*, and a public* $\mathsf{iv} \leftarrow \{0,1\}^{128}$*, and gives* iv *to a distinguisher* $\mathcal{D}$*. The distinguisher may query a tweak* twk *to an oracle* $\mathcal{G}_b$ *defined as follows:*

- $b = 0$**:** $\mathcal{G}_0(\mathsf{twk})$ *returns* $\mathsf{PRG}_{2n}(\mathsf{sd}, \mathsf{iv}, \mathsf{twk})$*.*
- $b = 1$**:** *For each previously unseen tweak,* $\mathcal{G}_1$ *samples* α *distinct* 128*-bit blocks uniformly from* $\{0,1\}^{128}$*, without replacement across all blocks returned so far, stores their concatenation for that tweak, and returns it. A repeated tweak receives the stored response.*

If $\mathcal{D}$ *makes at most* T *oracle queries, outputs* b'*, and wins when* $b' = b$*, then*

$$\mathsf{Adv}^{\mathsf{PRP}}_{\mathsf{PRG}_{2n},\mathcal{D}}[T] := |2\Pr[b' = b] - 1|.$$

Accordingly, $\mathsf{Adv}^{\mathsf{PRP}}_{\mathsf{PRG}_{2n}}[1]$ *in Theorem 1 denotes the advantage of the reduction's distinguisher in this game with one oracle query.*

B Detailed Specification of Lynxer

We follow the modular decomposition of the FAEST v2 specification [9], but instantiate the one way function with Lynx from Figure 3. Throughout this appendix, the VOLEitH security parameter is n, the padded VOLE length is $\hat{\ell} = \ell + 2n + B$, the first τ_1 vectors have size $N_i = 2^k$, the remaining $\tau_0 = \tau - \tau_1$ vectors have size $N_i = 2^{k-1}$, and $w_{\mathsf{grind}} = n - \sum_{i=0}^{\tau-1} k_i$ also denotes the zero-padding width in the concatenated VOLE matrices. Unlike FAEST, our BAVC leaf commitment uses the simpler $2n$-bit PRG expansion, like FAEST-EM, and the QuickSilver layer is specialized to degree 2, so only the coefficients a_0 and a_1 are needed.

B.1 Top-Level Algorithms

This subsection records the top-level APIs of Lynxer. It specifies how key generation samples a valid Lynx secret/public-key pair, how signing derives the VOLE, QuickSilver, and BAVC transcript components from a message and secret key, and how verification reconstructs the same transcript from the public key and signature. We elaborate on the QuickSilver part as it proves the input-output relationship of Lynx and makes explicit how the structure of Lynx aligns with the VOLEitH framework. We do not detail the sub-components of IT-MAC/VOLE generation, BAVC commitment, universal hash functions, and other auxiliary functions, which are common to the VOLEitH framework and can be found in the specification of FAESTv2 [9].

Algorithm 1: Lynxer Key Generation

Input: Security parameter 1^n, where $n \in \{128, 160, 192, 256, 384, 512\}$
Output: Secret key sk, public key pk

Sample $k \leftarrow \{0,1\}^n$;
repeat
  Sample $\mathsf{iv} \leftarrow \{0,1\}^n$;
  Derive the public linear maps L_0, L_1, L_2 and constants c_1, c_2 from iv;
  Set $x_1 := k + c_1$ and $x_2 := L_0(k) + c_2$;
until $x_1 \neq 0$, $x_2 \neq 0$, *and* $x_1 \neq x_2$;
Compute $t := \mathsf{Lynx}_{\mathsf{iv}}(k)$;
return $\mathsf{sk} := (\mathsf{iv}, k)$ *and* $\mathsf{pk} := (\mathsf{iv}, t)$;

B.2 QuickSilver

This subsection describes the QuickSilver proving process and constraints related to Lynx. The Lynx extended witness is $w = (k, v_1, v_2)$, ZK.OWFConstraints generates three degree-2 IT-MAC values to be zero-checked according to Section 4, and ZK.OWFProve/ ZK.OWFVerify define how the QuickSilver check integrates with the broader VOLE-in-the-Head framework. Because all Lynx checks are degree-2, the QuickSilver layer only uses the coefficient vectors a_0 and a_1; there is no a_2 term as in FAESTv2.

Algorithm 2: Lynxer Sign

Input: Message $m \in \{0,1\}^*$, secret key $\mathsf{sk} = (\mathsf{iv}, k)$, public key $\mathsf{pk} = (\mathsf{iv}, t)$
Output: Signature σ

Sample $\rho \leftarrow \{0,1\}^n$ and initialize $\mathsf{ctr} := 0$, $\mathsf{decom}_I := \bot$;
Compute $\mu := H_0(\mathsf{pk} \parallel m)$;
Derive $(r, \mathsf{iv}^{\mathsf{pre}}) := H_3(\mathsf{sk} \parallel \mu \parallel \rho)$ and set $\mathsf{iv}^\star := H_4(\mathsf{iv}^{\mathsf{pre}})$;
$(\mathsf{com}, \mathsf{decom}, \boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\tau-1}, \boldsymbol{u}, V) \leftarrow \mathsf{VOLECommit}(r, \mathsf{iv}^\star, \hat{\ell}; \mathsf{param})$;
$\mathsf{chall}_1 := H_1(\mu \parallel \mathsf{com} \parallel \boldsymbol{c}^1 \parallel \cdots \parallel \boldsymbol{c}^{\tau-1} \parallel \mathsf{iv}^{\mathsf{pre}})$;
Compute $\widetilde{\boldsymbol{u}} := \mathsf{VOLEHash}(\mathsf{chall}_1, \boldsymbol{u})$ and $\widetilde{V} := \mathsf{VOLEHash}(\mathsf{chall}_1, V)$;
Mask the witness by $\boldsymbol{d} := w \oplus \boldsymbol{u}[0, \ell)$, where

$$w := k \| v_1 \| v_2, v_1 := S_1(k + c_1), v_2 := S_2(L_0(k) + c_2) \; .$$

$\mathsf{chall}_2 := H_{2,0}(\mathsf{chall}_1 \parallel \widetilde{\boldsymbol{u}} \parallel \widetilde{V} \parallel \boldsymbol{d})$;
Let $\boldsymbol{u}^{\mathsf{qs}} := \boldsymbol{u}[\ell, \ell + n)$ and $V^{\mathsf{qs}} := V[0, \ell + n)$;
$(\widetilde{a}_0, \widetilde{a}_1) \leftarrow \mathsf{ZK.OWFProve}(w, \boldsymbol{u}^{\mathsf{qs}}, V^{\mathsf{qs}}, \mathsf{pk}, \mathsf{chall}_2; \mathsf{param})$;
while $\mathsf{decom}_I = \bot$ **do**
  $\mathsf{chall}_3 \leftarrow H_{2,3}(\mathsf{chall}_2 \parallel \widetilde{a}_0 \parallel \widetilde{a}_1 \parallel \mathsf{ctr}; n)$;
  if $\mathsf{CheckGrind}(\mathsf{chall}_3)$ *fails* **then**
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$;
    continue;
  $I \leftarrow \mathsf{DecodeAllChall3}(\mathsf{chall}_3[0, n - w_{\mathsf{grind}}); \mathsf{param})$;
  $\mathsf{decom}_I \leftarrow \mathsf{BAVC.Open}(\mathsf{decom}, I; \mathsf{param})$;
  if $\mathsf{decom}_I = \bot$ **then**
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$;
return $\sigma := (\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\tau-1}, \widetilde{\boldsymbol{u}}, \boldsymbol{d}, \widetilde{a}_1, \mathsf{decom}_I, \mathsf{chall}_3, \mathsf{iv}^{\mathsf{pre}}, \mathsf{ctr})$;

Algorithm 3: Lynxer Verify

Input: Message $m \in \{0,1\}^*$, public key $\mathsf{pk} = (\mathsf{iv}, t)$, signature σ
Output: Bit $b \in \{0,1\}$

Parse

$$\sigma = (\boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\tau-1}, \widetilde{\boldsymbol{u}}, \boldsymbol{d}, \widetilde{a}_1, \mathsf{decom}_I, \mathsf{chall}_3, \mathsf{iv}^{\mathsf{pre}}, \mathsf{ctr}) \; .$$

Reject if $\mathsf{CheckGrind}(\mathsf{chall}_3)$ fails;
Compute $\mu := H_0(\mathsf{pk} \parallel m)$ and $\mathsf{iv}^\star := H_4(\mathsf{iv}^{\mathsf{pre}})$;
$(\mathsf{com}, Q) \leftarrow \mathsf{VOLEReconstruct}(\mathsf{chall}_3[0, n - w_{\mathsf{grind}}), \mathsf{decom}_I, \boldsymbol{c}^1, \ldots, \boldsymbol{c}^{\tau-1}, \mathsf{iv}^\star, \hat{\ell}; \mathsf{param})$;
if $(\mathsf{com}, Q) = \bot$ **then**
  return 0;
$\mathsf{chall}_1 := H_1(\mu \parallel \mathsf{com} \parallel \boldsymbol{c}^1 \parallel \cdots \parallel \boldsymbol{c}^{\tau-1} \parallel \mathsf{iv}^{\mathsf{pre}})$;
Compute $\widetilde{Q} := \mathsf{VOLEHash}(\mathsf{chall}_1, Q)$ and $\Delta := \mathsf{ToField}(\mathsf{chall}_3, n)$;
Set $\widetilde{V}' := \widetilde{Q} - \Delta \cdot \widetilde{\boldsymbol{u}}$;
$\mathsf{chall}_2 := H_{2,0}(\mathsf{chall}_1 \parallel \widetilde{\boldsymbol{u}} \parallel \widetilde{V}' \parallel \boldsymbol{d})$;
$\widetilde{a}_0 \leftarrow \mathsf{ZK.OWFVerify}(\boldsymbol{d}, Q[0, \ell + n), \mathsf{pk}, \mathsf{chall}_2, \mathsf{chall}_3, \widetilde{a}_1; \mathsf{param})$;
$\mathsf{chall}_3' \leftarrow H_{2,3}(\mathsf{chall}_2 \parallel \widetilde{a}_0 \parallel \widetilde{a}_1 \parallel \mathsf{ctr}; n)$;
if $\mathsf{chall}_3' = \mathsf{chall}_3$ **then**
  return 1;
return 0;

Algorithm 4: ZK.OWFConstraints

Input: Authenticated witness $[\![w]\!] = ([\![k]\!], [\![v_1]\!], [\![v_2]\!])$, public key $\mathsf{pk} = (\mathsf{iv}, t)$
Output: Authenticated coefficient vectors (a_0, a_1) for the Lynx degree-2 constraints

Derive the public linear maps L_0, L_1, L_2 and constants c_1, c_2 from iv;
Initialize the authenticated constraint list $[\![\boldsymbol{z}]\!] := ()$;
Set $[\![x_1]\!] := [\![k]\!] + c_1$ and $[\![x_2]\!] := L_0([\![k]\!]) + c_2$;
for $i \in \{1, 2\}$ **do**
    Append $[\![x_i]\!] \cdot [\![v_i]\!] + 1$ to $[\![\boldsymbol{z}]\!]$;
Append $L_1([\![v_1]\!]) \cdot L_2([\![v_2]\!]) + L_3([\![k]\!] + [\![v_1]\!] + [\![v_2]\!]) + t$ to $[\![\boldsymbol{z}]\!]$;
Parse the degree-2 QuickSilver encoding of $[\![\boldsymbol{z}]\!]$ as the coefficient pair $(\boldsymbol{a}_0, \boldsymbol{a}_1)$; // $p_{\boldsymbol{z}}(x) = \boldsymbol{z}x^2 + \boldsymbol{a}_1 x + \boldsymbol{a}_0$.
return $(\boldsymbol{a}_0, \boldsymbol{a}_1)$;

Algorithm 5: ZK.OWFProve

Input: Witness $w \in \{0,1\}^\ell$, auxiliary mask $\boldsymbol{u} \in \{0,1\}^n$, VOLE rows $V \in \mathbb{F}_2^{(\ell+n)\times n}$, public key pk, challenge chall_2
Output: QuickSilver response $(\widetilde{a}_0, \widetilde{a}_1)$

for $i \in [0, \ell - 1]$ **do**
    Interpret the i-th witness bit as the authenticated value $[\![w[i]]\!] := (w[i], \mathsf{ToField}(V|_i, n))$;
Set $u^\star := \mathsf{ToField}(\boldsymbol{u}, n)$;
for $i \in [0, n - 1]$ **do**
    $v[i] := \mathsf{ToField}(V|_{\ell+i}, n)$;
Let $\alpha_n \in \mathbb{F}_{2^n}$ be the basis element over $\mathbb{F}_2$, and set $v^\star := \sum_{i\in[0,n-1]} v[i]\alpha_n^i$;
$(\boldsymbol{a}_0, \boldsymbol{a}_1) \leftarrow \mathsf{ZK.OWFConstraints}([\![w]\!], \mathsf{pk}; \mathsf{param})$;
$\widetilde{a}_0 \leftarrow \mathsf{ZKHash}(\mathsf{chall}_2, (\boldsymbol{a}_0, u^\star))$;
$\widetilde{a}_1 \leftarrow \mathsf{ZKHash}(\mathsf{chall}_2, (\boldsymbol{a}_1, v^\star))$;
return $(\widetilde{a}_0, \widetilde{a}_1)$;

Algorithm 6: ZK.OWFVerify

Input: Masked witness $\boldsymbol{d} \in \{0,1\}^{\ell}$, verifier rows $Q \in \mathbb{F}_2^{(\ell+n)\times n}$, public key pk, challenge chall_2, challenge chall_3, prover value $\widetilde{a}_1$

Output: The reconstructed QuickSilver value $\widetilde{a}_0$

Set $\Delta := \mathsf{ToField}(\mathsf{chall}_3, n)$;
for $i \in [0, \ell-1]$ **do**
  Reconstruct the authenticated bit $[\![w[i]]\!] := \mathsf{ToField}(Q|_i, n) + \boldsymbol{d}[i] \cdot \Delta$;
for $i \in [0, n-1]$ **do**
  $q[i] := \mathsf{ToField}(Q|_{\ell+i}, n)$;
Let $\alpha_n \in \mathbb{F}_{2^n}$ be the basis element over $\mathbb{F}_2$, and set $q^\star := \sum_{i\in[0,n-1]} q[i]\alpha_n^i$;
$\boldsymbol{b} \leftarrow \mathsf{ZK.OWFConstraints}([\![w]\!], \mathsf{pk}; \mathsf{param})$;
Set $\widehat{a} := \mathsf{ZKHash}(\mathsf{chall}_2, (\boldsymbol{b}, q^\star))$;
return $\widehat{a} - \widetilde{a}_1 \cdot \Delta$;