IMPERIAL

Improved signatures from instantiating HAWK with a remarkable lattice

Author

M.J. Adam

CID: 02286647

Supervised by

Prof. C. Ling

M. Lie

Second Marker

Dr S. Vlaski

A Thesis submitted in fulfillment of requirements for the degree of
Master of Engineering in Electronic and Information Engineering

Department of Electrical and Electronic Engineering
Imperial College London

2026

Abstract

The HAWK post-quantum signature scheme [1] provides a useful framework in which to ask whether signature generation can benefit from lattice structure beyond a standard module lattice setting. This thesis focuses on the remarkable lattice E_8, a highly structured eight-dimensional lattice. Its well understood structure and high degree of symmetry make it a good controlled setting for studying more efficient discrete Gaussian sampling. It is also relevant cryptographically, since E_8 admits efficient decoding and remarkable lattices have been proposed as promising for LIP and Module LIP based constructions [1], [2].

Starting from a congruence defined rank 2 $\mathbb{Z}[\zeta_8]$ module [3], we show that the underlying $\mathbb{Z}$ lattice is isometric to $2E_8$, give a clear $\mathbb{Z}[\zeta_8]$ basis, and then embed it in $R_n = \mathbb{Z}[X]/(X^n + 1)$, yielding a module whose underlying lattice is the direct sum $(2E_8)^{\oplus n/4}$.

However, an obstruction appears when this construction is compared with HAWK's signing interface. HAWK labels signing cosets using coefficientwise reduction modulo 2, but this reduction does not recover the full internal quotient of an E_8 block module. As a result, the existing HAWK sampler cannot be replaced directly by an E_8 sampler. To resolve this mismatch, we introduce a coset-matching interface based on internal module coordinates called "HAWK-E_8-CM". HAWK-E_8-CM is an E_8, coset-matched HAWK variant in which verification uses a modified public quadratic form. We also provide an associated proof of concept implementation that tests the arithmetic, coset handling, sampler interface, signing path, and verification path required by this variant.

Our implementation shows that an E_8 sampler is faster than HAWK's sampler in standalone sampler benchmarks, with selected parallel configurations giving median speedups of approximately $4.32\times$, $5.31\times$, and $7.12\times$, across $n = 256, 512, 1024$. Giving clear evidence that the remarkable structure of E_8 can lead to efficiency gains which may trickle down to signature generation.

Plain Language Summary

Digital signatures help prove that digital messages, files, software updates, and online transactions are genuine. Many of the signature methods used today may become insecure if large quantum computers are built, so researchers are developing new methods that are designed to remain secure in a post-quantum world.

This thesis studies whether one such modern post-quantum signature scheme can be improved by using a special lattice called E_8. A lattice is like a mathematical grid where any point in the grid can be expressed as an integer linear combination of a finite number of basis vectors. The E_8 lattice is especially interesting because it is highly regular, highly symmetric, and very well understood. These features make it a useful testing ground for studying how signatures can be generated more efficiently.

The idea we provide is to place repeated copies of E_8 inside the mathematical setting used by the signature scheme, showing that this can be done in a precise way. However, the fit is not as simple as directly replacing one part of the original scheme, as we show. The original signing method uses a simple way of tracking even and odd information, but this loses important information when applied to the new E_8 based structure.

To address this, we develop a new matching method that keeps track of the missing information and leads to a modified version of the original scheme. A proof of concept implementation has been produced to test the arithmetic, signing, and verification steps of this modified construction. This implementation also presents a faster method for sampling vectors by using the special structure of the E_8 lattice, within a similar framework to the original signature scheme, which could lead towards faster signature generation.

Declaration of Originality

I hereby declare that the work presented in this thesis is my own unless otherwise stated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced. LLMs have been used where appropriate for formatting, proofreading and research.

Copyright Declaration

Acknowledgments

I would like to thank my supervisors, Maja Lie and Professor Cong Ling, for their guidance and support throughout this project. Their feedback was especially valuable in helping to develop the initial mathematical direction of the thesis, what papers to research and in understanding how to approach the field of lattices.

I am also very grateful to Maja Lie and Franciele Silva who took the time to discuss aspects of lattice theory, provided feedback, notes, meetings and helped improve my general understanding throughout the project. These conversations, notes and feedback helped me work through several of the technical difficulties that arose, particularly around the E_8 embedding and its compatibility with HAWK.

Contents

1 Introduction

Contents

1.1 Motivation

Digital signatures are a basic tool for trust and authentication in digital systems. They are used to verify that software updates, messages, transactions, and other digital objects come from the claimed sender and have not been altered. Many widely deployed signature schemes rely on mathematical problems, such as integer factorisation and discrete logarithms, that would become vulnerable if large scale quantum computers were built. This motivates the development of post-quantum signature schemes, which are designed to remain secure against both classical adversaries and those with access to quantum computation.

Lattice-based cryptography is one of the leading approaches to post-quantum cryptography. It is attractive because lattice problems have strong worst-to-average case hardness foundations [4] and because lattice-based schemes can often be implemented efficiently. However, efficiency is not only a matter of fast arithmetic. In many lattice-based signature schemes, signing depends on

sampling short lattice vectors from carefully chosen probability distributions. If this sampling is too slow, the scheme becomes less practical. If the sampling distribution is wrong, even slightly, the security argument may no longer apply.

The HAWK signature scheme [1] is a useful framework in which to study this issue. HAWK relies on the Lattice Isomorphism Problem (LIP) and is designed around structured module lattices to provide compact and efficient post-quantum signatures. Its signing procedure relies almost entirely on discrete Gaussian sampling over structured lattice cosets, so the behaviour of the sampler is closely tied to both performance and correctness. This makes HAWK a natural place to ask whether additional lattice structure can make sampling more efficient.

This thesis explores that question through the remarkable lattice E_8. The E_8 lattice is a highly structured eight-dimensional lattice with strong symmetry and well understood geometry [5], [6]. It is called remarkable due to its structure which allows for efficient decoding in the Closest Vector Problem (CVP) [7]. Such remarkable lattices have been proposed as promising candidates for LIP and Module-LIP based constructions [1], [2]. Its properties also make it a good controlled setting for studying discrete Gaussian sampling. The structure of E_8 suggests useful coset decompositions, and its known shell structure provides low norm shell reference data against which sampler behaviour can be checked [5]. Our motivation therefore is that the structure of E_8 may expose a more organised way to handle the sampling problem.

At the same time, introducing such structure into a cryptographic scheme is delicate. A lattice that is useful for sampling must also fit the algebraic interface used by signing and verification, and it must not introduce structure that obviously weakens the underlying security assumptions.

1.2 Problem Statement

Stated more clearly, the problem addressed in this thesis is whether the structure of E_8 can be used inside the HAWK signature framework in a way that is mathematically coherent, compatible with signing and verification, and meaningful for sampling. At first sight, the idea appears straightforward: if E_8 has useful symmetry and a well understood coset structure, then one might try to replace HAWK's existing sampler with an E_8 sampler. However, this turns out not to be a simple substitution problem.

HAWK does not sample from an arbitrary lattice in isolation. Its signing procedure is tied to a specific algebraic setting, a specific mod-2 coset determined by the message hash and secret basis,

and a public quadratic form used during verification. Any replacement sampling structure must therefore satisfy several constraints at once. It must live naturally inside HAWK's cyclotomic ring, expose the right cosets to the signing algorithm, and preserve the relationship between the signer's sampled vector and the verifier's norm check.

A significant difficulty is that the E_8 module has its own internal mod-2 quotient, while HAWK labels signing cosets using coefficientwise reduction modulo 2. These two ways of reducing modulo 2 are not the same. Coefficientwise reduction loses most of the internal parity information of the E_8 module, so it cannot label the required E_8 cosets correctly. This creates a mismatch between the structure that makes E_8 useful for sampling and the interface that HAWK uses for signing.

We therefore need to determine whether the E_8 structure can be incorporated into HAWK without losing the properties that make it useful in the first place. This requires three separate questions to be addressed. Firstly whether an E_8 module can be embedded into HAWK's ambient module, then whether its cosets can be matched correctly to HAWK's signing target and finally whether the resulting construction can still be verified using an appropriate public quadratic form. And since changing the public form also changes the security formulation, the problem becomes a scheme level design issue as well as an implementation one.

1.3 Goal and Research Questions

We do not aim to produce a production ready replacement for HAWK. Instead, the goal is to understand what mathematical and implementation changes are required before an E_8 sampler can be used in a HAWK-like signature construction, and whether those changes create a plausible route towards improved signing performance. A further goal is that the lessons learned from this process may indicate how similar methods could be applied to the wider family of remarkable lattices.

The main research question is therefore:

> Can an E_8 lattice structure be integrated into the HAWK signature framework in a way that preserves the signing and verification interface needed for a correct HAWK-like variant, and what benefits could this provide?

We address this question through the following sub questions:

1. Can a congruence defined rank 2 $\mathbb{Z}[\zeta_8]$ module be identified with a scaled copy of E_8, and can it be given a clear module basis?

2. Can this E_8 based module be embedded into HAWK's cyclotomic ring $R_n = \mathbb{Z}[X]/(X^n + 1)$, producing repeated E_8 blocks inside the ambient module used by HAWK?

3. Does HAWK's coefficientwise mod-2 signing interface correctly label the cosets of the E_8 block module, or is a different coset-matching mechanism required?

4. If a direct sampler replacement is not possible, what changes to the public quadratic form and verification equation are needed to obtain a coherent HAWK-like variant?

5. Can the resulting construction be implemented as a proof of concept signing and verification path, sufficient to test the arithmetic, coset handling, norm consistency, and practical behaviour of our proposed variant?

6. Do the mathematical construction and proof of concept implementation indicate any plausible advantages for sampler and signing efficiency?

These questions guide the development of the thesis. The first two questions concern the algebraic embedding of E_8. The third question addresses the obstruction that arises from HAWK's mod-2 interface. The fourth question turns this obstruction into a modified construction, which we call HAWK-E_8-CM, where CM stands for Coset Matching. The final two questions connect the mathematical construction to our associated implementation and assess whether the resulting changes point towards a more efficient signature scheme.

1.4 Contributions

We answer the previous questions through the following contributions:

1. We identify the congruence defined rank 2 $\mathbb{Z}[\zeta_8]$ module as a scaled copy of E_8, and give a clear module basis for it.

2. We extend this construction into HAWK's cyclotomic ring, producing an E_8 block module with underlying lattice $(2E_8)^{\oplus n/4}$.

3. We show that HAWK's coefficientwise mod-2 reduction does not recover the full internal quotient of the E_8 block module.

4. We define HAWK-E_8-CM, an E_8 block, coset-matched HAWK variant with a modified public quadratic form.

5. We provide a proof of concept implementation of the uncompressed signing and verification path for this variant.

6. We show that the implemented cached and parallel E_8 block sampler can outperform HAWK's sampler in standalone sampler benchmarks. Achieving relatively stable median sampler speedups of approximately $4.32\times$, $5.31\times$, and $7.12\times$ for $n = 256, 512, 1024$, respectively.

1.5 Scope and Limitations

This thesis focuses on an E_8 case study rather than on all remarkable lattices. However, the challenges and design choices identified here may also inform how other remarkable lattices behave when placed inside HAWK-like signing frameworks.

The mathematical scope includes the E_8 block embedding, the mod-2 coset mismatch, the internal coset-matching map, and the modified public quadratic form. The implementation scope is limited to a proof of concept uncompressed signing and verification path. This code tests the arithmetic, coset handling, norm consistency, speed and other practical behaviour of the construction, but it is not constant time and is not side-channel hardened.

Several limitations remain in the current code implementation. We use a coset-matched Construction A sampler for the internal E_8 block cosets, but it remains a research prototype. It uses floating point probability computations, runtime generated probability tables, and data dependent control flow, so it is not constant time or side-channel hardened and is not suitable for deployment.

Compressed signatures are also outside the main scope. We describe the mathematical reconstruction rule, but final compression would require new bounds and thorough numerical error analysis. More cryptanalysis is also needed for the overall scheme, we address significant issues but the changed public form leaves questions open. Finally, we can not claim that the current HAWK-E_8-CM prototype gives a final practical improvement over HAWK as a full signature scheme. Results show a sampler level advantage for the E_8 sampler, while the full uncompressed signing path remains experimental and requires substantial further optimisation.

1

1.6 Thesis Structure

The remainder of this thesis is organised as follows.

Chapter 2: Introduces the background material needed to understand the field. It covers lattice-based cryptography such as signature schemes, discrete Gaussian sampling, module lattices, the HAWK signature scheme, and the role of remarkable lattices, with particular attention to E_8.

Chapter 3: Describes the design process. We explore how replacing HAWK's sampler with an E_8 sampler led to certain problems and the changes/mitigations made in response.

Chapter 4: Presents the main construction and implementation. It gives the E_8 block embedding, analyses the mod-2 coset mismatch, defines the HAWK-E_8-CM variant, and describes the code implementation.

Chapter 5: Sets out the testing and verification strategy. Describing how the arithmetic, coset handling, signing path, verification path, and sampler behaviour are tested.

Chapter 6: Presents the results obtained.

Chapter 7: Evaluates these results in relation to the research questions, including correctness, efficiency, sampler validation, and the limitations of the current prototype.

Finally, the thesis is concluded by a summarising of the main findings, the extent to which the research questions have been answered, and outlining directions for future work.

2 Background

Contents

2.1 Post-Quantum Signatures and the Role of Sampling

The prospect of large scale quantum computation motivates a transition away from classical public key cryptography whose security relies on integer factorisation or the discrete logarithm problem. Lattice-based signatures are leading candidates for new post-quantum signature schemes as lattice-based cryptography has strong worst-to-average case hardness foundations [4] and can be instantiated in a wide range of designs. However, they also expose a bottleneck as generating

signatures often requires sampling from carefully specified high dimensional distributions in a way that is both efficient and distributionally correct [8], [9].

In hash and sign lattice signatures, the signer must output a short lattice vector whose distribution is sufficiently close to a target. Deviations from the intended distribution can invalidate security arguments, while the mechanisms used to enforce the target distribution directly impact throughput and stability [1], [10]. In HAWK, signature generation entails discrete Gaussian sampling over a structured lattice coset, so the practicality of the scheme is closely tied to the efficiency and robustness of this sampling step [1]. More broadly, recent work on lattice Gaussian sampling has emphasised that obtaining polynomial time, implementation friendly samplers often requires leveraging structure in nested lattices and their quotients, rather than treating sampling as a generic black box [9], [11].

We therefore intended to show that E_8 can enable not only faster sampling, but also a more organised approach to the sampling problem. [5], [6] show that certain remarkable lattices have unusually rigid structure, explicit descriptions, and well understood shell behaviour, all of which can be useful when designing and testing samplers.

The challenge is that HAWK and related modern schemes are expressed over module lattices, whereas remarkable lattices are usually first encountered as Euclidean objects. Hence, a key question is how a remarkable lattice, with E_8 as the example, can be realised in a module lattice form compatible with HAWK [12]. In parallel, because HAWK is closely related to the module variant of the lattice isomorphism problem, the security landscape is informed by sampling theory as well as developing work on module LIP and related attacks, which constrains what kinds of additional structure can be introduced safely [2], [13], [14], [15].

2.2 Lattice Preliminaries

This section summarises the lattice notions to be used throughout. The emphasis is on definitions and invariants that appear in discrete Gaussian sampling, norm verification, and the description of remarkable lattices such as E_8. These notions also provide the bridge to the module lattice viewpoint used by HAWK. References include [5], [6], with the ideal and module lattice viewpoint treated in detail in [12].

2.2.1 Euclidean Lattices and Basic Invariants

A full rank Euclidean lattice $\Lambda \subset \mathbb{R}^n$ is a discrete additive subgroup generated by n linearly independent vectors. If $B \in \mathbb{R}^{n\times n}$ has columns $b_1, \ldots, b_n$ forming a basis, then

$$\Lambda = B\mathbb{Z}^n = \left\{ \sum_{i=1}^{n} z_i b_i \ : \ z \in \mathbb{Z}^n \right\}$$

The Gram matrix associated with B is $G = B^\top B$, where $\top$ denotes the transpose. The Gram matrix induces a norm in the following way: for a lattice vector $x = Bz \in \Lambda$ one has $\|x\|^2 = z^\top G z =: \|z\|_G^2$.

Throughout, lattices are assumed to be full rank unless stated otherwise. If $\Lambda = B\mathbb{Z}^n$, where the columns of B form a basis of Λ, a fundamental parallelepiped is

$$\mathcal{P}(B) = \left\{ \sum_{i=1}^{n} t_i b_i : 0 \le t_i < 1 \right\}$$

The determinant, or covolume, of Λ is the volume of this fundamental parallelepiped:

$$\det(\Lambda) = \mathrm{vol}(\mathcal{P}(B)) = |\det(B)|$$

This value is independent of the chosen basis. A lattice is integral if $\langle x, y\rangle \in \mathbb{Z}$ for all $x, y \in \Lambda$. And a lattice is unimodular if it is equal to its dual lattice or equivalently, for an integral full rank lattice, $\det(\Lambda) = 1$.

The dual lattice is given by $\Lambda^* = \{y \in \mathbb{R}^n : \langle y, x\rangle \in \mathbb{Z} \text{ for all } x \in \Lambda\}$. If $\Lambda = B\mathbb{Z}^n$, where B is a square full rank basis matrix, then $B^{-\top}$ is a basis matrix for Λ^*, so $\Lambda^* = B^{-\top}\mathbb{Z}^n$. A lattice is even if $\|x\|^2 \in 2\mathbb{Z}$ for all $x \in \Lambda$. Evenness and unimodularity are not what define remarkable lattices in general, but they are important structural properties of E_8. In particular, since E_8 is even and unimodular, its possible norm shells are strongly constrained [5], [6].

Duality is used here mainly as background. It appears in Gaussian sampling through smoothing type quantities, and in the ideal/module lattice setting where trace pairings and integrality conditions are described using dual objects [9], [12]. Throughout this thesis, norms are Euclidean unless stated otherwise.

2.2.2 The Lattice Isomorphism Problem

Informally, LIP asks whether two full rank Euclidean lattices are the same up to an isometry [16]. One convenient way to express this, that is also connected to the module setting, is through Gram matrices.

> **Gram-matrix formulation:** Given Gram matrices $G_1, G_2 \in \mathbb{R}^{n\times n}$ of full rank lattices $\Lambda_1, \Lambda_2 \subset \mathbb{R}^n$ (i.e. $G_i = B_i^\top B_i$ for some basis B_i of Λ_i), decide whether there exists a unimodular change of basis $M \in \mathrm{GL}_n(\mathbb{Z})$ such that $G_2 = M^\top G_1 M$. In the search version of the problem, one also asks to output such an M. If such an M exists, it gives a witness that the two quadratic forms, and hence the corresponding lattices, are isometric.

This relation is the standard algebraic expression of equivalence, or congruence, of quadratic forms under an integral change of basis. In cryptographic settings such as HAWK, this formulation is useful because public keys can be interpreted as structured quadratic forms whose hidden structure is known to the signer but should be hard for an attacker to recover [2], [5].

2.2.3 Ideal and Module Lattices

The ideal and module lattice viewpoint links Euclidean lattices with algebraic number theory, which is one of the main reasons why modern lattice-based cryptography can be implemented efficiently. Instead of working with an arbitrary basis of a lattice in $\mathbb{R}^n$, one works with algebraic objects such as ideals or modules over structured rings. This gives both a Euclidean lattice and an efficient arithmetic representation.

Let K be a number field of degree $d = r_1 + 2r_2$, with ring of integers $\mathcal{O}_K$. The Minkowski embedding sends elements of K into real Euclidean space $\Phi : K \longrightarrow \mathbb{R}^{r_1} \times \mathbb{C}^{r_2} \cong \mathbb{R}^d$. Under this embedding, the image of $\mathcal{O}_K$, and more generally of a fractional ideal $I \subset K$, is a full rank lattice in $\mathbb{R}^d$: $\Lambda(I) := \Phi(I)$. To turn this into a metric lattice, one also needs a positive definite bilinear or Hermitian form. In cyclotomic and CM (Complex Multiplication) settings, this is usually expressed using a trace pairing such as $\langle x, y\rangle = \mathrm{Tr}_{K/\mathbb{Q}}(x\overline{y})$ or a weighted version of it [12]. The important point for this thesis is that the algebraic object alone is not enough. The embedding and the chosen bilinear form determine the Euclidean norm that is used for sampling and verification.

A module lattice generalises this construction. If R is an order in K, and $M \subset K^r$ is an R-module, define the componentwise embedding

$$\Phi_r : K^r \to \mathbb{R}^{dr}, \qquad \Phi_r(x_1, \ldots, x_r) = (\Phi(x_1), \ldots, \Phi(x_r))$$

Then $\Phi_r(M) \subset \mathbb{R}^{dr}$ is the associated $\mathbb{Z}$ lattice. This is the setting used by many efficient lattice-based schemes, including HAWK, which works over the cyclotomic ring $R_n = \mathbb{Z}[X]/(X^n + 1)$. The ring structure makes polynomial arithmetic efficient, while the coefficient embedding, equivalently the normalised trace form, gives the Euclidean lattice on which norms and sampling distributions are defined [1].

This distinction between algebraic structure and Euclidean geometry is central to this thesis. The same underlying Euclidean lattice may have more than one algebraic description, and different descriptions can expose different cosets, automorphisms, or decompositions. These features may be useful for sampling, but they also constrain how the lattice can be inserted into a signature scheme.

For sampling, the key point is that the probability distribution is defined by the Euclidean norm induced by the chosen embedding and quadratic form. The algebraic module structure is useful because it provides efficient arithmetic and a way to label cosets, but the sampler must still produce vectors with the correct Euclidean distribution. This is exactly the issue that later appears in the comparison between HAWK's coefficientwise mod-2 interface and the internal mod-2 quotient of the E_8 block module.

2.2.4 Search Module LIP

The module version restricts the same idea of LIP to quadratic or Hermitian forms arising from modules over a fixed ring. This follows the quadratic form formalism for LIP based cryptography developed in [2], and the search module LIP formulation used for HAWK [1], [8]. Following the formulation used for HAWK, let K be a CM-field with ring of integers $R = \mathcal{O}_K$, and let $Q \in \mathcal{H}_r^{>0}(K)$ be a positive definite Hermitian $r \times r$ matrix over K. Here $\star$ denotes conjugate transpose. The R-equivalence class of Q is

$$[Q] = \left\{ U^\star Q U \ : \ U \in \mathrm{GL}_r(R) \right\}$$

The worst-case search module lattice isomorphism problem associated with Q, denoted wc-smLIP$^{Q}_{K,r}$ is the following problem:

> **Worst-case search module LIP:** Given an element $Q' \in [Q]$, find a matrix $U \in \mathrm{GL}_r(R)$ such that $Q' = U^\star Q U$.

Thus, the problem is defined by the field K, the rank r and the starting form Q, because the public instances live in the equivalence class $[Q]$. For direct key recovery HAWK works in the rank 2 class generated by the identity form:

$$Q_{\mathrm{HAWK}} = B^\star B = B^\star I_2 B$$

Direct key recovery for HAWK can therefore be viewed as search module LIP in the identity class $[I_2]$ [1], [8].

2.3 Discrete Gaussian Sampling on Lattices

The discrete Gaussian distribution over lattices is the probabilistic object we work with. This section fixes notation and summarises the parameters and correctness issues used later, particularly for HAWK-style signing [1] and for understanding how Gaussian mass, norm behaviour, and tail bounds affect sampler design and calibration [9].

2.3.1 Definitions and Key Parameters

Let $\Lambda \subset \mathbb{R}^n$ be a full rank lattice and let $s > 0$ be a width parameter. Define the Gaussian weight function as

$$\rho_s(x) \;=\; \exp\Big(-\pi\frac{\|x\|^2}{s^2}\Big), \qquad x \in \mathbb{R}^n$$

The associated Gaussian mass of Λ is $\rho_s(\Lambda) \;=\; \sum_{x\in\Lambda}\rho_s(x)$ and the discrete Gaussian distribution $D_{\Lambda,s}$ assigns probability mass

$$\Pr_{X\sim D_{\Lambda,s}}[X = x] \;=\; \frac{\rho_s(x)}{\rho_s(\Lambda)}, \qquad x \in \Lambda$$

More generally, for a coset $\Lambda + c$ with $c \in \mathbb{R}^n$, one defines $D_{\Lambda+c,s}$ by normalising the same Gaussian weight ρ_s over that coset:

$$\Pr[X = x + c] \;=\; \frac{\rho_s(x+c)}{\rho_s(\Lambda + c)}, \qquad x \in \Lambda$$

Coset Gaussians are particularly important in lattice-based signatures, where the signer samples from a lattice coset determined by a message dependent value. This is the setting in which HAWK performs its Gaussian sampling step [1].

In the definition above we use the lattice theory parameter s. In HAWK and in the implementation later in this thesis, we instead use the standard deviation style parameter σ, with weights of the form

$$\exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$$

These conventions are related by $s = \sqrt{2\pi}\,\sigma$.

The width parameter controls both the typical norm of samples and the analytic behaviour of the distribution. In many cryptographic analyses, it is related to the smoothing parameter [9]. With the convention $\rho_s(x) = \exp(-\pi\|x\|^2/s^2)$, the smoothing parameter is defined by

$$\eta_\varepsilon(\Lambda) = \inf\{s > 0 : \rho_{1/s}(\Lambda^* \setminus \{0\}) \le \varepsilon\}$$

Informally, this is a scale beyond which the Gaussian smooths out the dual lattice and gives the statistical closeness properties used in security proofs. Practical sampling depends on carefully chosen parameters like the sampling width, which affects security [4], [9]. In particular, the sampling width should be at least as large as the smoothing parameter of the lattice. If it is too small, the distribution can become highly sensitive to the coset and to implementation details, potentially leaking secret information about the lattice. If it is too large, samples have larger norm, which affects rejection behaviour, verification bounds, and signature size.

Tail behaviour is also important because signing and verification usually impose a norm threshold. For a threshold T, one is interested in quantities such as $\Pr_{X \sim D_{\Lambda+c,s}}\big[\|X\| > T\big]$, since this probability controls how often samples are rejected or fall outside the verification bound. In the results, this is why norm histograms, low-shell behaviour, and rejection rates are reported alongside timing measurements.

For later validation and analysis it is also useful to consider the radial distribution induced by

$D_{\Lambda+c,s}$, meaning the probability that $\|X\|^2$ lies on a given norm shell. In the coset case this is captured by the shifted theta series

$$\Theta_{\Lambda+c}(q) = \sum_{x\in\Lambda} q^{\|x+c\|^2}$$

When the squared norms occurring in the coset are integral, this can be written as

$$\Theta_{\Lambda+c}(q) = \sum_{m\geq 0} N_m(\Lambda+c)q^m, \qquad N_m(\Lambda+c) = \left|\{x\in\Lambda : \|x+c\|^2 = m\}\right|$$

Here $N_m(\Lambda+c)$ counts the number of vectors in the coset $\Lambda+c$ on the shell of squared norm m. Substituting $q = e^{-\pi/s^2}$ gives

$$\rho_s(\Lambda+c) = \Theta_{\Lambda+c}(e^{-\pi/s^2})$$

and therefore

$$\Pr_{X\sim D_{\Lambda+c,s}}\left[\|X\|^2 = m\right] = \frac{N_m(\Lambda+c)e^{-\pi m/s^2}}{\Theta_{\Lambda+c}(e^{-\pi/s^2})}$$

Under the σ-normalisation we use in the implementation, the equivalent substitution is $q = e^{-1/(2\sigma^2)}$.

This theta series viewpoint is useful for remarkable lattices and Construction A lattices because their shell structure is often enough to give predicted low-norm behaviour. We use it to motivate the norm and low-shell diagnostics used for the produced E_8 sampler. [5], [6].

2.3.2 Sampling and Correctness Considerations

For signature applications, the sampler usually targets a message dependent coset, so we focus on the coset Gaussian $D_{\Lambda+c,s}$. A sampler should therefore output vectors that are either exactly distributed according to this distribution, or statistically close to it. Efficient samplers exploit structure in the lattice, such as product decompositions, nested sublattices, module coordinates, cosets, or code-based descriptions, rather than relying on generic high dimensional rejection [9], [11].

The correctness requirements we specify for the sampler in HAWK-E_8-CM are support correctness, distributional correctness, and operational stability. Support correctness means that the sampler outputs vectors in the intended coset. Distributional correctness means that the output should match the intended coset Gaussian, or be negligibly close to it, otherwise repeated signa-

tures could introduce bias or leak information. While operational stability concerns the practical behaviour of the sampler, including rejection rates, norm-bound failures, runtime variation, and sensitivity to parameter choices.

2.4 Remarkable Lattices and Efficient Sampling

We use remarkable lattices as structured sources for discrete Gaussian sampling. In this context, a remarkable lattice is defined by having unusually explicit structure. This includes efficient encoding or decoding procedures, useful decompositions, large automorphism groups, and shell data [5], [6]. They are also relevant to LIP and Module LIP because this structure can affect the associated quadratic form and module-lattice problems. These features make such lattices attractive for sampling, while also meaning that their use in public cryptographic constructions must be treated carefully [2].

For sampling, the most useful feature is often a decomposition into smaller or more structured pieces. For example, a lattice may be described through cosets of a sublattice, a Construction A code, or a recursive family such as the Barnes–Wall lattices. Such descriptions can turn a high dimensional sampling problem into a collection of smaller sampling problems, provided the Gaussian mass of the pieces can be controlled [9], [11]. In the E_8 case studied later, this role is played by the internal mod-2 coset structure of the E_8 block module.

Remarkable lattices can also provide useful validation data. Known theta series, low-norm shell counts, and other invariants give reference values against which sampler output can be compared. This gives stronger diagnostics than simply checking that the sampler produces short vectors [5], [6]. This same structure can also become a cryptanalytic concern when it remains visible in a public cryptographic object. In particular, large automorphism groups or special decompositions may interact with LIP or module LIP type hardness assumptions.

Not every remarkable lattice is suitable for a HAWK-style construction. A useful candidate must admit a compatible module or ideal lattice realisation, expose usable sampling structure, provide enough data for validation, and avoid obvious conflicts with the module LIP security setting. These requirements make E_8 a natural first case study: it is small, highly structured, has well understood shell data, and admits algebraic descriptions related to cyclotomic module lattices [5], [6], [12].

2.5 E_8 as the Running Case Study

2

The previous section explains why remarkable lattices are relevant to sampling and validation. This section records the specific facts about E_8 that are needed later.

The lattice E_8 is the unique positive-definite even unimodular lattice in dimension 8. It is highly symmetric, has a well understood root system, and has clear shell structure, including 240 minimal vectors [5], [6], combining efficiently exploitable geometry and strong diagnostic data. These properties make it useful here because dimension 8 is small enough for blockwise sampling and detailed diagnostics, while still being rich enough to expose the compatibility issues that arise when Euclidean lattice structure is placed inside HAWK's module setting.

A standard coordinate description of E_8 is

$$E_8 = \left\{ x \in \mathbb{Z}^8 : \sum_{i=1}^{8} x_i \in 2\mathbb{Z} \right\} \cup \left\{ x \in (\mathbb{Z} + \tfrac{1}{2})^8 : \sum_{i=1}^{8} x_i \in 2\mathbb{Z} \right\}$$

This presents E_8 as the union of the D_8 root lattice and a shifted half-integer coset, showing its parity and coset structure [5], [6].

The description used later in this thesis follows the congruence defined cyclotomic module description of E_8 given by van Gent and van Woerden [3]. Let $O_8 = \mathbb{Z}[\zeta_8]$, we then have a rank-2 O_8-module $M \subset O_8^2$, whose elements are written as

$$\left(\sum_{i=0}^{3} x_i \zeta_8^i, \ \sum_{i=0}^{3} x_{i+4} \zeta_8^i \right), \qquad x_i \in \mathbb{Z}$$

subject to the congruence condition

$$\sum_{i=0}^{7} x_i \equiv 2x_j \pmod 4 \qquad \text{for all } j = 0, \ldots, 7$$

We use this cyclotomic module description as the starting point for embedding an E_8 block structure into HAWK's ring R_n. For now the key point is that the construction gives algebraic coordinates over O_8, rather than only a Euclidean description of E_8. These coordinates are what later allow the block cosets to be labelled internally.

This structure also has a consequence for sampler design in that the sampling problem becomes blockwise. Each block has dimension 8, so cosets, norms, and low-shell behaviour can be tested at the block level before being assembled into the full HAWK-E_8-CM module.

2.6 The HAWK Signature Scheme

We use HAWK as the cryptographic framework in which to test whether E_8 structure can be integrated into a lattice-based signature scheme. The purpose of this section is to focus on the parts of HAWK that matter for this thesis: the module setting, the signing coset, the Gaussian sampling step, and the public quadratic form used during verification.

2.6.1 HAWK Workflow and the Sampling Step

HAWK is a lattice-based signature scheme built over the cyclotomic ring $R_n = \mathbb{Z}[X]/(X^n + 1)$ with n a power of two. Its secret key contains a structured unimodular basis $B \in \mathrm{GL}_2(R_n)$ and its public key represents a Hermitian quadratic form $Q = B^\star B$. This public form allows the verifier to check the norm of a hidden vector without knowing the secret basis B.

At a high level, HAWK has the usual key generation, signing, and verification workflow of a signature scheme.

- **Key generation:** the scheme generates a secret basis B over R_n, together with public data derived from the form $Q = B^\star B$. The secret basis is used by the signer to move between the public verification coordinates and the sampling coordinates.
- **Signing:** given a message and salt, HAWK hashes to a binary target $h \in (R_n/2R_n)^2$. The signer computes the corresponding parity target $t = Bh \bmod 2$ and samples a short vector from the coefficientwise coset $2R_n^2 + t$. If the sampled vector is denoted by x, the signer sets

$$w = B^{-1}x, \qquad s = \frac{h - w}{2}$$

  The congruence condition $x \equiv Bh \pmod 2$ ensures that $w \equiv h \pmod 2$, so that $s \in R_n^2$.
- **Verification:** the verifier recomputes h, reconstructs $w = h - 2s$ and checks a norm bound using the public quadratic form Q. In other words, the verifier checks a quantity of the form

$$\|w\|_Q^2 = \langle w, Qw \rangle$$

HAWK's sampler is not an arbitrary black box sampler from a lattice coset. It is tied to three specific pieces of structure: the coefficientwise mod-2 target $t = Bh \bmod 2$, the ambient module

R_n^2, and the public quadratic form determined by $Q = B^\star B$. This form induces the verifier's public norm, $\|w\|_Q^2 = \langle w, Qw \rangle$, which is the norm checked during verification.

This tight connection between the sampler, the mod-2 target, and the verification norm is what makes replacing HAWK's sampler by an E_8 sampler non-trivial.

2.6.2 Integration Constraints for Modified Module-Lattice Instantiations

As HAWK's sampling step is tightly connected to its algebraic interface a modified lattice structure cannot therefore be introduced only by replacing the sampling algorithm. It must also fit the module, coset, and norm interfaces used by signing and verification.

The first requirement is module compatibility. HAWK works over $R_n = \mathbb{Z}[X]/(X^n + 1)$, so any replacement lattice structure must be expressible in a form compatible with R_n module arithmetic. This is why the E_8 construction used later is realised as a module inside R_n^2.

The second requirement is coset compatibility. HAWK signing is tied to the binary target $t = Bh \bmod 2$ and to coefficientwise cosets of the form $2R_n^2 + t$. A replacement sampler must therefore explain how this binary target labels the cosets from which it samples. In the E_8 construction, this becomes the coset-matching problem.

The third requirement is norm compatibility. HAWK verification checks shortness using the public form $Q = B^\star B$. If the signing algorithm samples using a different Euclidean geometry, then verification must check the corresponding norm. This is why the E_8 construction leads to a modified public quadratic form.

We therefore must first define a compatible module representation, a corrected coset interface, and a matching public norm to insert an E_8 sampler.

2.6.3 Security Constraints Relevant to HAWK and Structured Variants

HAWK is closely related to the module variant of the lattice isomorphism problem. Its public key represents a structured Hermitian form $Q = B^\star B$, where $B \in \mathrm{GL}_2(R_n)$ is the secret basis. Direct secret key recovery can therefore be viewed as a search module LIP problem in the identity class $[I_2]$: given the public form, recover a module change of basis explaining it [1], [2].

Another vector of attack to consider is signature forgery. Direct key recovery is quite different to signature forgery as a forger does not need to recover B, it only needs to produce a valid signature,

which corresponds to finding a sufficiently short vector in the correct message dependent coset. This is why HAWK's signature security is formulated using a one-more short-vector type problem, written as omSVP [1], [8]. The one-more aspect captures the fact that the adversary may have seen valid signatures before being asked to produce a new one.

This distinction matters for any HAWK-like modification. Changing the lattice geometry used during signing, or changing the public quadratic form used during verification, can change the corresponding module LIP class and the associated one-more short-vector problem. Such a variant therefore does not automatically inherit the security formulation of HAWK.

Additional visible structure must also be treated carefully. Features such as automorphisms, special decompositions, module descriptions, relations with dual or hull-like invariants, and particular choices of field or module rank may give an attacker more information than in a generic module LIP instance [13], [14], [15], [17]. This does not mean that every structured variant is insecure, but that the public form, visible symmetries, and chosen module setting must be part of security considerations.

For our case, the important point is that replacing HAWK's sampling geometry cannot be treated as just an implementation level change. If the modification changes the public norm or exposes additional structure, then the security problem must be restated for the HAWK-E_8-CM construction rather than copied unchanged from HAWK.

2.7 Related Work & Summary

We draw on three strands of related work: module LIP based signatures, discrete Gaussian sampling, and the use of highly structured lattices.

Module LIP signatures and HAWK: As previously stated HAWK is a lattice-based signature scheme built around a module LIP key recovery formulation, using structured module arithmetic to obtain compact signatures and efficient signing [1], [8]. Work on LIP and module LIP provides the security setting in which it should be understood [2], [16]. We build on that framework, but study a modified HAWK-like construction, HAWK-E_8-CM.

Discrete Gaussian sampling: Discrete Gaussian sampling is a central tool in lattice-based signatures. Classical sampling methods, including GPV, Klein, and Peikert-style approaches, provide

the general background for sampling from lattice cosets [10], [18], [19]. More recent work studies how sampling parameters, smoothing behaviour, and implementation constraints affect correctness and efficiency [9]. There is also recent work on samplers that exploit additional algebraic, recursive, and code related structure in specialised lattice families [11]. These works raise the question of whether additional lattice structure can make sampling more organised inside a signature workflow.

Remarkable lattices and their structure: The classical literature on remarkable lattices gives detailed descriptions of lattices such as E_8, including their constructions, automorphism groups, theta series, and shell counts [5], [6]. This is one reason they are useful for us as their structure gives information that can support sampler design and provide checks on sampler output. However, the same structure also has to be treated with care. Work on LIP and module LIP cryptanalysis shows that automorphisms, hull-style information, and particular choices of field or rank can change the hardness picture [13], [14], [15], [17].

Positioning of this thesis: We combine these strands into a case study using E_8 structure inside a HAWK-like signing framework. Starting from the congruence defined $O_8 = \mathbb{Z}[\zeta_8]$ module related to E_8 [3], we develop the module embedding, identify the coset mismatch with HAWK's original interface, and implement a proof of concept coset-matched variant. Our contribution therefore being an analysis of what changes are required for remarkable-lattice structure to fit coherently into a HAWK-style construction and the benefits that arise from this.

3

Analysis & Design

Contents

3.1 High Level Design Overview

The design process for this thesis started from a fairly simple idea: HAWK's signing step depends on sampling, and remarkable lattices such as E_8 have a lot of useful structure. Our original expectation was therefore that this structure might make the sampling stage faster and more organised. At that point, we were thinking mainly in terms of replacing part of HAWK's sampler with something based on a remarkable lattice.

As we developed this idea however, it became clear that this was not only a sampler design problem. HAWK's sampler is tied to the rest of the scheme through its module representation, its mod-2 coset interface, and its public verification norm. This meant that we could not treat the sampler as an isolated component. Before designing a faster sampler, we first had to understand whether an E_8 based lattice could be placed inside HAWK in a way that was compatible with signing and verification.

The final design that emerged is a HAWK-like variant which we call HAWK-E_8-CM. Here, "CM" refers to the coset-matching step that is needed to connect HAWK's binary signing target to the internal mod-2 quotient of the E_8 based module. So the use of E_8 became a scheme level design problem that we needed to solve, before we could begin optimising the sampler.

For notation, let $O_8 = \mathbb{Z}[\zeta_8]$. The E_8 module basis found later is written as the matrix

$$P = \begin{pmatrix} 2(1+\zeta_8) & 1-\zeta_8+\zeta_8^2+\zeta_8^3 \\ 0 & 1+\zeta_8+\zeta_8^2+\zeta_8^3 \end{pmatrix}$$

Inside HAWK's ring $R_n = \mathbb{Z}[X]/(X^n+1)$, with $Y = X^{n/4}$, this becomes

$$P_n = \begin{pmatrix} 2(1+Y) & 1-Y+Y^2+Y^3 \\ 0 & 1+Y+Y^2+Y^3 \end{pmatrix}$$

The detailed derivation of P, P_n, and the relation $M_n = P_n R_n^2$ is given in Chapter 4. In this chapter they are just used to describe the design decisions.

The final design has five main components.

Design component	**Role in the final construction**
E_8 based module embedding	Places repeated copies of a lattice isometric to $2E_8$ inside HAWK's ambient module R_n^2, giving the construction an exact algebraic object to work with.
Internal coset matching	Matches HAWK's target $t = Bh \bmod 2$ to a coset of the E_8 based module using internal P_n coordinates rather than ambient coefficientwise reduction.
Modified public form	Changes the verification form from $Q = B^\star B$ to $Q_{E_8} = B^\star P_n^\star P_n B$, so that the verifier measures the same Euclidean E_8 side norm as the signer.
Construction A block sampler	Samples from the internal $2E_8$ cosets required by the corrected coset interface, using the RM$(1,3)$ description of E_8.
Uncompressed prototype	Implements the arithmetic, coset handling, signing path, verification path, and sampler diagnostics needed to test the construction before compression or production optimisation.

Table 3.1: High level components of the final HAWK-E_8-CM design.

One important part of the design process was realising that the E_8 structure has to be handled blockwise. Initially, we were thinking more generally about how a remarkable lattice could improve HAWK's sampling step. The blockwise structure became clear only after embedding the E_8 based module into HAWK's ring. Since $R_n = \mathbb{Z}[X]/(X^n+1)$ contains copies of $O_8 = \mathbb{Z}[\zeta_8]$ through the

substitution $\zeta_8 \mapsto X^{n/4}$, the resulting module naturally decomposes into $n/4$ separate $2E_8$ blocks. This is what turns the sampler into a blockwise sampler, with one internal 8-bit coset label for each block.

The final signing flow can be summarised as

$$\boxed{h \longmapsto t = Bh \bmod 2 \longmapsto x_E \in C_t \longmapsto z = P_n^{-1} x_E \longmapsto w = B^{-1} z \longmapsto s = \frac{h-w}{2}}$$

Here C_t is the internally matched E_8 block coset. Verification then reconstructs $w = h - 2s$ and checks the modified public norm $\|w\|^2_{Q_{E_8}} = \langle w, Q_{E_8} w \rangle$.

This makes clear why our final design is not just inserting a faster sampler.

The rest of this chapter explains the constraints that caused this shift, the design decisions made in response, and the boundary between the completed prototype and the work left for future development.

3.2 Design Constraints and Key Decisions

The first task was deciding which remarkable lattice was worth studying. E_8 became the chosen candidate as it is small enough to compute with explicitly, has a well understood structure, has an efficient decoding algorithm, and has enough symmetry and shell information to make sampler design and validation possible.

After choosing E_8, the next question was how to fit it into HAWK's algebraic setting. HAWK works over the cyclotomic ring R_n, so we needed an E_8 based object that could be expressed as a module compatible with this ring. This led to researching and then the finding of a rank-two O_8 module from existing literature [3]. We then took this definition and identified its underlying $\mathbb{Z}$-lattice with $2E_8$, and then searched for an honest O_8 basis. Finding the basis was a turning point, because it gave a matrix P, and therefore a corresponding matrix P_n inside HAWK's ring.

At that point the problem changed from "can we use E_8?" to "what does using E_8 mean for HAWK?" The main constraints that emerged were the coset interface, the verification norm, the sampler structure, and the boundary of what we could reasonably implement in a prototype.

3.2.1 Coset Interface

The first major constraint after finding the E_8 module basis was the coset interface. Initially, we expected that once the E_8 based module had been embedded into HAWK's ring, HAWK's existing binary target could be used directly by a new sampler. This was a reasonable first assumption, because HAWK already computes a mod-2 target during signing.

The problem was that there are two different mod-2 structures involved. HAWK uses coefficientwise reduction in the ambient module, while the E_8 block module has its own internal quotient. These two quotient maps do not agree. This meant that the first sampler we designed was incomplete, being based on the wrong coset interface.

At this point we had two possible design directions. The first was to keep HAWK's coefficientwise interface and accept that the E_8 module would only be used in a limited way. This would have kept the construction closer to HAWK, but it would also lose the main coset information that makes the E_8 block structure useful. The ambient coefficientwise reduction would not recover the full internal quotient of the E_8 block module, so the sampler would not be given the correct E_8 coset labels.

The second option was to change the interface so that HAWK's target is interpreted through the internal coordinates of the E_8 block module. We chose this option because it preserves the full E_8 coset information and gives the sampler a clean blockwise target. This was more invasive from a scheme design point of view, but it was necessary if the sampler was going to make genuine use of the E_8 block structure rather than only using E_8 as a superficial coordinate description.

This decision changed the project significantly. The E_8 sampler could no longer be treated as a replacement for HAWK's existing sampler. Instead, the design needed a coset-matching layer between HAWK's signing target and the E_8 sampler. After extending the O_8 module into R_n, the construction decomposes into repeated $2E_8$ blocks, each with their own internal coset label.

3.2.2 Verification Norm

The second constraint was the verification norm. Once the coset interface had been corrected, we still had to decide what geometry the sampler should use. The verifier must check the same norm that the signer used when producing the sample.

There were two possible design choices here, the first being to keep HAWK's original public

form $Q = B^\star B$. This would have kept the construction closer to ordinary HAWK, which was appealing from an integration point of view. However, it would also mean that an embedded E_8 side vector $x_E = P_n z$ would not be measured using its ordinary Euclidean E_8 block norm. Instead, the relevant norm would be the pulled back norm $\|P_n^{-1} x_E\|^2$. This gives a skewed geometry rather than the clean Euclidean geometry of the E_8 blocks. Removing much of the original motivation for using E_8, as the sampler would no longer be exploiting the Euclidean E_8 geometry. It would mainly be using E_8 as a different coordinate system.

The second option was to let the sampler use the ordinary Euclidean norm of the embedded vector, $\|x_E\|^2 = \|P_n z\|^2$. This choice better matched the purpose of this thesis, since the aim is to investigate whether the structure of a remarkable lattice could give a more organised and efficient sampling problem. The cost is that the public verification form must change. The verifier must use

$$B^\star P_n^\star P_n B = Q_{E_8} = B^\star S_n B$$

We chose the second option. Although it moves the construction further away from HAWK, it makes the role of E_8 mathematically meaningful. The signer samples using the Euclidean E_8 block geometry, and the verifier checks exactly that same geometry through the modified public form. Unfortunately, the modified public form also leaves a separate theoretical task as the security argument for HAWK must be revisited for the HAWK-E_8-CM setting.

3.2.3 Sampler Structure

The sampler is the component that turns the internal binary coset label produced by the HAWK-E_8-CM interface into an actual lattice vector. So unlike HAWK which has a coordinatewise parity problem, it is a blockwise sampling problem over E_8 derived cosets. For each $2E_8$ block, the sampler receives an internal label $\tau \in (O_8/2O_8)^2$ and samples from the corresponding coset $C_\tau = 2M + P\tau$. The corrected coset interface gives labels in the internal P coordinates of the E_8 module.

The design choice was therefore which E_8 aware sampling strategy to use. One possible approach would have been to adapt a dedicated root-lattice sampler for E_8, such as the framework of Espitau, Wallet and Yu [9]. Their work includes a sampler for E_8, based on the decomposition $E_8 = D_8 \cup \left(\frac{1}{2}, \ldots, \frac{1}{2}\right) + D_8$, together with their general lattice extension sampling framework.

We chose a Construction A description instead because it gave a very clear and implementation friendly route from the corrected internal coset label to an actual block sample, as we require

samples from $2E_8$ cosets. In the normalisation we use, each E_8 block is described using the binary code RM$(1, 3)$, the same as the extended Hamming $[8, 4, 4]$ code. This represents the block as a union of 16 product-lattice cosets.

This structure also fits the global decomposition $M_n \cong (2E_8)^{\oplus n/4}$. Different E_8 blocks can be sampled independently, which gives a place to use parallelism effectively. Within each block, the sampler chooses one of the RM$(1, 3)$ code components according to its Gaussian mass, samples the shifted one-dimensional coordinates inside that component, and maps the result back through the E_8 module basis. The implementation adapts this procedure to the internal P coordinate cosets required by HAWK-E_8-CM.

3.2.4 Prototype Scope and Boundary

The final design decision was deciding what should be implemented within the scope of our code implementation. Some major difficulties stood out such as: a constant time implementation, compressed signatures, compact public-key encoding, side-channel analysis, and full signature level optimisations.

We therefore decided to treat the implementation as a prototype of our uncompressed scheme, rather than as a production signature scheme. We then separated the work into two, mathematical and engineering, making it more manageable and allowing us to flexibly achieve as much as we could. We started with implementing the mathematical and scheme interface: the E_8 module embedding, the corrected internal coset interface, the modified public form, and the corresponding signing and verification equations. This layer had to be tested and completely correct first, as performance measurements would not be meaningful if the underlying coset handling or verification norm were still incorrect.

The engineering work was then built around this. We initially used a simpler sampler to validate the algebraic path through signing and verification. After the interface was working, we replaced this with a Construction A sampler, based on a sampler we had first explored in SageMath at the beginning of our work on this thesis, and adapted it to sample from the corrected internal E_8 block cosets required by HAWK-E_8-CM.

Keeping our implementation uncompressed has a trade-off. It makes the prototype less compact and not production ready, but it also made the core scheme logic easier to test. In particular, it avoids mixing the mathematical questions with additional complications from signature com-

pression, fixed point reconstruction, and compact encoding. These components are important for a practical scheme, but they sit on top of the basic signing and verification interface and so we deemed them out of scope.

The resulting code therefore demonstrates an E_8 based module construction can in fact be connected to HAWK-style signing and verification correctly. We then build our testing around this. Compression, side-channel analysis and constant time hardening are left as future work.

Area	**Completed in this thesis**	**Left outside scope**
Lattice choice	Developed E_8 as the remarkable lattice case study.	No larger survey of remarkable lattices is completed.
Module embedding	Constructed $M \cong 2E_8$, found an O_8 basis, and extended it to $M_n = P_n R_n^2 \subset R_n^2$.	Other candidate lattices require separate module and embedding analyses.
Coset interface	Defined the internal P_n coordinate coset map for HAWK-E_8-CM.	A drop in replacement for HAWK's coefficientwise mod-2 sampler is not possible.
Verification form	Changed the public form to $Q_{E_8} = B^\star P_n^\star P_n B$, proved the norm identity, and stated the modified security setting.	A complete HAWK-style unforgeability proof and dedicated cryptanalysis of the public form class.
Implementation	Implemented an uncompressed signing and verification path with a coset matched Construction A E_8 block sampler.	Compressed signatures and production integration are not completed.
Engineering	Tested correctness, norms, rejection behaviour, and timing of the prototype.	A constant time, side-channel hardened production sampler is not implemented.

Table 3.2: Scope of the HAWK-E_8-CM prototype and work left beyond this thesis.

The next chapter gives the mathematical and implementation details behind these design decisions.

3

4 Implementation

Contents

4.1 The Cyclotomic E_8 Module and Its Extension to HAWK

4.1.1 The Starting Module and the Normalization

4

We begin with the cyclotomic field $K_8 := \mathbb{Q}(\zeta_8)$ and its ring of integers $O_8 := \mathbb{Z}[\zeta_8]$. The module we start from is the subset $M \subset O_8^2$ defined by the congruence condition [3].

$$M = \left\{ \left(\sum_{i=0}^{3} x_i \zeta_8^i,\ \sum_{i=0}^{3} x_{i+4} \zeta_8^i \right) \in O_8^2 \ \middle|\ \sum_{i=0}^{7} x_i \equiv 2x_0 \equiv 2x_1 \equiv \cdots \equiv 2x_7 \pmod 4 \right\}$$

So an element of M is really an element of O_8^2, but written in terms of its eight integer coefficients

$$(x_0, \ldots, x_7) \in \mathbb{Z}^8$$

The congruence condition says that these coefficients are not arbitrary and depend on the given condition which we will talk more about below.

This is the starting point of the construction. The goal of the following sections is to unpack this congruence defined object into a more explicit lattice description.

For normalization, we use the same convention as HAWK, where HAWK works over a power of two cyclotomic ring $R_n = \mathbb{Z}[X]/(X^n + 1)$. For $f, g \in R_n$, HAWK uses the involution $f \mapsto f^\star$, induced by $X \mapsto X^{-1}$, and the normalized trace inner product

$$\langle f, g \rangle = \frac{1}{n} \operatorname{Tr}(f^\star g)$$

Under this normalization, the trace inner product agrees with the usual Euclidean inner product on coefficient vectors. This is useful because it means that, when studying the coefficient vector $(x_0, \ldots, x_7)$ we can use the ordinary Euclidean geometry of $\mathbb{Z}^8$. [8]

Finally, M is not just a subset of O_8^2, it is an O_8 submodule. The only point that needs checking is that the congruence condition is stable under multiplication by ζ_8, since $O_8 = \mathbb{Z}[\zeta_8]$. Multiplication by ζ_8 cyclically shifts the coefficients in each block, with a sign change:

$$(x_0, x_1, x_2, x_3) \longmapsto (-x_3, x_0, x_1, x_2)$$

Applied to both components of O_8^2, this sends

$$(x_0, \ldots, x_7) \longmapsto y = (-x_3, x_0, x_1, x_2, -x_7, x_4, x_5, x_6)$$

Let $S = \sum_{i=0}^{7} x_i$. The defining condition says $S \equiv 2x_i \pmod 4$ for every $i = 0, \ldots, 7$. In particular, all the x_i have the same parity, since

$$2x_i \equiv 2x_j \pmod 4 \quad \Longleftrightarrow \quad x_i \equiv x_j \pmod 2$$

The sum of the transformed coefficients is $\sum_{i=0}^{7} y_i = S - 2x_3 - 2x_7$. Since x_3 and x_7 have the same parity, $x_3 + x_7$ is even, and hence $2x_3 + 2x_7 \equiv 0 \pmod 4$. Therefore

$$\sum_{i=0}^{7} y_i \equiv S \pmod 4$$

Also, each coefficient y_i is either one of the original x_j, or the negative of one of them. In either case $2y_i \equiv 2x_j \pmod 4$ for some j. Hence

$$\sum_{i=0}^{7} y_i \equiv 2y_i \pmod 4 \qquad \text{for every } i$$

So multiplication by ζ_8 preserves the defining congruence condition. Since the condition is also closed under addition and integer multiplication, M is stable under the action of $\mathbb{Z}[\zeta_8]$, and is therefore an O_8 submodule of O_8^2.

4.1.2 Recovering the Underlying $\mathbb{Z}$-lattice

Forget the O_8 module structure for a moment and look at the same object just as an ordinary $\mathbb{Z}$-lattice. An element of M can be written by its coefficient vector $x = (x_0, \ldots, x_7) \in \mathbb{Z}^8$ using the basis $1, \zeta_8, \zeta_8^2, \zeta_8^3$ on each copy of O_8. Since HAWK's normalized trace inner product agrees with the usual Euclidean inner product on coefficient vectors, the underlying $\mathbb{Z}$-lattice is exactly the subset of $\mathbb{Z}^8$ cut out by the congruence condition. [8]

To compare this lattice with E_8, we divide all coordinates by 2. So set $y_i = \frac{x_i}{2}$. From the congruence condition, all of the x_i must have the same parity. Indeed, for any i, j, $2x_i \equiv 2x_j \pmod 4$ and this is equivalent to $x_i \equiv x_j \pmod 2$. Therefore, after setting $y_i = \frac{x_i}{2}$ there are only

two possibilities:

$$y \in \mathbb{Z}^8 \qquad \text{or} \qquad y \in \left(\mathbb{Z} + \frac{1}{2}\right)^8$$

Now again let $S = \sum_{i=0}^{7} x_i$. Since all the x_i have the same parity and there are eight coordinates, S is always even. Hence $\sum_{i=0}^{7} y_i = \frac{S}{2}$ is an integer. The original congruence condition $S \equiv 2x_j \pmod 4$ is equivalent to $S - 2x_j \in 4\mathbb{Z}$. Dividing by 2, this becomes $\frac{S}{2} - x_j \in 2\mathbb{Z}$ or equivalently

$$\sum_{i=0}^{7} y_i \equiv x_j \pmod 2$$

This gives the two cases:

$$\begin{aligned} &\text{if all } x_i \text{ are even}, \quad x_j \in 2\mathbb{Z}, \text{ so } y \in \mathbb{Z}^8 \text{ and } \textstyle\sum_i y_i \in 2\mathbb{Z} \\ &\text{if all } x_i \text{ are odd}, \quad x_j \in 2\mathbb{Z}+1, \text{ so } y \in \left(\mathbb{Z}+\tfrac{1}{2}\right)^8 \text{ and } \textstyle\sum_i y_i \in 2\mathbb{Z}+1 \end{aligned}$$

Thus the divided lattice is

$$\Gamma_8' = \left\{ y \in \mathbb{Z}^8 \;\middle|\; \sum_i y_i \in 2\mathbb{Z} \right\} \cup \left\{ y \in \left(\mathbb{Z}+\tfrac{1}{2}\right)^8 \;\middle|\; \sum_i y_i \in 2\mathbb{Z}+1 \right\}$$

Conversely, every $y \in \Gamma_8'$ gives an element $x = 2y \in M$. Hence $M \cong 2\Gamma_8'$.

Now compare Γ_8' with the usual description of the E_8 lattice,

$$E_8 = \left\{ z \in \mathbb{Z}^8 \cup \left(\mathbb{Z}+\tfrac{1}{2}\right)^8 \;\middle|\; \sum_i z_i \in 2\mathbb{Z} \right\}$$

The integer part of Γ_8' already agrees with the integer part of E_8. The only difference is in the half-integer part: in Γ_8', the sum is odd, while in the standard model of E_8, the sum is even.

This difference is removed by changing the sign of one coordinate. We define

$$\phi : \mathbb{R}^8 \longrightarrow \mathbb{R}^8, \qquad \phi(y_0, y_1, \ldots, y_7) = (-y_0, y_1, \ldots, y_7)$$

This map preserves the Euclidean inner product, since for any $u, v \in \mathbb{R}^8$,

$$\langle \phi(u), \phi(v) \rangle = (-u_0)(-v_0) + \sum_{i=1}^{7} u_i v_i = \sum_{i=0}^{7} u_i v_i = \langle u, v \rangle$$

So ϕ is an isometry.

Then check what ϕ does to the parity condition. If $y \in \mathbb{Z}^8$, then $\phi(y) \in \mathbb{Z}^8$, and

$$\sum_i \phi(y)_i = -y_0 + \sum_{i=1}^{7} y_i = \sum_i y_i - 2y_0$$

Since $2y_0$ is even, the parity of the total sum is unchanged. Therefore the integer part with even sum is sent to itself.

Now suppose $y \in \left(\mathbb{Z} + \frac{1}{2}\right)^8$. Then $\phi(y)$ is still in $\left(\mathbb{Z} + \frac{1}{2}\right)^8$, but now

$$\sum_i \phi(y)_i = \sum_i y_i - 2y_0$$

Since $y_0 \in \mathbb{Z} + \frac{1}{2}$, we have $2y_0 \in 2\mathbb{Z} + 1$. So subtracting $2y_0$ changes the parity of the total sum. Therefore if $\sum_i y_i \in 2\mathbb{Z} + 1$ then $\sum_i \phi(y)_i \in 2\mathbb{Z}$. Thus ϕ sends the half-integer part of Γ'_8 exactly to the half-integer part of E_8.

Hence $\phi(\Gamma'_8) = E_8$. Since ϕ is an isometry, Γ'_8 is isometric to E_8. Therefore the original congruence module is E_8, scaled by a factor of 2:

$$\boxed{M \cong 2E_8}$$

Thus vector lengths are scaled by a factor of 2, and Gram matrices are scaled by a factor of 4.

4.1.3 An Honest O_8 basis

So far, M has been described by a congruence condition inside O_8^2. This is useful for recognising the lattice, but it is not yet a convenient module basis. We would like to at least get a pseudo-basis from generators in K_8^2 and coefficient ideals, but in fact we can do better here and find two explicit vectors $u, v \in O_8^2$ such that every element of M can be written uniquely as

$$au + bv, \qquad a, b \in O_8$$

In other words, we want to prove that M is a free rank-2 O_8 module with basis $\{u, v\}$.

We choose the two vectors

$$u := \big(2(1 + \zeta_8),\, 0\big), \quad \text{and} \quad v := \big(1 - \zeta_8 + \zeta_8^2 + \zeta_8^3,\, 1 + \zeta_8 + \zeta_8^2 + \zeta_8^3\big)$$

The reason for choosing u and v in this form is that they give an upper-triangular basis matrix:

$$P = \begin{pmatrix} 2(1+\zeta_8) & 1-\zeta_8+\zeta_8^2+\zeta_8^3 \\ 0 & 1+\zeta_8+\zeta_8^2+\zeta_8^3 \end{pmatrix}$$

This makes the determinant especially easy to compute, because $\det P = 2(1+\zeta_8)\big(1+\zeta_8+\zeta_8^2+\zeta_8^3\big)$ First, check that u and v really lie in M. Their coefficient vectors are

4

$$u \leftrightarrow (2,2,0,0,0,0,0,0), \quad \text{and} \quad v \leftrightarrow (1,-1,1,1,1,1,1,1).$$

Both satisfy the congruence condition for M. Since M is an O_8 submodule, this means that the O_8 span $L := O_8u + O_8v$ is contained in M: $L \subseteq M$.

But we still need to show that L is not a proper submodule of M. To do this, we compare covolumes as ordinary $\mathbb{Z}$-lattices. Multiplication by the matrix P gives the sublattice $L = PO_8^2 \subseteq O_8^2$. The corresponding $\mathbb{Z}$-index is given by the absolute norm of $\det P$:

$$[O_8^2 : L] = \big|N_{K_8/\mathbb{Q}}(\det P)\big|$$

Since P is upper triangular, $\det P = 2(1+\zeta_8)\big(1+\zeta_8+\zeta_8^2+\zeta_8^3\big)$. Therefore

$$\big|N_{K_8/\mathbb{Q}}(\det P)\big| = \big|N_{K_8/\mathbb{Q}}\big(2(1+\zeta_8)\big)\big| \cdot \big|N_{K_8/\mathbb{Q}}\big(1+\zeta_8+\zeta_8^2+\zeta_8^3\big)\big|$$

We now compute the two factors. Since $K_8 = \mathbb{Q}(\zeta_8)$ has degree $\varphi(8) = 4$ the norm of the rational integer 2 is $N_{K_8/\mathbb{Q}}(2) = 2^4 = 16$. This is because 2 is fixed by every embedding of K_8 into $\mathbb{C}$, so its norm is just the product $2 \cdot 2 \cdot 2 \cdot 2 = 16$.

Next, we compute $N_{K_8/\mathbb{Q}}(1+\zeta_8)$. The field $K_8 = \mathbb{Q}(\zeta_8)$ is a cyclotomic field, so its embeddings into $\mathbb{C}$ are determined by where they send ζ_8. An embedding must send ζ_8 to another primitive eighth root of unity. These are exactly

$$\zeta_8^a \qquad \text{for } a \in (\mathbb{Z}/8\mathbb{Z})^\times = \{1,3,5,7\}$$

Equivalently, $\mathrm{Gal}(K_8/\mathbb{Q}) \cong (\mathbb{Z}/8\mathbb{Z})^\times$ with automorphisms

$$\sigma_a(\zeta_8) = \zeta_8^a, \qquad a \in \{1,3,5,7\}$$

Therefore the conjugates of $1+\zeta_8$ are

$$\sigma_a(1+\zeta_8) = 1+\zeta_8^a, \qquad a \in \{1,3,5,7\}$$

So

$$N_{K_8/\mathbb{Q}}(1+\zeta_8) = \prod_{a\in\{1,3,5,7\}} (1+\zeta_8^a) = (1+\zeta_8)(1+\zeta_8^3)(1+\zeta_8^5)(1+\zeta_8^7)$$

Now pair the conjugates as complex conjugates:

$$(1+\zeta_8)(1+\zeta_8^7) = 1+\zeta_8+\zeta_8^7+\zeta_8^8$$

Since $\zeta_8^8 = 1$, this becomes

$$(1+\zeta_8)(1+\zeta_8^7) = 2+\zeta_8+\zeta_8^{-1}$$

Using $\zeta_8+\zeta_8^{-1} = 2\cos\left(\frac{\pi}{4}\right) = \sqrt{2}$ we get

$$(1+\zeta_8)(1+\zeta_8^7) = 2+\sqrt{2}$$

Similarly, $(1+\zeta_8^3)(1+\zeta_8^5) = 2+\zeta_8^3+\zeta_8^{-3} = 2+2\cos\left(\frac{3\pi}{4}\right) = 2-\sqrt{2}$.

Therefore

$$N_{K_8/\mathbb{Q}}(1+\zeta_8) = (2+\sqrt{2})(2-\sqrt{2}) = 4-2 = 2$$

Hence

$$\left|N_{K_8/\mathbb{Q}}\big(2(1+\zeta_8)\big)\right| = N_{K_8/\mathbb{Q}}(2)\, N_{K_8/\mathbb{Q}}(1+\zeta_8) = 16\cdot 2 = 32$$

For the second factor, use the geometric series identity

$$1+\zeta_8+\zeta_8^2+\zeta_8^3 = \frac{\zeta_8^4-1}{\zeta_8-1}$$

Since $\zeta_8^4 = -1$, this becomes

$$1+\zeta_8+\zeta_8^2+\zeta_8^3 = \frac{-2}{\zeta_8-1} = \frac{2}{1-\zeta_8}$$

We also need the norm of $1-\zeta_8$. Using the same conjugates as above,

$$N_{K_8/\mathbb{Q}}(1-\zeta_8) = (1-\zeta_8)(1-\zeta_8^3)(1-\zeta_8^5)(1-\zeta_8^7)$$

Pairing complex conjugates gives $(1-\zeta_8)(1-\zeta_8^7) = 2-\zeta_8-\zeta_8^{-1} = 2-\sqrt{2}$ and $(1-\zeta_8^3)(1-\zeta_8^5) = 2-\zeta_8^3-\zeta_8^{-3} = 2+\sqrt{2}$.

Hence

$$N_{K_8/\mathbb{Q}}(1-\zeta_8) = (2-\sqrt{2})(2+\sqrt{2}) = 2$$

Therefore

$$N_{K_8/\mathbb{Q}}\big(1+\zeta_8+\zeta_8^2+\zeta_8^3\big) = \frac{N_{K_8/\mathbb{Q}}(2)}{N_{K_8/\mathbb{Q}}(1-\zeta_8)} = \frac{16}{2} = 8$$

4

So altogether,

$$\big|N_{K_8/\mathbb{Q}}(\det P)\big| = 32 \cdot 8 = 256$$

Thus the O_8 span $L = O_8u + O_8v$ has $\mathbb{Z}$-covolume 256. From the previous subsection, we also know that $M \cong 2E_8$. Since E_8 is unimodular, its covolume is 1. Scaling an eight-dimensional lattice by 2 multiplies its covolume by 2^8, so $\mathrm{covol}(M) = 2^8 = 256$. Therefore L and M have the same covolume. Since we already know that $L \subseteq M$ they must be equal:

$$\boxed{M = O_8u \oplus O_8v = PO_8^2}$$

So u and v form an honest O_8 basis of M. This is stronger than only giving a pseudo-basis. In pseudo-basis language it is the special case where both coefficient ideals are the full ring $\big((u, O_8), (v, O_8)\big)$. The corresponding pseudo-matrix is $(P; (O_8, O_8))$. But in this construction the most accurate description is simply that

$$\boxed{\{u, v\} \text{ is an } O_8 \text{ basis of } M}$$

This also clarifies the scaling. The integral module inside O_8^2 naturally gives $2E_8$, not the unit normalized E_8. If we wanted the unscaled E_8 Gram matrix instead of $4C_{E_8}$, where C_{E_8} is an E_8 Cartan matrix, we would have to divide the generators by 2. That would move us out of the integral module O_8^2 and into the fractional vector space K_8^2 so it would no longer match the integral module interface needed for the later HAWK embedding, which we do not want.

4.1.4 Extending Scalars into HAWK's ring

We now want to move from the small cyclotomic ring $O_8 = \mathbb{Z}[\zeta_8]$ to the larger ring used in HAWK [8]. Let n be the HAWK degree, so $n \in \{256, 512, 1024\}$ and let $R_n = \mathbb{Z}[X]/(X^n+1)$.

The key point is that R_n contains a copy of O_8. To see this more clearly we set $k := \frac{n}{4}$. Then X^k behaves like an eighth root of unity inside R_n, because

$$(X^k)^4 + 1 = X^n + 1 = 0 \qquad \text{in } R_n$$

Therefore we can embed O_8 into R_n by sending $\zeta_8 \longmapsto X^k$. So we have a ring embedding

$$\iota_n : O_8 \hookrightarrow R_n, \qquad \iota_n(\zeta_8) = X^k$$

This is the algebraic bridge between the O_8 module M and HAWK's actual cyclotomic ring.

Next, we describe R_n as an O_8 module. Every exponent $0 \le m < n$ can be written uniquely in the form

$$m = qk + r, \qquad 0 \le q \le 3, \qquad 0 \le r < k$$

Using this, every element of R_n can be written uniquely as

$$f(X) = \sum_{r=0}^{k-1} X^r a_r(X^k)$$

where

$$a_r(T) \in \mathbb{Z}[T]/(T^4 + 1) \cong O_8$$

In other words, R_n is a free O_8 module of rank k, with basis $1,\ X,\ X^2,\ \ldots,\ X^{k-1}$. This means that R_n consists of k copies of O_8, shifted by powers of X.

Now we extend the module M from O_8 to R_n. A clean way to do this is by extension of scalars:

$$M_n := R_n \otimes_{O_8} M$$

Since $M \subset O_8^2$, this gives $M_n \subset R_n \otimes_{O_8} O_8^2 \cong R_n^2$. So M_n is now a rank 2 module inside the same ambient space R_n^2 that HAWK uses.

From the previous subsection, we know that $M = PO_8^2$ where

$$P = \begin{pmatrix} 2(1+\zeta_8) & 1 - \zeta_8 + \zeta_8^2 + \zeta_8^3 \\ 0 & 1 + \zeta_8 + \zeta_8^2 + \zeta_8^3 \end{pmatrix}$$

To obtain the corresponding matrix over R_n simply apply the embedding $\zeta_8 \mapsto X^k$. This gives

$$P_n = \begin{pmatrix} 2(1+X^k) & 1 - X^k + X^{2k} + X^{3k} \\ 0 & 1 + X^k + X^{2k} + X^{3k} \end{pmatrix}$$

Therefore the extended module is $M_n = P_n R_n^2$.

4

Equivalently, M_n is generated over R_n by the two vectors

$$u_n = \big(2(1+X^k),\, 0\big)$$

and

$$v_n = \big(1 - X^k + X^{2k} + X^{3k},\; 1 + X^k + X^{2k} + X^{3k}\big)$$

So the same two generator structure from O_8^2 carries over directly to HAWK's ring.

A useful way to view this construction is that since R_n is a free O_8 module with basis

$$1,\; X,\; X^2,\; \ldots,\; X^{k-1}$$

the extended module decomposes as

$$M_n = \bigoplus_{r=0}^{k-1} X^r M \subset R_n^2$$

So, as an ordinary $\mathbb{Z}$-lattice, M_n is just k shifted copies of M. Since $M \cong 2E_8$ we get $M_n \cong M^{\oplus k} \cong (2E_8)^{\oplus k}$. Because $k = n/4$, this gives $M_n \cong (2E_8)^{\oplus n/4}$.

For the standard HAWK parameter sets, this becomes

$$(2E_8)^{64} \subset R_{256}^2, \qquad (2E_8)^{128} \subset R_{512}^2, \qquad (2E_8)^{256} \subset R_{1024}^2$$

For example, in the HAWK-512 case, $n = 512, \quad k = \frac{512}{4} = 128$. So the two R_{512} module generators are

$$\big(2(1+X^{128}), 0\big), \quad \big(1 - X^{128} + X^{256} + X^{384},\; 1 + X^{128} + X^{256} + X^{384}\big)$$

This also fits naturally with HAWK's geometry. As established above, HAWK's normalized trace inner product agrees with the Euclidean inner product on coefficient vectors. The public quadratic form is $Q = B^\star B$ so $\|Bf\|^2 = \|f\|_Q^2$. Thus lengths measured by the public form

correspond to ordinary Euclidean lengths after applying B. For this reason, the extended module M_n should be viewed as an E_8 block lattice on the ambient coefficient side, or x-side, of HAWK's geometry. [8]

So the main result we have so far is

$$\boxed{M_n = P_n R_n^2 \subset R_n^2, \qquad \text{with underlying } \mathbb{Z}\text{-lattice } (2E_8)^{\oplus n/4}}$$

4.1.5 Outcome of the Module Embedding

The construction above gives the algebraic object we needed. The congruence module has underlying lattice $M \cong 2E_8$, admits the O_8 basis $M = PO_8^2$, and extends to $M_n = P_n R_n^2 \subset R_n^2$. As a $\mathbb{Z}$-lattice, this extended module satisfies $M_n \cong (2E_8)^{\oplus n/4}$. Thus the E_8 block embedding inside HAWK's ambient module is fixed.

However, this does not yet make the sampler a replacement for HAWK's sampler. HAWK signing is tied to the target $t = Bh \bmod 2$ and to coefficientwise cosets $2R_n^2 + t$. The remaining problem is to interpret this target as a compatible internal coset label for $M_n/2M_n$. The next section separates this internal quotient from ambient coefficientwise reduction, explains why these two maps are not the same, and defines the coordinate map based on P_n for labelling the E_8 block cosets.

4.2 The Modulo 2 Coset Interface for the E_8 Embedding

The previous section leaves us with one specific problem: the lattice embedding has been constructed, but the mod-2 interface with HAWK's signing procedure has not yet been matched.

Something that is easy to miss is that there are two different ways to reduce modulo 2. The first is the internal quotient $M_n/2M_n$. This is the quotient that belongs naturally to the E_8 block module itself and it remembers the mod-2 coordinates of a vector inside M_n.

The second is ambient coefficientwise reduction after viewing $M_n \subset R_n^2$. This simply reduces the coefficients of the embedded vector modulo 2 in the larger HAWK module R_n^2. These two reductions seem similar, but they are not the same map.

This distinction is already visible in one $2E_8$ block. $M/2M$ has 2^8 classes, as you would expect

for an eight-dimensional lattice. But if we reduce the embedded coefficient vectors of $M \subset O_8^2 \cong \mathbb{Z}^8$ coefficient by coefficient modulo 2, then the original congruence condition forces all coordinates to have the same parity. So coefficientwise reduction only sees the two patterns

$$(0, \ldots, 0), \qquad (1, \ldots, 1)$$

Thus coefficientwise reduction loses most of the mod-2 information. The aim of this section is to fix this mismatch:

$$\text{Given } t = Bh \bmod 2, \text{ which coset of } M_n/2M_n \text{ should the } E_8 \text{ block sampler use?}$$

The answer is to use the coordinates coming from the basis $M_n = P_n R_n^2$, rather than ambient coefficientwise reduction in R_n^2. That is, the correct mod-2 label is obtained by writing the vector in its P_n coordinates and reducing those coordinates modulo 2. This gives the correct mod-2 quotient for the E_8 block module.

4.2.1 The Correct Mod-2 Quotient on M_n

Writing $\overline{R}_n := R_n/2R_n$. Since P_n gives an actual R_n basis of M_n, every element of M_n can be written uniquely in the form

$$x = P_n z, \qquad z \in R_n^2$$

So z should be thought of as the coordinate vector of x with respect to the P_n-basis. This gives a mod-2 map

$$\pi_M : M_n \longrightarrow \overline{R}_n^2, \qquad \pi_M(P_n z) := z \bmod 2$$

π_M first writes x in P_n coordinates and then reduces those coordinates modulo 2.

This map is well defined because the expression $x = P_n z$ is unique. In other words, P_n is injective as a map $P_n : R_n^2 \longrightarrow M_n$. So there is no ambiguity in which z should be used.

The kernel of π_M is exactly $2M_n$. To see this, suppose $x = P_n z$ and $\pi_M(x) = 0$ then $z \equiv 0 \pmod 2$, so $z = 2u$ for some $u \in R_n^2$. Therefore $x = P_n(2u) = 2(P_n u) \in 2M_n$. Conversely, every element of $2M_n$ clearly maps to 0, because if $x = 2P_n u$ then $x = P_n(2u)$ and the coordinate vector $2u$ is zero modulo 2. Hence $\ker(\pi_M) = 2M_n$.

Therefore π_M gives the quotient $M_n/2M_n \cong \overline{R}_n^2$. This is the important point. The internal

mod-2 quotient of M_n has the same size as HAWK's usual parity space:

$$|M_n/2M_n| = \left|\overline{R}_n^2\right| = 2^{2n}$$

So, the E_8 block module has exactly the right number of mod-2 cosets. The problem was not that $M_n/2M_n$ was too small. The problem was only that ambient coefficientwise reduction does not see the full quotient.

There is also a useful blockwise way to view the same quotient. Since

$$R_n \cong \bigoplus_{r=0}^{k-1} X^r O_8 \qquad k = \frac{n}{4}$$

reducing modulo 2 gives

$$\overline{R}_n \cong \bigoplus_{r=0}^{k-1} X^r (O_8/2O_8)$$

Using $M_n = P_n R_n^2$ and reducing in P_n coordinates, we get

$$M_n/2M_n \cong \bigoplus_{r=0}^{k-1} X^r (M/2M)$$

So HAWK's $2n$ parity bits can be interpreted block by block. There are $k = n/4$ independent E_8 block labels. Each label lies in $M/2M \cong (O_8/2O_8)^2$ so each $2E_8$ block receives one 8-bit coset label.

For example: HAWK-512: $k = 128$ so there are 128 independent E_8 block labels, and HAWK-1024: $k = 256$ so there are 256 independent E_8 block labels.

We now have the correct mod-2 quotient for the embedded E_8 block module. The next step is to compare it with ambient coefficientwise reduction and show exactly why that second reduction map loses most of the parity information.

4.2.2 Why Coefficientwise Reduction Cannot Work

We have now shown that the correct mod-2 quotient on M_n is obtained by reducing the P_n coordinates: $\pi_M(P_n z) = z \bmod 2$. The obstruction comes from the other mod-2 map, namely ambient coefficientwise reduction

$$\rho_{\text{amb}} : R_n^2 \longrightarrow \overline{R}_n^2, \qquad x \mapsto x \bmod 2$$

Restrict this map to $M_n = P_n R_n^2$. Writing $k = n/4$ and $Y = X^k$, then

$$P_n = \begin{pmatrix} 2(1+Y) & 1 - Y + Y^2 + Y^3 \\ 0 & 1 + Y + Y^2 + Y^3 \end{pmatrix}$$

Reducing modulo 2 gives

4

$$\overline{P}_n = \begin{pmatrix} 0 & q \\ 0 & q \end{pmatrix}, \qquad q := 1 + Y + Y^2 + Y^3$$

Therefore, for $x = P_n(a, b) \in M_n$,

$$\rho_{\text{amb}}(x) = \overline{P}_n(\bar{a}, \bar{b}) = (q\bar{b},\ q\bar{b})$$

Equivalently,

$$\rho_{\text{amb}}\,|_{M_n} = \overline{P}_n \circ \pi_M$$

This single equation is the whole issue in a compressed form. The correct parity map on M_n is π_M, but the ambient coefficient reduction seen by HAWK is obtained only after composing with the singular map $\overline{P}_n$. $\overline{P}_n$ forgets the first internal coordinate completely and only keeps the second coordinate after multiplication by q.

To compute the dimension of this image, consider the structure of HAWK's ring modulo 2. Since the characteristic is 2, we have $1 = -1$, and hence $X^n + 1 = X^n - 1$ over $\mathbb{F}_2$. Also, n is a power of two, say $n = 2^m$. In characteristic 2, the Frobenius identity gives

$$(X + 1)^{2^m} = X^{2^m} + 1$$

Therefore $X^n + 1 = (X + 1)^n$ in $\mathbb{F}_2[X]$, and so

$$\overline{R}_n \cong \mathbb{F}_2[X]/(X^n + 1) = \mathbb{F}_2[X]/(X + 1)^n$$

Now compute q. Since signs disappear modulo 2, $q = 1 + Y + Y^2 + Y^3$. In characteristic 2,

$$(1 + Y)^3 = 1 + 3Y + 3Y^2 + Y^3 = 1 + Y + Y^2 + Y^3$$

because $3 \equiv 1 \pmod 2$. Hence $q = (1 + Y)^3$. Since $Y = X^k$, this gives $q = (1 + X^k)^3$. Finally,

$k = n/4$ is also a power of two, so the same Frobenius identity gives $1 + X^k = (1 + X)^k$. Therefore

$$q = (1 + X^k)^3 = (1 + X)^{3k}$$

In terms of $\varepsilon = X + 1$, this is simply $q = \varepsilon^{3k}$.

Multiplication by q therefore has image $\varepsilon^{3k}\overline{R}_n$, an $\mathbb{F}_2$-space of dimension

$$n - 3k = n - \frac{3n}{4} = \frac{n}{4}$$

Hence

$$\dim_{\mathbb{F}_2} \rho_{\text{amb}}(M_n) = \frac{n}{4}$$

whereas

$$\dim_{\mathbb{F}_2}(M_n/2M_n) = 2n$$

So ambient coefficientwise reduction collapses $2n$ abstract parity bits down to only $n/4$ visible bits. This is a large scale version of the single block observation that one $2E_8$ block has 2^8 abstract mod-2 classes but only two coefficientwise parity patterns.

This then gives us a no-go statement for the drop in replacement. There is no map assigning to every HAWK residue $t \in \overline{R}_n^2$ an element $x_t \in M_n$ such that $x_t \equiv t \pmod{2R_n^2}$, because that would force $\rho_{\text{amb}}(M_n) = \overline{R}_n^2$, contradicting the dimension count above. So there is a mod-2 mismatch, and it cannot be repaired by merely choosing a different $\mathbb{Z}$-basis of the same embedded module.

4.2.3 The Internal Coset-Matching Map

We have shown that ambient coefficientwise reduction cannot give the right matching map. The correct quotient is instead the internal map

$$\pi_M : M_n \longrightarrow \overline{R}_n^2, \qquad \pi_M(P_n z) = z \bmod 2$$

Once this quotient map is used, the coset-matching problem becomes straightforward.

HAWK already fixes a binary section $s : \overline{R}_n \to R_n$ that chooses the representative with coefficients in $\{0, 1\}$, and the same section is applied coefficientwise to $\overline{R}_n^2$. For a target class

$t = Bh \bmod 2 \in \overline{R}_n^2$ define the E_8 block signing coset by

$$\boxed{\mathcal{C}_t := \pi_M^{-1}(t) = P_n\big(2R_n^2 + s(t)\big) = 2M_n + P_n s(t)}$$

This definition is independent of the chosen lift. Indeed, replacing $s(t)$ by $s(t) + 2u$ changes the right hand side by $2P_n u \in 2M_n$. This gives the desired translation $2R_n^2 + t \longmapsto \mathcal{C}_t \subset M_n$ and it is a bijection on cosets. More precisely, $P_n : \ 2R_n^2 + s(t) \xrightarrow{\sim} \mathcal{C}_t$ is a one-to-one correspondence, because P_n is an R_n-module isomorphism from R_n^2 onto M_n. Thus the correct coset of $M_n/2M_n$ is obtained by asking that the P_n coordinates have the prescribed residue t.

4

The block form is especially clean. If

$$t = \sum_{r=0}^{k-1} X^r \tau_r, \qquad \tau_r \in (O_8/2O_8)^2$$

then

$$\mathcal{C}_t = \bigoplus_{r=0}^{k-1} X^r \, \mathcal{C}_{\tau_r}^{(8)}, \qquad \mathcal{C}_{\tau_r}^{(8)} := 2M + Ps(\tau_r)$$

So an E_8 based sampler should consume HAWK's parity data blockwise. Not one centre bit per coefficient, but one 8-bit coset label per $2E_8$ block. This is the mathematical interface that the current HAWK sampler is missing.

4.2.4 What this Means for the Sampler and the Public Norm

At this point the coset-matching problem, as a module theoretic problem, is solved. What remains is a design choice we need to make about the norm to be sampled from. Let

$$x_E \in \mathcal{C}_t, \qquad z := P_n^{-1} x_E \in 2R_n^2 + s(t)$$

If one wants to keep HAWK's existing public form $Q = B^\star B$, then the relevant weight is still the HAWK weight on z, namely

$$\exp\Big(-\frac{\|z\|^2}{2\sigma^2}\Big) = \exp\Big(-\frac{\|P_n^{-1}x_E\|^2}{2\sigma^2}\Big)$$

So on the E_8 side the correct distribution is not the ordinary Euclidean discrete Gaussian on the coset $\mathcal{C}_t$, it is the pulled-back, anisotropic Gaussian induced by P_n^{-1}. In other words, if the public key remains the HAWK key Q, then the sampler is no longer a plain Euclidean E_8 block sampler.

If, instead, we insist on sampling with the ordinary Euclidean norm on the E_8 block side, then the public quadratic form must change. Writing $S_n := P_n^\star P_n$ the corresponding public form becomes

$$\boxed{Q_{E_8} := B^\star S_n B = B^\star P_n^\star P_n B}$$

Indeed, for $x_E = P_n B w$, $\|x_E\|^2 = \langle w,\, B^\star P_n^\star P_n B\, w\rangle = \langle w, Q_{E_8} w\rangle$. This produces a legitimate E_8 preconditioned variant, but it is no longer HAWK as specified, because the HAWK specification assumes a secret unimodular basis $B \in GL_2(R_n)$, whereas our P_n generates a proper submodule $M_n \subset R_n^2$ rather than an ambient unimodular basis. Any such variant would therefore require a separate correctness and security analysis.

This also clarifies the sampler architecture. HAWK's current sampler uses two one-dimensional tables T_0, T_1 and chooses between them using a single bit $t[r]$ for each coordinate. Under the E_8 interface, the natural centre data are no longer scalar bits but 8-bit labels in $M/2M$ for each block, so the replacement sampler must be blockwise. The correct mathematical object to tabulate is therefore a family of distributions on the 256 cosets of $2M$ inside $M \cong 2E_8$.

4.2.5 Outcome of the Coset Interface

This section solves the coset matching problem at the module level. The correct label for an E_8 block coset is obtained by reducing the P_n coordinates modulo 2, not by reducing the embedded vector coefficientwise in R_n^2. Ambient coefficientwise reduction loses most of the internal quotient $M_n/2M_n$, so a sampler replacement inside HAWK's original coefficientwise interface is not possible.

We have also fixed the geometric choice we will use. If HAWK's original public form $Q = B^\star B$ were kept, the sampler would be measured by the preimage norm $\|P_n^{-1} x_E\|^2$, rather than by the Euclidean E_8 block norm. Since the aim is to use the E_8 block geometry itself, the construction instead uses the internal coset $\mathcal{C}_t = 2M_n + P_n s(t)$ and changes the public form to $Q_{E_8} = B^\star P_n^\star P_n B$. Thus HAWK-$E_8$-CM is a HAWK like variant with an internal P_n coordinate coset interface and a modified public quadratic form. The next section explains how the rest of the signing and verification equations change under this choice.

4.3 HAWK-E_8-CM

We now state the resulting HAWK-E_8-CM construction.

4.3.1 The Uncompressed Scheme

The cleanest way to state our new construction is first to ignore signature compression. So for this version the signature contains the full vector

$$s = (s_0, s_1) \in R^2$$

This is not a perfect final practical format, but is nice for proving that the mathematics of the new construction is consistent.

The resulting scheme which we call HAWK-E_8-CM is as follows.

Key generation

1. Generate a HAWK-style unimodular secret basis

$$B = \begin{pmatrix} f & F \\ g & G \end{pmatrix} \in GL_2(R), \qquad fG - gF = 1$$

2. Compute the fixed preconditioning form $S_n = P_n^\star P_n$.
3. Compute the modified public form $\widetilde{Q} = B^\star S_n B$.
4. Store the secret key as the HAWK-style data needed to recover or use B
5. Store the public key as an encoding of the independent public entries

$$\tilde{q}_{00}, \qquad \tilde{q}_{01}$$

(current prototype stores the expanded form for testing.)

Signing

Given a message m, signing proceeds in the same outer shape as HAWK, but with the E_8 sampler in the middle.

1. Hash the message and salt as in HAWK to obtain $h \in (R/2R)^2$.
2. Compute the target residue $t = Bh \bmod 2$.

3. Use the internal coset-matching map from the previous section and sample

$$x_E \in \mathcal{C}_t = 2M_n + P_n s(t)$$

This sample is taken with respect to the ordinary Euclidean norm on the embedded E_8 block lattice.

4. Convert back to R^2 coordinates: $z = P_n^{-1} x_E$. Then $z \in 2R^2 + s(t)$.

5. Set $w = B^{-1} z$.

6. Accept the sample only if $\|x_E\|^2 \leq 8n(\sigma_{\text{verify}}^{E_8})^2$, otherwise restart.

7. Apply the HAWK symmetry-breaking rule to w_1. If necessary, replace w by $-w$.

8. Output

$$s = \frac{h - w}{2}$$

Importantly $s \in R^2$. This follows because the sampled coset ensures $z \equiv Bh \pmod 2$ and hence, since B^{-1} has entries in R, $w = B^{-1} z \equiv h \pmod 2$. Therefore $h - w$ is divisible by 2.

Verification

Given a public key encoding $\widetilde{Q}$, a message m, and a signature (salt, s), the verifier does the following.

1. Recompute $h \in (R/2R)^2$ from the message and salt.

2. Reconstruct $w = h - 2s$.

3. Check the symmetry-breaking condition on w_1.

4. Check the modified norm bound $\|w\|_{\widetilde{Q}}^2 \leq 8n(\sigma_{\text{verify}}^{E_8})^2$.

This is exactly the same style of verification as HAWK, but with the public form $\widetilde{Q} = B^\star S_n B$ replacing $Q = B^\star B$.

The bound used above is written in the same radius convention as HAWK where the verification radius is expressed as $L = 2\sqrt{2n}\,\sigma$, so the squared norm bound is $L^2 = 8n\sigma^2$. We keep the same convention here, but apply it to the new E_8 side norm.

We now provide the basic completeness statement for the uncompressed version.

Claim. Assume key generation outputs a secret basis B and public form $\widetilde{Q} = B^\star S_n B$. Assume also that signing outputs without restarting, and that the verifier recomputes the same hash value h. Then the uncompressed verifier accepts.

4

Proof. The signer samples

$$x_E \in \mathcal{C}_t = P_n(2R^2 + s(t)), \qquad t = Bh \bmod 2$$

Therefore there exists some $z \in 2R^2 + s(t)$ such that $x_E = P_n z$. Since $s(t) \equiv t \pmod 2$, and $t = Bh \bmod 2$, we have $z \equiv Bh \pmod 2$. Now define $w = B^{-1}z$. Since $B \in GL_2(R)$, we also have $B^{-1} \in M_2(R)$. Hence $w \equiv h \pmod 2$. If the symmetry-breaking rule replaces w by $-w$, then this congruence is unchanged modulo 2. Thus in either case

$$s = \frac{h - w}{2}$$

is an element of R^2, and the signature is well formed

When the verifier receives s, it reconstructs $w' = h - 2s = h - 2\left(\frac{h-w}{2}\right) = w$. Thus the verifier recovers exactly the same representative w that the signer encoded in the signature. The symmetry-breaking check passes because the signer enforces this canonical choice before output.

It remains to compare the norm checked by the verifier with the norm used by the signer. Since $z = Bw$ we have $x_E = P_n z = P_n B w$. Therefore

$$\|x_E\|^2 = \langle P_n Bw, P_n Bw\rangle = \langle w, B^\star P_n^\star P_n B\, w\rangle = \langle w, \widetilde{Q}w\rangle = \|w\|_{\widetilde{Q}}^2$$

So the signer's condition

$$\|x_E\|^2 \le 8n(\sigma_{\text{verify}}^{E_8})^2$$

is exactly the verifier's condition

$$\|w\|_{\widetilde{Q}}^2 \le 8n(\sigma_{\text{verify}}^{E_8})^2$$

Thus the verifier accepts.

4.3.2 The Fixed Preconditioning Form and Public Key Derivation

We begin with expanding S_n:

$$a = 2(1+Y), \qquad b = 1 - Y + Y^2 + Y^3, \qquad c = 1 + Y + Y^2 + Y^3$$

Then

$$P_n = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$$

and therefore

$$S_n = P_n^\star P_n = \begin{pmatrix} a^\star a & a^\star b \\ b^\star a & b^\star b + c^\star c \end{pmatrix}$$

Using $Y^4 = -1$, and $Y^\star = Y^{-1} = -Y^3$, this gives

$$S_n = \begin{pmatrix} 8 + 4(Y - Y^3) & 4Y^2 \\ -4Y^2 & 8 \end{pmatrix}$$

The off-diagonal entries are conjugates of each other, since

$$(4Y^2)^\star = 4Y^{-2} = 4(-Y^2) = -4Y^2$$

Thus S_n is Hermitian, as required for a public quadratic form.

The determinant of S_n is also fixed: $\Delta_n := \det(S_n)$. From the expression above,

$$\Delta_n = \big(8 + 4(Y - Y^3)\big)8 - (4Y^2)(-4Y^2)$$

Since $Y^4 = -1$, this simplifies to

$$\boxed{\Delta_n = 48 + 32(Y - Y^3)}$$

This determinant replaces the determinant-one relation that appears in HAWK. This is a small looking change, but it affects public key derivation, verification, and the compressed reconstruction formula.

Now let us move on to the public key derivation. The secret basis generation can remain like

in HAWK. We may still generate a unimodular matrix

$$B = \begin{pmatrix} f & F \\ g & G \end{pmatrix} \in GL_2(R), \qquad fG - gF = 1$$

This is useful because it means the NTRU-style trapdoor equation itself does not have to be changed. The signer still needs B in order to compute $t = Bh \bmod 2$, and later to map the sampled vector back to the w-side.

4

The change is in the public form. Instead of publishing data derived from $Q = B^\star B$. We now derive the public key from $\widetilde{Q} = Q_{E_8} = B^\star S_n B$. Writing

$$\widetilde{Q} = \begin{pmatrix} \tilde{q}_{00} & \tilde{q}_{01} \\ \tilde{q}_{10} & \tilde{q}_{11} \end{pmatrix}, \qquad \tilde{q}_{10} = \tilde{q}_{01}^\star$$

Because B is unimodular, the determinant of $\widetilde{Q}$ is just the determinant of S_n:

$$\det(\widetilde{Q}) = \det(B^\star)\det(S_n)\det(B) = \Delta_n$$

So the public key can still be thought of as two independent polynomials, $\tilde{q}_{00}$ and $\tilde{q}_{01}$, but the reconstruction relation is no longer determinant-one. Instead it is $\tilde{q}_{00}\tilde{q}_{11} - \tilde{q}_{01}\tilde{q}_{10} = \Delta_n$. Since $\tilde{q}_{10} = \tilde{q}_{01}^\star$, this can also be written as $\tilde{q}_{00}\tilde{q}_{11} - \tilde{q}_{01}\tilde{q}_{01}^\star = \Delta_n$

The verifier needs the full form $\widetilde{Q}$ to check the signature norm, but a HAWK-style compressed public key should not have to store every entry independently. Since the determinant is fixed, $\tilde{q}_{11}$ is determined by $\tilde{q}_{00}$ and $\tilde{q}_{01}$:

$$\tilde{q}_{11} = \frac{\Delta_n + \tilde{q}_{01}\tilde{q}_{10}}{\tilde{q}_{00}}$$

whenever $\tilde{q}_{00}$ is invertible in the relevant ring used for verification.

This is one of the places where our new scheme would need new parameter work. HAWK's current key generation checks and public key encoding bounds are designed for the entries of $B^\star B$. Here the verifier would instead see entries of $B^\star S_n B$. Since S_n is not the identity matrix, the sizes and distributions of the public coefficients are changed. So the same shape of public key may still be possible, but the actual coefficient bounds need to be re-derived [8].

4.3.3 The Verification Norm Formula

For implementation, the verifier does not want to multiply by the full 2×2 matrix in the most naive way. HAWK uses a completion of squares formula based on the two public entries q_{00}, q_{01}. We use the same idea here, but the determinant term changes from 1 to Δ_n.

Write

$$\widetilde{Q} = \begin{pmatrix} \tilde{q}_{00} & \tilde{q}_{01} \\ \tilde{q}_{10} & \tilde{q}_{11} \end{pmatrix}, \qquad \tilde{q}_{10} = \tilde{q}_{01}^{\star}, \qquad \det(\widetilde{Q}) = \Delta_n$$

For

$$w = (w_0, w_1) \in R_n^2$$

the verification norm is defined by the Hermitian form $\|w\|_{\widetilde{Q}}^2 = \langle w, \widetilde{Q}w \rangle$. Using the normalized trace convention, this means

$$n\|w\|_{\widetilde{Q}}^2 = \mathrm{Tr}(w^{\star}\widetilde{Q}w)$$

Expanding the matrix product gives $w^{\star}\widetilde{Q}w = \tilde{q}_{00}w_0w_0^{\star} + \tilde{q}_{01}w_1w_0^{\star} + \tilde{q}_{01}^{\star}w_0w_1^{\star} + \tilde{q}_{11}w_1w_1^{\star}$.
We want to compute this same expression using $\tilde{q}_{00}$, $\tilde{q}_{01}$, and the fixed determinant Δ_n, rather than using $\tilde{q}_{11}$.

Define

$$d = \frac{w_1}{\tilde{q}_{00}}, \qquad e = w_0 + \tilde{q}_{01}d$$

Then $w_1 = \tilde{q}_{00}d, \quad w_0 = e - \tilde{q}_{01}d$. Since $\widetilde{Q}$ is Hermitian, $\tilde{q}_{00}^{\star} = \tilde{q}_{00}$, so $w_1^{\star} = (\tilde{q}_{00}d)^{\star} = d^{\star}\tilde{q}_{00}$. Also, the determinant relation gives $\Delta_n = \tilde{q}_{00}\tilde{q}_{11} - \tilde{q}_{01}\tilde{q}_{01}^{\star}$.

We introduce e as it completes the square in the w_0 coordinate. This is the same idea as the scalar identity

$$ax^2 + 2bxy + cy^2 = a\left(x + \frac{b}{a}y\right)^2 + \left(c - \frac{b^2}{a}\right)y^2$$

Here the role of x is played by w_0, the role of y is played by w_1, and the determinant term replaces the final correction term.

Now expand the completed-square expression:

$$\tilde{q}_{00}ee^{\star} = \tilde{q}_{00}(w_0 + \tilde{q}_{01}d)(w_0^{\star} + d^{\star}\tilde{q}_{01}^{\star}) \; = \tilde{q}_{00}w_0w_0^{\star} + \tilde{q}_{00}\tilde{q}_{01}d\,w_0^{\star} + \tilde{q}_{00}w_0d^{\star}\tilde{q}_{01}^{\star} + \tilde{q}_{00}\tilde{q}_{01}\tilde{q}_{01}^{\star}dd^{\star}$$

Using $w_1 = \tilde{q}_{00} d$ and $w_1^\star = d^\star \tilde{q}_{00}$, the two middle terms become

$$\tilde{q}_{00}\tilde{q}_{01} d\, w_0^\star = \tilde{q}_{01} w_1 w_0^\star \quad \text{and} \quad \tilde{q}_{00} w_0 d^\star \tilde{q}_{01}^\star = \tilde{q}_{01}^\star w_0 w_1^\star$$

Therefore

$$\tilde{q}_{00} e e^\star = \tilde{q}_{00} w_0 w_0^\star + \tilde{q}_{01} w_1 w_0^\star + \tilde{q}_{01}^\star w_0 w_1^\star + \tilde{q}_{00}\tilde{q}_{01}\tilde{q}_{01}^\star d d^\star$$

4

It remains to recover the final $\tilde{q}_{11} w_1 w_1^\star$ term. The completed square has already produced the term $\tilde{q}_{00}\tilde{q}_{01}\tilde{q}_{01}^\star d d^\star$ but the Hermitian form requires $\tilde{q}_{11} w_1 w_1^\star$. Since $w_1 = \tilde{q}_{00} d$ and $w_1^\star = d^\star \tilde{q}_{00}$, this required term is

$$\tilde{q}_{11} w_1 w_1^\star = \tilde{q}_{00}^2 \tilde{q}_{11} d d^\star$$

The difference between the required coefficient and the one produced by the completed square is therefore

$$\tilde{q}_{00}^2 \tilde{q}_{11} - \tilde{q}_{00}\tilde{q}_{01}\tilde{q}_{01}^\star = \tilde{q}_{00}\left(\tilde{q}_{00}\tilde{q}_{11} - \tilde{q}_{01}\tilde{q}_{01}^\star\right)$$

Using the determinant relation $\Delta_n = \tilde{q}_{00}\tilde{q}_{11} - \tilde{q}_{01}\tilde{q}_{01}^\star$ this difference is $\tilde{q}_{00}\Delta_n d d^\star$. But $d w_1^\star = d d^\star \tilde{q}_{00}$, so this correction can be written as $\Delta_n d w_1^\star$.

Adding this determinant correction gives

$$\tilde{q}_{00} e e^\star + \Delta_n d w_1^\star = \tilde{q}_{00} w_0 w_0^\star + \tilde{q}_{01} w_1 w_0^\star + \tilde{q}_{01}^\star w_0 w_1^\star \; + \tilde{q}_{00}\tilde{q}_{01}\tilde{q}_{01}^\star d d^\star + \Delta_n d d^\star \tilde{q}_{00}$$

The last two terms combine as $\tilde{q}_{00}(\tilde{q}_{01}\tilde{q}_{01}^\star + \Delta_n) d d^\star$ Since $\tilde{q}_{01}\tilde{q}_{01}^\star + \Delta_n = \tilde{q}_{00}\tilde{q}_{11}$ this becomes $\tilde{q}_{00}^2 \tilde{q}_{11} d d^\star$. But $w_1 w_1^\star = (\tilde{q}_{00} d)(d^\star \tilde{q}_{00}) = \tilde{q}_{00}^2 d d^\star$ so the final term is $\tilde{q}_{11} w_1 w_1^\star$.

Hence

$$\tilde{q}_{00} e e^\star + \Delta_n d w_1^\star = \tilde{q}_{00} w_0 w_0^\star + \tilde{q}_{01} w_1 w_0^\star + \tilde{q}_{01}^\star w_0 w_1^\star + \tilde{q}_{11} w_1 w_1^\star$$

which is the quadratic expression defined by $\widetilde{Q}$. Taking the trace therefore gives

$$\boxed{n\|w\|_{\widetilde{Q}}^2 = \mathrm{Tr}\left(\tilde{q}_{00} e e^\star + \Delta_n\, d\, w_1^\star\right)}$$

This is analogue of the HAWK verification formula, except that the final term is now multiplied by Δ_n. In HAWK the determinant is 1, so this factor is invisible [8].

4.3.4 Compressed Reconstruction and the Signature Format

When HAWK compresses signatures only s_1 is stored, so verification fixes $w_1 = h_1 - 2s_1$ and must reconstruct the missing component s_0, equivalently $w_0 = h_0 - 2s_0$. The reconstruction rule is therefore obtained by choosing the w_0 that minimises the public norm for this fixed w_1, and then rounding the corresponding value of s_0.

Therefore the compressed reconstruction step can already be settled at the purely mathematical level.

$$\widetilde{Q} = \begin{pmatrix} \tilde{q}_{00} & \tilde{q}_{01} \\ \tilde{q}_{01}^{\star} & \tilde{q}_{11} \end{pmatrix}, \qquad \det(\widetilde{Q}) = \Delta_n, \qquad w = (w_0, w_1) \in R^2$$

Using the completed square calculation from the previous subsection, the $\widetilde{Q}$-norm can be written, for fixed w_1, as

$$\|w\|_{\widetilde{Q}}^2 = \left\langle w_0 + \frac{\tilde{q}_{01}}{\tilde{q}_{00}} w_1,\, \tilde{q}_{00}\left(w_0 + \frac{\tilde{q}_{01}}{\tilde{q}_{00}} w_1\right)\right\rangle + \left\langle w_1,\, \frac{\Delta_n}{\tilde{q}_{00}}\, w_1 \right\rangle$$

The second term depends only on w_1, so it has no effect on where the minimum occurs as w_0 varies. Therefore the w_0-dependent part is minimised when the completed square vanishes:

$$w_0 + \frac{\tilde{q}_{01}}{\tilde{q}_{00}} w_1 = 0$$

Hence the unique real minimiser in the w_0 direction is

$$w_0^{\min} = -\frac{\tilde{q}_{01}}{\tilde{q}_{00}}\, w_1$$

So if we keep the compressed signature format similar to HAWK then the mathematically correct reconstruction rule is unchanged in form and becomes

$$\boxed{\hat{s}_0 = \left\lfloor \frac{h_0}{2} + \frac{\tilde{q}_{01}}{\tilde{q}_{00}} \left(\frac{h_1}{2} - s_1 \right) \right\rceil}$$

with coefficientwise rounding. The determinant term Δ_n changes the value of the norm, but it does not change the location of the minimiser. This is the analogue of HAWK's reconstruction formula, with q_{00}, q_{01} replaced by $\tilde{q}_{00}, \tilde{q}_{01}$. [8]

What does not carry over automatically is the negligible failure analysis. In HAWK, compressed verification is supported by a key generation check on $(1/q_{00})[0]$ against the parameter

β_0, together with public key and signature coefficient bounds chosen so that the fixed point rebuilding procedure returns the mathematically correct s_0 with negligible probability of error. The current HAWK specification [8] reports reconstruction failure estimates below 2^{-105} and 2^{-315} for HAWK-512 and HAWK-1024, respectively. For HAWK-E_8-CM, the same style of proof will require new bounds for $\tilde{q}_{00}^{-1}$, new coefficient ranges for $\tilde{q}_{00}, \tilde{q}_{01}, \tilde{q}_{11}$, and a new fixed point error budget for the implementation. Therefore for simplicity we only implement the uncompressed scheme, while compression is an optimisation for future development.

This follows the same separation that already appears in HAWK itself. Where the uncompressed signature is the clean cryptographic object, and compression is an implementation oriented encoding layer on top of it. In the present variant, all algebraic correctness statements can already be made at the uncompressed level, so there is no reason to hold the scheme definition hostage to a compressed reconstruction proof.

4.3.5 Public Form Scaling

From

$$\widetilde{Q} = B^\star S_n B, \qquad S_n = \begin{pmatrix} 8 + 4(Y - Y^3) & 4Y^2 \\ -4Y^2 & 8 \end{pmatrix}, \qquad Y = X^{n/4}$$

one gets

$$\tilde{q}_{00} = 8(ff^\star + gg^\star) + 4(Y - Y^3)ff^\star + 4Y^2(f^\star g - fg^\star),$$

$$\tilde{q}_{01} = 8(Ff^\star + Gg^\star) + 4(Y - Y^3)Ff^\star + 4Y^2(Gf^\star - Fg^\star),$$

$$\tilde{q}_{11} = 8(FF^\star + GG^\star) + 4(Y - Y^3)FF^\star + 4Y^2(F^\star G - FG^\star)$$

Using the HAWK coefficient bounds $|f_i|, |g_i| \leq \eta, \quad |F_i|, |G_i| \leq 127$ [8] this yields the coefficient bounds

$$\boxed{|\tilde{q}_{00}[i]| \leq 32n\eta^2, \qquad |\tilde{q}_{01}[i]| \leq 32n\eta \cdot 127, \qquad |\tilde{q}_{11}[i]| \leq 32n \cdot 127^2}$$

For $n \leq 1024$ and $\eta \leq 8$, these worst-case bounds stay below half of the 31-bit prime

$$p_1 = 2147473409$$

used by HAWK for NTT arithmetic, and hence also below half of the companion prime

$$p_2 = 2147389441$$

So the same NTT moduli remain sufficient for HAWK-E_8-CM, no new modular arithmetic architecture is forced by the public form change.

The fixed preconditioning form $S_n = P_n^\star P_n$ is not negligible though. To see its scale, we evaluate S_n under the four complex embeddings of the relation $Y^4 = -1$. These embeddings send

$$Y \longmapsto y \in \{\zeta_8, \zeta_8^3, \zeta_8^5, \zeta_8^7\}$$

For such a value of y, the embedded 2×2 Hermitian block is

$$S(y) = \begin{pmatrix} 8 + 4(y - y^3) & 4y^2 \\ -4y^2 & 8 \end{pmatrix}$$

Keeping in mind $\zeta_8 = e^{\pi i/4}$, $y - y^3 = \pm\sqrt{2}$ and $|4y^2| = 4$. Therefore the eigenvalues of this Hermitian block are

$$\lambda_\pm = \frac{(8 + 4r) + 8}{2} \pm \frac{1}{2}\sqrt{(4r)^2 + 4 \cdot 4^2}, \qquad r = y - y^3 \in \{\sqrt{2}, -\sqrt{2}\}$$

Equivalently, $\lambda_\pm = 8 + 2r \pm 2\sqrt{6}$. For $r = \sqrt{2}$ and $r = -\sqrt{2}$, this gives

$$0.272593\ldots, \quad 5.929448\ldots, \quad 10.070552\ldots, \quad 15.727407\ldots$$

Since $S_n = P_n^\star P_n$, these are squared singular values of P_n, so P_n has singular values between approximately

$$\sqrt{0.272593} \approx 0.5221 \qquad \text{and} \qquad \sqrt{15.727407} \approx 3.9658$$

The eigenvalues give a useful scale estimate for the effect of S_n. For each embedded 2×2 block, they imply the operator bounds

$$0.2725\ldots \, \|v\|^2 \le \langle v, S_n v\rangle \le 15.7274\ldots \, \|v\|^2$$

The average eigenvalue is about 8, so if the secret basis coefficients behave roughly isotropically, one expects the typical size of the public form $B^\star S_n B$ to be around 8 times the size of the corresponding HAWK form $B^\star B$. And the largest eigenvalue is just below 16, so the worst-case scale estimate is about 16.

Multiplication of coefficient sizes by 8 corresponds to $\log_2(8) = 3$ extra bits, while a conservative

16 corresponds to $\log_2(16) = 4$ extra bits. Therefore public key coefficient budgets should be roughly +3 or +4 bits above HAWK for a compressed version.

HAWK's compression analysis depends on the constant coefficient of q_{00}^{-1}. In HAWK-E_8-CM the analogous quantity is the constant coefficient $(\tilde{q}_{00}^{-1})[0]$. If $\tilde{q}_{00} \approx 8q_{00}$, then the inverse scales in the opposite direction

$$\tilde{q}_{00}^{-1} \approx \frac{1}{8} q_{00}^{-1}$$

So for future compression work β_0 should initially be scaled down by about a factor of 8. This is not a final compression proof since the actual bounds must come from the distribution of $\tilde{q}_{00}^{-1}$, the fixed point reconstruction error, and measured key generation histograms.

A useful side effect however ends up appearing on the security side. HAWK spec [8] notes that its published omSVP reduction becomes vacuous for the deployed parameter sets because the public basis vector e_1 is already short enough. In HAWK-E_8-CM, the same vector is measured using the modified public form $Q_{E_8} = B^\star S_n B$. Since e_1 selects the top left entry of this public form, its squared norm is the normalized trace of $\tilde{q}_{00}$, equivalently the constant coefficient

$$\|e_1\|_{Q_{E_8}}^2 = \tilde{q}_{00}[0]$$

For the unchanged HAWK secret distribution, the leading part of $\tilde{q}_{00}$ is scaled by the average size of S_n. Since the average eigenvalue of S_n is 8, while the cross-correlation terms in $\tilde{q}_{00}[0]$ should average out heuristically, one expects $\tilde{q}_{00}[0] \approx 8q_{00}[0]$. Using the HAWK estimate $q_{00}[0] \approx 2n\sigma_{\text{krsec}}^2$ this gives

$$\tilde{q}_{00}[0] \approx 16n\sigma_{\text{krsec}}^2$$

Thus the public vector e_1 falls under the verification threshold $8n(\sigma_{\text{verify}}^{E_8})^2$ only if $16n\sigma_{\text{krsec}}^2 \lesssim 8n(\sigma_{\text{verify}}^{E_8})^2$ or equivalently when

$$\sigma_{\text{verify}}^{E_8} \gtrsim \sqrt{2}\,\sigma_{\text{krsec}}$$

which is a much stricter condition than the one that makes the HAWK reduction vacuous. This is only a heuristic scaling argument, not a proof, as it depends on the expected scaling of $\tilde{q}_{00}[0]$.

4.3.6 Security Formulation and Proof

At the scheme definition level, the direct key recovery problem can be stated directly. In HAWK [1], the public key lives in the equivalence class $[I_2(K)]$, and direct key recovery is rank 2 search

module LIP in that class. In HAWK-E_8-CM, the public key lives instead in the fixed class

$$[S_n], \qquad S_n = P_n^\star P_n$$

and direct key recovery is the corresponding worst-case problem

$$\text{wc-smLIP}_{K,2}^{S_n}: \quad \text{given } \widetilde{Q} \in [S_n], \text{ find } U \in GL_2(R) \text{ such that } \widetilde{Q} = U^\star S_n U$$

For actual keys, $U = B$, so the public key is an instance generated from this fixed class. HAWK's own specification [8] already frames direct recovery as smLIP within a fixed equivalence class, and the only change we make here is the replacement of the identity form by S_n.

This distinction must be made explicit because recent work on distinguish LIP and on HAWK's lattice invariants shows that one cannot safely argue security by pretending that an auxiliary lattice with visibly different invariants is hidden among identity-form instances. At the most basic level, decisional LIP instances are only meaningful between lattices whose efficiently computable invariants agree, since otherwise invariants such as the determinant already give a trivial distinguisher [2]. Recent search to distinguish results for direct sums further reinforce that the auxiliary lattice used in the problem formulation must be treated as part of the fixed class, rather than as an interchangeable representative of the identity form class [3].

Separately, recent HAWK work studies both genus and special genus invariants within the HAWK setting. Genus alone gives an enormous number of isometry classes for the HAWK lattice, while special genus distinguishability is polynomial time for HAWK-like Hermitian lattices but does not yet appear to lead to a practical decision-LIP advantage by itself [20], [21]. Therefore the right way to view HAWK-E_8-CM is as a HAWK-like signature scheme whose public keys lie in the fixed class $[S_n]$.

The signing side hardness statement must likewise be rewritten at the class $[S_n]$ level. A valid forged signature allows the verifier to reconstruct $w = h - 2s$. Therefore the forged vector satisfies

$$w \equiv h \pmod 2, \qquad \text{or equivalently} \qquad w \in h + 2R^2$$

as well as the required symmetry condition and the verification norm bound

$$\|w\|_{Q_{E_8}}^2 = \langle w, B^\star S_n B\, w \rangle \le 8n(\sigma_{\text{verify}}^{E_8})^2$$

Equivalently, if $x_E = P_n Bw$ then $Bw \equiv Bh = t \pmod 2$, so $x_E \in 2M_n + P_n s(t)$. Thus a valid forgery can also be viewed as producing a short vector in the correct internally matched E_8 block coset.

So the relevant one-more problem is no longer HAWK's original omSVP problem in the identity class. We have to re state it for the preconditioned public form class $[S_n]$. In other words, direct key recovery is described by search module LIP in the class $[S_n]$, while signature forgery should be viewed as a one-more short-vector problem in the same preconditioned class. This does not prove security by itself, but it states the problem class for HAWK-E_8-CM.

What remains delicate is the set of "trivial" isometries that must be quotiented out in that one-more game. HAWK's current proof uses a publicly computable group generated by multiplication by powers of X and a symplectic automorphism, and recent automorphism work shows that missing such publicly computable transformations can quadratically speed up key recovery and effectively halve security bits. In HAWK-E_8-CM, multiplication by powers of X certainly remains public because $X^\star X = 1$ and S_n commutes with scalar multiplication by X. By contrast, HAWK's determinant-one symplectic formula does not carry over exactly, since S_n is not the identity form and $\det(S_n) = \Delta_n \neq 1$. This does not mean that the symplectic symmetry disappears. Rather, it means that any surviving anti linear public isometry must include the determinant contribution coming from the embedding matrix P_n. Therefore there is a proof obligation we must now undertake. Before stating the final omSVP-style assumption, we need to identify the efficiently computable public isometries of Q_{E_8} that are forced by the HAWK-style scalar rotations and symplectic structure. We treat the possible existence of further non-scalar public isometries as a separate cryptanalytic question [8], [22]

There is also a positive point here where HAWK's own specification says that, for its deployed parameter sets, the omSVP reduction is only a soundness argument and that the published parameters are ultimately justified by cryptanalysis, not by a non-vacuous formal reduction. In our current HAWK-E_8-CM, S_n raises the scale of obvious public coordinate vectors. So although a full reduction still has to be rewritten, the new form is less exposed to the exact vacuity mechanism that HAWK itself identifies in the identity form setting. Our conclusion is therefore that HAWK-E_8-CM's security story should be built around the fixed public class $[S_n]$, automorphism-aware one-more short vectors, and lattice reduction experiments on Q_{E_8}, rather than around an identity class omSVP statement copied verbatim from HAWK.

Let us now begin the proof. The point of this proof is that the ordinary HAWK formula does

not disappear completely, but has to be modified by the determinant of S_n. We now show that the corrected map is still public, integral, and norm preserving.

Write

$$R = R_n = \mathbb{Z}[X]/(X^n + 1), \qquad k = \frac{n}{4}, \qquad Y = X^k$$

Then $Y^4 = -1$, and the embedding matrix is

$$P_n = \begin{pmatrix} 2(1+Y) & 1 - Y + Y^2 + Y^3 \\ 0 & 1 + Y + Y^2 + Y^3 \end{pmatrix}$$

Its determinant is

$$\delta_n := \det(P_n) = 2(1+Y)(1+Y+Y^2+Y^3) = 4(Y + Y^2 + Y^3)$$

Since $S_n = P_n^\star P_n$ we also have $\det(S_n) = \det(P_n^\star)\det(P_n) = \delta_n^\star \delta_n =: \Delta_n$. In this construction,

$$\Delta_n = 48 + 32(Y - Y^3)$$

The useful replacement for HAWK's symplectic map comes from the matrix

$$\Omega_n := P_n^{-1} J_2 \overline{P_n}, \qquad J_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Although P_n^{-1} appears in the definition, the expression simplifies in R to

$$\Omega_n = \begin{pmatrix} -1 + Y - Y^3 & -2Y + 2Y^2 - 2Y^3 \\ Y + Y^3 & -1 + Y - Y^3 \end{pmatrix} \in M_2(R)$$

This integrality is important. It shows that the embedding matrix P_n still transports the symplectic symmetry to an integral map over the HAWK ring, but not in the same untwisted form as in HAWK.

Now let

$$Q_{E_8} = B^\star S_n B, \qquad B \in GL_2(R), \qquad \det(B) = 1$$

and set $C := P_n B$. Then

$$Q_{E_8} = C^\star C, \qquad \det(C) = \det(P_n)\det(B) = \delta_n$$

On the C side, the map $z \longmapsto J_2\overline{z}$ preserves the ordinary Euclidean norm. Pulling this map back to the w coordinates gives

$$w \longmapsto C^{-1} J_2 \overline{C}\,\overline{w}$$

A direct 2×2 adjugate calculation gives

$$C^{-1} J_2 \overline{C} = \delta_n^\star Q_{E_8}^{-1} J_2$$

Therefore the public anti-linear map is

$$\boxed{\omega_{E_8,Q}(w) := \delta_n^\star Q_{E_8}^{-1} J_2\,\overline{w}}$$

Here $Q_{E_8}^{-1}$ is taken over the fraction field, but the resulting map is integral. Indeed,

$$C^{-1} J_2 \overline{C} = B^{-1} P_n^{-1} J_2 \overline{P_n}\,\overline{B} = B^{-1}\Omega_n \overline{B}$$

Since

$$B^{-1},\ \Omega_n,\ \overline{B} \in M_2(R)$$

the map sends R^2 to itself. It is also bijective, because it is obtained by conjugating the bijection $z \mapsto J_2\overline{z}$. Hence $\omega_{E_8,Q}$ is a well defined automorphism of the underlying module R^2.

It remains to check that it preserves the public norm. For any $w \in R^2$, $C\,\omega_{E_8,Q}(w) = J_2\overline{Cw}$. Therefore

$$\|C\,\omega_{E_8,Q}(w)\|^2 = \|J_2\overline{Cw}\|^2 = \|Cw\|^2$$

Since $Q_{E_8} = C^\star C$, this is $\|\omega_{E_8,Q}(w)\|_{Q_{E_8}}^2 = \|w\|_{Q_{E_8}}^2$. So $\omega_{E_8,Q}$ is a public isometry of Q_{E_8}.

The relations with the scalar rotations are the same as expected from the HAWK symplectic symmetry, except that the square of the anti-linear generator is the central sign. First, $\Omega_n\overline{\Omega_n} = -I_2$ and therefore $\omega_{E_8,Q}^2 = -1$. Second, since $\omega_{E_8,Q}$ is conjugate-linear and $X^\star = X^{-1}$, $\omega_{E_8,Q}(X^i w) = X^{-i}\omega_{E_8,Q}(w)$. Thus the public group generated by the surviving scalar rotations

and the determinant twisted symplectic map is

$$\boxed{G_{E_8,Q} = \langle X \cdot I_2,\ \omega_{E_8,Q} \rangle = \{\, X^i,\ X^i \omega_{E_8,Q} : 0 \le i < 2n \,\}}$$

It has $4n$ elements and satisfies

$$X^{2n} = 1, \qquad \omega_{E_8,Q}^2 = X^n = -1, \qquad \omega_{E_8,Q} X \omega_{E_8,Q}^{-1} = X^{-1}$$

Equivalently, if $r = X$ and $\omega = \omega_{E_8,Q}$, then

$$G_{E_8,Q} = \langle r, \omega \mid r^{2n} = 1,\ \omega^2 = r^n,\ \omega r \omega^{-1} = r^{-1} \rangle$$

After quotienting by the central subgroup $\{\pm 1\}$, this gives the usual dihedral action on rotation orbits. This is the same HAWK-style orbit structure, but with the symplectic generator replaced by its determinant twisted E_8 preconditioned version.[8], [22]

It remains to justify that the scalar part has not gained any extra public isometries. If a scalar $\alpha \in R^\times$ preserves every Hermitian form in this setting, then it must satisfy $\alpha^\star \alpha = 1$. Under every complex embedding this gives

$$|\sigma(\alpha)| = 1$$

By Kronecker's theorem, such an algebraic integer is a root of unity. In the power of two cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$, the roots of unity are

$$\{1, X, X^2, \ldots, X^{2n-1}\}$$

So the scalar public isometries are the powers of X, as in HAWK.[8]

Therefore the public orbit that should be removed in the HAWK-E_8-CM one-more formulation is

$$G_{E_8,Q} \cdot w = \{\, X^i w,\ X^i \omega_{E_8,Q}(w) : 0 \le i < 2n \,\}$$

A one-more statement for this is then: given Q_{E_8}, samples $w_1, \ldots, w_N \leftarrow D_{Q_{E_8},\sigma}$ and the public group $G_{E_8,Q}$, output a short vector $w^\star \in R^2$ such that

$$w^\star \notin \bigcup_{j=1}^{N} G_{E_8,Q} \cdot w_j$$

This completes the construction of the uniform public isometry group forced by the scalar rotations and the determinant twisted symplectic map. The statement does not prove that this group exhausts all efficiently computable non-scalar public isometries for generic keys, nor does it rule out accidental extra automorphisms for special sampled public keys. Thus the HAWK-E_8-CM one-more problem should quotient at least by the group generated by powers of X and the determinant twisted symplectic isometry above. Treating this as the complete public orbit requires the additional assumption that no further efficiently computable public isometries exist for generic public forms in the class $[S_n]$ [22].

This also does not prove full signature security. The rest of HAWK's reduction would still have to be redone in the fixed public form class $[S_n]$, including the mod-2 bad-event and fixed coset counting arguments for the determinant twisted map $\omega_{E_8,Q}$, rather than copying them directly from the identity form setting.

4.3.7 Outcome of the HAWK-E_8-CM Construction

The final mathematical proposal is:

$$\boxed{\begin{aligned} x_E \in \mathcal{C}_t = 2M_n + P_n s(t), \qquad t = Bh \bmod 2, \\ x_E = P_n B w, \qquad Q_{E_8} = B^\star P_n^\star P_n B \end{aligned}}$$

This defines the uncompressed HAWK-E_8-CM construction. The signer samples using the Euclidean E_8 block norm, the parity target is matched through the internal P_n coordinate quotient, and the verifier checks the same norm through the modified public form.

The construction therefore changes the scheme interface, not only the sampling algorithm. Its public keys lie in the fixed form class $[S_n]$, so key recovery and forgery assumptions must be stated for that class rather than copied from the identity form setting. The public isometry analysis above constructs the uniform public symmetries forced by the scalar rotations and the determinant twisted symplectic map, which should be quotiented at least in the corresponding one-more formulation. Proving that these exhaust all efficiently computable public isometries, and giving a full HAWK-style security proof, remains outside the scope of this thesis.

The implementation described below realises this uncompressed path and tests our sampler, the modified public form, the verifier, and the signing path. The remaining issues are compression, compact encoding, full security analysis, and production hardening.

4.4 Software Implementation of HAWK-E_8-CM

The previous sections described the mathematical construction needed, we now describe the experimental implementation of the uncompressed HAWK-E_8-CM path. The purpose of the code is to validate the arithmetic, coset interface, sampler interface, signing equation, and verification equation defined above, while also enabling initial benchmarking of our new scheme. It is not a production implementation of HAWK, and it is not constant time.

We keep our implementation separate from HAWK. The original HAWK public API, compressed signature format, signing path, verification path, and standard tests remain unchanged. The E_8 code is enabled only for experimental targets through the compile time flag `HAWK_ENABLE_E8_EXPERIMENTAL`. We reuse HAWK's ring representation and secret basis data where useful, but the HAWK-E_8-CM path is a separate experimental construction.

4.4.1 Repository Integration and Data Flow

The implementation is organised around five experimental source files.

File	Role in the prototype
`hawk_e8_inner.h`	Internal declarations for the experimental E_8 path, guarded by `HAWK_ENABLE_E8_EXPERIMENTAL`.
`e8_math.c`	Arithmetic helpers for applying P_n, applying $S_n = P_n^\star P_n$, constructing Δ_n, and computing the expanded public form $Q_{E_8} = B^\star S_n B$.
`e8_sampler.c`	Coset matched Construction A sampler for the internal E_8 block cosets, including runtime probability tables, cache logic, and optional block parallelism.
`e8_sign.c`	Uncompressed signing path. It computes $t = Bh \bmod 2$, calls the E_8 sampler, maps back through B^{-1}, applies the sign rule, and outputs $s = (h - w)/2$.
`e8_vrfy.c`	Uncompressed signature decoding, sign rule checking, and the completion of squares verifier for the modified public form.

Table 4.1: Main experimental source files for HAWK-E_8-CM.

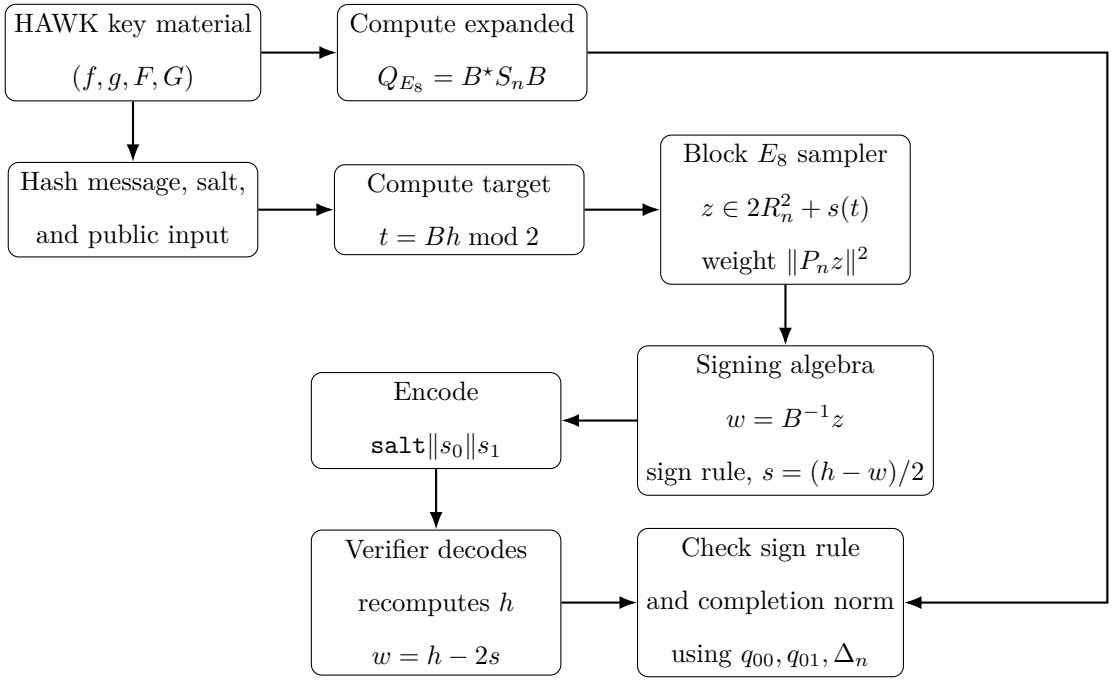

Figure 4.1: Implemented data flow for the uncompressed HAWK-E_8-CM path.

Figure 4.1 shows the data flow. The left hand side reuses HAWK key material and message hashing, while the right hand side is the E_8 path. The difference from HAWK is that the sampler outputs z coordinates whose length is measured after applying P_n. Verification must therefore use the modified public form $Q_{E_8} = B^\star S_n B$.

4.4.2 Arithmetic and Public Form Helpers

The implementation works over

$$R_n = \mathbb{Z}[X]/(X^n + 1), \qquad n \in \{256, 512, 1024\}, \qquad k = \frac{n}{4}, \qquad Y = X^k$$

In code, multiplication by powers of $Y = X^k$ is implemented as a signed coefficient permutation in R_n, rather than as a full polynomial multiplication. This is used to apply both P_n and S_n.

The helper `e8_apply_P` applies P_n to a pair of coefficient arrays (z_0, z_1). Conceptually, it implements

$$x_0 = 2(z_0 + Yz_0) + z_1 - Yz_1 + Y^2 z_1 + Y^3 z_1,$$
$$x_1 = z_1 + Yz_1 + Y^2 z_1 + Y^3 z_1$$

The corresponding core loop is a signed shift and add computation.

Listing 4.1: Core idea behind applying P_n.

```
for (size_t u = 0; u < n; u++) {
    z0_y0 = z0[u];
    z0_y1 = X_shift(z0, u, k);

    z1_y0 = z1[u];
    z1_y1 = X_shift(z1, u, k);
    z1_y2 = X_shift(z1, u, 2*k);
    z1_y3 = X_shift(z1, u, 3*k);

    out0[u] = 2*(z0_y0 + z0_y1)
            + z1_y0 - z1_y1 + z1_y2 + z1_y3;
    out1[u] = z1_y0 + z1_y1 + z1_y2 + z1_y3;
}
```


Given HAWK secret basis data B, $fG - gF = 1$, we compute the expanded public form $Q_{E_8} = B^\star S_n B$. The four entries $q_{00}, q_{01}, q_{10}, q_{11}$ are stored in the prototype. This is deliberately less compact than HAWK's public key representation, but it keeps the verifier and diagnostics simple while validating the modified norm formula. Compressed E_8 public key reconstruction is not implemented.

4.4.3 Coset Layout and Sampler Contract

The implementation follows the internal P_n coordinate interface from Section 4.2. For a signing target $t = Bh \bmod 2$, we sample $z \in 2R_n^2 + s(t)$ and measure the sample by $\|P_n z\|^2$. Equivalently, the embedded vector is

$$x_E = P_n z \in 2M_n + P_n s(t)$$

This matches the internal quotient map $\pi_M(P_n z) = z \bmod 2$.

The full n-coefficient problem is reorganised into $k = n/4$ independent eight-dimensional blocks. For block index r, the implementation extracts the label

$$\tau_r = \big(t_0[r], t_0[r+k], t_0[r+2k], t_0[r+3k], t_1[r], t_1[r+k], t_1[r+2k], t_1[r+3k]\big) \bmod 2$$

Thus each $2E_8$ block receives one full 8 bit internal coset label. This is the code level counterpart

of the decomposition

$$M_n/2M_n \cong \bigoplus_{r=0}^{k-1} X^r(M/2M)$$

4.4.4 Construction A Sampler

The sampler is a coset matched Construction A sampler. For a block label τ, it samples $z_{\text{blk}} \in 2\mathbb{Z}^8 + \tau$ with weight proportional to

$$\exp\left(-\frac{\|Pz_{\text{blk}}\|^2}{2\sigma_{\text{sign}}^2}\right)$$

The corresponding E_8 side vector is $x_{\text{blk}} = Pz_{\text{blk}}$, so the block contract is $C_\tau = 2M + P\tau = P(2\mathbb{Z}^8 + \tau)$.

Internally, the sampler writes $z_{\text{blk}} = \tau + 2y$ and decomposes $y \bmod 2$ using the 16 components labelled by RM$(1, 3)$, equivalently the extended binary Hamming $[8, 4, 4]$ code. For each component, the sampler computes its Gaussian mass under the E_8 side norm, selects a component according to these masses, samples the eight shifted one dimensional coordinates inside the selected component, and reconstructs the coordinate vector z_{blk}. Debug builds check the coset condition and the block norm identity.

To avoid recomputing the same probabilities for every block, the sampler uses a runtime cache keyed by the σ_{sign} value. The cache stores the one dimensional shifted CDF tables, the component masses for the 16 RM$(1, 3)$ components, compact component selection CDFs, hot path tables for the most common one dimensional values, tail fallback data, and affine reconstruction data for block outputs.

Because these tables have finite support, the implemented distribution is an approximation to the ideal real valued discrete Gaussian. In the implementation, finite Gaussian weights, normalisers, component masses, and cached probability values are computed and stored using `double` arithmetic. The hot path CDF thresholds are stored as 64-bit integer values, and runtime sampling compares these thresholds against 64-bit uniform targets.

The total approximation error can therefore be separated into finite tail truncation, floating point table construction error, and CDF discretisation error. Writing these contributions as $\varepsilon_{\text{tail}}$,

$\varepsilon_{\text{arith}}$, and ε_{cdf}, the distance from the ideal discrete Gaussian can be estimated heuristically as

$$\varepsilon_{\text{DG}} \lesssim \varepsilon_{\text{tail}} + \varepsilon_{\text{arith}} + \varepsilon_{\text{cdf}}$$

This follows the standard floating point error modelling viewpoint, where rounding effects are treated separately from mathematical truncation and sampling discretisation errors [23], [24].

Since the Construction A product basis has squared norm 32, the one dimensional coordinate width is $\sigma_{\text{coord}} = \frac{\sigma_{\text{sign}}^{E_8}}{\sqrt{32}}$. Thus the worst case in the current parameter set is $\sigma_{\text{coord}} \leq \frac{3.669}{\sqrt{32}} \approx 0.649$. The table support radius is $R = 20\sigma_{\text{coord}} + 32$, so in the worst case $R \approx 44.97$. For any shifted one dimensional coset, there is an allowed integer point within distance $1/2$ of the centre. The denominator of the one dimensional Gaussian mass is therefore at least $\exp(-1/(8\sigma_{\text{coord}}^2))$, while the omitted mass is bounded by a crude two sided Gaussian tail beyond radius R. This gives

$$\eta_{\text{tail,1D}} \lesssim 4\exp\left(-\frac{R^2 - \frac{1}{4}}{2\sigma_{\text{coord}}^2}\right)$$

At $n = 1024$, this gives $\eta_{\text{tail,1D}} < 2^{-3465}$. Since a full n-degree sample uses $2n$ one dimensional selections, the finite tail contribution is bounded by $\varepsilon_{\text{tail}} \lesssim 2n\,\eta_{\text{tail,1D}} < 2^{-3454}$ $(n = 1024)$. Thus finite tail truncation remains negligible compared with the other numerical effects.

The CDF discretisation contribution comes from two runtime paths. The main path uses 64-bit integer thresholds for the component choice and for the hot one dimensional choices. Since each E_8 block performs one component choice and eight one dimensional choices, and there are $n/4$ blocks, the fixed point contribution is therefore bounded crudely by $\frac{9n}{4} \cdot 2^{-64}$. At $n = 1024$, this is about $2^{-52.8}$. The second contribution comes from the floating point fallback path for one dimensional values outside the hot path. With $\sigma_{\text{coord}} \leq 0.649$, the probability of entering this fallback path is about $2^{-16.1}$ per one dimensional draw in the worst case. Since this fallback uses ordinary floating point CDFs, a conservative estimate for its contribution is on the order of $2n \cdot 2^{-16.1} \cdot 2^{-53} < 2^{-58}$ $(n = 1024)$ up to a small factor for the number of tail entries. Therefore, the CDF discretisation estimate is better summarised as $\varepsilon_{\text{cdf}} \lesssim 2^{-52.8}$.

The remaining term is floating point arithmetic error from constructing the tables. Since the probability tables are constructed using `double` arithmetic, under the round to nearest model the unit roundoff is taken as $u_{\text{D}} \approx 2^{-53}$. This is an engineering estimate based on the standard floating point error model [23]. Each table entry, normaliser, component mass, and CDF threshold is obtained from a small number of arithmetic operations together with evaluations of the Gaussian

exponential. Absorbing both the operation count and the numerical error from these exponential evaluations into a conservative constant C_{arith}, we estimate

$$\varepsilon_{\text{arith}} \lesssim \frac{9n}{4} C_{\text{arith}} u_{\text{D}}$$

Here $9n/4$ counts one component selection and eight one dimensional selections for each of the $n/4$ independent E_8 blocks. Taking C_{arith} between 2^8 and 2^{10} gives, for $n = 1024$,

$$\varepsilon_{\text{arith}} \lesssim 2^{-34} \text{ to } 2^{-32}$$

This is deliberately conservative and should be read as an implementation error estimate.

Combining the three estimates, the current implementation is therefore heuristically within about

$$\varepsilon_{\text{DG}} \lesssim 2^{-3454} + (2^{-34} \text{ to } 2^{-32}) + 2^{-52.8} \approx 2^{-32}$$

of the ideal real valued discrete Gaussian at $n = 1024$, under the above floating point error model.

Finally, in our implementation the serial sampler is the default reference path while we also have a performance oriented path using parallelisation. Since there are $k = n/4$ independent blocks, we can process contiguous block ranges using a pthread worker pool. Randomness is derived from independent SHAKE streams, with no shared mutable RNG state between worker ranges. This affects timing and reproducibility of particular random traces, but it does not change the mathematical sampler contract.

The sampler remains experimental as although it has very small tail error and small CDF discretisation error, it is still based on floating point table construction, data dependent table access, and data dependent control flow.

4.4.5 Signing, Verification, and Encoded Objects

The sampler backed signing path implements the uncompressed construction from 4.3.1. After hashing the message and salt to obtain $h = (h_0, h_1)$, the signer computes $t = Bh \bmod 2$. The sampler returns $z \in 2R_n^2 + s(t), \quad x_E = P_n z$. The signing code then computes

$$w = B^{-1}z, \qquad B^{-1} = \begin{pmatrix} G & -F \\ -g & f \end{pmatrix}$$

and outputs $s = \frac{h-w}{2}$. The division by 2 is well defined because $z \equiv Bh \pmod 2$, and hence $w \equiv h \pmod 2$.

Before encoding, we apply a sign based symmetry break. It compares w and $-w$, scans the coefficients of w_1 from index 0 upward, then scans w_0 only as a fallback, and keeps the representative whose first nonzero coefficient is positive, then the all zero vector is accepted.

The verifier decodes the uncompressed signature, recomputes h, and reconstructs $w = h - 2s$. It checks the same sign representative rule and then evaluates the modified completion of squares norm.

$$n\|w\|^2_{Q_{E_8}} = \mathrm{Tr}(q_{00}ee^\star + \Delta_n dw_1^\star)$$

where $d = \frac{w_1}{q_{00}}, \quad e = w_0 + q_{01}d$. The acceptance condition is $\|w\|^2_{Q_{E_8}} \le 8n\sigma^2_{\mathrm{verify},E_8}$.

Only an uncompressed signature format is implemented: $\sigma_{E_8} = \texttt{salt} \,\|\, s_0 \,\|\, s_1$. The arrays s_0 and s_1 contain n signed 16 bit coefficients each, stored in little endian form. Thus s_0 and s_1 contribute $4n$ bytes in total. The salt lengths follow the HAWK parameter sizes, so the current uncompressed sizes are

n	salt length	uncompressed signature size (bytes)
256	14	$14 + 4n = 1038$
512	24	$24 + 4n = 2072$
1024	40	$40 + 4n = 4136$

4.4.6 Implementation Boundary

The implementation is a research prototype. Its purpose is to exercise the uncompressed HAWK-E_8-CM arithmetic and data flow. The following parts remain outside the current implementation:

- constant time sampling and signing.
- side channel hardening.
- an AVX2 optimised E_8 sampler.
- compressed E_8 public keys, compressed E_8 signatures and a production E_8 key format.
- a full security reduction or qROM style unforgeability proof for the modified public form class.

The tests and diagnostics used to support the implementation are described in Chapter 5. The performance and calibration measurements are reported in Chapter 6. Further repository details, including build flags, experimental targets, and helper interfaces, are given in the appendix.

The implementation is available at

`https://github.com/Maximilian-Adam/hawk_e8`

The results in chapter 6 were generated from commit `2301763`.

5

Test Plan & Verification

Contents

This chapter describes how HAWK-E_8-CM was tested and what level of correctness is supported by the current implementation. The purpose of these tests is to check that the implemented arithmetic, coset handling, sampler interface, signing path, verification path, and diagnostic tools match the construction described in Chapter 4. The tests do not prove full cryptographic security or deployment readiness.

5.1 Testing Objectives

The main risks introduced by HAWK-E_8-CM are the modified public form, the internal P_n coordinate coset interface, and the blockwise E_8 sampler. The tests were therefore designed to check the following points:

- The fixed matrices P_n, $S_n = P_n^\star P_n$, and the determinant factor Δ_n are implemented consistently with the formulas in Chapter 4.
- The expanded public form

$$Q_{E_8} = B^\star S_n B$$

is computed correctly from the HAWK secret basis.

- The verifier checks the modified E_8 side norm.

- The uncompressed signing path produces signatures satisfying

$$s = \frac{h - w}{2} \in R_n^2$$

so that verification reconstructs the same $w = h - 2s$.

- The sampler uses the internal coset interface

$$C_t = P_n(2R_n^2 + s(t)), \qquad t = Bh \bmod 2$$

- $w = h - 2s$ satisfies the same symmetry-breaking rule enforced by the signer, and the verifier rejects the opposite representative $-w$.

- Valid signatures verify, while controlled corruptions of the message, signature, salt, public hash input, or public form are rejected.

- Record enough norm, rejection, public form size, sampler timing, and full signing timing data to support results.

5.2 Test Organisation

The main E_8 correctness target is

```
make -C Reference_Implementation test-e8
```

This runs the E_8 arithmetic, public form, verification, signing, sampler, and sampler backed signing tests.

Test	Checks	Result
`test_e8_math`	P_n, $S_n = P_n^\star P_n$, and Δ_n helpers.	Passes
`test_e8_public`	Expanded public form algebra for $Q_{E_8} = B^\star S_n B$.	Passes
`test_e8_verify`	Uncompressed verifier and agreement between completion norm and direct Q_{E_8}-norm.	Passes
`test_e8_sign`	Dummy signing algebra and uncompressed signature encoding.	Passes
`test_e8_sampler`	Block/full sampler support, internal cosets, and norm consistency.	Passes
`test_e8_sign_sampler`	Sampler backed uncompressed signing and verification.	Passes

Table 5.1: E8-specific correctness tests run by the `test-e8` target.

These tests establish that the implemented uncompressed E_8 path is internally consistent at the arithmetic, verification, sampler, and sign/verify interface level.

The histogram diagnostics are generated with

```
make -C Reference_Implementation e8-histograms
```

These produce the public form and signature diagnostic CSVs used to record expanded public form coefficient ranges, valid signature outcomes, tampering rejection, norm checks, coset checks, and signature level diagnostic data.

Additional sampler validation diagnostics are generated with

```
make -C Reference_Implementation e8-validation-summary
```

This produces the low-shell, all coset, global norm, and stability diagnostic CSVs. These are used to check the internal E_8 coset interface, blockwise norm factorisation, and low-shell sampler behaviour.

The rejection summary is generated with

```
make -C Reference_Implementation e8-rejection-summary
```

This records aggregate signing, rejection, norm-bound, norm-margin, and failure statistics over larger runs.

The parameter sweep diagnostics are generated with

```
make -C Reference_Implementation e8-sigma-sign-sweep
make -C Reference_Implementation e8-sigma-verify-sweep
```

These record how the sampler and signing path behave as $\sigma_{\text{sign}}^{E_8}$ and $\sigma_{\text{verify}}^{E_8}$ are varied.

Sampler isolated timing is generated through

```
make -C Reference_Implementation sampler-bench
```

Although for accurate timing operate in isolated-matrix mode with debug checks and sampler profiling disabled.

Full signing path timing is generated through

```
make -C Reference_Implementation sign-bench
```

5

5.3 Coset and Norm Verification

The most important interface test is the internal coset check. For a signing target $t = Bh \bmod 2$ the sampler should output a vector whose P_n coordinate preimage satisfies

$$z \bmod 2 = t$$

The histogram diagnostics record this through fields such as

`coset_check_success`, `piM_matches_t`, `ambient_mod2_matches_t`, `coset_error_count`, `ambient_error_count`

The expected condition for valid signatures is

$$\texttt{piM\_matches\_t} = 1, \qquad \texttt{coset\_check\_success} = 1, \qquad \texttt{coset\_error\_count} = 0$$

The norm tests check that the E_8 side norm and the public verification norm agree. The diagnostic CSV records fields such as:

`pnorm`, `qnorm`, `norm_equal`, `norm_margin`, and `verify_bound`

For valid signatures, the expected condition is

$$\texttt{pnorm} = \texttt{qnorm}, \qquad \texttt{norm\_equal} = 1, \qquad \texttt{norm\_margin} \geq 0$$

This verifies that the signer and verifier are measuring the same E_8 preconditioned norm.

5.4 Sampler Validation and Timing Diagnostics

The sampler is tested both inside the signing path and in isolation. The signing path tests check that sampled vectors can produce valid signatures under the modified public form, while the isolated sampler benchmark measures the cost of the E_8 block sampler separately from hashing, reconstruction, rejection, and verification.

Additional validation diagnostics check three sampler specific properties: that sampled blocks lie in the intended internal $2E_8$ cosets, that the sum of the block norms agrees with the full E_8 side norm, and that the observed low-shell behaviour is consistent with the finite reference distribution. The outcomes of these diagnostics are reported in Chapter 6.

The rejection summary records aggregate signing behaviour, including total attempts, rejected attempts, empirical rejection rate, maximum attempts, norm statistics, norm margins, coset failures, norm mismatches, and verification failures. These results are also reported in Chapter 6.

5.5 What Has and Has Not Been Verified

The current tests support the following conclusions:

- the E_8 tests compile and pass through the `test-e8` target.
- the implemented P_n, S_n, Δ_n, and Q_{E_8} arithmetic is consistent with the tested formulas.
- the verifier's completion of squares formula agrees with the direct Q_{E_8} norm.
- the uncompressed sampler backed signing path produces signatures that verify.
- the symmetry-breaking rule is tested, including rejection of the negated representative.
- controlled tampering of valid signatures is rejected.
- the internal P_n coordinate coset condition is checked and passes in the diagnostic CSVs.
- sampler, rejection, public form size, and timing diagnostics are produced for the measurements analysed in Chapter 6.

We have not yet verified the following:

- a formal negligible distance bound from the target distribution.
- constant time or side-channel safe behaviour.
- cryptographic security beyond implementation level correctness. Have not yet proven unforgeability or resistance to structural attacks.
- performance conclusions beyond the benchmark configurations reported in Chapter 6.

These limitations define the boundary between the functional correctness evidence in this chapter and the empirical performance analysis reported in Chapter 6.

6

Results

Contents

6.1 Experimental Setup

All results in this chapter were obtained from the same HAWK-E_8-CM prototype, built from commit `2301763`. It is not constant time, uses floating-point probability computations in the E_8 sampler, and keeps the HAWK-E_8-CM signature uncompressed. The measurements in this chapter should therefore be read as prototype evidence for correctness, validation behaviour, rejection behaviour, sampler speed, and signature size overhead.

Item	Value
Repository commit	`2301763`
Operating system	Arch Linux
Kernel	`6.19.8-arch1-1`
Architecture	`x86_64`
Processor	AMD Ryzen 9 9950X3D 16-Core Processor
CPU layout	16 cores, 32 logical CPUs, 2 threads per core
Cache	L2: 16 MiB, L3: 128 MiB
Frequency scaling	`amd-pstate-epp`, governor `powersave`, boost enabled
Compiler	GCC 16.1.1

Table 6.1: Machine and build configuration used for experimental results.

6.2 Parameter Calibration

Before the final measurements, the two Gaussian parameters were calibrated separately to be in line with the updated HAWK-E_8-CM scheme.

The signing width $\sigma_{\text{sign}}^{E_8}$ controls the E_8 side Gaussian distribution, so it was chosen before tuning the verification radius. The sampling width should be at least the smoothing parameter for the full fixed signing coset [11]. For a fixed target $C_t = 2M_n + P_n s(t)$ the relevant period lattice is $2M_n$. Since $M_n \cong (2E_8)^{\oplus n/4}$, this gives $2M_n \cong (4E_8)^{\oplus n/4}$. We therefore use the smoothing estimate for the full signing space.

The smoothing parameter of E_8 can be estimated by

$$\eta_\varepsilon(E_8) \approx \frac{1}{\sqrt{2}} \sqrt{\frac{1}{\pi} \log\left(\frac{240}{\varepsilon}\right)}$$

[9]. For the full signing space there are $n/4$ independent E_8 blocks, so the number of shortest nonzero vectors changes from 240 to $240(n/4) = 60n$. The fixed coset period lattice is also scaled by a factor of 4, since $2M_n \cong (4E_8)^{\oplus n/4}$.

We use the standard deviation convention $\exp\left(-\frac{\|x\|^2}{2\sigma^2}\right), \quad s = \sqrt{2\pi}\sigma$, so the corresponding smoothing estimate in σ-normalisation is

$$\sigma_{\text{smooth}}(n, \varepsilon) \approx \frac{2}{\pi} \sqrt{\log\left(\frac{60n}{\varepsilon}\right)}$$

Using $\varepsilon = 2^{-32}$, the smoothing estimates were rounded upward to three decimal places, giving

$$\sigma_{\text{sign}}^{E_8} = 3.592,\ 3.631,\ 3.669$$

for $n = 256, 512, 1024$, respectively.

A sigma sign sweep was then run with a loose verification radius so that rejection behaviour did not determine the signing width. The diagnostics checked low shell concentration, internal coset behaviour, and Construction A component entropy. The closest smoothing admissible values already had near maximal component entropy, low first shell concentration, and no diagnostic failures, so the smallest smoothing admissible widths were selected.

After fixing $\sigma_{\text{sign}}^{E_8}$, the verification parameter $\sigma_{\text{verify}}^{E_8}$ was tuned. This parameter controls the public norm bound

$$L^2 = \left\lfloor 8n(\sigma_{\text{verify}}^{E_8})^2 \right\rfloor$$

The aim was to choose the smallest radius with retry pressure on the order of 10^{-5} to have similar behaviour to HAWK and able to be feasibly tested and observed. The values were estimated from the block norm statistics of the sigma sign sweep. If μ_{blk} and v_{blk} denote the empirical mean and variance of one sampled E_8 block norm, then the full norm over $n/4$ blocks was approximated by

$$\frac{n}{4}\mu_{\text{blk}} + z_{1-p}\sqrt{\frac{n}{4}v_{\text{blk}}}$$

where $p = 10^{-5}$ and $z_{1-p} \approx 4.265$. This gives the estimate

$$\sigma_{\text{verify}}^{E_8} \approx \sqrt{\frac{\frac{n}{4}\mu_{\text{blk}} + 4.265\sqrt{\frac{n}{4}v_{\text{blk}}}}{8n}}$$

This suggested verification parameters of

$$\sigma_{\text{verify}}^{E_8} = 2.03,\ 1.99,\ 1.95$$

for $n = 256, 512, 1024$, respectively.

The selected values were then confirmed using 1,000,000 accepted signatures for each parameter set. The observed retry rates can be seen in table 6.7. All three runs had zero verifier failures, zero coset failures, and zero norm mismatches. These experiments therefore provide empirical support for the parameter set in Table 6.2, which we use for the remaining measurements.

Parameter	E_8**-256**	E_8**-512**	E_8**-1024**
Centered binomial parameter η	2	4	8
$\sigma_{\text{sign}}^{E_8}$	3.592	3.631	3.669
$\sigma_{\text{verify}}^{E_8}$	2.030	1.990	1.950
$L^2 = \lfloor 8n(\sigma_{\text{verify}}^{E_8})^2 \rfloor$	8439	16220	31150
Salt length	14 bytes	24 bytes	40 bytes
Hashed public-key length	16 bytes	32 bytes	64 bytes
Uncompressed signature size	1038 bytes	2072 bytes	4136 bytes
Expanded public verification material	4112 bytes	8224 bytes	16448 bytes
Expanded private signing material incl. h_{pub}	1040 bytes	2080 bytes	4160 bytes

Table 6.2: Prototype HAWK-E_8-CM parameters used for the final experiments.

6.3 Correctness Status Before Measurement

Before collecting performance and calibration data, we re ran the HAWK reference tests and the HAWK-E_8-CM tests described in Table 5.1. This was done to ensure that the implementation was in a consistent state before recording timing, norm, rejection, and histogram diagnostics.

The HAWK reference tests completed successfully, including the SHAKE, codec, self-test, and sampler tests. The E_8 tests also completed successfully for $\log n \in \{8, 9, 10\}$, covering the arithmetic helpers, public form construction, uncompressed verification, signing path, Construction A coset-matched sampler, and sampler backed signing path.

We also tested whether modified signatures and public data were rejected by the verifier. For each valid signature, we generated six tampered cases: modifying the message, s_0, s_1, salt, public hash/input data, or public form.

$\log n$	n	Keys	Trials/key	Valid accepted	Tamper rejected	Result
8	256	10	100	1000/1000	6000/6000	Pass
9	512	10	100	1000/1000	6000/6000	Pass
10	1024	10	100	1000/1000	6000/6000	Pass

Table 6.3: Signature histogram correctness outcomes.

In addition to the tests described above, three extra validation diagnostics were run. These tests were designed to check the internal coset interface, the blockwise norm decomposition, and

the low-shell behaviour of the sampler.

Coset interface	**Global norm**	**Low-shell diagnostic**
All 256 internal block labels τ were tested, with 8 samples per label, for parity, coset membership, and norm consistency.	The full E_8 side norm was checked against the sum of the independent block norms for $n = 256, 512, 1024$.	Observed low-shell frequencies were compared against a finite reference distribution for all 256 internal labels.
No parity, coset, or norm failures were observed.	No block-sum or full-norm failures were observed over 1000 trials for each n.	The median probability error was approximately 5.09×10^{-3}.

Table 6.4: Additional validation diagnostics for the E_8 sampler and norm interface.

These tests confirm that the E_8 coset and norm interfaces behave consistently in the tested code paths, and that the low-shell diagnostic shows no obvious finite range distortion.

6.4 Norm Behaviour

The previous subsection checked the E_8 coset and norm interfaces at the pass/fail level. Now we report the numerical norm behaviour of accepted signatures under the prototype parameters. We now use five keys and 1000 accepted signatures per key for each value of $\log n$, giving 5000 accepted signatures at each parameter size.

$\log n$	n	σ_{sign}	σ_{verify}	Bound	Mean norm	Max norm	Min margin
8	256	3.592	2.03	8439	6611.9	8208	231
9	512	3.631	1.99	16220	13493.9	16096	124
10	1024	3.669	1.95	31150	27561.0	30856	294

Table 6.5: Observed norm behaviour under the prototype parameters.

The observed means are close to the expected scale of a high dimensional Gaussian sample. Using the rough continuous approximation $\|x_E\|^2 \approx \sigma_{\text{sign}}^2 \chi_{2n}^2$, the expected value would be $2n\sigma_{\text{sign}}^2$. For the three parameter sizes this gives approximately 6606.1, 13500.6, and 27569.3, respectively. The observed means, 6611.9, 13493.9, and 27561.0, remain very close to these values. This supports the expected norm scale of the sampler.

$\log n$	n	Observed mean	$2n\sigma_{\text{sign}}^2$	Ratio
8	256	6611.9	6606.1	1.001
9	512	13493.9	13500.6	1.000
10	1024	27561.0	27569.3	1.000

Table 6.6: Comparison between observed norms and the continuous Gaussian norm estimate.

6.5 Rejection Behaviour

With the current verification bounds, rejected signing attempts were observed, but only rarely. We used 1000000 accepted signatures for each value of $\log n$. The measured retry pressure remains low, with observed rejection rates between about 1.1×10^{-5} and 3.2×10^{-5}.

$\log n$	n	Accepted	Attempts	Rejected attempts	Rejection rate
8	256	1000000	1000018	18	1.80×10^{-5}
9	512	1000000	1000011	11	1.10×10^{-5}
10	1024	1000000	1000032	32	3.20×10^{-5}

Table 6.7: Observed rejection behaviour under the current prototype bounds.

Accepted signatures remained below the verification bounds, with no verifier failures, coset failures, or norm mismatches recorded.

6.6 Public Form Coefficient Sizes

The modified public form

$$Q_{E_8} = B^\star S_n B$$

has larger coefficients than HAWK's identity preconditioned form. We record the maximum absolute coefficient sizes for $\tilde{q}_{00}$, $\tilde{q}_{01}$, and $\tilde{q}_{11}$. The largest observed bit lengths are shown in Table 6.8.

$\log n$	n	Keys	$\max \lvert\tilde{q}_{00}\rvert$	bits	$\max \lvert\tilde{q}_{01}\rvert$	bits	$\max \lvert\tilde{q}_{11}\rvert$	bits
8	256	10	4696	13	6040	13	118336	17
9	512	10	18376	15	26116	15	811040	20
10	1024	10	69088	17	95160	17	6139912	23

Table 6.8: Observed maximum absolute coefficients in the expanded public form $Q_{E_8} = B^\star S_n B$, 10 keys, 10000 trials for each parameter size.

The dependent entry $\tilde{q}_{11}$ is larger, as expected, because it contains the entries involving F and G, whose coefficients are much larger than those of f and g. The largest observed $\tilde{q}_{11}$ coefficient used 17 bits for $n = 256$, 20 bits for $n = 512$, and 23 bits for $n = 1024$. These values are comfortably below the 31-bit NTT primes used by the reference implementation in HAWK. A final compressed public key format would still need a dedicated coefficient bound and encoding analysis.

6.7 Sampler Timing

The sampler timing benchmark uses separate single configuration invocations so each E_8 thread count is measured with its own worker pool, cache, and scheduler state. A total of 10 million recorded rows were collected per configuration using the command sequence seen here in the appendix. These timings correspond to the implementation using `double` probability tables and 64-bit hot path CDF thresholds.

The HAWK baseline is the serial reference sampler. The selected HAWK-E_8-CM configurations use spin workers with per worker randomness. We compare HAWK's sampler, amortised to a unit of eight coordinates, with the HAWK-E_8-CM block sampler. This is the closest direct standalone comparison because both samplers are measured inside the same benchmark harness and both are reported in cycles per amortised eight coordinate unit.

n	E_8 threads	HAWK med	HAWK p95	E_8 med	E_8 p95	Med speedup
256	8	751	763	174	274	4.32×
512	8	759	774	143	202	5.31×
1024	16	755	782	106	135	7.12×

Table 6.9: Standalone sampler timing comparison.

n	E_8 blocks	E_8 threads	HAWK mean	E_8 mean	Mean speedup
256	64	8	756.004	195.473	3.87×
512	128	8	763.351	153.210	4.98×
1024	256	16	758.911	112.269	6.76×

Table 6.10: Mean sampler costs.

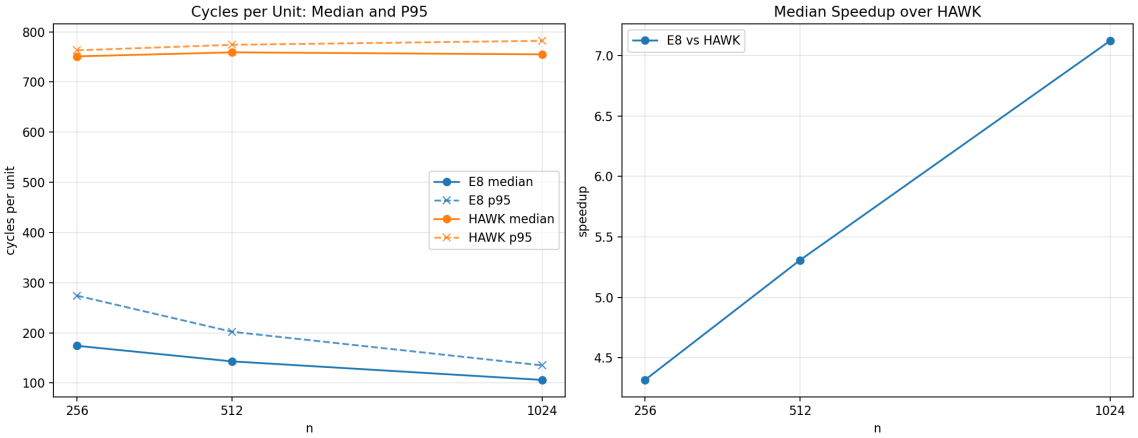

Figure 6.1: Summary of the selected E_8 configurations against the serial HAWK sampler.

The speedup increases with n because the E_8 sampler has more independent blocks to distribute across workers: 64, 128, and 256 blocks for $n = 256, 512, 1024$, respectively. The fixed overheads of worker scheduling, RNG setup, and cache use are therefore amortised over more block samples. By contrast, the HAWK sampler cost per eight coordinate unit stays roughly flat across these sizes, so the parallel E_8 advantage becomes larger as n grows.

The selected thread counts were chosen from a separate thread count sweep of five separate runs of 100,000 trials for each configuration, seen here. The best median settings in that sweep were 8 threads for $n = 256$, 8 threads for $n = 512$, and 16 threads for $n = 1024$. Larger thread counts did not improve the median cost and sometimes increased the tail cost. This drop off is expected once the available block level parallelism has mostly been exploited, because additional workers add synchronisation, scheduling, RNG state, and cache pressure, while each worker receives fewer E_8 blocks to amortise those overheads.

The focused runs at the chosen thread counts above can be seen to be slightly faster than they were in the broader thread count sweep. This is most likely because each selected configuration was

run for much longer and in isolation, so the CPU frequency, worker pool, caches, branch predictors, and benchmark state had more time to settle.

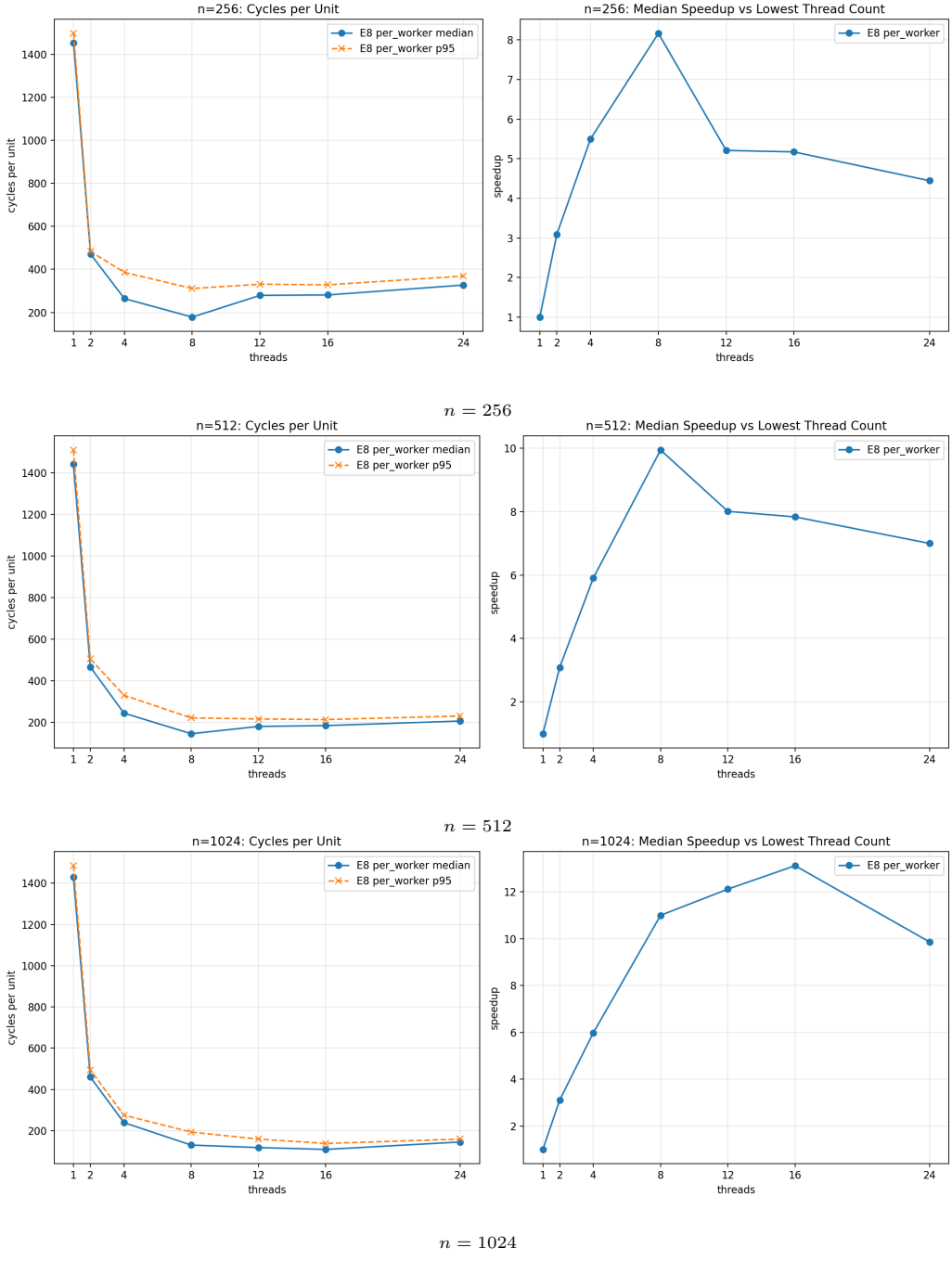

Figure 6.2: Thread count comparison used to choose the final E_8 sampler configurations.

The one thread E_8 value in the sweep should not be interpreted as an optimised serial sampler. It uses the same worker oriented structure and per block randomness as the parallel path, rather than a tight serial RNG loop.

These results show that the E_8 sampler is substantially faster than HAWK's sampler at the standalone sampler level. The improvement comes from both the direct sum decomposition $M_n \cong (2E_8)^{\oplus n/4}$ which exposes many independent eight-dimensional blocks, and the underlying E_8 lattice structure, which gives a compact finite coset geometry suitable for cached block sampling. Combined with per worker randomness and parallel sampling, this gives a sampler level speedup.

6.8 Signature Timing

The full signing benchmark was run using the same optimised build as the sampler benchmark, with debug checks and sampler profiling disabled. Each result pools 10 runs of 10000 trials, giving 100,000 signing trials. The E_8 configurations use the same selected thread counts as the sampler benchmark: 8 threads for $n = 256$, 8 threads for $n = 512$, and 16 threads for $n = 1024$. All reported trials completed successfully, with no failed signatures.

n	HAWK med	E_8 med	Med slowdown	HAWK p95	E_8 p95	Mean slowdown
256	93 611	658 502	7.03×	146 329	694 020	6.43×
512	196 166	2 587 611	13.191×	200 638	2 688 661	13.193×
1024	434 257	10 605 047	24.42×	442 857	11 188 471	24.51×

Table 6.11: Full signature generation timing. Values are cycles per signature.

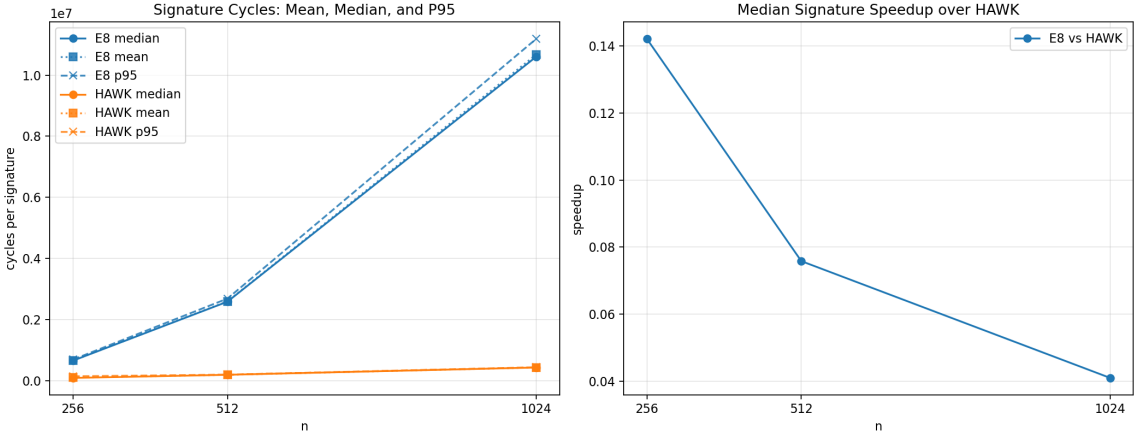

Figure 6.3: Full signing benchmark summary for HAWK and HAWK-E_8-CM.

These timings are used only for within benchmark comparison against the HAWK-E_8-CM prototype. The HAWK baseline here is the reference signing path, rather than the optimised AVX2 timings reported for HAWK-512 and HAWK-1024 in the HAWK specification [8]. For $n = 256$, $n = 512$ and $n = 1024$, our HAWK signing path is about 1.7×, 2.3×, and 2.4× slower than HAWK's stated timing. The specification also reports a fast signing mode, but that mode changes the signing interface by using a decoded private key and a faster RNG, so the normal signing figures are the more appropriate comparison here.

Also it can be seen that relative slowdown between HAWK and HAWK-E_8-CM increases with n. We believe this is due to parts of the prototype scaling poorly. Although the E_8 sampler benefits from having more independent blocks at larger n, the full signer also pays for reconstruction, modified public form arithmetic, and uncompressed output handling over larger polynomials. These costs grow with the ring dimension and dominate the saved sampler time, so the end to end gap widens as n increases.

The results show that the standalone sampler speedup does not yet translate into faster end to end signing. The current HAWK-E_8-CM signer is still an uncompressed prototype and performs substantially more work around the sampler than HAWK. Thus the E_8 sampler is faster as an isolated component, while the full signing path still needs engineering before it can compete with HAWK end to end.

6.9 Signature Size

The HAWK-E_8-CM prototype stores the full uncompressed signature $\texttt{salt} \,\|\, s_0 \,\|\, s_1$, with s_0 and s_1 stored as signed 16-bit coefficient arrays. Therefore the signature size is

$$\text{salt length} + 4n$$

The size overhead is expected, because compression is deliberately outside the current implementation boundary.

n	Salt bytes	HAWK compressed signature bytes	Current E_8 signature bytes	Ratio
256	14	249	1038	4.17
512	24	555	2072	3.73
1024	40	1221	4136	3.39

Table 6.12: Signature sizes for HAWK and the uncompressed HAWK-E_8-CM prototype.

For context, an uncompressed HAWK implementation using the same storage convention would likely have the same signature sizes and expanded public verification material as shown in Table 6.2. The signature would store $\texttt{salt} \,\|\, s_0 \,\|\, s_1$, and the verifier would store four 32 bit public form polynomials plus h_{pub}.

6.10 Summary

The results show that the implemented HAWK-E_8-CM path behaves coherently as an uncompressed prototype. The public form norm agrees with the sampled E_8 side norm, and no coset, norm mismatch, or verification failures were observed. The E_8 structure also made validation direct, since coset labels, block norms, and low-shell behaviour could be checked separately at the level of individual $2E_8$ blocks.

The performance results are very positive at the sampler level. In the sampler benchmark, selected E_8 configurations outperform the HAWK sampler by median factors of approximately $4.32\times$, $5.31\times$, and $7.12\times$ for $n = 256, 512, 1024$, respectively. This supports the central implementation claim that the structure of E_8 can lead to a faster sampling component. However, this benefit has not yet turned into an end to end signing speedup, because the HAWK-E_8-CM signing path is still an uncompressed diagnostic prototype. The current signature format is also several times larger than HAWK's compressed signatures. However, we have still shown a clear sampler level advantage, which could translate into an end to end signing speedup in an optimised HAWK-E_8-CM implementation.

7

Evaluation

Contents

7.1 Evaluation Against Initial Objectives

The original aim of this thesis was to determine whether a remarkable lattice could be embedded correctly into the HAWK setting and used as the basis for a structured sampling interface that could improve efficiency. The results show that this aim was largely met, as summarised in Table 7.1.

Objective	Outcome	Status
Identify a congruence defined O_8-module with E_8.	The module $M \subset O_8^2$ was shown to have underlying $\mathbb{Z}$-lattice isometric to $2E_8$, and an explicit O_8 basis was found.	Achieved
Extend the construction into HAWK's ring R_n.	The basis was extended through $\zeta_8 \mapsto X^{n/4}$, giving $M_n = P_n R_n^2$ with underlying lattice $(2E_8)^{\oplus n/4}$.	Achieved
Understand the mod-2 signing interface.	Have shown that HAWK's ambient coefficientwise reduction does not recover the internal quotient $M_n/2M_n$.	Achieved
Design a coherent HAWK-like variant if direct replacement fails.	The internal coset map $\pi_M(P_n z) = z \bmod 2$ was used to define the cosets $C_t = 2M_n + P_n s(t)$, and verification was changed with the modified public form $Q_{E_8} = B^\star P_n^\star P_n B$.	Achieved
Implement a proof of concept signing and verification path.	The uncompressed HAWK-E_8-CM path was implemented and tested for $n \in \{256, 512, 1024\}$, including signing, verification, coset checks, norm checks, rejection diagnostics, and timing.	Achieved
Assess possible advantages for efficiency.	The final standalone sampler benchmark shows a sampler level speedup over HAWK, selected configurations giving median speedups of approximately 4.32×, 5.31×, and 7.12×. This is not yet an end to end signing speedup, because the full HAWK-E_8-CM signing path is still an uncompressed prototype.	Achieved at the sampler level

Table 7.1: Evaluation of the final outcome against the objectives stated in the Introduction.

7.2 How the Objectives Changed

At the start of the work on this thesis, the main idea was fairly simple: remarkable lattices might help improve the sampling stage of HAWK. In that early version, the work looked like it might

mostly be a sampler optimisation problem where we would find a useful E_8 based sampler and try to place it inside the signing algorithm.

As the work developed, it became clear that this was not enough. HAWK's signing procedure is not just a standalone sampler. It is tied to a specific coefficientwise mod-2 coset and to the public quadratic form used during verification. After embedding the E_8 block module into R_n^2, we found that the natural quotient $M_n/2M_n$ is not the same as ambient coefficientwise reduction. This meant that the original idea of simply replacing the sampler was too narrow. The real problem was to understand whether the signing interface itself could be changed correctly.

Keeping HAWK's original interface would have made the role of E_8 much less meaningful. If the sampler had to measure the pulled-back norm $\|P_n^{-1}x_E\|^2$, then the E_8 module would mostly be acting as a different coordinate system, rather than as the actual Euclidean geometry used for sampling. Our final construction instead samples using the Euclidean E_8 block norm and changes the public form so that the verifier checks the same norm. This made the work more scheme level than we first expected, but it also made the final construction much more meaningful.

7.3 Evaluation of Correctness and Implementation Quality

The implementation achieved what it needed to. Producing a working uncompressed signing and verification path for the proposed HAWK-E_8-CM construction. The tests in Chapter 5 cover the arithmetic helpers, construction of the public form, coset handling, signing, verification, and the sampler backed signing path. The results in Chapter 6 then show the expected behaviour in the measured runs: valid signatures were accepted, tampered inputs were rejected, and the public verification norm agreed with the norm measured on the E_8 side.

These results give confidence that the algebraic interface has been implemented consistently. In particular, the agreement between the sampled norm and the verification norm supports the choice of public form $Q_{E_8} = B^\star P_n^\star P_n B$. The coset diagnostics also support the use of the internal P_n coordinate quotient.

The tests establish implementation consistency for the exercised cases. They do not prove full cryptographic security or exact sampler distribution. The sampler remains a prototype, with a production version requiring a fixed point, constant time implementation and side-channel hardening.

7.3.1 Evaluation of Performance

The performance results support a sampler level advantage for the HAWK-E_8-CM construction, but not yet a full signing path advantage. In the sampler benchmark, the cached and parallel E_8 sampler was faster than the ordinary HAWK sampler for all three tested parameter sizes. Using the selected stable configurations from Table 6.9, the median speedups were approximately 4.32× for $n = 256$, 5.31× for $n = 512$, and 7.12× for $n = 1024$. The improvement comes from the fact that M_n decomposes into $n/4$ independent $2E_8$ blocks and the structure of the E_8 lattice.

The full signing path result is less favourable. The current HAWK-E_8-CM signer is uncompressed and diagnostic, so it performs more work around the sampler than HAWK. It also stores both signature polynomials and uses the modified public form machinery needed to test the construction. Consequently, the end to end signing benchmark should not be treated as a production comparison. In conclusion the E_8 sampler has shown a clear standalone speed benefit, while further engineering would be required for that benefit to carry through to full signature generation.

7

7.4 Evaluation of Signature and Public Key Practicality

The current uncompressed signature format is not practically competitive with HAWK. It stores $\texttt{salt} \,\|\, s_0 \,\|\, s_1$. As shown in Chapter 6, this makes the signature several times larger than the corresponding HAWK signatures. This would need to be corrected for a final signature scheme.

The public form results are more positive. The observed coefficients of $\tilde{q}_{00}$, $\tilde{q}_{01}$, and $\tilde{q}_{11}$ all stayed well within the existing 31-bit NTT arithmetic range. This is useful, because it means that changing the public form does not immediately require a new modular arithmetic setup. At the same time, this is still only experimental evidence from the measured keys. A final public key format would need a proper coefficient bound argument and a clear compression strategy.

A similar situation applies to compressed signatures. The mathematical reconstruction rule is already clear, and the determinant term does not change where the s_0 reconstruction minimiser occurs. However, the full compression argument from HAWK cannot simply be reused. The modified public form changes the coefficient distributions and the fixed point error analysis, so compression is best treated as future work. In other words, currently we have established the right mathematical shape of compression, but not yet a complete compressed implementation or proof.

7.5 Comparison with HAWK

Compared with HAWK [1], [8], the current strength of HAWK-E_8-CM is that it gives a more structured and efficient way to think about the sampling problem. The construction exposes a blockwise E_8 geometry, together with a clean internal coset interface. Each block has a well defined E_8 side target coset, and the verifier checks the same E_8 side norm through the modified public form. This is encouraging, because it turns the sampling problem into something more organised and testable.

The trade-off is that this is no longer HAWK with one component swapped out. The public form changes from $Q = B^\star B$ to $Q_{E_8} = B^\star S_n B, \quad S_n = P_n^\star P_n$.

This means that the security setting has to be stated in the modified public form class. In Chapter 4, we already reformulated direct key recovery as search module LIP in the class $[S_n]$, and we described the corresponding one-more short-vector problem for the preconditioned form. We have also identified the public isometry group generated by powers of X and the determinant twisted symplectic map, which gives the orbit that should be quotiented out in the one-more formulation.

What remains is a complete security analysis around it as HAWK's reduction cannot be copied unchanged. The mod-2 bad event arguments, fixed coset counting, and parameter supporting cryptanalysis would need to be repeated for the HAWK-E_8-CM public form class [14], [15]. This is still a substantial task, but we have already clarified the correct security setting in which that future analysis should take place.

7.6 Relation to Existing Sampling and Lattice Work

This thesis differs quite significantly from standalone work on discrete Gaussian sampling over structured lattices [9], [11]. A standalone E_8 sampler only has to sample from the right Euclidean lattice or coset. In a signature scheme, the sampler must also fit the signing target, the secret basis transformation, the verification norm, and the public key format. We have shown here that these interfaces determine whether the sampler can be used at all.

Compared with broader remarkable-lattice or lattice-isomorphism work [2], [5], we have a greater focus on implementation. Rather than trying to classify every remarkable lattice that

might fit into HAWK, we use E_8 as a controlled test case and follow the idea all the way through to a working prototype. This made it possible to see not only the attractive parts of the structure, such as the block geometry and coset information, but also the extra constraints it creates once it is placed inside a real signature framework.

The E_8 case we have produced suggests a general methodology exists rather than a universal construction. For any remarkable lattice one would first need to find a compatible module realisation inside the HAWK ring, then analyse the induced mod-2 quotient, the sampler norm, and the resulting public quadratic form. Some lattices may have embeddings whose reduction modulo 2 is invertible, in which case the coset-matching problem would be much simpler than in the E_8 case. Similarly, if the associated preconditioning form is publicly equivalent to the identity form, the public key class may remain close to that of ordinary HAWK. However, when the embedding defines a proper submodule or a new Euclidean geometry, one should expect a modified public form and hence a separate security formulation, otherwise it becomes less clear whether the introduced lattice has given genuinely new geometry rather than just rewriting HAWK in another basis. Thus we do not believe there exists a universal recipe that works unchanged for every remarkable lattice, but have created here a template for the checks that any such construction must pass.

7.7 Threats to Validity

There are also some boundaries to keep in mind when interpreting these results. They do not undermine the main contribution we provide, but they clarify what the current evidence can support.

First, the sampler is still experimental. The implementation uses floating point probability computations, data dependent table access, and data dependent control flow.

Second, the parameters are provisional. Final parameters would need to balance rejection rate, security level, compression failure probability, implementation cost, and side channel constraints.

Third, the security analysis is not yet a full scheme proof. We identify the correct public form class, reformulate the key recovery and one-more short-vector setting, and describe the relevant public isometries. A complete analysis would still need to repeat the HAWK arguments for the modified public form and support them with dedicated cryptanalysis.

Finally, we only focus on E_8 and cannot state how other remarkable lattices may perform.

Conclusions and Future Directions

This thesis set out to test whether the structure of a remarkable lattice, chosen to be E_8, could be brought into the HAWK setting in a mathematically meaningful way. The outcome was encouraging. Starting from a congruence defined $\mathbb{Z}[\zeta_8]$-module, we showed that the underlying lattice is isometric to $2E_8$, found an explicit module basis, and extended it into HAWK's ring to obtain repeated E_8 blocks inside R_n^2. This gives an algebraic route from a remarkable Euclidean lattice to the module lattice framework used by HAWK.

The most important lesson was that this idea cannot be treated as a simple sampler replacement. HAWK's coefficientwise mod-2 interface does not recover the internal quotient of the E_8 block module. Resolving this led to the coset-matched variant HAWK-E_8-CM, where signing uses internal P_n coordinates and verification uses the modified public form $Q_{E_8} = B^\star P_n^\star P_n B$. This is, in our view, the central achievement of the project. Identifying the obstruction and turning it into a clean new scheme design.

Our code implementation supports this design at the proof of concept level. The uncompressed signing and verification path works across the tested parameter sizes, the coset and norm diagnostics behave as expected, and the public form coefficients remain within the existing NTT arithmetic range. At the standalone sampler level, the selected E_8 configurations are also faster than the HAWK sampler, with median speedups of approximately (4.32×), (5.31×), and (7.12×) for (n=256, 512, 1024) respectively. The results also clearly show the limitations. The current full prototype is slower than HAWK, uses uncompressed signatures, relies on an experimental floating point sampler, and does not yet provide a full security proof. However, our implementation displays that a future optimised signing path may lead towards a full scheme speedup.

Overall, we consider the project successful because it answers the main research question in a useful way. Future work should now focus on a constant time sampler, compressed signatures and public keys, complete parameter calibration, and a full security analysis in the modified public form class. With those pieces in place, HAWK-E_8-CM could become a strong basis for further study of remarkable lattices in post-quantum signatures.

A

Appendix

A.1 Implementation Appendix

This appendix section documents the software interface of the E8 prototype.

A.1.1 Repository Structure

The implementation is split between the original HAWK code and the experimental HAWK-E_8-CM path. The ordinary HAWK public API, compressed signature format, and original tests are not changed by the E_8 code. The E_8 specific code is compiled only when

$$\texttt{HAWK\_ENABLE\_E8\_EXPERIMENTAL} = 1$$

is defined by the E_8 specific reference Makefile targets.

The implementation is available at

`https://github.com/Maximilian-Adam/hawk_e8`

The results in this thesis were generated from commit `2301763`.

File	Role
`src/hawk_e8_inner.h`	Gated declarations for the experimental E_8 helpers.
`src/e8_math.c`	Implements P_n, S_n, Δ_n, and Q_{E_8}.
`src/e8_vrfy.c`	Implements the uncompressed codec, sym-break, and completion verifier.
`src/e8_sampler.c`	Implements the coset matched Construction A E_8 sampler, including runtime cache logic and optional block level parallelism.
`src/e8_sign.c`	Implements dummy and sampler backed uncompressed signers.
`Reference_Implementation/`	Contains synced copies used by the reference test targets.
`tests/test_e8_*.c`	Unit and integration tests for the E_8 path.
`tests/e8_*.c`	CSV validation, histogram, rejection calibration, sampler benchmark, and signing benchmark harnesses for the experimental E_8 path.
`docs/HAWK_E8_VARIANT.md`	Repository level description of the current variant.

Table A.1: Repository structure for the experimental HAWK-E_8-CM implementation.

This separation keeps the baseline HAWK implementation intact while allowing the experimental E_8 preconditioned path to be built, tested, and benchmarked separately.

A.1.2 Build and Test Commands

All commands in this appendix are run from the repository root. The ordinary HAWK implementation, public API, compressed signature format, and original tests remain unchanged. The E_8 code is experimental and is enabled only for E_8 specific reference targets.

Build with

```
make build
```

Then the reference implementation can be built with

```
make -C Reference_Implementation
```

The HAWK reference tests are

```
Reference_Implementation/bin/test_self
Reference_Implementation/bin/test_codec
Reference_Implementation/bin/test_sampler
```

The full E_8 correctness test suite is run with

```
make -C Reference_Implementation test-e8
```

This target runs the tests for the P_n, S_n, and Δ_n helpers, the expanded public form Q_{E8}, the uncompressed verifier, dummy signing algebra, block/full sampler coset checks, and the sampler backed uncompressed sign/verify path.

The main diagnostic and benchmark targets used are:

```
make -C Reference_Implementation e8-histograms
make -C Reference_Implementation e8-validation-summary
make -C Reference_Implementation e8-rejection-summary
make -C Reference_Implementation e8-sigma-verify-sweep
make -C Reference_Implementation e8-sigma-sign-sweep
make -C Reference_Implementation sampler-bench
make -C Reference_Implementation sign-bench
make -C Reference_Implementation profile-sign-bench
```


These targets produce the following CSV outputs:

Output file	**Purpose**
`e8_hist_public.csv`	Public form coefficient histograms.
`e8_hist_signatures.csv`	Signature and signing output histograms.
`e8_validation_summary.csv`	Manifest for the validation diagnostics.
`e8_validation_shell.csv`	Low-shell sampler sanity check against a finite reference.
`e8_validation_coset.csv`	All 256 block-coset parity, coset, and norm checks.
`e8_validation_global.csv`	Global norm-factorisation checks.
`e8_validation_stability.csv`	Serial timing stability diagnostic.
`e8_rejection_summary.csv`	Signing rejection and norm-bound summary.
`e8_sampler_bench.csv`	Standalone HAWK and E_8 sampler timings.
`e8_sign_bench.csv`	Full signing path timing benchmark.

Table A.2: Main diagnostic and benchmark outputs produced by the reference implementation.

For sampler timing, the commands we used to generate the results are structured as follows:

```
for trial in $(seq 1 10); do
  out="Reference_Implementation/e8_sampler_bench_Trial${trial}.csv"

  Reference_Implementation/bin/e8_sampler_bench \
    --single-config --logn 8 --threads 8 \
    --worker-mode spin --rng-mode per_worker \
    --trials 1000000 > "$out"

  Reference_Implementation/bin/e8_sampler_bench \
    --single-config --logn 9 --threads 8 \
    --worker-mode spin --rng-mode per_worker \
    --trials 1000000 --no-header >> "$out"

  Reference_Implementation/bin/e8_sampler_bench \
    --single-config --logn 10 --threads 16 \
    --worker-mode spin --rng-mode per_worker \
    --trials 1000000 --no-header >> "$out"

  Reference_Implementation/bin/e8_sampler_bench \
    --single-hawk-sampler --logn 8 \
    --trials 1000000 --no-header >> "$out"

  Reference_Implementation/bin/e8_sampler_bench \
    --single-hawk-sampler --logn 9 \
    --trials 1000000 --no-header >> "$out"

  Reference_Implementation/bin/e8_sampler_bench \
    --single-hawk-sampler --logn 10 \
    --trials 1000000 --no-header >> "$out"
done
```

Before each timing run the following was run to collect accurate results by removing unneeded diagnostics:

```
make -C Reference_Implementation clean
```

```
make -B -C Reference_Implementation bin/e8_sampler_bench \
E8_CFLAGS='-O2 -DHAWK_ENABLE_E8_EXPERIMENTAL=1 -DHAWK_E8_DEBUG_CHECKS=0 -DHAWK_E8_PROFILE_SAMPLER=
```

The thread count sweep was run using this command sequence:

```
trials=100000

for trial in 1 2 3 4 5; do
  out="Reference_Implementation/e8_sampler_bench_thread_compare_Trial${trial}.csv"
  rm -f "$out"

  for logn in 8 9 10; do
    for threads in 1 2 4 8 12 16 24; do
      mode=spin
      if [ "$threads" -eq 1 ]; then
        mode=serial
      fi

      no_header=
      if [ -f "$out" ]; then
        no_header=--no-header
      fi

      Reference_Implementation/bin/e8_sampler_bench \
        --single-config \
        --logn "$logn" \
        --threads "$threads" \
        --worker-mode "$mode" \
        --rng-mode per_worker \
        --trials "$trials" \
        $no_header >> "$out"
    done
  done
done
```

the benchmark binary can also be run like so:

```
make -C Reference_Implementation bin/e8_sampler_bench
```

A

```
Reference_Implementation/bin/e8_sampler_bench --selected-configs
Reference_Implementation/bin/e8_sampler_bench --isolated-matrix
Reference_Implementation/bin/e8_sampler_bench --single-hawk-sampler --logn 10
Reference_Implementation/bin/e8_sampler_bench \
  --single-config --logn 10 --threads 8 \
  --worker-mode spin --rng-mode per_worker
```

The validation summary is only a diagnostic sanity check. It checks shell behaviour, coset support, and norm consistency, but it is not a formal statistical proof of the sampler distribution.

The rejection summary target also accepts

```
E8_REJECTION_KEYS=<positive integer>
E8_REJECTION_TRIALS=<positive integer>
E8_REJECTION_LOGN=8|9|10
```

A

A.1.3 Internal E8 Helper API

The most important internal helper functions are grouped in Tables A.3–A.5. These functions are reference helpers only and should not be interpreted as a stable public API.

Function	Purpose
`e8_apply_P`	Applies the preconditioner P_n to two coefficient arrays.
`e8_apply_S`	Applies $S_n = P_n^\star P_n$.
`e8_make_delta`	Constructs $\Delta_n = \det(S_n)$.
`e8_compute_qform`	Computes the expanded public form $Q_{E_8} = B^\star S_n B$.
`e8_qnorm_direct`	Evaluates the expanded Q_{E_8} norm directly, mainly for tests.
`e8_qnorm_completion`	Evaluates the verifier's completion of squares norm formula.
`e8_verify_bound_from_sigma`	Computes the bound $\lfloor 8n\sigma^2_{\text{verify}} \rfloor$.

Table A.3: Core arithmetic and norm helpers.

Function	Purpose
`e8_extract_tau`	Extracts the 8-bit internal P coordinate coset label for one block.
`e8_block_apply_P`	Applies the one-block P map.
`e8_block_norm2`	Computes $\|Pz_{\text{block}}\|^2$ for one block.
`e8_block_solve_P_checked`	Applies the checked P^{-1} bridge from an E_8 side block back to z coordinates.
`e8_rm13_codeword`	Returns RM$(1,3)$ code data used by the Construction A sampler.
`e8_rm13_syndrome`	Computes the corresponding RM$(1,3)$ syndrome.
`e8_sample_block_construction_a_cm`	Samples one coset-matched Construction A E_8 block.
`e8_sample_z_construction_a_cm`	Samples all $n/4$ coset-matched Construction A blocks.

Table A.4: Block sampler and coset helper API.

Function	Purpose
`e8_salt_len`	Returns the salt length for the chosen parameter size.
`e8_sig_uncompressed_size`	Returns the current uncompressed signature size.
`e8_encode_sig_uncompressed`	Encodes `salt` $\| s_0 \| s_1$.
`e8_decode_sig_uncompressed`	Decodes the uncompressed signature format.
`e8_sym_break`	Applies the sign based symmetry-breaking rule used by the uncompressed path.
`e8_verify_uncompressed`	Verifies an uncompressed E_8 signature with the default verifier parameter.
`e8_verify_uncompressed_with_sigma`	Verification variant that accepts an explicit σ_{verify}.
`e8_sign_dummy_uncompressed`	Dummy signer used to exercise algebra and encoding only.
`e8_sign_sampler_uncompressed`	Sampler backed uncompressed signer using the coset-matched Construction A CM sampler.
`e8_sign_sampler_trace_uncompressed`	Signing variant with trace outputs for tests and histogram logging.
`e8_sign_sampler_trace_timed_uncompressed`	Trace signing variant that also records timing data.

Table A.5: Signature, verification, and signing helper API.

A.1.4 Block Coset Extraction

The block sampler translates HAWK's two parity polynomials into one 8-bit internal P coordinate coset label per E_8 block. The implemented mapping is:

Listing A.1: Extracting the 8-bit coset label for block r

```
tau = 0;
for (unsigned u = 0; u < 4; u++) {
    tau |= (t0[r + u*k] & 1) << u;
    tau |= (t1[r + u*k] & 1) << (u + 4);
}
```

This is the implementation version of $\tau_r = \big(t_0[r], t_0[r+k], t_0[r+2k], t_0[r+3k], t_1[r], t_1[r+k], t_1[r+2k], t_1[r+3k]\big) \bmod 2$

τ_r is the internal P coordinate label. The sampler therefore returns a block $z_{\text{block}} \in 2\mathbb{Z}^8 + \tau_r$ and the measured E_8 side block is $x_{\text{block}} = Pz_{\text{block}}$. The block norm used by the sampler is $\|x_{\text{block}}\|^2 = \|Pz_{\text{block}}\|^2$

A

A.1.5 Norm Consistency Test

The arithmetic tests verify that applying S_n gives the same quadratic form as applying P_n and taking the Euclidean norm: $\|P_n z\|^2 = \langle z, S_n z\rangle$. The following simplified fragment shows the logic of the test.

Listing A.2: Test identity $\|P_n z\|^2 = \langle z, S_n z\rangle$.

```
e8_apply_P(pz0, pz1, z0, z1, logn);
e8_apply_S(sz0, sz1, z0, z1, logn);
lhs = 0;
rhs = 0;

for (size_t u = 0; u < n; u++) {
    lhs += pz0[u]*pz0[u] + pz1[u]*pz1[u];
    rhs += z0[u]*sz0[u] + z1[u]*sz1[u];
}

assert(lhs == rhs);
```

This test connects the two views used throughout the implementation. The sampler measures $\|P_n z\|^2$, while the public verification form is induced by $S_n = P_n^\star P_n$. In the verifier, the implemented norm check uses the completion of squares formula with the determinant term Δ_n. The slower expanded Q_{E_8} norm computation is kept in tests so that the completion formula can be checked against the direct public form norm while the path remains experimental.

A.1.6 Experimental Parameter and Storage Map

Table A.6 gives the parameter map from HAWK for the current uncompressed HAWK-E_8-CM prototype. Entries marked N/A are not finalised, usually because a compressed or serialised E_8 format is yet to be defined. The expanded storage rows describe the current test implementation, not deployment key sizes.

Table A.6: Full prototype parameter and storage map for the current uncompressed HAWK-E_8-CM implementation.

Parameter	**E8-256**	**E8-512**	**E8-1024**
Targeted security	Prototype	Prototype	Prototype
Bit security λ	N/A	N/A	N/A
Final serialised private key size	N/A	N/A	N/A
Underlying HAWK encoded private key size	96 bytes	184 bytes	360 bytes
Expanded private basis arrays (f, g, F, G)	1024 bytes	2048 bytes	4096 bytes
Expanded private signing material including h_{pub}	1040 bytes	2080 bytes	4160 bytes
Final serialised public key size	N/A	N/A	N/A
Expanded public verification material	4112 bytes	8224 bytes	16448 bytes
Signature size	1038 bytes	2072 bytes	4136 bytes
Degree n	256	512	1024
Transcript size limit q_s	2^{32}	2^{64}	2^{64}
Centered binomial parameter η	2	4	8
Signature standard deviation $\sigma_{\text{sign}}^{E8}$	3.592	3.631	3.669
Verification standard deviation $\sigma_{\text{verify}}^{E8}$	2.030	1.990	1.950
Verification bound $L^2 = \lfloor 8n(\sigma_{\text{verify}}^{E8})^2 \rfloor$	8439	16220	31150

A

Parameter	**E8-256**	**E8-512**	**E8-1024**
Key-recovery standard deviation σ_{krsec}	N/A	N/A	N/A
Salt length	112 bits / 14 bytes	192 bits / 24 bytes	320 bits / 40 bytes
Keygen seed length	128 bits / 16 bytes	192 bits / 24 bytes	320 bits / 40 bytes
Hashed public-key length h_{pub}	128 bits / 16 bytes	256 bits / 32 bytes	512 bits / 64 bytes
Stored q_{00} representation	1024 bytes	2048 bytes	4096 bytes
Stored q_{01} representation	1024 bytes	2048 bytes	4096 bytes
Stored q_{10} representation	1024 bytes	2048 bytes	4096 bytes
Stored q_{11} representation	1024 bytes	2048 bytes	4096 bytes
Stored s_0 representation	512 bytes	1024 bytes	2048 bytes
Stored s_1 representation	512 bytes	1024 bytes	2048 bytes
Signature decompression bound β_0	N/A	N/A	N/A

Bibliography

[1] L. Ducas, E. W. Postlethwaite, L. N. Pulles, and W. van Woerden, *Hawk: Module lip makes lattice signatures fast, compact and simple*, Cryptology ePrint Archive, Paper 2022/1155, 2022. [Online]. Available: https://eprint.iacr.org/2022/1155.

[2] L. Ducas and W. P. J. van Woerden, *On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography*, Cryptology ePrint Archive, Paper 2021/1332, 2021. [Online]. Available: https://eprint.iacr.org/2021/1332.

[3] D. M. H. van Gent and W. P. J. van Woerden, *A search to distinguish reduction for the isomorphism problem on direct sum lattices*, Cryptology ePrint Archive, Paper 2025/1204, 2025. [Online]. Available: https://eprint.iacr.org/2025/1204.

[4] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*, Springer, 2009, pp. 147–191. DOI: 10.1007/978-3-540-88702-7_5.

[5] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 3rd ed. New York: Springer, 1999. DOI: 10.1007/978-1-4757-6568-7.

[6] J. Martinet, *Perfect Lattices in Euclidean Spaces* (Grundlehren der mathematischen Wissenschaften). Berlin, Heidelberg: Springer, 2003, vol. 327. DOI: 10.1007/978-3-662-05167-2.

[7] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 227–232, 1982. DOI: 10.1109/TIT.1982.1056484.

[8] J. W. Bos et al., "HAWK specification document, version 1.1," National Institute of Standards and Technology, Tech. Rep., 2025, Round 2 submission to the NIST Additional Digital Signature Schemes project. Also available from the HAWK project website. [Online]. Available: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/hawk-spec-round2-web.pdf.

[9] T. Espitau, A. Wallet, and Y. Yu, *On gaussian sampling, smoothing parameter and application to signatures*, Cryptology ePrint Archive, Paper 2023/1654, 2023. [Online]. Available: https://eprint.iacr.org/2023/1654.

[10] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, ser. STOC '08, ACM, 2008. DOI: 10.1145/1374376.1374407.

[11] M. F. Bollauf, M. Lie, and C. Ling, *On gaussian sampling for q-ary lattices and linear codes with lee weight*, Cryptology ePrint Archive, Paper 2025/087, 2025. [Online]. Available: https://eprint.iacr.org/2025/087.

[12] A. Lemire-Paquin, "Ideal lattices in cyclotomic fields," M.S. thesis, McGill University, 2014. [Online]. Available: https://escholarship.mcgill.ca/concern/theses/6108vf681.

[13] G. Mureau, A. Pellet-Mary, H. Pliatsok, and A. Wallet, *Cryptanalysis of rank-2 module-lip in totally real number fields*, Cryptology ePrint Archive, Paper 2024/441, 2024. [Online]. Available: https://eprint.iacr.org/2024/441.

[14] L. Ducas and S. Gibbons, *Hull attacks on the lattice isomorphism problem*, Cryptology ePrint Archive, Paper 2023/194, 2023. [Online]. Available: https://eprint.iacr.org/2023/194.

[15] F. C. Silva, M. Lie, and C. Ling, *On hull attacks on the module lattice isomorphism problem*, Cryptology ePrint Archive, Paper 2025/1376, 2025. [Online]. Available: https://eprint.iacr.org/2025/1376.

[16] H. Bennett, A. Ganju, P. Peetathawatchai, and N. Stephens-Davidowitz, *Just how hard are rotations of $\mathbb{Z}^n$? algorithms and cryptography with the simplest lattice*, Cryptology ePrint Archive, Paper 2021/1548, 2021. [Online]. Available: https://eprint.iacr.org/2021/1548.

[17] K. Jiang, A. Wang, H. Luo, G. Liu, Y. Yu, and X. Wang, *Exploiting the symmetry of $\mathbb{Z}^n$: Randomization and the automorphism problem*, Cryptology ePrint Archive, Paper 2023/1735, 2023. [Online]. Available: https://eprint.iacr.org/2023/1735.

[18] P. Klein, "Finding the closest lattice vector when it's unusually close," in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '00, ACM-SIAM, 2000, pp. 937–941. [Online]. Available: https://dl.acm.org/doi/10.5555/338219.338661.

[19] C. Peikert, "An efficient and parallel gaussian sampler for lattices," in *Advances in Cryptology – CRYPTO 2010*, T. Rabin, Ed., ser. Lecture Notes in Computer Science, vol. 6223, Berlin, Heidelberg: Springer, 2010. DOI: 10.1007/978-3-642-14623-7_5.

[20] D. M. H. van Gent, *A note on the genus of the HAWK lattice*, Cryptology ePrint Archive, Paper 2025/215, 2025. [Online]. Available: https://eprint.iacr.org/2025/215.

[21] G. Mureau, *Special genera of hermitian lattices and applications to HAWK*, Cryptology ePrint Archive, Paper 2025/940, 2025. [Online]. Available: https://eprint.iacr.org/2025/940.

[22] D. M. H. van Gent and L. N. Pulles, *HAWK: Having automorphisms weakens key*, Cryptology ePrint Archive, Paper 2025/928, 2025. [Online]. Available: https://eprint.iacr.org/2025/928.

[23] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia: SIAM, 2002, Chapter 2 covers floating point arithmetic and the standard model of arithmetic. DOI: 10.1137/1.9780898718027.

[24] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, 1991. DOI: 10.1145/103162.103163.