A Compact Signature Scheme Based on QC-MDGM Codes

Alessandro Annechini[0009−0009−4452−802X], Alessandro Barenghi[0000−0003−0840−6358], and Gerardo Pelosi[0000−0002−3812−5429]

Department of Electronics, Information and Bioengineering - DEIB, Politecnico di Milano, Milano, Italy
{alessandro.annechini, alessandro.barenghi, gerardo.pelosi}@polimi.it

Abstract. Constructing a post-quantum signature scheme that is simultaneously compact and efficient remains a central challenge in code-based cryptography. Existing schemes based on turning a zero-knowledge identification scheme into a signature exhibit either large signatures or slow verification procedures, while the design of hash-and-sign schemes has led to constructions such as Wave and MIRANDA, having small signatures at the cost of massive public key sizes and time-consuming signature algorithms. In this work, we present ASTRA-Sign: a quASi-cyclic code-based full-distance decoding TRApdoor Signature Scheme, combining the hash-and-sign paradigm with quasi-cyclic moderate density generator matrix codes to obtain small signatures and small public keys. The security of our scheme is based on the hardness of finding low weight codewords in quasi-cyclic codes, and on the hardness of finding a codeword that has full Hamming distance from a given random vector. We analyse key recovery and signature forgery attacks against ASTRA, and we propose several parameter sets achieving 128-, 192- and 256-bit security. Our scheme exhibits public keys and signatures below 1 kiB for 128 bits of security, with 40 μs verification times.

Keywords: Post-quantum cryptography · Post-quantum digital signature · Code-based cryptography · QC-MDGM codes

1 Introduction

The transition to post-quantum cryptography (PQC) is one of the most critical challenges in modern information security. In the area of digital signatures, the National Institute of Standards and Technology (NIST) has standardized the lattice-based scheme ML-DSA [40] (formerly CRYSTALS-Dilithium [29]) and the conservative hash-based scheme SLH-DSA [41] (formerly SPHINCS$^+$ [7]), while FALCON has been selected for standardization as FN-DSA [49]. These schemes provide practical and well-studied options, but they also leave open an important design question: constructing a post-quantum digital signature scheme based on non-lattice assumptions, while simultaneously achieving compact public keys, compact signatures, and very efficient verification.

This question is especially relevant in asymmetric deployment scenarios. For example, in the Transport Layer Security (TLS) protocol, certificate chains, including certificates for intermediate Certification Authorities (CAs), are transmitted during the handshake. Reducing both the size of the public keys embedded in such certificates and the size of the corresponding signatures can reduce communication overhead, while fast verification directly affects handshake latency. Likewise, in embedded secure-boot environments, devices must authenticate firmware before execution, making fast verification crucial and favoring schemes with compact signatures. In these settings, a moderately more expensive signing procedure can be an acceptable trade-off when signing is performed offline or in controlled environments, provided that the resulting scheme offers compact public keys, compact signatures, and fast verification.

Code-based cryptography is a natural candidate for assumption diversity. The hard problems underlying code-based cryptography have been studied for several decades, and code-based Key Encapsulation Mechanisms (KEMs) have demonstrated strong security margins and competitive performance. Digital signatures, however, remain significantly more difficult to construct from error-correcting codes. Existing code-based signatures typically follow one of two design paradigms. Fiat-Shamir signatures derived from zero-knowledge identification protocols can rely on randomly picked hard problem instances, but their signatures include non-interactive proof transcripts, making them comparatively large. Conversely, hash-and-sign schemes can produce compact signatures and simple verification algorithms, but they require a trapdoor that must be embedded without leaking exploitable structure.

Classical hash-and-sign code-based signatures, such as the CFS scheme [24], rely on a hidden decodable code and on the ability of the signer to invert a syndrome map. However, the density of decodable syndromes is very small for the high-rate Goppa codes used in CFS, causing expensive signing procedures and large public keys. Moreover, recent cryptanalysis has reported an attack against the CFS setting based on very high-rate binary Goppa codes [13]. More recent hash-and-sign approaches, inspired by the Gentry-Peikert-Vaikuntanathan (GPV) framework [32], include Wave [26] and MIRANDA [25]. These schemes demonstrate the flexibility of trapdoor-sampling techniques in coding theory, but their public keys and signing costs remain large for several practical use cases.

A successful approach to reducing key sizes and improving efficiency in code-based cryptography is the adoption of Quasi-Cyclic (QC) codes. QC codes allow each circulant block of a structured matrix to be represented by one row, or equivalently by one polynomial, thereby reducing storage and supporting efficient arithmetic. This compact representation is central to state-of-the-art KEMs such as HQC [38] and BIKE [6]. However, using QC codes in digital signatures is delicate. Previous attempts [46,52] have suffered from structural weaknesses, where the sparsity of the secret code or statistical biases in the signature outputs leaked information on the secret key [43,47].

Contributions. In this work, we propose ASTRA-Sign: a quASi-cyclic code-based full-distance decoding TRApdoor Signature Scheme, a code-based hash-

and-sign signature scheme built from Quasi-Cyclic Moderate Density Generator Matrix (QC-MDGM) codes. It is designed to combine the compact representation and efficient arithmetic provided by quasi-cyclic structure with a trapdoor inversion procedure tailored to full-weight error vectors over a field $\mathbb{F}_q$, with $q > 2$. In designing the scheme, the key contributions are: *i)* the proposal of a new computationally hard problem, the full-distance codeword finding, related to the canonical codeword finding in arbitrary linear codes, of which we prove the NP-hardness; *ii)* the design of trapdoored instances of the aforementioned problem, constituted by a QC-MDGM code, together with an efficient algorithm to invert the trapdoor one-way function, employing a sparse generator matrix of the QC-MDGM code as the trapdoor information; *iii)* an analysis of the uniformity of the outputs of our signature scheme, providing numerical evidence of the closeness of the distribution of the signature to an ideal (uniform) sampler; *iv)* the evaluation of current attack strategies and the proposal of concrete parameter sets to obtain ASTRA instances providing λ bit security, $\lambda \in \{128, 192, 256\}$. We report public key and signature sizes which are both below 1 kB for $\lambda = 128$, allowing them to fit in tight scenarios such as IPv6 packets, with signatures being $3\times$ smaller and public keys being $1.4\times$ smaller than the Dilithium (now ML-DSA) signature scheme selected by NIST for standardization. We evaluate the performance of a C implementation of ASTRA and verification latencies outperforming deployment-grade implementations of SPHINCS$^+$ and competing with Falcon, two NIST standardization selected signature schemes.

2 Background

Notation. Scalars are denoted by lowercase letters (e.g., p, k, n), column vectors by lowercase boldface letters (e.g., $\mathbf{w} \in \mathbb{F}_q^\ell$), matrices by uppercase boldface letters (e.g., $\mathbf{A} \in \mathbb{F}_q^{\ell_1 \times \ell_2}$), and sets by uppercase calligraphic letters (e.g., $\mathcal{V}$). The Hamming weight, i.e., the number of non-zero entries, of a vector $\mathbf{w}$ is denoted by $\mathtt{wt}(\mathbf{w})$. When the same notation is used for a row/column-regular circulant matrix $\mathbf{A}$, $\mathtt{wt}(\mathbf{A})$ denotes its row or column weight. A right-shifting square circulant matrix $\mathbf{B} \in \mathbb{F}_q^{\ell \times \ell}$ is denoted by $\mathbf{circ}(\mathbf{B}[0,:])$, where $\mathbf{B}[0,:]$ is its first row. Via the known isomorphism between circulant matrices and the quotient ring $\mathbb{F}_q[x]/\langle x^\ell - 1\rangle$, the matrix $\mathbf{circ}(\mathbf{w})$ for $\mathbf{w} = [w_0, \ldots, w_{\ell-1}] \in \mathbb{F}_q^\ell$ is represented by the polynomial $\mathsf{w}(x) = \sum_{i=0}^{\ell-1} w_i x^i$. We use $:=$ for assignment or definition, and $\leftarrow$ for uniform sampling from a set.

Digital signatures. A digital signature scheme enables a signer holding a private key sk to produce a signature σ on a message m, such that any party holding the corresponding public key pk can verify the authenticity and integrity of m.

Definition 1 (Signature Scheme). *A digital signature scheme Σ is a triple of probabilistic polynomial-time (PPT) algorithms* (KEYGEN, SIGN, VERIFY) *associated with a message space $\mathcal{M} = \{0,1\}^*$:*

- KEYGEN$(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: On input the security parameter λ in unary, it outputs a public verification key pk and a private signing key sk.*

– $\textsc{Sign}(\mathsf{sk}, m) \to \sigma$: *On input a signing key* sk *and a message* $m \in \mathcal{M}$*, it outputs a signature* σ*.*
– $\textsc{Verify}(\mathsf{pk}, m, \sigma) \to b$: *A deterministic algorithm that, on input a verification key* pk*, a message* $m \in \mathcal{M}$*, and a signature* σ*, outputs a bit* $b \in \{0, 1\}$*, where* 1 *denotes acceptance and* 0 *denotes rejection.*

The standard security notion of Existential Unforgeability under Chosen-Message Attack (EUF-CMA) models an adversary's capability to adaptively obtain signatures on messages of its choosing, and subsequently attempt to forge a valid signature on a fresh message. This is formalized via a game-based experiment.

Definition 2 (EUF-CMA Security). *Let* $\Sigma = (\textsc{Keygen}, \textsc{Sign}, \textsc{Verify})$ *be a signature scheme with message space* $\mathcal{M}$*, and let* A *be a PPT adversary. Consider the experiment* $Exp^{\text{EUF-CMA}}_{\Sigma,\mathsf{A}}(\lambda)$ *defined as follows:*

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{Keygen}(1^\lambda)$.
2. $(m^*, \sigma^*) \leftarrow \mathsf{A}^{\mathcal{O}_{\textsc{Sign}}(\mathsf{sk},\cdot)}(\mathsf{pk})$*, where* $\mathcal{O}_{\textsc{Sign}}(\mathsf{sk}, \cdot)$ *is a signing oracle that, on input a message* $m \in \mathcal{M}$*, returns* $\sigma \leftarrow \textsc{Sign}(\mathsf{sk}, m)$ *and adds* m *to the query set* $\mathcal{Q}$*, which is initially empty.*
3. *The experiment outputs* 1 *if* $\textsc{Verify}(\mathsf{pk}, m^*, \sigma^*)=1$ *and* $m^* \notin \mathcal{Q}$*,* 0 *otherwise.*

The EUF-CMA advantage of the adversary A *against* Σ *is defined as follows:* $Adv^{\text{EUF-CMA}}_{\Sigma,\mathsf{A}}(\lambda) = \Pr[Exp^{\text{EUF-CMA}}_{\Sigma,\mathsf{A}}(\lambda) = 1]$*. The scheme* Σ *is EUF-CMA secure if for every PPT adversary* A*,* $Adv^{\text{EUF-CMA}}_{\Sigma,\mathsf{A}}(\lambda)$ *is negligible in* λ*.*

In many concrete constructions, underlying hash functions are analyzed in idealized settings. In the Random Oracle Model (ROM), a hash function is modeled as a publicly accessible oracle implementing a uniformly random function. In the post-quantum setting, this is generalized to the Quantum Random Oracle Model (QROM), where the adversary is granted the capability to query the hash oracle in quantum superposition. While transitioning from the ROM to the QROM often invalidates classical proof techniques, many cryptographic paradigms have been studied and proven secure in the QROM. These include the Fiat-Shamir (FS) transform [27,28,36] and GPV-style hash-and-sign constructions [35,60].

2.1 Linear Codes

Let $\mathbb{F}_q$ denote the finite field with q elements. A q-ary linear code of length n and dimension k is a k-dimensional linear subspace $\mathcal{C} \subseteq \mathbb{F}_q^n$. It can be represented by a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ as $\mathcal{C} = \{\mathbf{G}^\intercal \mathbf{u} : \mathbf{u} \in \mathbb{F}_q^k\}$, or by a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ as $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_q^n : \mathbf{Hc} = \mathbf{0}\}$. Vectors $\mathbf{u} \in \mathbb{F}_q^k$ are called information words, and the elements of $\mathcal{C}$ are called codewords. A generator matrix is said to be in systematic form if $\mathbf{G} = \left[\mathbf{I}^{k\times k} \mid \mathbf{A}\right]$ for some $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$. The Hamming distance between $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ is $d_H(\mathbf{x}, \mathbf{y}) = \mathtt{wt}(\mathbf{x}-\mathbf{y})$. The minimum distance of $\mathcal{C}$ is $d_{\min}(\mathcal{C}) = \min\{d_H(\mathbf{c}_1, \mathbf{c}_2) : \mathbf{c}_1, \mathbf{c}_2 \in \mathcal{C}, \mathbf{c}_1 \neq \mathbf{c}_2\}$. Equivalently, since $\mathcal{C}$ is linear, $d_{\min}(\mathcal{C}) = \min\{\mathtt{wt}(\mathbf{c}) : \mathbf{c} \in \mathcal{C}, \mathbf{c} \neq \mathbf{0}\}$. A code with minimum distance $d_{\min}$ can uniquely correct up to $\lfloor (d_{\min} - 1)/2 \rfloor$ errors under bounded-distance decoding.

Given a parity-check matrix $\mathbf{H}$, the syndrome of $\mathbf{y} \in \mathbb{F}_q^n$ is $\mathbf{s} = \mathbf{H}\mathbf{y} \in \mathbb{F}_q^{n-k}$. Thus, $\mathbf{y}$ is a codeword if and only if its syndrome is zero. In syndrome decoding, if $\mathbf{y} = \mathbf{c} + \mathbf{e}$ for some transmitted codeword $\mathbf{c} \in \mathcal{C}$ and an error vector $\mathbf{e}$, then $\mathbf{H}\mathbf{y} = \mathbf{H}\mathbf{e}$; decoding amounts to recovering an error vector compatible with the observed syndrome, typically within the error-correcting capability of the code or within a prescribed cryptographic weight regime.

Quasi-cyclic codes. Structured code families are commonly used in code-based cryptography to reduce representation size and improve arithmetic efficiency. In particular, *quasi-cyclic* (QC) codes admit a block-circulant representation with codewords composed as a sequence of n_0 ℓ-symbol blocks, obtained from an information word composed by a sequence of k_0 blocks, each including ℓ symbols. Accordingly, a generator matrix can be viewed as a $k_0 \times n_0$ array of $\ell \times \ell$ circulant blocks, and a parity-check matrix as an $(n_0 - k_0) \times n_0$ array of $\ell \times \ell$ circulant blocks. Each circulant block is fully determined by one row, or equivalently by a polynomial in $\mathbb{F}_q[x]/\langle x^\ell - 1\rangle$. This allows QC matrices to be represented as polynomials over this quotient ring, reducing storage requirements and supporting efficient arithmetic.

The density of the matrices defining the code is also a relevant design parameter. Here, density refers to the weight of rows, columns, or circulant blocks of the generator matrices (GMs) or parity-check matrices (PCs), not to the weight of arbitrary codewords. Low-density (LD) constructions, such as LDPC or LDGM-type codes, use matrices with very small row and column weights, often constant or logarithmic in the code length $n_0\ell$. Moderate-density constructions use larger but still sub-linear weights, typically in the $\Theta(\sqrt{n})$ regime, or equivalently in the corresponding $\Theta(\sqrt{\ell})$ regime when n_0 is fixed and ℓ grows.

A Quasi-Cyclic Moderate-Density Generator Matrix (QC-MDGM) code is a q-ary linear code whose generator matrix combines a QC block-circulant structure with moderate row or block density. Compared with LDGM-like constructions, a QC-MDGM generator matrix is less sparse, while remaining much less dense than a uniformly randomly drawn matrix of the same size. QC-MDGM codes are employed to retain the compactness and efficient arithmetic of QC representations while avoiding some of the fragility associated with codes admitting an extremely sparse description.

Code-based computationally hard problems. The foundational computational hardness in code-based cryptography relies on the intractability of the *Syndrome Decoding Problem* (SDP). Let $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ be a parity-check matrix of a linear code, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ a syndrome vector, and $t \in \mathbb{N} \setminus \{0\}$ a target Hamming weight. The search version of SDP asks to find an error vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathtt{wt}(\mathbf{e}) = t$ and $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}$. The corresponding decision problem asks to determine whether such a vector exists. While the decision version in the Hamming metric is known to be NP-complete [17], the search variant is polynomially reducible to it under standard formulations and is assumed to be computationally hard also in the average case. Concretely, security margins are evaluated as the cost of the most efficient non-structural decoding algorithms, namely Information-Set Decoding (ISD) and its modern variants [10,37,44,45,48].

A closely related formulation is the *Codeword Finding Problem* (CFP), which represents the zero-syndrome variant of SDP. Given a parity-check matrix $\mathbf{H}$ and a target weight t, CFP requires finding a nonzero codeword $\mathbf{c} \in \mathbb{F}_q^n$ in the kernel of $\mathbf{H}$ such that $\mathtt{wt}(\mathbf{c}) = t$. The decision version of CFP is also NP-complete, and its search variant is largely employed in code-based cryptography.

2.2 Code-Based Hash-and-Sign Paradigms

The hash-and-sign approach relies on the inversion of a public deterministic map $\mathsf{F} : \mathcal{X} \to \mathcal{Y}$ whose security depends on an underlying hard problem. While the verification key describes F, the signing key contains a secret trapdoor that allows the legitimate signer to compute or sample preimages. To sign a message $m \in \{0,1\}^*$, the signer determines a target $y = \mathsf{Hash}(m) \in \mathcal{Y}$, optionally incorporating a random salt as $y = \mathsf{Hash}(m \parallel \mathsf{salt})$, and uses the trapdoor to output a preimage $\sigma \in \mathcal{X}$ satisfying $\mathsf{F}(\sigma) = y$. Verification evaluates $\mathsf{F}(\sigma)$ and checks if the result equals y, together with any weight or structural constraints on σ.

In a code-based setting, F is typically instantiated as: $\mathsf{F}_{\mathbf{H}} : \mathbf{e} \longmapsto \mathbf{H}\mathbf{e}^{\mathsf{T}} = \mathbf{s}$, where $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ is a public parity-check matrix, $\mathbf{s}$ is the target syndrome, and the domain is restricted to vectors $\mathbf{e}$ of a target weight or prescribed structure. Without knowledge of the trapdoor, recovering $\mathbf{e}$ reduces to a decoding problem. This paradigm encompasses two main design methodologies.

Trapdoor Inversion (CFS-style). Introduced in the classical Courtois-Finiasz-Sendrier (CFS) framework [23], this approach utilizes a secret bounded-distance decoder. Since decodable syndromes are extremely sparse for the high-rate codes used in CFS, the signer repeatedly hashes the message with an incremented counter until a decodable syndrome is found. This restricts CFS-like schemes to highly structured codes, introduces structural risks, and leads to high signing latencies and large public keys.

Trapdoor Sampling (GPV-style). Modern designs adapt the framework introduced by Gentry-Peikert-Vaikuntanathan (GPV) [32] to coding theory, moving from deterministic inversion to randomized preimage sampling. The signer samples a valid error vector $\mathbf{e}$ matching the target syndrome from a distribution intended to hide the trapdoor. The goal is to make the output signature distribution independent, or sufficiently independent, of the secret trapdoor. Two notable instantiations of this methodology are *Wave* [26] and *MIRANDA* [25], that provide small signatures at the cost of public keys in the order of megabytes for the proposed parameters, and computationally heavy signing procedures.

Fiat-Shamir signatures from identification protocols (such as CROSS [8] and LESS [9]) can use conservative public instances and avoid embedded trapdoors, but they carry a transcript overhead due to the transmission of commitments, challenges, and responses across multiple rounds. Conversely, hash-and-sign schemes can achieve very compact signatures, since the output is essentially a constrained preimage vector, and verification can be as simple as a matrix-vector multiplication together with a weight or structure check. Their

KEYGEN():	SIGN(sk,m):	VERIFY(pk,m,$\boldsymbol{\sigma}$):
1 $\mathsf{seed} \leftarrow \{0,1\}^\lambda$	1 $\mathtt{r} \leftarrow \{0,1\}^\lambda$	1 $(\mathbf{u}, \mathtt{r}) := \boldsymbol{\sigma}$
2 $\mathbf{G} :=$ QC-MDGM(seed)	2 $\mathtt{h} := \mathcal{H}(m \Vert \mathtt{r})$	2 $\mathtt{h} := \mathcal{H}(m \Vert \mathtt{r})$
// $\mathbf{G} = [\mathbf{G_0} \vert ... \vert \mathbf{G_{n_0-1}}]$	3 $\mathbf{v} :=$ SAMPLEVEC($\mathtt{h}$)	3 $\mathbf{v} :=$ SAMPLEVEC($\mathtt{h}$)
3 **for** $i := 1$ **to** $n_0 - 1$ **do**	4 $\mathbf{u} :=$ FULLDISTDEC($\mathbf{G}, \mathbf{v}$)	4 $\mathbf{c} := \mathbf{M}^{\mathrm{T}}\mathbf{u}$
4 $\quad \mathbf{M}_i := \mathbf{G_0}^{-1}\,\mathbf{G_i}$	5 $\boldsymbol{\sigma} := (\mathbf{u}, \mathtt{r})$	
5 $\mathbf{M} := [\mathbf{I} \mid \mathbf{M}_1 \vert ... \vert \mathbf{M}_{n_0-1}]$		
6 $\mathsf{pk} := \{\mathbf{M}\}$		5 **if** $\mathtt{wt}(\mathbf{v} - \mathbf{c}) = pn_0$ **then**
7 $\mathsf{sk} := \{\mathbf{G}\}$		6 $\quad$ **return** 1 // ok
8 **return** $(\mathsf{pk}, \mathsf{sk})$	6 **return** $\boldsymbol{\sigma}$	7 **return** 0 // ko

Fig. 1: ASTRA-Sign primitives: KEYGEN, SIGN, VERIFY.

main challenge lies in constructing a compact, efficient, and non-leaking trapdoor. The goal of ASTRA is to improve this latter design point by combining a quasi-cyclic representation with moderate-density generator matrices and a trapdoor decoding procedure for full-weight errors.

3 Design and Specification of ASTRA

In this section, we describe the ASTRA signature scheme defined by three primitives, KEYGEN, SIGN, and VERIFY, of which we report the algorithms in Figure 1.

The KEYGEN procedure starts by uniformly sampling a λ-bit binary string $\mathtt{seed}$ (line 1). The seed is employed to derive a QC-MDGM generator matrix $\mathbf{G} \in \mathbb{F}_q^{p \times pn_0}$ by expanding it deterministically with a Cryptographically Secure Pseudorandom Number Generator (CSPRNG) in the QC-MDGM procedure (line 2). The generator matrix $\mathbf{G}$ is built through tiling n_0 sparse circulant blocks of size $p \times p$, each with column Hamming weight equal to a fixed value $v \in \Theta(\sqrt{pn_0})$: $\mathbf{G} = [\mathbf{G_0} \vert \mathbf{G_1} \vert \ldots \vert \mathbf{G_{n_0-1}}] \in \mathbb{F}_q^{p \times pn_0}$. The expansion via CSPRNG is done obtaining the positions and values of the $n_0 v$ non null elements of the first row of the QC-MDGM matrix $\mathbf{G}$ via rejection sampling. The systematic form generator matrix of the same code, $\mathbf{M} = [\mathbf{I} \vert \mathbf{M_1} \vert \ldots \vert \mathbf{M_{n_0-1}}] \in \mathbb{F}_q^{p \times pn_0}$ is computed by multiplying each block of $\mathbf{G}$ by the inverse of the first block $\mathbf{G_0^{-1}}$ (lines 3–4). An efficient sampler for an invertible circulant block is obtained by choosing its width p as a prime such that 2 generates the multiplicative group of $\mathbb{F}_p$, and ensuring that the polynomial $\mathsf{g_0}(x)$ in $\mathbb{F}_q[x]/\langle x^p - 1\rangle$ is such that the sum of its coefficient is non-null, i.e., such that $\mathsf{g_0}(1) \neq 0$ [11]. The KEYGEN procedure ends by returning the private key $\mathsf{sk} = \mathbf{G}$ and public key $\mathsf{pk} = \mathbf{M}$ (lines 6–8).

The SIGN algorithm takes as input the private key sk and a message $m \in \{0,1\}^*$ taking the form of an arbitrary binary string to be signed. First, the message is concatenated with a random λ-bit salt $\mathtt{r}$ (line 1), and the resulting bitstring is hashed to produce a digest $\mathtt{h} = \mathcal{H}(m \parallel \mathtt{r})$ via a cryptographic hash function $\mathcal{H}$ having digest length at least 2λ (line 2). This digest acts as the seed for the SAMPLEVEC routine, which employs a CSPRNG to output a pseudo-

Algorithm 4: FullDistDec($\mathbf{G}, \mathbf{v}$) Algorithm

Input: $\mathbf{G}$: $p \times pn_0$ sparse generator matrix over $\mathbb{F}_q$; $\mathbf{v}$: random vector in $\mathbb{F}_q^{pn_0}$
Output: $\mathbf{u}$: information word s.t. $\mathbf{u}^\mathsf{T}\mathbf{G}$ is at full distance from $\mathbf{v}$

1 $\beta \leftarrow \mathbb{F}_q^*$; $a \leftarrow \{0, \ldots, p-1\}$
2 $\mathbf{G} := \beta x^a \cdot \mathbf{G}$
3 **repeat**
4 $\mathbf{u}' \leftarrow \mathbb{F}_q^p$; $\mathbf{c} := \mathbf{G}^\mathsf{T}\mathbf{u}'$
5 **for** $\mathtt{it} := 1$ **to** $\mathtt{iterMax}$ **do**
6 **for** $i := 0$ **to** $p-1$ **do**
7 $\mathtt{ctr} := \mathbf{0}_{1\times q}$
8 **for** $j := 0$ **to** $pn_0 - 1$ **do**
9 **if** $\mathbf{G}_{i,j} \neq 0$ **then**
10 $\alpha := (\mathbf{v}_j - \mathbf{c}_j)\mathbf{G}_{i,j}^{-1}$ // $\alpha \in \mathbb{F}_q$
11 $\mathtt{ctr}[\alpha] \leftarrow \mathtt{ctr}[\alpha] + 1$
12 $\alpha^* := \arg\min_i\{\mathtt{ctr}[i]\}$ // $\alpha \in \mathbb{F}_q$
13 $\mathbf{c} := \mathbf{c} + \alpha^* \cdot (\mathbf{G}_{i,:})^\mathsf{T}$
14 **if** $\mathtt{wt}(\mathbf{v} - \mathbf{c}) = pn_0$ **then**
15 **break**
16 **until** $\mathtt{wt}(\mathbf{v} - \mathbf{c}) = pn_0$
17 $\mathbf{u} := [\mathbf{c}_j]_{0\leq j<p}$ // first p elements of $\mathbf{c}$
18 **return** $\mathbf{u}$

random target vector $\mathbf{v} \in \mathbb{F}_q^{pn_0}$ by means of performing rejection sampling on its coordinates (line 3). The Sign algorithm then proceeds to find a codeword of the code generated by $\mathbf{G}$ at full Hamming distance from $\mathbf{v}$, i.e., such that $\mathtt{wt}(\mathbf{v} - \mathbf{c}) = pn_0$, exploiting the knowledge of the private key $\mathbf{G}$. To do so, the FullDistDec procedure (line 4, described in Section 3.1) leverages the knowledge of the sparse private matrix $\mathbf{G}$. The output of FullDistDec is the p elements long information word $\mathbf{u}$ such that its image through the systematic generator matrix $\mathbf{M}$ (acting as the public key) is the codeword $\mathbf{c}$, i.e. $\mathbf{M}^\mathsf{T}\mathbf{u} = \mathbf{c}$. The signature $\boldsymbol{\sigma}$ for the message m returned by the Sign algorithm (line 6) is the pair consisting of the information word $\mathbf{u}$ and the seed $\mathtt{r}$, $\boldsymbol{\sigma} = (\mathbf{u}, \mathtt{r})$ (line 5).

The verification routine, Verify, checks that the signature is valid, i.e. that the information word $\mathbf{u}$ is mapped by $\mathbf{M}$ into a codeword $\mathbf{c}$ at Hamming distance pn_0 from the pseudorandomly sampled vector $\mathbf{v}$, starting from the random seed $\mathtt{r}$ and the message $\mathbf{m}$. To do so, given m and $\boldsymbol{\sigma} = (\mathbf{u}, \mathtt{r})$, it recomputes the $\mathbf{v}$ from m and $\mathtt{r}$ (lines 1–2), computes the codeword $\mathbf{c}$ as $\mathbf{c} = \mathbf{M}^\mathsf{T}\mathbf{u}$ (line 3), and checks that $\mathtt{wt}(\mathbf{v} - \mathbf{c}) = \mathtt{wt}(\mathbf{v} - \mathbf{M}^\mathsf{T}\mathbf{u}) = pn_0$.

3.1 The FullDistDec Procedure

The signature primitive of ASTRA relies on an efficient solver which inverts a function $f_\mathbf{M}$ parametric in a systematic form generator matrix $\mathbf{M}$ of a QC-MDGM code, employing a low weight generator matrix $\mathbf{G}$ representing the same code. In particular, the function $f_\mathbf{M}$ takes an information word $\mathbf{u}$ and a full

Hamming weight error $\mathbf{e}$ and computes the error affected codeword $\mathbf{v} = f_{\mathbf{M}}(\mathbf{u}, \mathbf{e})$:

$$f_{\mathbf{M}} : \mathbb{F}_q^p \times \{\mathbf{x} \in \mathbb{F}_q^{pn_0} | \mathtt{wt}(\mathbf{x}) = pn_0\} \to \mathbb{F}_q^{pn_0}, \quad f_{\mathbf{M}}(\mathbf{u}, \mathbf{e}) := \mathbf{M}^{\mathsf{T}}\mathbf{u} + \mathbf{e}$$

The SIGN procedure must find a valid error vector $\mathbf{e}$ of full Hamming weight ($\mathtt{wt}(\mathbf{e}) = pn_0$) and an information word $\mathbf{u}$ such that $f_{\mathbf{M}}(\mathbf{u}, \mathbf{e}) = \mathbf{M}^{\mathsf{T}}\mathbf{u} + \mathbf{e} = \mathbf{v}$. Equivalently, this requires to find a codeword $\mathbf{c}$ generated by $\mathbf{M}$ such that $\mathbf{c}$ lies at full Hamming distance from $\mathbf{v}$. Since the public matrix $\mathbf{M}$ and the private sparse matrix $\mathbf{G}$ generate the exact same code $\mathcal{C}$, finding a codeword of the code represented by $\mathbf{G}$ is equivalent, but the sparsity of $\mathbf{G}$ makes the problem computationally tractable, similarly to the case where one desires to decode a sparse parity check matrix code (e.g., QC-LDPC, QC-MDPC), via an iterative decoder. Indeed such decoding strategy relies on a sparse parity check matrix of the code to perform efficient decoding, as it is computationally hard to do so when given only the systematic form of the parity check matrix itself [6].

The FULLDISTDEC procedure, shown in Algorithm 4, operates as an iterative, fixed point decoder, as follows. First, it applies a random multiplication by a scalar $\beta \in \mathbb{F}_q$ and a random rotation by a positions, $0 \leq a < p$, to the circulant blocks composing the sparse QC generator matrix (lines 1–2): this avoids any bias caused by specific *multiples* or specific *rotations* of the private key (the output distribution of the procedure is discussed in the next section). Then, it draws a random information word $\mathbf{u}$ and it computes the corresponding codeword $\mathbf{c} = \mathbf{G}^{\mathsf{T}}\mathbf{u}'$ (line 4), which acts as a starting point. The goal is to update $\mathbf{c}$ to increase its Hamming distance from the target $\mathbf{v}$, until $\mathtt{wt}(\mathbf{v} - \mathbf{c}) = pn_0$. The decoder iterates over the rows of $\mathbf{G}$ (i.e., the set of known sparse codewords of $\mathcal{C}$), computing, for each one of them, an array `ctr` that is initialized to zero and meant to be indexed with elements of the field $\mathbb{F}_q$ (line 5). The rationale behind the computation of `ctr` (lines 7–11) is to quantify the number of coordinates in which $\mathbf{v}$ and $\mathbf{c} + \alpha \cdot (\mathbf{G}_{i,:})^{\mathsf{T}}$ have the same value, for every value of $\alpha \in \mathbb{F}_q$. The element $\mathtt{ctr}[\alpha]$ of the vector `ctr` stores such element, i.e., $\mathtt{ctr}[\alpha] = n_0 p - \mathtt{wt}(\mathbf{v} - (\mathbf{c} + \alpha \cdot (\mathbf{G}_{i,:})^{\mathsf{T}}))$. As a consequence of the definition of `ctr`, we have that $\mathtt{ctr}[0]$ simply corresponds to the number of coordinates having the same value in $\mathbf{v}$ and the current codeword $\mathbf{c}$. The decoder exploits the fact that if the value of $\mathtt{ctr}[\alpha^*]$ is smaller than $\mathtt{ctr}[0]$ for a given non null field element α^*, then the number of coordinates with the same value between $\mathbf{v}$ and $(\mathbf{c} + \alpha^* \cdot (\mathbf{G}_{i,:})^{\mathsf{T}})$ is lower than the one between $\mathbf{v}$ and $(\mathbf{c} + 0 \cdot (\mathbf{G}_{i,:})^{\mathsf{T}}) = \mathbf{c}$. This implies that adding $\alpha^* \cdot (\mathbf{G}_{i,:})^{\mathsf{T}}$ to $\mathbf{c}$ reduces the number of matching coordinates between the codeword and the target $\mathbf{v}$, i.e. it computes a codeword with increased Hamming distance from $\mathbf{v}$. Willing to maximize this effect, the decoder picks α^* as the index of the minimum valued element of `ctr` (line 12). Note that, whenever α^* is 0 and thus the codeword $\mathbf{c}$ is not updated, no addition of a row of $\mathbf{G}$ can increase the Hamming distance between $\mathbf{c}$ and $\mathbf{v}$, and thus $\mathbf{c}$ is not modified. Whenever multiple coefficients achieve the smallest value in `ctr` (meaning that $\arg\min_i$ is not uniquely defined), we opt as a tie-breaking criterion the selection of the *smallest non-zero* value of α^*, where the ordering is trivially defined over $\mathbb{Z}$ for values in $\mathbb{F}_q$ when q is a prime number, but can also be

determined for other finite fields. In the case where the process fails to compute a full distance codeword after a bounded number of iterations (`iterMax`) of the loop at line 5, the decoder restarts the procedure by drawing a different starting point. The decoder repeats the random starting information word sampling process and the decoding attempt until a codeword matching the desired distance is found (loop enclosing lines 3–16). Once the codeword has been found, Algorithm 4 obtains the corresponding information word under the encoding defined by the public generator matrix $\mathbf{M}$. Since the public key $\mathbf{M}$ is in systematic form, the information word $\mathbf{u}$ that maps to the selected codeword $\mathbf{c}$ amounts to the first p coordinates of the vector $\mathbf{c}$ (line 17).

If the rows of the generator matrix $\mathbf{G}$ are sparse enough, then the process eventually converges to a codeword $\mathbf{c}$ at full Hamming distance from $\mathbf{v}$. We note that selecting cryptosystem parameters such that the decoder has a good decoding performance (i.e., a small probability of decoding failures), will yield a more efficient signature algorithm. We handle decoding failures by restarting the process from a different $\mathbf{u}'$ in the loop at lines 3–16, while the side channel security implications of the early termination at line 14 are discussed in Section 6.3.

We now provide an intuitive argument for why this decoder works when $\mathbf{G}$ is sparse, but cannot be applied to a dense representation of the same code. We do so by deriving an approximate probabilistic model for the values in the `ctr` vector, and showing how the resulting decoder behaves under different parameter regimes. Consider the case where the distance between $\mathbf{v}$ and the current codeword $\mathbf{c}$ is $\mathtt{wt}(\mathbf{v}-\mathbf{c}) = \delta$ (where the goal is to obtain $\delta = n_0 p$). The distribution of $\mathtt{ctr}[0]$ when a row $\bar{\mathbf{g}}$ of weight $w = n_0 v$ of $\mathbf{G}$ is being considered is $\Pr_\delta(\mathtt{ctr}[0] = x) = \binom{\delta}{n_0 v - x}\binom{n_0 p - \delta}{x} / \binom{n_0 p}{n_0 v}$. On the other hand, when considering a scalar multiple $\alpha \neq 0$ of such codeword, the probability that $\mathbf{v}_j - (\mathbf{c}_j + \alpha \cdot \bar{\mathbf{g}}_j) = 0$ for any non-zero coordinate j is $\mathtt{p}_{0|\delta} := (1 - \frac{\delta}{n_0 p}) + (\frac{\delta}{n_0 p})\frac{q-2}{q-1}$, and the distribution of $\mathtt{ctr}[\alpha]$ can thus be approximated as $\Pr_\delta(\mathtt{ctr}[\alpha] = x, \alpha \neq 0) = \binom{n_0 v}{x}\mathtt{p}_{0|\delta}^x(1 - \mathtt{p}_{0|\delta})^{n_0 v - x}$. Now, the probability that $\arg\min_i\{\mathtt{ctr}[i]\} = 0$ is:

$$\sum_{x=0}^{n_0 v} \Pr_\delta(\mathtt{ctr}[0] = x) \left(\Pr_\delta(\mathtt{ctr}[\alpha] > x, \alpha \neq 0) \right)^{q-1} \tag{1}$$

On the other hand, the probability that the smallest value in `ctr` is equal to $\mathtt{ctr}[0] - m$, meaning that in line 13 of Algorithm 4 the distance between $\mathbf{v}$ and $\mathbf{c}$ gets increased by m, is:

$$\sum_{x=m}^{n_0 v} \Pr_\delta(\mathtt{ctr}[0] = x) \left[\left(\Pr_\delta(\mathtt{ctr}[\alpha] \geq x - m, \alpha \neq 0) \right)^{q-1} - \left(\Pr_\delta(\mathtt{ctr}[\alpha] > x - m, \alpha \neq 0) \right)^{q-1} \right] \tag{2}$$

Starting from the distribution of the initial distance δ_0 between $\mathbf{v}$ and $\mathbf{c} = \mathbf{G}^\intercal \mathbf{u}'$ ($\binom{n_0 p}{\delta_0}\left(\frac{q-1}{q}\right)^{\delta_0}\left(\frac{1}{q}\right)^{n_0 p - \delta_0}$), the decoding procedure can be modeled via a Markov chain, whose state is the distance between $\mathbf{v}$ and $\mathbf{c}$, and the transition probabilities are determined by Equation 1 and Equation 2. The probability $\mathtt{p}_{\mathtt{sign}}$ of obtaining a valid signature after $\leq p \cdot \mathtt{iterMax}$ iteration of the inner loop is the

probability that $\delta = n_0 p$ after $p \cdot \texttt{iterMax}$ steps of such Markov process. Figure 2 shows the value of $\texttt{p}_{\texttt{sign}}$ for instances of Algorithm 4 employing QC-MDGM codes with different codeword weights $w = n_0 v$. The dashed lines delimit the range for which solving the codeword finding problem (i.e., finding codewords of weight w starting from the public key) is computationally infeasible, while the dotted line indicates the Gilbert-Varshamov bound of random QC codes with the same length and rate [22], corresponding to the weight of the hardest instances of codeword finding. As it can be seen, for $w = 132$, the probability of finding a solution to the full distance decoding problem is close to 1, and becomes exponentially lower for larger weights. Starting from the public representation of the QC-MDGM code, an attacker can derive in polynomial time codewords with weight at least $w \approx 2100$, which is however insufficient for solving the problem via iterative decoding. We discuss more efficient attacking techniques in Section 6.2.

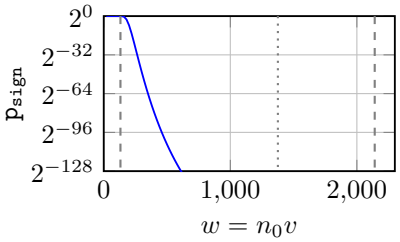


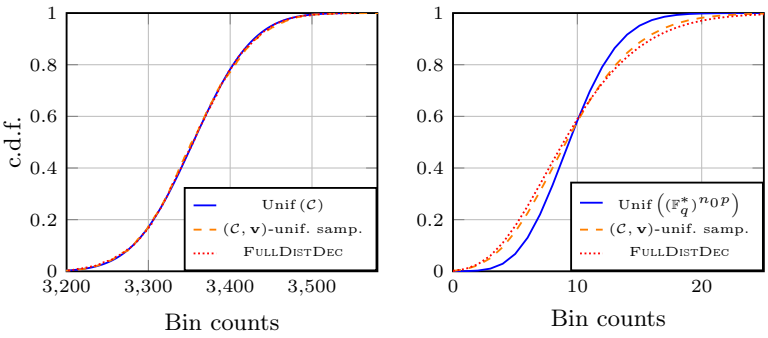
Fig. 2: Probability $\texttt{p}_{\texttt{sign}}$ for QC-MDGM codes over $\mathbb{F}_{23}$ with $n_0 = 2$ (i.e., rate $\frac{1}{2}$), $p = 2237$, $\texttt{iterMax} = 3$, and different codeword weights $w = n_0 v$.

4 Output Distribution of FullDistDec

The primary security requirement for the FullDistDec procedure is to ensure that output signatures do not reveal information about the sparse codewords in the private key $\mathbf{G}$. We begin with an analysis on the expected output distribution of an ideal uniform trapdoor sampler, that we will denote as $(\mathcal{C}, \mathbf{v})$-*uniform sampler*, i.e., one that, given $\mathbf{v}$, returns a codeword $\mathbf{c}$ of the code $\mathcal{C}$ extracted uniformly at random from the set of solutions to the decoding problem. Then, we analyze the output distribution of FullDistDec for small parameters, where we are able to observe experimentally the whole distribution of the output codewords $\mathbf{c}$ and error vectors $\mathbf{e}$. Finally, we discuss potential leakages in the output samples, providing evidence that the output distribution of FullDistDec resists statistical attacks for cryptographic-grade parameters.

Given a fixed QC-MDGM code $\mathcal{C}$ over $\mathbb{F}_q$ and a random vector $\mathbf{v} \in \mathbb{F}_q^{n_0 p}$, let $S_{\mathcal{C}}(\mathbf{v}) = \{\mathbf{c} \in \mathcal{C} | \texttt{wt}(\mathbf{v} - \mathbf{c}) = n_0 p\}$ be the set of codewords which are solutions of this instance, and let $T_{\mathcal{C}}(\mathbf{v}) = \left\{\mathbf{e} \in (\mathbb{F}_q^*)^{n_0 p} | \mathbf{v} - \mathbf{e} \in \mathcal{C}\right\}$ be the set of differences (error vectors) between $\mathbf{v}$ and each one of the solutions in $S_{\mathcal{C}}(\mathbf{v})$. Moreover, let $\mathcal{X}$ be the random variable corresponding to the input to a $(\mathcal{C}, \mathbf{v})$-uniform sampler, and $\mathcal{Y}$ be the random variable corresponding to the codeword returned by the $(\mathcal{C}, \mathbf{v})$-uniform sampler. We now prove the following.

Lemma 1. *Given a fixed QC-MDGM code $\mathcal{C}$, the set of codewords $\mathbf{c}$ returned by a $(\mathcal{C}, \mathbf{v})$-uniform sampler, queried with random vectors $\mathbf{v}$ uniformly drawn*



(a) Codewords $\mathbf{c}$ (b) Errors $\mathbf{e}$

Fig. 3: Cumulative distribution function the repetition in occurrences of vectors given by 10^7 output samples of a $(\mathcal{C}, \mathbf{v})$-*uniform sampler* and FullDistDec, employing a QC-MDGM code over $\mathbb{F}_5$ with $n_0 = 2$, $p = 5$, $v = 2$, compared with the expected c.d.f. of the uniform distribution over the same domain with the same amount of samples. Figure (a): c.d.f. for the output codewords $\mathbf{c}$ over $\mathbb{F}_5^p$. Figure (b) c.d.f. for the output errors $\mathbf{e} = \mathbf{v} - \mathbf{c}$ over $(\mathbb{F}_5^*)^{n_0 p}$.

from $(\mathbb{F}_q)^{n_0 p}$ *is uniform over* $\mathcal{C}$ *(i.e.,* $\mathcal{Y} \sim \mathit{Unif}(\mathcal{C})$*). On the other hand, the distribution of the error vectors* $\mathbf{e} = \mathbf{v} - \mathbf{c}$ *is not uniform over* $(\mathbb{F}_q^*)^{n_0 p}$ *(i.e.,* $\mathcal{X} - \mathcal{Y} \nsim \mathit{Unif}\big((\mathbb{F}_q^*)^{n_0 p}\big)$*).*

Proof. By definition of $(\mathcal{C}, \mathbf{v})$-uniform sampler, we have that:

$$\Pr(\mathcal{Y} = \mathbf{c} \mid \mathcal{X} = \mathbf{v}) = \begin{cases} \frac{1}{|S_{\mathcal{C}}(\mathbf{v})|} & \text{if } \mathbf{c} \in S_{\mathcal{C}}(\mathbf{v}) \\ 0 & \text{if } \mathbf{c} \notin S_{\mathcal{C}}(\mathbf{v}) \end{cases}$$

Therefore, the probability that a specific codeword $\mathbf{c}$ is returned is:

$$\Pr(\mathcal{Y} = \mathbf{c}) = \frac{1}{q^{n_0 p}} \sum_{\mathbf{v} \mid \mathbf{c} \in S_{\mathcal{C}}(\mathbf{v})} \frac{1}{|S_{\mathcal{C}}(\mathbf{v})|} = \frac{1}{q^{n_0 p}} \sum_{\mathbf{e} \in (\mathbb{F}_q^*)^{n_0 p}} \frac{1}{|S_{\mathcal{C}}(\mathbf{c} + \mathbf{e})|}$$

Now, we have that for every $\mathbf{c_0} \in \mathcal{C}$:

$$\mathbf{c} \in S_{\mathcal{C}}(\mathbf{v}) \Leftrightarrow \mathtt{wt}(\mathbf{v} - \mathbf{c}) = n_0 p \Leftrightarrow \mathtt{wt}((\mathbf{v} + \mathbf{c_0}) - (\mathbf{c} + \mathbf{c_0})) = n_0 p \Leftrightarrow \mathbf{c} + \mathbf{c_0} \in S_{\mathcal{C}}(\mathbf{v} + \mathbf{c_0})$$

Which implies that $|S_{\mathcal{C}}(\mathbf{v})| = |S_{\mathcal{C}}(\mathbf{v} + \mathbf{c_0})|$, and thus:

$$\Pr(\mathcal{Y} = \mathbf{c}) = \frac{1}{q^{n_0 p}} \sum_{\mathbf{e} \in (\mathbb{F}_q^*)^{n_0 p}} \frac{1}{|S_{\mathcal{C}}(\mathbf{c} + \mathbf{e})|} = \frac{1}{q^{n_0 p}} \sum_{\mathbf{e} \in (\mathbb{F}_q^*)^{n_0 p}} \frac{1}{|S_{\mathcal{C}}((\mathbf{c} + \mathbf{c_0}) + \mathbf{e})|} = \Pr(\mathcal{Y} = \mathbf{c} + \mathbf{c_0})$$

The fact that $\Pr(\mathcal{Y} = \mathbf{c}) = \Pr(\mathcal{Y} = \mathbf{c} + \mathbf{c_0})$ for any $\mathbf{c_0} \in \mathcal{C}$ proves that $\mathcal{Y} \sim \mathrm{Unif}(\mathcal{C})$. Moving to the error distribution, we have:

$$\Pr(\mathcal{X} - \mathcal{Y} = \mathbf{e}) = \frac{1}{q^{n_0 p}} \sum_{\mathbf{v} \mid \mathbf{e} \in T_{\mathcal{C}}(\mathbf{v})} \frac{1}{|T_{\mathcal{C}}(\mathbf{v})|} = \frac{1}{q^{n_0 p}} \sum_{\mathbf{c} \in \mathcal{C}} \frac{1}{|T_{\mathcal{C}}(\mathbf{e} + \mathbf{c})|}$$

Table 1: Deviations from Unif $\left((\mathbb{F}_q^*)^{\bar{w}}\right)$ for different subsets of coordinates. *Overlap* indicates the largest intersection between a coordinate subset of size $\bar{w}$ and the support of any weight-w codeword ($w = n_0 v$) in the QC-MDGM matrix $\mathbf{G}$. Code parameters: $q = 5$, $n_0 = 2$, $p = 5$, $v = 2$, $w = n_0 v = 4$.

Subset Size $\bar{w}$	**Overlap**	χ^2**-divergence**	**Total Variation**
2	1	0.000136	0.004684
	2	0.001413	0.015758
3	2	0.002134	0.018903
	3	0.007150	0.035204
4	2	0.005199	0.028983
	3	0.009929	0.040555
	4	0.021289	0.060309

Since $T_{\mathcal{C}}(\mathbf{e}+\mathbf{c})\left\{\tilde{\mathbf{e}} \in (\mathbb{F}_q^*)^{n_0 p} | \mathbf{e}+\mathbf{c}-\tilde{e} \in \mathcal{C}\right\} = \left\{\tilde{\mathbf{e}} \in (\mathbb{F}_q^*)^{n_0 p} | \mathbf{e}-\tilde{\mathbf{e}} \in \mathcal{C}\right\} = T_{\mathcal{C}}(\mathbf{e})$, we obtain $\Pr(\mathcal{X}-\mathcal{Y}=\mathbf{e}) = \frac{1}{q^{(n_0-1)p}\cdot|T_{\mathcal{C}}(\mathbf{e})|}$. The set $T_{\mathcal{C}}(\mathbf{e})$ does not necessarily have the same size for different values of $\mathbf{e}$, meaning that the distribution of $\mathcal{X}-\mathcal{Y}$ is not uniform over $(\mathbb{F}_q^*)^{n_0 p}$. □

Figure 3 shows the result of a batch of experiments, performed by collecting 10^7 output samples of FullDistDec with a QC-MDGM code $\mathcal{C}$ over $\mathbb{F}_5$ with $n_0 = 2$, $p = 5$, $v = 2$, and 10^7 output samples of a $(\mathcal{C}, \mathbf{v})$-*uniform sampler* that we implemented for the aforementioned small code by enumerating all valid solutions for each one of the vectors $\mathbf{v} \in (\mathbb{F}_5^*)^{2\cdot 5}$ and sampling uniformly from them. Willing to observe if some output is sampled more or less frequently, therefore potentially giving rise to an information source, we counted in each of the two experiments the number of appearances of each codeword and error vector, and computed the histogram of such appearance counts. The plots show the cumulative probability of obtaining a given bin value on the output codewords $\mathbf{c}$ (Figure 3a) and the derived errors $\mathbf{e} = \mathbf{v} - \mathbf{c}$ (Figure 3b) over their respective domains (i.e., the probability that a codeword or an error vector appears $\leq x$ times out of 10^7). The output codeword distribution of both the $(\mathcal{C}, \mathbf{v})$-uniform sampler and FullDistDec match the uniform distribution over $\mathcal{C}$, while their respective output error distribution deviates from the uniform distribution over $(\mathbb{F}_5^*)^{n_0 p}$, as expected by Lemma 1. Even on such small parameters, the output error distribution of the heuristic procedure FullDistDec matches the one of a $(\mathcal{C}, \mathbf{v})$-uniform sampler extremely closely.

To understand if the non-uniformity of the error distribution of FullDistDec is related to the secret key $\mathbf{G}$, we analyze the distribution of the output error vectors along subsets of coordinates of different sizes $\bar{w}$ (with domain $(\mathbb{F}_q^*)^{\bar{w}}$). Table 1 shows the resulting deviation from the uniform distribution over $(\mathbb{F}_q^*)^{\bar{w}}$ for the same code employed for Figure 3. We separate different coordinate subsets according to the largest overlap with the support of any row of the secret sparse matrix $\mathbf{G}$, and we quantify the deviation from uniformity in terms of

χ^2-divergence ($\sum_x \frac{(P(x)-U(x))^2}{U(x)}$) and Total Variation ($\frac{1}{2}\sum_x |P(x)-U(x)|$). As it can be seen, the largest deviation from uniformity is obtained in coordinate subsets that align with the support of the sparse codewords. This fact suggests two different statistical key recovery attack strategies. In the first, the attacker collects multiple samples from FullDistDec, then start from a random support with size $\bar{w} = w = n_0 v$ to perform a stochastic ascent procedure, where each element of the support is iteratively adapted to maximize the deviation from uniformity along the selected coordinates. In the second one, the attacker collects multiple samples from FullDistDec, then analyze subsets of $\bar{w} < n_0 v$ coordinates (e.g., $\bar{w} = 2$) and reconstruct the distance spectrum of the secret key (e.g., for $\bar{w} = 2$, the set of pairwise distances in one of the secret supports) by enumerating all possible coordinate subsets and retaining the ones with the largest deviation from uniform. The first attack is clearly computationally infeasible for cryptographic-grade parameters, as the domain size of the distribution to be analyzed is $(q-1)^{n_0 v}$. On the other hand, the second attack acts on a much smaller domain, and remains feasible even for larger values of q, p and v.

We now discuss the number of samples Q_s required for the second attack to work. A necessary condition is that Q_s must be large enough to at least *distinguish* the distribution along any subset of $\bar{w}$ coordinates from the uniform distribution over $(\mathbb{F}_q{}^*)^{\bar{w}}$ with non-negligible probability. This can be done efficiently (and optimally) via a coincidence-based test [42], requiring $Q_s \approx \frac{\sqrt{(q-1)^{\bar{w}} \cdot \ln(\frac{1}{\delta})}}{\epsilon^2}$ samples, with ϵ being the Total Variation between the target distribution $P(\cdot)$ and the uniform distribution, and δ the allowed failure rate ($0 < \delta < \frac{1}{2}$). A tighter bound for the number of required samples, in case the target distribution $P(\cdot)$ is not adversarially selected (although still unknown to the attacker), is $Q_s \approx \frac{\sqrt{2(q-1)^{\bar{w}} \cdot \ln(\frac{1}{\delta})}}{\chi^2}$, where χ^2 is the χ^2-divergence w.r.t. the uniform distribution, since $\chi^2 \geq 4\epsilon^2$.

Figure 4 shows the number of samples (in log-scale) required to distinguish the distribution of $\bar{w}$ error coordinates lying in the support of the private key from the uniform distribution. We employed a QC-MDGM code with the smallest parameter set proposed for ASTRA targeting $\lambda = 80$ bits of security (see Table 6), and increasing values of v (and thus $w = 2v$), in order to observe how the distinguishing complexity scales w.r.t. the density of the QC code. As it can be seen, the number of samples required to solve the task with non-negligible advantage (e.g., with success probability $1 - \delta = \frac{2}{3}$ for each possible set of coordinates) grows exponentially as v increases, since the logarithm of such quantity shows a (super) linear trend for every value of $\bar{w} \in \{2, 3, 4, 5\}$ we tested. According to NIST guidelines [39], a signature scheme can achieve EUF-CMA security if 2^{64} signatures are not enough to provide information about the private key. The results reported in Figure 4 suggest that the number of samples required to distinguish the distribution of any subset of $\bar{w}$ coordinates from the uniform distribution over $(\mathbb{F}_q{}^*)^{\bar{w}}$ is much larger than 2^{64} for cryptographic-grade parameters. We provide additional results on different parameter sets in Appendix A.

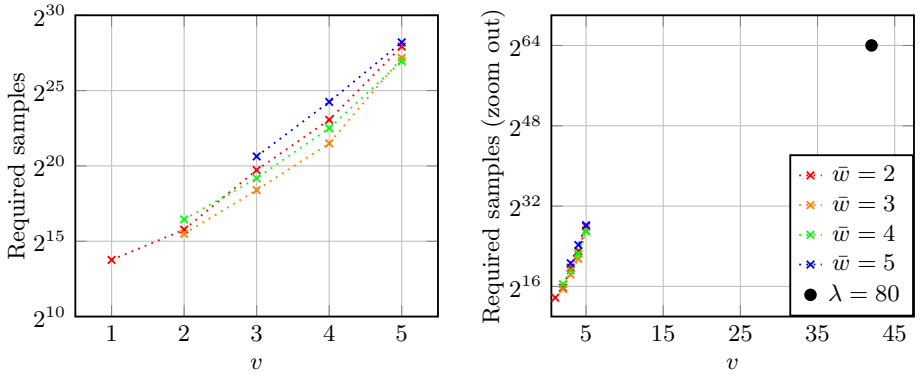


Fig. 4: Number of samples Q_s required to distinguish $\bar{w}$ error coordinates lying in the support of one of the rows of the secret matrix $\mathbf{G}$ with success probability $\frac{2}{3}$, for different values of column weight v. Code parameters: $n_0 = 2$, $p = 827$, $q = 13$, achieving 80b security at $v = 42$. The black marker shows the lower bound on the number of samples required for the scheme to be secure. Data obtained by generating $2 \cdot 10^8$ signatures per data point; y-axis in log-scale.

While experimental data shows that ASTRA is likely secure under statistical attacks on the output of FullDistDec, providing a closed-form proof that the distribution of the generated signatures is computationally indistinguishable from the uniform over the valid preimages remains an open problem for future work. Consequently, the resistance of ASTRA to structural leakage currently relies on an assumption (supported by our numerical experiments) on the output distribution of the sampler, which we define in the following section.

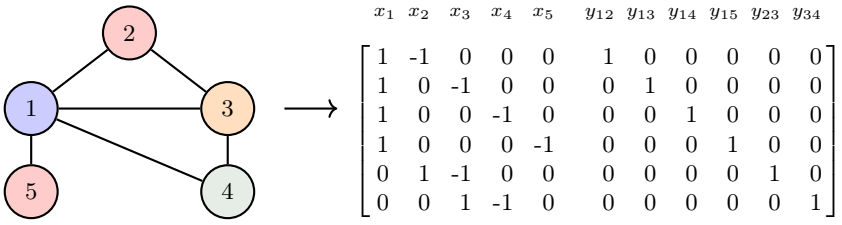
5 Security Analysis

In this section, we first describe the computational problems underlying the hardness of key recovery and signature forgery for ASTRA. Then we describe the set of assumptions behind the security of ASTRA. Finally, we prove that, under such assumptions, ASTRA is EUF-CMA secure.

5.1 Hard Problems

Recovering a private key from the public key of ASTRA amounts to solving the following computational search problem:

Definition 3 (Low Weight Codeword Finding (LWC-F($\mathbf{M}$,q,w))). *Given a generator matrix $\mathbf{M}$ in systematic form, and an integer w, find a vector $\mathbf{c} \in \mathbb{F}_q^{pn_0}$ such that $\mathbf{c}$ is a codeword of the code generated by $\mathbf{M}$ having $\mathtt{wt}(\mathbf{c}) = w$.*



Graph Coloring with $\ell = 4$ **Parity-check matrix H over $\mathbb{F}_5$**

Fig. 5: Reduction from **GC-D**(V,E,k) with $\ell = 4$ to **FDC-D**(**M**,ℓ,**0**). The valid coloring of the graph corresponds, for example, to the codeword $\mathbf{c} = [1,2,3,4,2,\ 1,2,3,1,1,1]^{\mathsf{T}}$ of the code described by the parity-check matrix **H**, where $\mathtt{wt}(\mathbf{c}-\mathbf{0}) = \mathtt{wt}(\mathbf{c}) = n$.

The **LWC-F**(**M**,q,w) problem is NP-hard for arbitrary codes, given that its decision version is NP-complete [18,53], and its hardness has been extended to arbitrary quasi-cyclic codes [16,5] (let **QC-LWC-F**(**M**,q,w) denote the codeword finding problem restricted to quasi-cyclic codes). This problem is the same underlying the security of some code-based KEMs such as BIKE and LEDAcrypt [6,11] and has resisted decades of cryptanalysis.

The hardness of forging a signature in ASTRA reduces to either breaking a cryptographic hash function, or solving the following hard problem:

Definition 4 (Full Distance Codeword Finding (FDC-F(M,q,v))). *Given an arbitrary generator matrix* **M** *over* $\mathbb{F}_q$ *in systematic form, and a vector* $\mathbf{v} \in \mathbb{F}_q^{pn_0}$*, find a vector* $\mathbf{c} \in \mathbb{F}_q^{pn_0}$ *such that* **c** *is generated by* **M** *and* $\mathtt{wt}(\mathbf{v}-\mathbf{c}) = pn_0$.

The **FDC-F**(**M**,q,**v**) problem is equivalent to the Full Weight Syndrome Decoding problem (**FWSD-F**(**M**,q,**v**)). Indeed, solving an instance of **FDC-F**(**M**,q,**v**) amounts to finding a codeword **c** such that $\mathbf{e} = \mathbf{c}-\mathbf{v}$ has full Hamming weight, meaning that $\mathbf{He} = \mathbf{H}(\mathbf{c}-\mathbf{v}) = -\mathbf{Hv}$, where **H** is a parity check matrix of the code described by **M**. Conversely, finding a full weight vector **e** such that $\mathbf{He} = \mathbf{s}$ for a given syndrome **s** and parity check matrix **H** is equivalent to performing the following steps: *i)* finding *any* vector that satisfies $\mathbf{Hx} = -\mathbf{s}$ (using Gaussian elimination); *ii)* finding a codeword **c** at full distance from **x**; *iii)* computing $\mathbf{e} = \mathbf{c} - \mathbf{x}$.

Let **FDC-D**(**M**,q,**v**) denote the decision version of **FDC-F**(**M**,q,**v**), i.e., the problem of determining whether a full Hamming distance codeword exists. We now prove the following.

Theorem 1. ***FDC-D***(**M**,q,**v**) *is* NP*-complete for* $q \geq 4$.

Proof. It is easy to observe that **FDC-D**(**M**,q,**v**) is in NP: given a vector **c**, a deterministic Turing Machine can test if the vector has full Hamming distance from **v**, and that it belongs to the code generated by the systematic matrix **M** (for instance, by performing a multiplication over $\mathbb{F}_q$ between $\mathbf{M}^{\mathsf{T}}$ and the first

k coordinates of $\mathbf{c}$, and checking whether the result equals $\mathbf{c}$). The computation takes polynomial time and a non-deterministic Turing machine can run it on all the possible information words simultaneously.

Let **GC-D**(V,E,ℓ) indicate the decisional graph coloring problem with ℓ colors over a graph (V,E), where $\ell \geq 3$. **GC-D**(V,E,ℓ) asks whether there exists a valid vertex coloring function $\phi : V \to \{1,2,\ldots,\ell\}$ such that, for every edge $\{u,v\} \in E$, it holds that $\phi(u) \neq \phi(v)$. **GC-D**(V,E,ℓ) is a known NP-complete problem [34]. We now show that **GC-D**(M,ℓ) Karp-reduces to **FDC-D**$(\mathbf{M},\ell+a,\mathbf{v})$, which in turn implies that **FDC-D**$(\mathbf{M},\ell+a,\mathbf{v})$ is NP-complete. For the sake of clarity in description, we provide a graphical representation of the reduction in Figure 5.

Consider an instance of the graph coloring decision problem, with input vertexes V, edges E and $\ell \geq 3$ (Figure 5, on the left). We first consider the case where $\ell+1$ is a prime power $q = p^m$, and thus there is a finite field $\mathbb{F}_{\ell+1}$ containing ℓ non-null elements. We construct a parity-check matrix $\mathbf{H}$ having rows with $|V|+|E|$ coordinates (Figure 5, on the right). For each vertex $v \in V$, label a coordinate of the rows of $\mathbf{H}$ as x_v, and for each edge $(u,v) \in E$, label another coordinate of the rows of $\mathbf{H}$ as y_{uv}. All labels must be applied to separate coordinates of the rows, thus univocally labelling the $|V|+|E|$ coordinates. Each row of $\mathbf{H}$ corresponds to an equation $x_u - x_v + y_{uv} = 0$, i.e., it has the coordinates labelled by x_u and y_{uv} set to 1 and the one labelled by x_v set to -1. We can now compute the generator matrix $\mathbf{M}$ of the code which admits $\mathbf{H}$ as a parity check matrix (this computation can be done in polynomial time), and call a solver for **FDC-D**$(\mathbf{M},\ell+1,\mathbf{0})$. If the solution exists, there is a codeword $\mathbf{c}$ which has full distance from the null vector, that is, $\mathbf{c}$ has full (equal to $n_0 p$) Hamming weight. The values of the first $|V|$ coordinates of the codeword $\mathbf{c}$ can be interpreted as the colors to be assigned to the nodes of the graph so that no two adjacent nodes have an equal coloring. Indeed, since $\mathbf{c}$ is a codeword, all the parity check equations in $\mathbf{H}$ are satisfied by its values. This in turn implies that, for each edge (u,v) the colors corresponding to the nodes u and v, i.e., the values x_u, x_v satisfy the equation $x_u - x_v + y_{uv} = 0$, where y_{uv} is non null (as it is labelling a coordinate of $\mathbf{c}$). Since $x_u - x_v + y_{uv} = 0$ and $y_{uv} \neq 0$ imply $x_u \neq x_v$, if **FDC-D**$(\mathbf{M},\ell,\mathbf{0})$ admits a solution, then there is a function $\phi : V \to \{1,2,\ldots,\ell\}$, $\phi(v) = (x_v)_{(\mathbb{F}_q \setminus \{0\}) \to (\mathbb{Z}_q \setminus \{0\})}$, such that $\phi(u) \neq \phi(v)$ for every edge $(u,v) \in E$, meaning that also **GC-D**(V,E,ℓ) has a positive answer. With the same line of reasoning we have that if **FDC-D**$(\mathbf{M},\ell+1,\mathbf{0})$ returns a negative answer, then also **GC-D**(V,E,ℓ) has a negative answer.

To handle the cases where $\ell+1$ is not a prime power, we construct a different decisional graph coloring instance which has a solution if and only if the original one **GC-D**(V,E,ℓ) does. To do so, we construct the instance **GC-D**(V',E',ℓ'), where $\ell' = \ell + a$ and $\ell'+1$ is a prime power p^m. We build V' adding a distinct nodes to V, and we obtain E' from E adding edges from the added nodes to all the others but themselves. This construction can be done in polynomial time, as the value of a is upper bounded by the distance between $\ell+1$ and the successive power of 2, i.e., smaller than 2ℓ. Given the structure of the graph (V',E'), its

coloring will have dedicated colors to the a added nodes due to the fact that they are adjacent to any other node, leaving $\ell' - a = \ell$ colors to be assigned to the nodes which are shared with (V, E). □

The hardness of **FDC-D**($\mathbf{M}$,q,$\mathbf{v}$), and thus of **FDC-F**($\mathbf{M}$,q,$\mathbf{v}$), can be extended to the family of quasi-cyclic codes (**QC-FDC-D**($\mathbf{M}$,q,$\mathbf{v}$)) by employing the same code tiling and coordinate permutation proof technique used in [5,16]. This implies that the search problem **QC-FDC-F**($\mathbf{M}$,q,$\mathbf{v}$) is NP-hard.

5.2 EUF-CMA Security of ASTRA

The EUF-CMA security of ASTRA relies on the following two assumptions.

Assumption 1 *The distribution of public keys (i.e., QC-MDGM generator matrices $\mathbf{M} \in \mathbb{F}_q^{p \times pn_0}$ in systematic form) returned by the* KEYGEN *procedure is computationally indistinguishable from the uniform distribution of QC generator matrices in $\mathbb{F}_q^{p \times pn_0}$ admitting a systematic form.*

Assumption 2 *Consider $Q_s = 2^{64}$ signatures, that contain a codeword $\mathbf{c}$ and allow to compute an error vector $\mathbf{e} = \mathbf{v} - \mathbf{c}$ as the difference between the message-related vector $\mathbf{v}$ and the codeword $\mathbf{c}$, as returned by the* FULLDISTDEC *procedure. The distribution of any subset of $0 \leq \bar{w} \leq n_0 p$ coordinates of $\mathbf{e}$ is computationally indistinguishable from the uniform distribution over $(\mathbb{F}_q^*)^{\bar{w}}$.*

The bound $Q_s = 2^{64}$ is selected according to NIST guidelines [39], and is the same query limit employed in the standardized schemes Falcon [49] and SPHINCS+ [41], along with the hash-and-sign scheme Wave [12]. A direct implication of Assumption 2 is that the output of FULLDISTDEC is computationally indistinguishable from the output distribution of a $(\mathcal{C}, \mathbf{v})$-uniform sampler (i.e., the uniform distribution over the set of valid preimages), since both distributions are consequently computationally indistinguishable from the uniform distribution over $(\mathbb{F}_q^*)^{n_0 p}$. We now prove the following.

Theorem 2. *Under Assumption 1 and Assumption 2, ASTRA is secure in the EUF-CMA model.*

Proof. ASTRA is an instantiation of the GPV framework [32], with the one way function being $f_{\mathbf{M}} : \mathbb{F}_q^p \times \{\mathbf{x} \in \mathbb{F}_q^{pn_0} | \mathtt{wt}(\mathbf{x}) = pn_0\} \to \mathbb{F}_q^{pn_0}$ defined as $f_{\mathbf{M}}(\mathbf{u}, \mathbf{e}) = \mathbf{M}^\intercal \mathbf{u} + \mathbf{e}$, and the trapdoor being a sparse generator matrix $\mathbf{G}$ that allows the usage of the FULLDISTDEC procedure. Under Assumption 2, this setting describes an *average trapdoor preimage sampleable function* [21], meaning that the EUF-CMA security of our scheme can be tightly reduced to the problem of inverting $f_{\mathbf{M}}$ without the trapdoor. Under Assumption 1, computing $f_{\mathbf{M}}^{-1}(\mathbf{v})$ amounts to solving an instance of **QC-FDC-F**($\mathbf{M}$,q,$\mathbf{v}$), as the attacker cannot distinguish a public key of ASTRA $\mathbf{M}$ from a random QC code. □

6 Quantification of Attack Strategies

In this section, we quantify the computational effort of a set of attack approaches that can be applied to the ASTRA signature scheme. We describe procedures aimed at retrieving the private key (key recovery attacks), creating a valid signature without the knowledge of the private key (signature forgery attacks), and highlight potential attack avenues through side channel leakage.

6.1 Key Recovery Attacks

Performing a key recovery attack corresponds to solving an instance of the well-studied low weight codeword finding problem on QC-MDGM codes given a generator matrix in systematic form. The currently most efficient approach to solve such instances is the use of Information Set Decoding (ISD) techniques [57]. The complexity of key recovery is $\frac{\mathtt{C}_{\mathtt{ISD}}(q,pn_0,p,vn_0)}{p}$, where $\mathtt{C}_{\mathtt{ISD}}(q,n,k,t)$ denotes the complexity of an ISD solver over for arbitrary codes over $\mathbb{F}_q$, and the denominator is due to the Decoding One Out of Many (DOOM) [50] speedup induced by the quasi-cyclicity of our codes.

6.2 Signature Forgery Attacks

Forging a signature requires solving the Full Distance Codeword Finding (**QC-FDC-F**($\mathbf{M}$,q,$\mathbf{v}$)) problem. The hardness of this problem (in its syndrome-decoding version) has been previously analyzed in the context of the Wave signature scheme [12], for the case where $q = 3$ (i.e., over the ternary field $\mathbb{F}_3$). For generic larger fields $\mathbb{F}_q$, $q > 3$, it has not been studied as thoroughly, as ours is, to our knowledge, the first cryptographic application of it. We note that, while NP-completeness is a worst-case notion, we do not expect a general polynomial time solver for the problem. We highlight in the following a set of techniques to solve such problem in super-polynomial time.

Prange-style Sampling. Our baseline approach resembles the one of Prange's ISD solver [48], which tries to decode an arbitrary code by guessing enough error-free coordinates of a corrupted codeword to be able to compose an information set. If the guess is correct, the corresponding information word is recovered, and re-encoded; the error is derived by subtracting the obtained codeword from the corrupted one. We adapt the technique by starting from the target vector $\mathbf{v} \in \mathbb{F}_q^{pn_0}$, and choosing an information word $\mathbf{u} \in \mathbb{F}_q^p$ so that p coordinates of the corresponding codeword through the systematic generator matrix $\mathbf{M}$ fit the full-distance constraint with respect to the first p coordinates of $\mathbf{v}$. Indeed, it suffices that the values of the coordinates of $\mathbf{u}$ are element-wise different from the ones of $\mathbf{v}$, as their value under a systematic encoding matches the one of the first coordinates of the codeword $\mathbf{c}$. Then, the remaining $p(n_0 - 1)$ entries of the codeword are computed by encoding $\mathbf{u}$ through $\mathbf{M}$, and checked for the full-distance constraint. Assuming that the codeword symbols behave as pseudo-randomly sampled elements in $\mathbb{F}_q$, the probability that none of them collide with

$\mathbf{v}$ is $\left(\frac{q-1}{q}\right)^{p(n_0-1)}$. Therefore, finding a forgery through this approach requires evaluating $\approx \left(\frac{q}{q-1}\right)^{p(n_0-1)}$ candidates, and the computational cost of evaluating each candidate matches (n_0-1) block multiplication computations (although most attempts can be interrupted after the first multiplication).

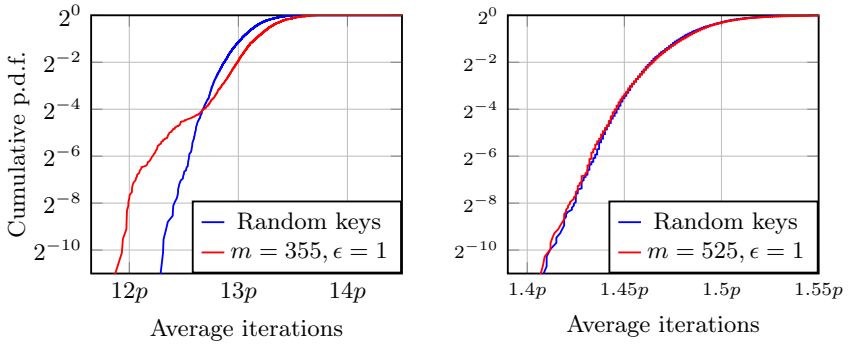
ISD with Early Pruning. The cost of evaluating candidates during random sampling can be improved by doing the following. First, the attacker samples a random codeword $\mathbf{c}$. For each row $\mathbf{G}_{i,:}$ of the generator matrix, the attacker keeps a binary array `val` of length q, initialized to $\texttt{val}[v] = 0$ for $v = \mathbf{v}_i$ and 1 otherwise. The attacker scans the row along each index j $(p \leq j \leq pn_0 - 1)$, updating `val` to rule out scalar multipliers that would cause a collision in the parity section (i.e., setting $\texttt{val}[(\mathbf{v}_j - \mathbf{c}_j) \cdot \mathbf{G}_{i,j}^{-1}] = 0$ if $\mathbf{G}_{i,j} \neq 0$), and stopping when `val` is full of zeros, which happens on average after $q \cdot \ln(q)$ steps. The process is repeated for all the p rows of $\mathbf{G}$, and restarted by drawing another random codeword. Thus, the attacker can effectively test $p(q-1)$ candidates in $(n_0-1)\mathtt{C}_{\texttt{mul}} + p \cdot q \cdot \ln(q)$ steps ($> \ln(q)$ bit operations per candidate), where $\mathtt{C}_{\texttt{mul}}$ denotes the cost of a block-wise multiplication. We very conservatively assume that this approach can be optimized down to 1 bit operation per candidate, leading to a forgery operational cost equal to $\left(\frac{q}{q-1}\right)^{p(n_0-1)}$.

Signature Forgery with MITM Techniques. The random sampling strategy strongly resembles a collision finding problem, which may suggest a Meet-In-The-Middle (MITM) technique or Wagner's Generalized Birthday Algorithm [54] exploiting time-memory trade-offs to reduce the attack runtime. This strategy, however, requires an efficient data structure to check whether, in a collection of vectors in $\mathbb{F}_q^{pn_0}$, at least one has full Hamming distance from a new given vector $\mathbf{c}$. Unlike strict equality (which allows for fast hash-table look-ups), checking for full-distance inequalities incurs either an exponential memory cost for dataset creation, a linear-time complexity per query, or a prohibitive false-negative rate. Therefore, we do not consider MITM and Wagner-style approaches as optimal signature forgery techniques for our scheme.

6.3 Side Channel Attacks

An *active* attacker can attempt to gather information on the private key by measuring the number of iterations (lines 5 and 6 of Algorithm 4) required to produce a signature. We start by noting that decoding failures attacks devised for QC-MDPC code based cryptosystems [33,56] do not apply to our scheme, since the attacker cannot know the vector that the FullDistDec procedure is trying to sign (indeed, while $\mathcal{H}(m\|\mathtt{r})$ and the target $\mathbf{v}$ are public, the starting random codeword $\mathbf{c}'$ is not). The only information an active attacker can gather is the average number of iterations required to compute a valid signature.

Figure 6 shows the average number of iterations required to sign a message by random keys, for two different parameter sets. Between different keys, the fluctuation of the average number of iterations is small. As an additional test,



(a) $n_0 = 2, q = 17, p = 1637, v = 66$ (b) $n_0 = 2, q = 23, p = 2237, v = 66$

Fig. 6: Average number of iterations required to generate a signature for random keys, and for weak keys satisfying the (m, ϵ)-gathering property [56]. Data obtained by generating 10^4 keys and performing 10^3 signature actions per key.

we considered the largest family of known, pathological weak keys in QC code based cryptosystems, namely, the set of matrices satisfying the (m, ϵ)-gathering property [56]. The codes from this family, characterized by having $v - \epsilon$ non-zero entries in one of the circulant blocks of the secret sparse matrix packed in the first m positions, show (in general) poor decoding capabilities, and are numerous enough to enable a key recovery attack if detected. To test if such property holds even for ASTRA, we tested the worst possible scenario, selecting codes with the smallest (m, ϵ) gathering such that they appear with probability at least 2^{-128}. As it can be seen in Figure 6, the cumulative behavior of such codes deviates slightly from the one of random keys, but not enough for them to be detected solely by the average number of signature iterations. Therefore, even if an *active* attacker were to measure such quantity precisely, the amount of leaked information may be considered negligible.

7 Parameters and Performance

In this section we propose parameter sets to instantiate ASTRA-Sign providing λ bits of security against key recovery and signature forgery attacks, for $\lambda \in \{128, 192, 256\}$, meeting the requirements of NIST security levels 1, 3 and 5, respectively, i.e. having the computational effort of breaking the scheme match the one of breaking AES-λ. Then, we provide performance figures from our preliminary C implementation of the scheme, and compare the performance of our scheme against schemes standardized by NIST, schemes that are currently under evaluation by NIST, and other code-based schemes with a publicly available implementation. Finally, we provide additional compact parameter sets for 80-bit security, meant to highlight how far the performance and public key/signature sizes of ASTRA can be pushed for embedded systems scenarios.

Table 2: Values of θ employed for the public key in our implementation.

q	$\log_2(q)$	θ	bits/coeff	overhead
13	3.70	4	3.75	1.33%
17	4.08	7	4.14	1.28%
19	4.24	4	4.25	0.05%
23	4.52	7	4.57	1.02%
29	4.85	1	5.00	2.92%
31	4.95	1	5.00	0.92%
53	5.72	4	5.75	0.52%
61	5.93	1	6.00	1.16%

Table 3: Values of θ employed for the signature in our implementation.

$q-1$	$\log_2(q-1)$	θ	bits/coeff	overhead
12	3.58	3	3.66	2.09%
16	4.00	1	4.00	0.00%
18	4.16	5	4.20	0.72%
22	4.45	4	4.50	0.90%
28	4.80	6	4.83	0.47%
30	4.90	1	5.00	1.89%
52	5.70	4	5.75	0.87%
60	5.90	1	6.00	1.57%

7.1 Parameter Selection and Encoding Strategy

The parameters defining an instance of ASTRA are the security level λ, the number n_0 of circulant blocks in the generator matrix, the size q of the base field $\mathbb{F}_q$, the size p of the circulant blocks, and the Hamming weight v of the rows and columns of each circulant block. From a practical standpoint, free parameters are λ, n_0 and q; once these are chosen, the most convenient values for p and v are the smallest ones such that the computational cost of key recovery and signature forgery exceeds the one to break AES-λ.

Given a security level λ, choosing n_0 and q allows to obtain different trade-offs with respect to public key size, signature size, and computational requirements. Increasing q improves signature time, as it allows the FullDistDec to converge in a smaller number of iterations, but requires an increase in p to thwart signature forgery attacks, in turn enlarging public key and signature sizes. Increasing n_0 makes signature forgery harder, allowing for a reduction in p and signature size at the cost of increasing public key sizes. On the other hand, decreasing the rate $\frac{1}{n_0}$ of the code makes low weight codeword finding (and thus key recovery by an attacker) easier, requiring a corresponding increase in v and consequently in signature time. Willing to provide a first set of parameters, we explored parameter sets with $2 \leq n_0 \leq 7$ and $q \in \{11, 13, 17, 19, 23, 29, 31, 53, 61\}$. We note that ours is not an exhaustive search of the parameter space, and exploring other values of q (e.g., more primes between 31 and 61, or prime powers such as 16 and 32) will yield additional parameter sets with unique performance profiles. We consider a full design space exploration of the scheme as an interesting direction for an engineering future work.

Encoding of Public Keys and Signatures. The public key and the largest element in the signature of ASTRA are composed by polynomials in $\mathbb{F}_q/\langle x^p - 1\rangle$, which can be represented as sequences of elements in $\mathbb{F}_q^p$. Given the values of p at hand, each field element fits in a byte, which would yield a public key size of $(n_0 - 1)p$ bytes. A first boost to efficiency is obtained by bit-packing the coefficient, thus employing only $\lceil \log_2(q) \rceil$ bits per coefficient, lowering the key size to $\left\lceil \frac{p \cdot \lceil \log_2(q) \rceil}{8} \right\rceil$. To further compress the size of the public key, we employ

a combination of arithmetic encoding and bit-packing. Given a value of q, we choose a number of coefficients θ such that $q^\theta \lesssim 2^k$ for some $k \leq 32$. Then, given θ coefficients $\{c_0, c_1, \ldots, c_{\theta-1}\}$, we can represent them with the binary encoding of the value $Q = \sum_{i=0}^{\theta-1} c_i \cdot q^i$, which is then stored using k bits. The reason for picking $k \leq 32$ is that the encoding operations are expected to be single-precision architectural operations on the overwhelming majority of computing platforms. A last optimization concerns the encoding of the signature. Since the information word $\mathbf{u} \in \mathbf{F}_q^p$ has full Hamming distance with the first p entries of $\mathbf{v}$ (that is known to the verifier), we can compute the vector $\mathbf{d} = ([\mathbf{v}_j]_{0 \leq j < p} - \mathbf{u} - \mathbf{1})$ (containing only values between 0 and $q-2$), and encode it exploiting the reduced range of the coefficients, thus reducing signature size. Table 2 and Table 3 show the values of θ we employed for the vectors in the public key and signature, along with an estimate of the amount of bits required for each coefficient: we observe that our techniques allow us to encode the elements with an overhead which is $< 3\%$ w.r.t. the optimal encoding.

7.2 Parameters for ASTRA-Sign

In the following, we report parameter sets for our scheme matching the computational cost of key recovery and signature forgery to be equal or greater than the cost of performing 2^λ AES-λ computations, for λ equal to 128, 192, and 256, respectively. To estimate the computational complexity of key recovery attacks based on ISD techniques (reported in Section 6.1), we employ the public attack complexity estimators available at [31,30], using a logarithmic memory access cost. The complexity of signature forgery has been evaluated with the criterion presented in Section 6.2. We selected parameters exhibiting either small signatures, a small combined size of public key and signature, a competitive signature time, or a balanced performance profile.

Table 4 shows our proposed parameter sets for ASTRA, along with the computational complexities of key recovery and signature forgery attacks, and the size of public keys and signatures. We report sub-kB sizes for $\lambda = 128$ parameter sets for both the signature and the public key, with the possibility of pushing the signature size down to 642B, smaller than all the current NIST standardization selected algorithms, trading off public key size. Regarding the value of `iterMax`, we point out that it has no impact on the security of the scheme, and has only a minor influence on its performance (unless it is set too low). We report in Table 4 the values of `iterMax` we employed for benchmarking.

7.3 Performance Comparison

We compare the performance of our C implementation with existing public implementations of other post-quantum signature schemes. We note that our implementation has some degree of preliminary optimizations for the Intel AVX2 ISA extensions, although we expect that more performance can be gained with a dedicated engineering effort, while the codebases we compare to are, for the

Table 4: Proposed parameter sets providing $\lambda = 128, 192, 256$ security bits, respectively. Bit complexities in equivalent AES-λ runs, $\lambda \in \{128, 192, 256\}$. Public key and signature sizes are in Bytes.

Instance	n_0	q	p	v	`iterMax`	**KRA**	**SFA**	|pk|	|sig|
ASTRA-128	2	17	1,637	66	35	129.23	128.17	848	835
	2	23	2,237	66	7	128.84	128.45	1,279	1,275
	3	31	1,523	73	7	128.62	129.09	1,904	968
	7	53	877	79	23	129.46	129.60	3,786	647
ASTRA-192	2	23	3,253	98	50	193.41	193.61	1,859	1,854
	2	29	4,091	99	20	195.28	192.11	2,557	2,496
	4	61	2,909	115	17	193.59	193.11	6,546	2,206
	5	61	2,213	117	17	192.83	196.09	6,640	1,684
ASTRA-256	2	29	5,387	130	25	257.56	257.72	3,367	3,287
	2	31	5,741	130	15	257.49	256.58	3,589	3,621
	3	53	4,933	146	50	256.92	256.12	7,092	3,578
	4	61	3,797	153	25	257.43	256.63	8,545	2,880

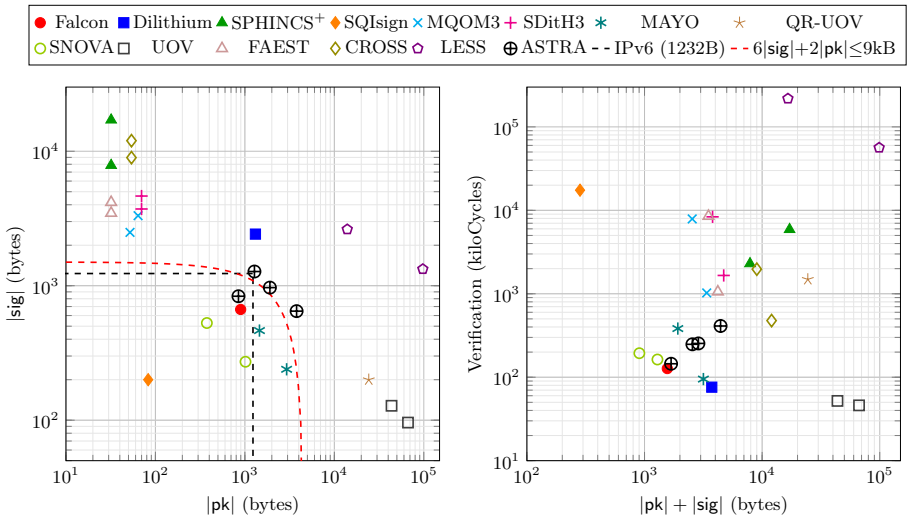
overwhelming majority, either production-grade or highly optimized. All benchmarks are run on an Intel Core i7-12700K, at 3.8 GHz with TurboBoost disabled, and the instantiations of ASTRA used for comparison are implemented in a constant-time manner. Timings in clock-cycle counts are obtained via the `rtdscp` clock counter lookup instruction, and averaged over 10^3 runs of each primitive (KEYGEN, SIGN, VERIFY) running on a single P-core.

Our comparison is made with the NIST standardized baselines, namely lattice-based Falcon [49] and Dilithium (now ML-DSA) [29], alongside the hash-based SPHINCS$^+$ (now SLH-DSA) [7]. We also include the schemes which have been selected by NIST to proceed onto the third round in the ongoing on-ramp additional call for signature schemes [4]. These schemes are meant to provide an alternative with superior performance to SPHINCS$^+$, while relying on computational trapdoors not related to lattices. The considered candidates are relying on problems from isogeny theory (SQIsign [1]), MPC-in-the-Head techniques (MQOM [15], SDitH [2]), multivariate quadratic systems (MAYO [19], QR-UOV [3], SNOVA [55], UOV [20]) and symmetric encryption (FAEST [14]). We additionally consider in our comparison the code-based schemes CROSS [8] and LESS [9] for their performance and signature sizes, along with the only unbroken hash-and-sign scheme with a publicly available, optimized implementation, Wave [26]. For schemes offering multiple parameter sets for any given security level λ, we included the most compact and the fastest options in our evaluation. Table 5 shows the performance of all the cited signature schemes, in terms of (thousands of) cycles required for key generation, signature generation, and verification, along with the size of public keys, signatures, and the combination of the

Table 5: Performance metrics for different signature schemes standardized or currently under evaluation by NIST, and for other code-based schemes, for NIST category 1, requiring $\lambda = 128$ security bits. Times are expressed in kilo-Clock Cycles (kCC), public key and signature sizes are in Bytes.

Scheme parameter set	**Keygen (kCC)**	**Sign (kCC)**	**Verify (kCC)**	$\lvert\mathsf{pk}\rvert$ **(B)**	$\lvert\mathsf{sig}\rvert$ **(B)**	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{sig}\rvert$ **(B)**
Falcon512	19,146	833	127	897	666	1,563
Dilithium2	73	211	76	1,312	2,420	3,732
SPHINCS$^+$-128s	226,117	1,697,023	2,295	32	7,856	7,888
SPHINCS$^+$-128f	3,527	82,161	5,896	32	17,088	17,120
ASTRA-128-2-17	4,535	80,912	145	848	835	1,683
ASTRA-128-2-23	12,096	19,394	248	1,279	1,275	2,554
ASTRA-128-3-31	4,959	19,696	254	1,904	968	2,872
ASTRA-128-7-53	2,016	469,814	411	3,786	647	4,433
Wave822	17,561,008	1,479,523	264,299	$3.68 \cdot 10^6$	822	$3,68 \cdot 10^6$
CROSS-R-SDP(G)-s	26	2,831	1,972	54	8,960	9,014
CROSS-R-SDP(G)-f	26	729	477	54	11,980	12,034
LESS-252-192	881	233,016	218,227	13,940	2,625	16,565
LESS-252-45	5,037	54,752	56,346	97,484	1,329	98,813
SQIsign	44,375	132,750	17,427	83	200	283
MQOM3-L1-sr-ct	206	11,244	7,888	52	2,492	2,544
MQOM3-L1-f-ct	82	1,079	1,024	64	3,316	3,380
SDitH3-L1-s-cp	74	8,453	8,359	70	3,721	3,791
SDitH3-L1-f-cp	74	1,869	1,658	70	4,643	4,713
MAYO-1	813	2,457	383	1,456	464	1,920
MAYO-2	934	887	95	2,928	239	3,167
QR-UOV	17,651	2,627	1,488	24,255	200	24,455
SNOVA_I_K	192	697	194	376	528	904
SNOVA_I_S	241	470	163	1,016	272	1,288
uov-Is	3,349	76	46	66,576	96	66,672
uov-Ip	2,208	72	52	43,576	128	43,704
FAEST-EM-128s	4	9,625	8,512	32	3,466	3,498
FAEST-EM-128f	4	1,250	1,057	32	4,170	4,202

two, taking as a security level $\lambda = 128$ (NIST category 1). We provide analogous tables for security levels $\lambda \in \{192, 256\}$ (NIST categories $3, 5$) in Appendix B. From a computational standpoint, ASTRA provides a significantly fast verification procedure, taking between 145 kCC and 411 kCC (between 40 μs and 114 μs), while signatures clock in at 5 ms to 127 ms. For reference, the highly optimized OpenSSL implementation of a (pre-quantum) RSA-4096 signature takes 2.7 ms, while a RSA-7680 signature takes 24ms to be computed on the same machine. This computational profile makes ASTRA essentially match the signature verification performance of the NIST standard chosen Falcon signature,



(a) Public key and signature size

(b) |pk| + |sig| size and verification time

Fig. 7: Public key sizes, signature sizes and verification times of different signature schemes for NIST security level 1.

while retaining acceptable signing times. The signing computation requirements of, e.g., ASTRA-128-2-23, are 4× to 87× smaller than the ones of the NIST standard chosen SPHINCS$^+$ for the same security margin, while gaining 6.1× in signature size and 3× in the combined size of public key and signature, the typical metric for the use in digital certificates. Considering the use in digital certificates, ASTRA compensates for the somewhat larger signature size with respect to MQ-based schemes (MAYO, QR-UOV, SNOVA, UOV) by providing a very compact public key, in turn obtaining better public key plus signature size figures than almost all of them.

When comparing the parameter set from ASTRA attaining the smallest public key plus signature size, ASTRA-128-2-17, with the only other code-based hash-and-sign signature scheme for which an optimized implementation is available, Wave, we obtain a slight increase in the attainable signature size (+13 B), while reducing the public key size from 3.68 MB down to 848 B, an ≈ 4330× reduction. We also reduce the signing time by 18× and verification time by 1822×. With respect to other signatures leveraging problems from coding theory, namely CROSS and LESS, ASTRA obtains a significant improvement in the signature size (about an order of magnitude), while retaining a faster verification than both, and faster signing than LESS. Compared to SQIsign, ASTRA trades off a larger public key and signature (about 6× when comparing with ASTRA-128-2-17) for a 120× gain in verification time and a 1.64× gain in signature time. Finally, considering signature schemes applying techniques from secure multiparty computation (MQOM, SDitH) ASTRA provides very com-

Table 6: Proposed parameter sets for $\lambda = 80$ security bits. Bit complexities in equivalent AES-128 runs. Public key and signature sizes are in Bytes.

Instance	n_0	q	p	v	iterMax	**KRA**	**SFA**	|pk|	|sig|
ASTRA-80	2	13	827	42	27	81.00	80.49	388	396
	2	17	1,091	42	8	81.07	80.42	565	562
	3	19	613	46	18	81.15	80.63	652	338
	4	23	509	47	26	80.42	82.92	873	303

petitive signature sizes (about 5× smaller) with a significant gain in verification time (7× to more than 57×).

Willing to provide a global overview of the public key and signature sizes we report in Figure 7(a) a scatter plot with all the values from Table 5. We also depict the cutoff (dashed black line) at 1232 bytes, which becomes crucial whenever single-packet interaction protocols are run over IPv6. Indeed, IPv6 does not allow packet fragmentation, therefore all the splitting logic should be pushed onto higher levels of the network stack. This becomes problematic whenever the transported application protocol runs over UDP, and does not provide any fragmentation support. The minimum value of the Packet Data Unit (PDU) size which must be guaranteed in IPv6 is 1280B, which, removing the headers, brings the room in the packet down to 1232B. As it can be seen, only few signature schemes, among which ASTRA, are able to achieve the stringent target of fitting a public key or a signature into a single IPv6 PDU. Furthermore, the same figure also reports a boundary on the size of six signatures plus two public keys, which was indicated by CloudFlare [59] to be the amount of cryptographic signature material exchanged during a TLS connection setup. The same technical white paper suggests that 9 kB is a reasonable upper bound for the overall size of such material, to avoid excessive latency in connections.

To provide a further global outlook on the current signature scenario, we depict in Figure 7(b) the verification computational requirements of the considered signature schemes, together with the combined public key and signature size for $\lambda = 128$ (NIST category 1). As it can be seen, ASTRA provides a very competitive profile, with only the lattice-based Falcon Pareto dominating it.

7.4 Parameters for 80-bit Security

In addition to the parameter sets for NIST security levels, we also report viable parameter sets to obtain compact cryptographic objects whenever 80 bits of security ($\lambda = 80$) are sufficient. While modern standards call for higher security margins for long-term data, 80-bit security is still useful for specific devices that have limited memory and processing power. For example, these parameters can be employed for short-lived authentication tokens in low-power sensor networks, where it is beneficial to have a small combined size for the public key and signature. Since public keys and signatures data are only valid for a short

Table 7: Performance metrics of ASTRA for $\lambda = 80$ security bits. Times are expressed in kiloCycles, public key and signature sizes are in Bytes.

ASTRA Instance	**Keygen (kCC)**	**Sign (kCC)**	**Verify (kCC)**	$\lvert\mathsf{pk}\rvert$ **(B)**	$\lvert\mathsf{sig}\rvert$ **(B)**	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{sig}\rvert$ **(B)**
ASTRA-80-2-13	2,314	15,330	80	388	396	784
ASTRA-80-2-17	2,125	3,608	98	565	562	1,127
ASTRA-80-3-19	1,595	16,394	83	652	338	990
ASTRA-80-4-23	931	31,748	84	873	303	1,176

time, this security level provides enough protection, as the transmitted information becomes useless long before an attacker can successfully perform an attack. Another highly constrained scenario where compactness is critical is DNSSEC, which faces strict UDP payload limits to avoid fragmentation over IPv6 (1232B) or standard Ethernet (1452B). As noted at a RIPE meeting [51], while DNSSEC typically employs long-lived public keys, a security level below $\lambda = 128$ could be deployed to improve network performance. In [51] it is also noted that, currently, only SQIsign and a reduced-security-version of HAWK targeting $\lambda = 64$ (whose security has been further reduced [58]) are able to fit.

Table 6 details the proposed parameters and their theoretical security, while Table 7 summarizes their performance. The proposed parameters are able to achieve very promising figures for $\lvert\mathsf{pk}\rvert + \lvert\mathsf{sig}\rvert$, namely between 784 and 1,176 bytes. For ASTRA-80-2-13 parameter set, the maximum operational DNSSEC payload (an NXDOMAIN authenticated denial containing two signatures) requires only 980 bytes, comfortably fitting the strict 1232B IPv6 limit without fragmentation. While the DNSKEY response requires 1,260 bytes, it successfully avoids TCP fallback on any standard network adhering to the 1452B limit.

8 Concluding Remarks

In this work, we present ASTRA: a quASi-cyclic code-based full-distance decoding TRApdoor Signature Scheme, a code-based hash-and-sign signature scheme built from QC-MDGM codes. We base our scheme on a new computationally hard problem, the full-distance codeword finding, proving that this problem is NP-hard, and employing it to build a trapdoor one-way function. We design an efficient decoding algorithm which acts as the trapdoor-aided solver for the one-way-function with trapdoor, leveraging the knowledge of a sparse generator matrix of an arbitrary QC-MDGM code as the trapdoor information, and provide numerical evidence that its output is not computationally distinguishable from a private-key-unaware sampler.

We propose concrete parameter sets for ASTRA building instances targeting λ-bit security, $\lambda \in \{128, 192, 256\}$. For $\lambda = 128$, we achieve public keys and signatures below 1 kB, with 3× signature size reduction and 1.4× public key

size reduction with respect to Dilithium. Our scheme achieves small verification times, around 40μs, with signature times between 5ms to 127ms, faster than other code-based hash-and-sign alternatives.

References

1. Aardal, M.A., Adj, G., Aranha, D.F., Basso, A., Martínez, I.A.C., Chávez-Saab, J., Santos, M.C.R., Dartois, P., Feo, L.D., Duparc, M., Eriksen, J.K., Fouotsa, T.B., Filho, D.L.G., Hess, B., Kohel, D., Leroux, A., Longa, P., Meyer, M., Panny, L., Patranabis, S., Petit, C., Rodríguez-Henríquez, F., Schaeffler, S., Wesolowski, B.: SQIsign: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://sqisign.org/
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.: Syndrome decoding in the head (SD-in-the-Head) signature scheme (version 3.0). Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://sdith.org/
3. Akiyama, R., Furue, H., Ikematsu, Y., Hoshino, F., Kinjo, K., Kosuge, H., Nakamura, S., Orihara, S., Takagi, T., Yamakoshi, K.: QR-UOV: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://info.isl.ntt.co.jp/crypt/qruov/
4. Alagic, G., Bros, M., Ciadoux, P., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Silberg, H., Smith-Tone, D., Waller, N.: Status report on the second round of the additional digital signature schemes for the nist post-quantum cryptography standardization process. Tech. Rep. NIST IR 8610 (April 2026). https://doi.org/10.6028/NIST.IR.8610
5. Annechini, A., Barenghi, A., Pelosi, G.: On the Hardness of Decoding Quasi-Cyclic Codes and the Security of Code-Based Public-Key Cryptosystems. In: Pietro, R.D., Renaud, K., Mori, P. (eds.) Proceedings of the 12th International Conference on Information Systems Security and Privacy, ICISSP 2026 - Volume 2, Marbella, Spain, March 04-06, 2026. pp. 49–60. SCITEPRESS (2026). https://doi.org/10.5220/0014296400004061, iSBN 978-989-758-800-6, ISSN 2184-4356
6. Aragon, N., Barreto, P.S.L.M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., Melchor, C.A., Misoczki, R., Persichetti, E., Richter-Brockmann, J., Sendrier, N., Tillich, J.P., Vasseur, V., Zémor, G.: BIKE: Bit Flipping Key Encapsulation. Round 4 Submission - Version 5.2 - 10/10/2024. [Online] Available: https://bikesuite.org/files/v5.2/BIKE_Spec.2024.10.10.1.pdf (2025)
7. Aumasson, J.P., Bernstein, D.J., Beullens, W., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Westerbaan, B.: SPHINCS+. Submission to the NIST Post-Quantum Cryptography Standardization Process (2020), https://sphincs.org/
8. Baldi, M., Barenghi, A., Battagliola, M., Bitzer, S., Gianvecchio, M., Karl, P., Manganiello, F., Pavoni, A., Pelosi, G., Pintore, F., Santini, P., Schupp, J., Sig-

norini, E., Slaughter, F., Wachter-Zeh, A., Weger, V.: CROSS: Codes and restricted objects signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2023), https://www.cross-crypto.com/
9. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J.F., Chou, T., Esser, A., Gaj, K., Karl, P., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M., Santini, P., Wallace, R., Zweydinger, F.: LESS: Linear Equivalence Signature Scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2023), https://www.less-project.com/
10. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: A Finite Regime Analysis of Information Set Decoding Algorithms. Algorithms **12**(10), 209 (2019). https://doi.org/10.3390/A12100209
11. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: LEDAcrypt - version 3.0 Specification (2020). [Online] Available: https://www.ledacrypt.org/documents/LEDAcrypt_v3.pdf (2025)
12. Banegas, G., Carrier, K., Chailloux, A., Couvreur, A., Debris-Alazard, T., Gaborit, P., Karpman, P., Loyer, J., Niederhagen, R., Sendrier, N., Smith, B., Tillich, J.P.: WAVE Specification Document. NIST PQC Additional Digital Signature Schemes, Round 1 specification (Jun 2023), https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/wave-spec-web.pdf
13. Bardet, M., Lemoine, A., Tillich, J.: An attack on the CFS scheme and on TII mceliece challenges (2026). https://doi.org/10.1007/978-3-032-35398-6_13, https://doi.org/10.1007/978-3-032-35398-6_13
14. 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: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://faest.info/
15. Benadjila, R., Bouillaguet, C., Feneuil, T., Rivain, M.: MQOM: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://mqom.org/
16. Berger, T.P., Cayrel, P., Gaborit, P., Otmani, A.: Reducing Key Length of the McEliece Cryptosystem. In: Preneel, B. (ed.) AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5580, pp. 77–97. Springer (2009). https://doi.org/10.1007/978-3-642-02384-2_6
17. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory **24**(3), 384–386 (1978). https://doi.org/10.1109/TIT.1978.1055873
18. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.A.: On the inherent intractability of certain coding problems (Corresp.). IEEE Trans. Inf. Theory **24**(3), 384–386 (1978). https://doi.org/10.1109/TIT.1978.1055873
19. Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M.J.: MAYO: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://pqmayo.org/
20. Beullens, W., Chen, M.S., Ding, J., Gong, B., Kannwischer, M.J., Patarin, J., Peng, B.Y., Schmidt, D., Tao, C., Yang, B.Y.: UOV: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://www.uovsig.org/

21. Chailloux, A., Debris-Alazard, T.: Tight and optimal reductions for signatures based on average trapdoor preimage sampleable functions and applications to code-based signatures. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II. pp. 453–479. Lecture Notes in Computer Science, Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_16
22. Chen, C.L., Peterson, W.W., Weldon Jr., E.J.: Some Results on Quasi-Cyclic Codes. Inf. Control. **15**(5), 407–423 (1969). https://doi.org/10.1016/S0019-9958(69)90497-5
23. Courtois, N.T., Finiasz, M., Sendrier, N.: Achieving AndSign Signatures in Code-Based Cryptography. In: International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT). pp. 157–174. Springer (2001). https://doi.org/10.1007/3-540-45682-1_10
24. Courtois, N.T., Finiasz, M., Sendrier, N.: How to Achieve a McEliece-Based Digital Signature Scheme. In: Boyd, C. (ed.) Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings. pp. 157–174. Lecture Notes in Computer Science, Springer (2001). https://doi.org/10.1007/3-540-45682-1_10
25. Couvreur, A., Debris-Alazard, T., Gaborit, P., Vinçotte, A.: MIRANDA: short signatures from a leakage-free full-domain-hash scheme. Cryptology ePrint Archive, Paper 2025/1878 (2025), https://eprint.iacr.org/2025/1878
26. Debris-Alazard, T., Sendrier, N., Tillich, J.: Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. pp. 21–51. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-34578-5_2
27. Don, J., Fehr, S., Majenz, C.: The Measure-and-Reprogram Technique 2.0: Multi-round Fiat-Shamir and More. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. pp. 602–631. Lecture Notes in Computer Science, Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_21
28. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. pp. 356–383. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_13
29. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(1), 238–268 (2018). https://doi.org/10.13154/TCHES.V2018.I1.238-268
30. Esser, A., Verbel, J.A., Zweydinger, F., Bellini, E.: CryptographicEstimators. https://github.com/Crypto-TII/CryptographicEstimators/commit/071cce7a942af423422cb0c6d6af55778c72e4d1
31. Esser, A., Verbel, J.A., Zweydinger, F., Bellini, E.: Sok: Cryptographicestimators - a software library for cryptographic hardness estimation. In: Zhou, J., Quek,

T.Q.S., Gao, D., Cárdenas, A.A. (eds.) Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024. ACM (2024). https://doi.org/10.1145/3634737.3645007
32. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 197–206. ACM (2008). https://doi.org/10.1145/1374376.1374407
33. Guo, Q., Johansson, T., Wagner, P.S.: A Key Recovery Reaction Attack on QC-MDPC. IEEE Trans. Inf. Theory **65**(3), 1845–1861 (2019). https://doi.org/10.1109/TIT.2018.2877458
34. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
35. Katsumata, S., Yamada, S., Yamakawa, T.: Tighter Security Proofs for GPV-IBE in the Quantum Random Oracle Model. J. Cryptol. **34**(1), 5 (2021). https://doi.org/10.1007/S00145-020-09371-Y
36. Liu, Q., Zhandry, M.: Revisiting Post-quantum Fiat-Shamir. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. pp. 326–355. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_12
37. May, A., Meurer, A., Thomae, E.: Decoding Random Linear Codes in $\mathcal{O}(2^{0.054n})$. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. pp. 107–124. Lecture Notes in Computer Science, Springer (2011). https://doi.org/10.1007/978-3-642-25385-0_6
38. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Bos, J., Deneuville, J.C., Dion, A., Gaborit, P., Lacan, J., Persichetti, E., Robert, J.M., Véron, P., Zémor, G.: Hamming Quasi-Cyclic (HQC) - Fourth round version. https://pqc-hqc.org/doc/hqc-specification_2024-02-23.pdf (2025)
39. National Institute of Standards and Technology: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. Tech. rep. (2016), https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf, section 4.A.4: Security Definition for Digital Signatures
40. National Institute of Standards and Technology: Fips 204: Module-lattice-based digital signature standard. Tech. rep., U.S. Department of Commerce (Aug 2024). https://doi.org/10.6028/NIST.FIPS.204
41. National Institute of Standards and Technology: Fips 205: Stateless hash-based digital signature standard. Tech. rep., U.S. Department of Commerce (Aug 2024). https://doi.org/10.6028/NIST.FIPS.205
42. Paninski, L.: A coincidence-based test for uniformity given very sparsely sampled discrete data. IEEE Trans. Inf. Theory **54**(10), 4750–4755 (2008). https://doi.org/10.1109/TIT.2008.928987, https://doi.org/10.1109/TIT.2008.928987

43. Pellegrini, A., Tognolini, G.: Breaking HWQCS: a code-based signature scheme from high weight QC-LDPC codes. IACR Cryptol. ePrint Arch. p. 117 (2024), https://eprint.iacr.org/2024/117
44. Perriello, S., Barenghi, A., Pelosi, G.: Improving the Efficiency of Quantum Circuits for Information Set Decoding. ACM Trans. Quantum Comput. **4**(4), 25:1–25:40 (2023). https://doi.org/10.1145/3607256
45. Perriello, S., Barenghi, A., Pelosi, G.: Quantum Circuit Design for the Lee-Brickell Based Information Set Decoding. In: Andreoni, M. (ed.) Applied Cryptography and Network Security Workshops - ACNS 2024 Satellite Workshops, AIBlock, AIHWS, AIoTS, SCI, AAC, SiMLA, LLE, and CIMSS, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II. pp. 8–28. Lecture Notes in Computer Science, Springer (2024), https://doi.org/10.1007/978-3-031-61489-7_2
46. Picozzi, C., Meneghetti, A., Tognolini, G.: A post-quantum digital signature scheme from QC-LDPC codes. IACR Cryptol. ePrint Arch. p. 1477 (2022), https://eprint.iacr.org/2022/1477
47. Prabowo, T.F., Tan, C.H.: Attack on a code-based signature scheme from QC-LDPC codes. In: Hajji, S.E., Mesnager, S., Souidi, E.M. (eds.) Codes, Cryptology and Information Security - 4th International Conference, C2SI 2023, Rabat, Morocco, May 29-31, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13874, pp. 136–149. Springer (2023). https://doi.org/10.1007/978-3-031-33017-9_9
48. Prange, E.: The Use of Information Sets in Decoding Cyclic Codes. IEEE Transactions on Information Theory **8**(5), 5–9 (sep 1962). https://doi.org/10.1109/TIT.1962.1057777
49. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Submission to the NIST Post-Quantum Cryptography Standardization Process (2020), https://falcon-sign.info/
50. Sendrier, N.: Decoding one out of many. In: Yang, B. (ed.) Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7071, pp. 51–67. Springer (2011). https://doi.org/10.1007/978-3-642-25405-5_4
51. Surý, O.: DNSSEC Post-Quantum Crypto Algorithms in BIND. DNS Working Group Minutes - RIPE 90 (2022), https://ripe90.ripe.net/archives/video/1608
52. Tan, C.H., Prabowo, T.F.: High weight code-based signature scheme from QC-LDPC codes. In: Seo, H., Kim, S. (eds.) Information Security and Cryptology - ICISC 2023 - 26th International Conference on Information Security and Cryptology, ICISC 2023, Seoul, South Korea, November 29 - December 1, 2023, Revised Selected Papers, Part I. Lecture Notes in Computer Science, vol. 14561, pp. 306–323. Springer (2023). https://doi.org/10.1007/978-981-97-1235-9_16
53. Vardy, A.: The intractability of computing the minimum distance of a code. IEEE Trans. Inf. Theory **43**(6), 1757–1766 (1997). https://doi.org/10.1109/18.641542
54. Wagner, D.A.: A generalized birthday problem. In: Yung, M. (ed.) Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. pp. 288–303. Lecture Notes in Computer Science, Springer (2002). https://doi.org/10.1007/3-540-45708-9_19
55. Wang, L.C., Chou, C.Y., Ding, J., Kuan, Y.L., Leegwater, J.A., Li, M.S., Tseng, B.S., Tseng, P.E., Wang, C.C.: SNOVA: Digital signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process (Additional Digital Signature Schemes) (2026), https://snova.pqclab.org/

56. Wang, T., Wang, A., Wang, X.: Exploring Decryption Failures of BIKE: New Class of Weak Keys and Key Recovery Attacks. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 70–100. Springer (2023). https://doi.org/10.1007/978-3-031-38548-3_3
57. Weger, V., Gassner, N., Rosenthal, J.: A Survey on Code-Based Cryptography, https://arxiv.org/pdf/2201.07119.pdf
58. Weis, S.: HAWK-n Key Recovery Reduces to SVP in Dimension $n/2+1$. Message to the NIST PQC Forum mailing list (2026), https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/2r2u6SbHun4/m/S-ZUsHNICQAJ
59. Westerbaan, B.: NIST's pleasant post-quantum surprise. CloudFlare Blog (2022), https://blog.cloudflare.com/nist-post-quantum-surprise/
60. Zhandry, M.: Secure Identity-Based Encryption in the Quantum Random Oracle Model. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. pp. 758–775. Lecture Notes in Computer Science, Springer (2012). https://doi.org/10.1007/978-3-642-32009-5_44

A Additional FullDistDec Output Data

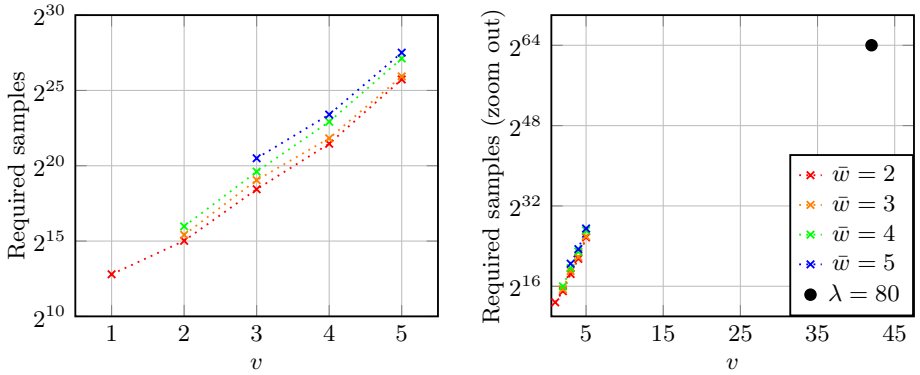


Fig. 8: Number of samples required to distinguish $\bar{w}$ error coordinates lying in the support of one of the rows of the secret matrix $\mathbf{G}$ with success probability $\frac{2}{3}$, for different values of column weight v. Code parameters: $n_0 = 2$, $p = 701$, $q = 11$, achieving 80b security at $v = 42$. The black marker shows the lower bound on the number of samples required for the scheme to be secure. Data obtained by generating $2 \cdot 10^8$ signatures per data point; y-axis in log-scale.

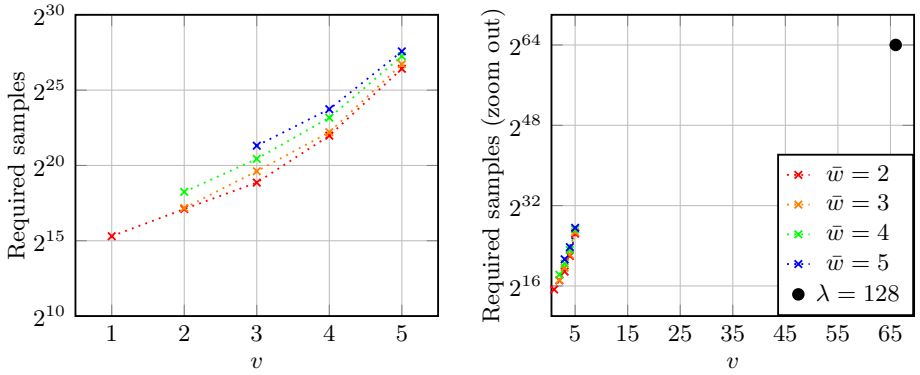


Fig. 9: Number of samples required to distinguish $\bar{w}$ error coordinates lying in the support of one of the rows of the secret matrix $\mathbf{G}$ with success probability $\frac{2}{3}$, for different values of column weight v. Code parameters: $n_0 = 2$, $p = 1637$, $q = 17$, achieving 128b security at $v = 66$. The black marker shows the lower bound on the number of samples required for the scheme to be secure. Data obtained by generating $2 \cdot 10^8$ signatures per data point; y-axis in log-scale.

B Full Performance Comparison

Table 8: Performance metrics for different signature schemes standardized or currently under evaluation by NIST, and for other code-based schemes, for NIST security level 3, requiring $\lambda = 192$ security bits. Times are expressed in kiloCycles, public key and signature sizes are in Bytes.

Scheme parameter set	**Keygen (kCC)**	**Sign (kCC)**	**Verify (kCC)**	$\lvert\mathsf{pk}\rvert$ **(B)**	$\lvert\mathsf{sig}\rvert$ **(B)**	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{sig}\rvert$ **(B)**
Falcon1024	56,438	1,670	252	1,793	1,280	3,073
Dilithium3	126	343	123	1,952	3,293	5,245
SPHINCS$^+$-192s	330,735	2,878,206	3,057	16,224	48	16,272
SPHINCS$^+$-192f	5,209	131,012	8,591	48	35,664	35,712
ASTRA-192-2-23	23,451	441,639	423	1,859	1,854	3,713
ASTRA-192-2-29	31,699	87,633	626	2,557	2,496	5,053
ASTRA-192-4-61	25,664	89,074	995	6,546	2,206	8,752
ASTRA-192-5-61	15,273	372,112	904	6,640	1,684	8,324
Wave1249	56,147,864	4,695,964	562,998	$7.86 \cdot 10^6$	1,249	$7.86 \cdot 10^6$
CROSS-R-SDP(G)-s	53	4,034	2,961	83	20,452	20,535
CROSS-R-SDP(G)-f	53	1,694	1,132	83	26,772	26,855
LESS-400-220	2,617	662,623	628,436	35,074	6,329	41,403
LESS-400-102	6,741	305,948	298,896	105,174	4,131	109,305
SQIsign	174,540	429,741	50,575	129	306	435
MQOM3-L3-sr-ct	1,596	36,352	30,702	78	5,932	6,010
MQOM3-L3-f-ct	467	5,542	5,495	96	7,564	7,660
SDitH3-L3-s-cp	114	41,600	41,204	98	8,484	8,582
SDitH3-L3-f-cp	114	10,235	8,852	98	10,452	10,550
MAYO-3	2,470	4,914	1,118	2,986	681	3,667
QR-UOV	57,160	6,875	4,868	71,891	292	72,183
SNOVA_III_K	507	1,506	404	912	688	1,600
SNOVA_III_S	675	1,574	401	2,032	456	2,488
uov-III	12,397	208	148	189,232	200	189,432
FAEST-EM-192s	8	38,796	39,119	48	7,874	7,922
FAEST-EM-192f	8	4,871	5,124	48	9,818	9,866

Table 9: Performance metrics for different signature schemes standardized or currently under evaluation by NIST, and for other code-based schemes, for NIST security level 5, requiring $\lambda = 256$ security bits. Times are expressed in kiloCycles, public key and signature sizes are in Bytes.

Scheme parameter set	**Keygen (kCC)**	**Sign (kCC)**	**Verify (kCC)**	$\lvert\mathsf{pk}\rvert$ **(B)**	$\lvert\mathsf{sig}\rvert$ **(B)**	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{sig}\rvert$ **(B)**
Falcon1024	56,438	1,670	252	1,793	1,280	3,073
Dilithium5	200	427	196	2,592	4,595	7,187
SPHINCS$^+$-256s	277,410	2,476,835	4,405	64	29,792	29,856
SPHINCS$^+$-256f	13,668	268,566	8,808	64	49,856	49,920
ASTRA-256-2-29	52,182	1,110,453	935	3,367	3,287	6,654
ASTRA-256-2-31	65,888	451,908	918	3,589	3,621	7,210
ASTRA-256-3-53	39,545	333,370	1,544	7,092	3,578	10,670
ASTRA-256-4-61	41,363	1,166,286	1,489	8,544	2,880	11,424
Wave1644	154,473,662	11,907,355	973,679	$13.63 \cdot 10^6$	1,644	$13.63 \cdot 10^6$
CROSS-R-SDP(G)-s	86	6,134	4,314	106	36,454	36,560
CROSS-R-SDP(G)-f	86	2,814	1,908	106	48,102	48,208
LESS-548-345	5,725	2,288,990	2,080,197	65,793	10,680	76,473
LESS-548-137	15,121	845,373	856,135	197,315	7,436	204,751
SQIsign	253,376	828,568	95,077	169	406	575
MQOM3-L5-sr-ct	2,673	54,589	49,652	104	10,836	10,940
MQOM3-L5-f-ct	826	9,572	8,710	128	13,380	13,508
SDitH3-L5-s-cp	175	65,287	64,608	132	15,147	15,279
SDitH3-L5-f-cp	173	13,899	13,195	132	19,144	19,276
MAYO-5	6,484	11,303	2,425	5,554	964	6,518
QR-UOV	179,430	16,259	11,606	173,676	392	174,068
SNOVA_V_K	1,086	3,480	982	1,216	896	2,112
SNOVA_V_S	1,342	4,788	821	2,716	591	3,307
uov-V	31,751	389	277	446,992	260	447,252
FAEST-EM-256s	15	61,003	63,655	64	14,554	14,618
FAEST-EM-256f	14	8,969	8,910	64	18,084	18,148