University of Bergen
Department of Informatics

Learning a Hawk Parallelepiped: Cryptanalysis of the Hawk Digital Signature Scheme

Author: Eirik Djupvik Skjerve

Supervisors: Igor Aleksandrovich Semaev & Martin Feussner

UNIVERSITY OF BERGEN
Faculty of Science and Technology

April, 2025

Abstract

In this work, we do cryptanalysis on the Hawk digital signature scheme using the *Learning a Parallelepiped* method which broke the GGH and basic NTRU digital signature schemes. This work shows that this method does not pose a threat to Hawk, as it is not possible / requires too many signature samples.

Acknowledgements

Acknowledgements here

`Eirik D. Skjerve`

Contents

1 Introduction **1**

1.1 Context and motivation 1

1.2 Objectives 2

1.3 Thesis outline 3

1.4 Scope and Limitations 3

1.5 Detailed tentative roadmap 3

1.5.1 Schedule 4

2 Background **5**

2.1 Overview 5

2.2 Cryptography 5

2.3 Cryptanalysis 6

2.4 Digital Signatures 8

2.4.1 Hash-and-Sign 8

2.4.2 Security of Digital Signatures 8

2.4.3 GGH 8

2.4.4 NTRU 8

2.5 Algebra 8

2.5.1 Polynomials 8

2.5.2 Polynomial rings 8

2.5.3 Number fields 8

2.6 Linear Algebra and Lattices 8

2.7 Probability Theory 9

2.8 Gradient Search 9

2.8.1 Overview 9

2.8.2 Gradients 9

2.8.3 Gradient descent and optimization 10

3 Hawk and HPP **11**
3.1 Overview . . . 11
3.2 Hawk . . . 12
3.2.1 Overview . . . 12
3.2.2 Parameters . . . 13
3.2.3 Hawk key pairs and key pair generation . . . 13
3.2.4 Solving the NTRU-equation . . . 15
3.2.5 Discrete Gaussian Distribution . . . 15
3.2.6 Hawk signature generation . . . 16
3.2.7 Hawk signature verification . . . 17
3.2.8 Hawk security . . . 18
3.3 Implementation of Hawk . . . 18
3.3.1 Smaller degree Hawk . . . 18
3.4 HPP . . . 19
3.4.1 Setup and idealized version . . . 19
3.4.2 Covariance matrix estimation . . . 20
3.4.3 Hidden parallelepiped to hidden hypercube transformation . . . 21
3.4.4 Moments and Gradient Descent . . . 22
3.5 HPP against a non-uniform distribution . . . 25
3.6 HPP against normal distribution . . . 26

4 Cryptanalysis of Hawk **29**
4.1 Overview . . . 29
4.2 Redefining the problem and solution . . . 29
4.2.1 Covariance matrix and hypercube transformation . . . 30
4.2.2 Moments and gradient search . . . 31
4.2.3 Gradient search overview . . . 32
4.3 Practical method . . . 32
4.3.1 Measuring method . . . 33
4.4 Estimating μ_4 . . . 34
4.4.1 Estimating μ_4 via direct sampling . . . 34
4.4.2 Computing μ_4 analytically . . . 35
4.5 Alternative method . . . 38

5 Results and discussion **40**
5.1 Results . . . 40
5.2 Conclusion . . . 41

Acronyms **43**

Bibliography **44**

A Generated code from Protocol buffers **46**

List of Figures

3.1 Hidden parallelepiped problem in dimension 2. Given only signature samples, recover rows of $\mathbf{V}$, here marked as vectors in red and blue 20

3.2 Hidden hypercube problem in dimension 2 22

3.3 HPP in dimension 2 for normal distribution 27

3.4 Hidden hypercube problem in dimension 2 for normal distribution 28

4.1 Hidden parallelepiped in dimension 2 for discrete Gaussian distribution ($\sigma = 2.02$) 36

4.2 Hidden hypercube in dimension 2 for discrete Gaussian distribution ($\sigma = 2.02$) 37

List of Tables

3.1 Parameter sets for **Hawk** 13

4.1 Measure of μ_4 36

5.1 Closeness measure for Hawk attack 41

Listings

A.1 Source code of something 46

Chapter 1

Introduction

1.1 Context and motivation

Digital signatures are an integral part of secure communication today, as they enable communicating parties to mathematically verify that each party is indeed who they say they are, and that messages communicated between them have not been changed during transit. The widely used Digital Signature Algorithm (DSA) and the Rivest, Shamir, Adleman (RSA) signature scheme are in peril due to the potential emergence of quantum computers which can easily solve the hard problems DSA and RSA-sign are based upon.

Whether practical quantum computers with these powers will emerge any time soon is debatable. Nevertheless, measures against the looming threat has already begun. In 2016, the National Institute of Standards and Technology (NIST) announced a process for selecting new standard schemes for Key Encapsulation Methods (KEMs) and digital signatures that are resilient against quantum attacks (https://www.nist.gov/pqcrypto). Many of the submissions to this process, including KRYSTALS-Dilithium which is to be standardized, are based on lattice problems that are believed to be hard to solve for both classical and quantum computers.

Asymmetric cryptographic schemes based on lattice problems are not an entirely new phenomenon, however. NTRU-Sign [4], the signature counterpart of the NTRU cryptosystem, is a digital signature scheme based upon the hardness of solving the Closest Vector Problem (CVP) [3]. The original scheme was broken by Phong. Q. Nguyen & Oded Regev in 2006 [7]; not by solving the CVP, but by observing that each signature leaks some information about the secret key. The title of their paper and the name of the attack is *Learning a Parallelepiped*, and the problem to solve in this attack will henceforth be denoted as the Hidden Parallelepiped Problem (HPP). Countermeasures in light of this attack were proposed, but these countermeasures were attacked in 2012 by a more advanced extension of the original attack [2].

Formulate more exactly what zonotope method does

Hawk [5] is a digital signature scheme submitted to NIST's standardization process and is a viable candidate for standardization due to its speed and small signature- and keysizes. It is also a lattice-based signature scheme akin to NTRU-sign, but with some significant changes, and a different underlying hard problem on which its security is based upon. This thesis will investigate if the *Learning a Parallelepiped* attack can be modified and aimed at Hawk to retrieve information about the secret key, and possibly break the scheme.

1.2 Objectives

The objective of this thesis consists of two main parts:

- **Implementation of Hawk in Rust**. As the first part of this thesis I implement the Hawk digital signature scheme according to the Hawk specification paper [5] in the Rust programming language. [1] Implementing a scheme and its algorithms is a good way to more deeply learn how it works. I chose to do the implementation in Rust for the sake of becoming more adept at this particular programming language as a personal bonus objective of the thesis. Moreover, having ones own implementation of a scheme makes it easier to experiment on, run simulations with, adjust, and modify to ones need. It would in any case be challenging to understand and work with dense, long, and complicated source code someone else has written. For the Hawk teams source code in C and a reference implementation in Python see https://github.com/hawk-sign.
- **Cryptanalysis and experimentation**. The second part of this thesis is cryptanalysis of Hawk. The goal is to use the *Learning a parallelepiped* attack and do suitable modifications to attack Hawk. This requires both theoretical and practical work, and experiments will, like the Hawk implementation itself, be implemented in Rust (unless stated otherwise).

[1] Disclaimer: this implementation is not meant to be comparable with the Hawk teams implementation for real life usage, as it is not highly optimized and not all formal requirements are met.

1.3 Thesis outline

Chapter 2 will introduce important notions and mathematical background used in this thesis. Chapter 3 will introduce Hawk and its implementation, and the *Learning a Parallelepiped* attack. In Chapter 4 the cryptanalysis of Hawk is presented. The final chapter will discuss results and future work.

1.4 Scope and Limitations

Even though quantum computing and quantum algorithms like Shor's algorithm is the main reason lattice based cryptography is being developed, nothing related to quantum computing will be considered in this thesis, as all computations, algorithms, analyses and attacks are done in the domain of classical computing.

Hawk has three different parameter sets, where one is a so-called "Challenge" parameter. This will mainly be the target for our cryptanalysis.

1.5 Detailed tentative roadmap

1. Introduce idea of a digital signature
2. Introduce lattice facts and lattice problems used in digital signatures
3. Introduce other linear algebra and statistics / probability theory stuff
4. Introduce notion of gradient search and variations
5. Describe Hawk in detail
6. Describe Hawk implementation in detail
7. Describe basic HPP attack using notation from original paper
8. Proof and discussion of HPP against normally distributed samples, still using notation from original paper

9. Describe general application of HPP against Hawk using Hawk notation and conventions (e.g. column vectors instead of row vectors, matrix B instead of V, etc.)

10. Describe measuring of DGD properties, and implementation of this

11. Detailed description of attack in practice, discuss implementation challenges w.r.t. memory, runtime, etc.

12. Results and discussion of these, limitations, considerations, etc.

1.5.1 Schedule

- **Week 9, 28.02:** Concluding that experiments were unsuccessful. Further experiments code runs will be to produce measurements that will be reported in the thesis
  Still nice to have 512 GB instance in NREC
- **Week 10, 07.03**
- **Week 11, 14.03**
- **Week 12, 21.03**
- **Week 13, 28.03**
- **Week 14, 04.04**
- **Week 15, 11.04**
- **Week 16, 18.04**
- **Week 17, 25.04**
- **Week 18, 02.05**
- **Week 19, 09.05**
- **Week 20, 16.05**
- **Week 21, 23.05**

Chapter 2

Background

2.1 Overview

In this chapter, the field of cryptography and cryptanalysis will be introduced, with an emphasis on digital signatures and cryptanalysis. The chapter will also introduce some necessary facts and notions related to algebra, linear algebra and lattices, and probability theory and statistics. Lastly, we introduce the notion of Gradient Descent, which will be a central tool in this thesis.

2.2 Cryptography

Cryptography is the study of tools and techniques that enable secure and, in part, reliable communication. For centuries, this entailed creative systems of codes and techniques used predominantly by military and governments, and the creation and breaking of such systems were considered an art. From the 1970s and onwards, however, mathematics increasingly becomes the backbone of cryptography, as mathematical theory provides provable security (and insecurity) of cryptographic systems. In the recent 50 years or so, cryptography has become an enabler of secure communication, verification, and integrity of information, all of which are integral aspects in our digital world. [6]

Symmetric key cryptography enables secure communication between two or more parties and utilizes a shared secret key, denoted $\mathcal{K}_{\mathsf{priv}}$. The system defines two functions, $\mathsf{ENC}_{\mathcal{K}_{\mathsf{priv}}}(\mathcal{M})$ and $\mathsf{DEC}_{\mathcal{K}_{\mathsf{priv}}}(\mathcal{C})$ that encrypt and decrypt a message, respectively. These functions depend on the input message and the secret key. Anyone without knowledge of $\mathcal{K}_{\textsf{priv}}$ should be unable to extract any meaningful information about either $\mathcal{K}_{\textsf{priv}}$ or $\mathcal{M}$ based only on observing $\mathcal{C}$.

Asymmetric key cryptography on the other hand utilizes two related, but distinct keys; one private and one public, denoted $\mathcal{K}_{\textsf{priv}}$ and $\mathcal{K}_{\textsf{pub}}$. Anyone with access to $\mathcal{K}_{\textsf{pub}}$ can encrypt a message $\mathcal{M}$ as $\mathcal{C} = \textsf{ENC}_{\mathcal{K}_{\textsf{pub}}}(\mathcal{M})$, rendering it unreadable. Only the holder of $\mathcal{K}_{\textsf{priv}}$ is able to decrypt the message as $\mathcal{M} = \textsf{DEC}_{\mathcal{K}_{\textsf{priv}}}(\mathcal{C})$. A crucial security property of such a system is that one should not be able to somehow deduce what $\mathcal{K}_{\textsf{priv}}$ is if one has access only to $\mathcal{K}_{\textsf{pub}}$.

The security of asymmetric cryptography is usually based on hard mathematical problems for which an effective algorithm to solve is either unknown to exist, or proven *not* to exist. For example, to decrypt a message encrypted in the RSA scheme [9] without possession of the secret key, one would need to find two large prime numbers p and q such that $p \cdot q = n$, where n is the RSA modulus used in its encryption algorithm. For specific and large enough values of n, this is considered a hard problem. [3] Other asymmetric cryptography schemes may utilize different hard problems as a basis for their security, such as the ”Discrete Logarithm Problem”, ”Closest Vector Problem”, and ”Learning With Errors”, to name a few.

One of the main applications of asymmetric cryptography are digital signatures. A digital signature $\mathcal{S}$ computed from a message $\mathcal{M}$ enable a receiver of the message-signature pair $(\mathcal{M}, \mathcal{S})$ to mathematically verify that the sender is in possession of the private key $\mathcal{K}_{\textsf{priv}}$ that relates to the receivers public key $\mathcal{K}_{\textsf{pub}}$. In addition, a signature $\mathcal{S}$ guarantees that the original message $\mathcal{M}$ is intact and has not changed since the computation of $\mathcal{S}$. Section 2.4 will go in further detail about the building blocks for digital signature schemes, and what constitutes a secure or insecure scheme.

2.3 Cryptanalysis

Cryptanalysis is the study of analyzing and breaking cryptographic systems, and it plays an important role in strengthening the cryptography we use today. By analyzing a system and exposing potential flaws and weaknesses, the designers of such as system can make suitable changes to it (or in extreme cases, discard it entirely) to further increase its security against adversaries with malicious intent, or altogether avoid the release of an insecure piece of technology.

The notion of a cryptanalytic *attack* refers to a methodical attempt of retrieving secret information from an instance of a cryptographic scheme. This can be either attempting to recover parts of, or the entire, secret key, deduce some information about an encrypted message, or somehow produce some information (e.g. creating a valid digital signature) as if one possesses the secret key. Generally, if any degree of the mentioned points is possible for a scheme, the scheme should be considered insecure or broken, and the attack can be considered successful.

An important consideration when designing and analyzing the security of an asymmetric crypto-system, is that a scheme is generally only as strong as its weakest part. In practice, even if a scheme is based on a provably hard problem, there may be other components of the scheme that renders it unsafe. As an example, the security of the NTRU-sign digital signature scheme is based upon the CVP. This means that without possession of the secret key, forging a signature would require one to solve an instance of CVP, which is generally considered hard. However, as we will see in section 3, each generated signature leaks some information about the secret key, which ultimately enables signature forgeries, regardless of whether one can solve the CVP or not.

Maybe find some direct source for this?

2.4 Digital Signatures

2.4.1 Hash-and-Sign

2.4.2 Security of Digital Signatures

2.4.3 GGH

2.4.4 NTRU

2.5 Algebra

2.5.1 Polynomials

2.5.2 Polynomial rings

2.5.3 Number fields

2.6 Linear Algebra and Lattices

Denote by $\mathbf{v}$ an $n \times 1$ column vector on the form

$$\mathbf{v} = \begin{bmatrix} v_0 \\ v_1 \\ ... \\ v_{n-1} \end{bmatrix}$$

and by $\mathbf{B}$ an $n \times m$ matrix on the form

$$\mathbf{B} = \begin{bmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ b_{m-1,0} & b_{m-1,1} & \cdots & b_{m-1,n-1} \end{bmatrix}$$

Generally, entries v_i and $b_{i,j}$ are integers unless stated otherwise. Some places the thesis will use row notation instead of column notation for the vectors, so that $\mathbf{v}$ is a $1 \times n$ row vector on the form

$$\mathbf{c} = [v_0, v_1, ..., v_{n-1}]$$

In these cases this will be pointed out.

We denote by $\langle \cdot, \cdot \rangle$ the dot-product of two vectors of equal dimensions as $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^t \mathbf{y} = \sum_{i=0}^{n-1} x_i y_i$

2.7 Probability Theory

2.8 Gradient Search

2.8.1 Overview

Used in this section: [10] Gradient descent is a widely used method in optimization and learning problems.

The general method works by measuring the gradient of the target function $\nabla f(\theta)$ w.r.t. to its parameters θ to get the direction in which θ $f(\theta)$ changes the most. One then takes a "step" based on this direction (for a descent, one moves against the gradient, for an ascent, with the gradient), influenced by the hyperparameter δ, which determines the magnitude of the step.

Before going more in depth of the methods, we quickly define the gradient of a function.

2.8.2 Gradients

Let $\theta = [\theta_1, \theta_2, \cdots, \theta_n]$ The gradient of a multivariate function $f(\theta)$ is defined as the vector of partial derivatives evaluated at θ

$$\nabla f(\theta) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1}(\theta) \\ \frac{\partial f}{\partial \theta_2}(\theta) \\ \cdots \\ \frac{\partial f}{\partial \theta_n}(\theta) \end{bmatrix}$$

2.8.3 Gradient descent and optimization

In Algorithm 1 the so-called "vanilla" gradient descent is described.

Algorithm 1 Vanilla gradient descent

```
Require: differentiable target function $f$
initialize $\theta_0$ as a starting point
while $\theta_t$ is not optimal do
    $\theta_t \leftarrow \theta_{t-1} - \delta \cdot \nabla f_\theta(\theta_{t-1})$
```

Note that for vanilla gradient descent as described in Algorithm 1, to compute the gradient one needs the entire dataset for each iteration, which sometimes can be too large to fit in memory or be computationally efficient.

Due to the popularity of gradient descent in especially machine learning, many optimization techniques exist that improve upon the vanilla method. In this thesis, however, we stick to the vanilla version.

Chapter 3

Hawk and HPP

3.1 Overview

In this chapter, the Hawk digital signature scheme and the *Learning a Parallelepiped* cryptanalytic attack will be presented. This will lay the foundation for chapter 4, which will employ the *Learning a Parallelepiped* attack against Hawk

In section 3.2, an introduction on how Hawk works on a high level will be given. The algorithms for key generation, signature generation and verification and their most important subroutines will be provided with enough detail in order to analyze them properly. Section 3.3 will discuss some considerations for implementing the scheme. The *Learning a Parallelepiped* attack is described in section 3.4.

3.2 Hawk

In this section we introduce the digital signature scheme Hawk [5]. As mentioned in the introduction, Hawk is a lattice based signature scheme that shares some key points with the digital signature scheme NTRU-sign [4], the target for the original HPP attack. The first part of this chapter, section 3.2, will give an introduction on how the Hawk scheme works on a high level. After this, the implementation of Hawk will be described in section 3.3, where more detail about specific subroutines and parts of the procedures will be given.

In these sections, some parts of Hawk as described in [5] are left out due to the focus of this thesis. For example, Hawk utilizes compression of keys and signatures, and some of their security considerations regarding a practical implementation are based on side channel attacks, both of which is not a concern in this thesis. Moreover, their security analysis is based on experimental lattice reduction and security proofs of the underlying hard problems (namely the *Lattice Isomorphism Problem* and the *One More Shortest Vector Problem*), which, again, is not the focus of this thesis.

3.2.1 Overview

In Hawk, we work with polynomials defined over the cyclotomic number field $\mathcal{K}_n = \mathbb{Q}[X]/(X^n+1)$ and its corresponding ring of integers $\mathbb{Z}[X]/(X^n+1)$. Such a polynomial $f \in \mathcal{K}_n$ can also be represented by a (column) vector of its coefficients in $\mathbb{Q}^n$ (in practical setting, coefficients are in $\mathbb{Z}$ instead of $\mathbb{Q}$), denoted $\mathsf{vec}(f)$, where the index of each coefficient corresponds to its power of X in f. [1] Explicitly, if $f = a_0X^0 + a_1X^1 + \cdots + a_{n-1}X^{n-1}$

$$\mathsf{vec}(f) = \begin{bmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{bmatrix}$$

We define a mapping

$$\mathsf{rot} : \mathcal{K}_n \to \mathbb{Q}^{n \times n}, \mathsf{rot}(f) = \begin{bmatrix} \mathsf{vec}(f) & \mathsf{vec}(fX) & \mathsf{vec}(fX^2) & \cdots & \mathsf{vec}(fX^{n-1}) \end{bmatrix}$$

which enables one to construct a matrix in $\mathbb{Q}^{n \times n}$ from a single polynomial in $\mathcal{K}_n$. A

This should maybe be in the background section...

[1] This is a polynomial representation that is frequent in programming implementations when a dedicated algebra library is not used.

vector $\mathsf{vec}(fX^k)$ to any power k will henceforth be referred to as a ”negacyclic shift” of f, since all entries with powers X^{n-t} for $n-t+k < n-1$ are shifted k times, and entries with powers X^{n-s} where $n-s+k \geq n$ will ”go around” and be negated as $-X^{n-s+k \mod n}$.

The hash function used in Hawk is SHAKE256, an extendable output function (XOF) which enables arbitrary output length, unlike hash functions like SHA256 which has a static output size of 256 bits. This property is useful since SHAKE256 can be used in all three degrees of Hawk.

3.2.2 Parameters

Below is the parameters for Hawk degree 256, 512, and 1024. Note that this list only contains the relevant parameters for this thesis, and that many parameters relevant to compression and decompression are omitted.

Table 3.1: Parameter sets for **Hawk**

Name	**HAWK-256**	**HAWK-512**	**HAWK-1024**
Targeted security	Challenge	NIST-I	NIST-V
Bit security λ	64	128	256
Degree n	256	512	1024
Transcript size limit	2^{32}	2^{64}	2^{64}
Centered binomial η for sampling (f, g)	2	4	8
Signature std. dev. σ_{sign}	1.010	1.278	1.299
Verification std. dev. σ_{verify}	1.042	1.425	1.571

Insert more relevant parameters here

3.2.3 Hawk key pairs and key pair generation

A Hawk private key is represented by the matrix

$$\mathbf{B} = \begin{bmatrix} f & F \\ g & G \end{bmatrix} \in \mathcal{K}_n^{2\times 2}$$

Here, entries are chosen such that f and g have small coefficients, and the equation $fG - Fg = 1 \mod X^n + 1$ holds, and consequently $\det(\mathbf{B}) = 1$, making $\mathbf{B}$ invertible over $\mathcal{K}_n^{2\times 2}$. The inverse of $\mathbf{B}$ is

$$\mathbf{B}^{-1} = \begin{bmatrix} G & -F \\ -g & f \end{bmatrix} \in \mathcal{K}_n^{2\times 2}$$

A Hawk public key is defined as

$$\mathbf{Q} = \begin{bmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{bmatrix} = \mathbf{B}^*\mathbf{B} = \begin{bmatrix} f^*f + g^*g & f^*F + g^*G \\ F^*f + G^*g & F^*F + G^*G \end{bmatrix}$$

where f^* denotes the Hermitian adjoint of f, explicitly $f^* = f_0 - f_{n-1}X - \cdots - f_1X^{n-1}$, and $\mathbf{B}^*$ is the transpose of $\mathbf{B}$, $\mathbf{B}^T$ where each entry is taken the Hermitian adjoint of. Explicitly:

$$\mathbf{B}^* = \begin{bmatrix} f^* & g^* \\ F^* & G^* \end{bmatrix}$$

We can also define rot on a matrix as the mapping

$$\mathsf{rot} : \mathcal{K}_n^{2\times 2} \to \mathbb{Q}^{2n\times 2n}, \ \mathsf{rot}(\begin{bmatrix} f & F \\ g & G \end{bmatrix}) = \begin{bmatrix} \mathsf{rot}(f) & \mathsf{rot}(F) \\ \mathsf{rot}(g) & \mathsf{rot}(G) \end{bmatrix}$$

Note that when working with $\mathsf{rot}(\mathbf{B})$ instead of $\mathbf{B}$, the Hermitian adjoint $\mathbf{B}^*$ corresponds to the transpose of $\mathsf{rot}(\mathbf{B})$, as $\mathsf{rot}(\mathbf{B}^*) = \mathsf{rot}(\mathbf{B})^T$. This is easy to show by observing that the first row of $\mathsf{rot}(\mathbf{B}^T)$ is equal to the first column of $\mathsf{rot}(\mathbf{B}^*)$, and so on. There is also an isomorphism going on here that needs to be shown. $\mathsf{rot}(\mathbf{Q}) = \mathsf{rot}(\mathbf{B}^*\mathbf{B}) = \mathsf{rot}(\mathbf{B}^*)\mathsf{rot}(\mathbf{B}) = \mathsf{rot}(\mathbf{B})^T\mathsf{rot}(\mathbf{B})$ due to $\mathsf{rot}()$ being an isomorphism.

Maybe prove this better?

The secret key $\mathbf{B}$ serves as a good basis for the private lattice in which a (hash digest of a) message will be signed in the signature generation step. A simplified version of the key generation is described in the following algorithm:

Algorithm 2 Simplified Hawk Key Generation

```
Sample $f, g \in \mathbb{Z}[X]/(X^n + 1)$ from $\mathsf{bin}(\eta)$
Compute $F, G$ s.t. $fG - Fg = 1 \mod \mathbb{Z}[X]/(X^n + 1)$ holds
$\mathbf{B} \leftarrow \begin{bmatrix} f & F \\ g & G \end{bmatrix}$
$\mathbf{Q} \leftarrow \mathbf{B}^*\mathbf{B}$
return private key $\mathbf{B}$, public key $\mathbf{Q}$
```

Polynomials f and g are sampled from a centered binomial distribution with η depending on Hawk degree (see table 3.1)

3.2.4 Solving the NTRU-equation

An important point in the key generation process is finding proper polynomials F and G that satisfy the NTRU-equation $fG - Fg = 1 \mod \mathbb{Z}[X]/(X^n + 1)$. Importantly, coefficients of F and G need to be rather small, and the procedure of solving the equation is a time-consuming part of the key generation process. In the original NTRU system (both encryption and signature) resultants are used. Hawk uses the method proposed by Pornin and Prest in [8] which improves upon the existing resultant method in terms of both time- and memory complexity.

Need some explanation of resultants here

3.2.5 Discrete Gaussian Distribution

Before introducing the Hawk signature generation we describe the procedure to sample from the Discrete Gaussian Distribution, as described in [5].
Denote by $\mathcal{D}_{2\mathbb{Z}+c,\sigma}$ the Discrete Gaussian Distribution with parameter c and σ. Let $\rho_\sigma : \mathbb{Z} \to \mathbb{R}, \rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$. The Probability Density Function (PDF) of $\mathcal{D}_{2\mathbb{Z}+c,\sigma}$ is defined as:

$$\Pr[\mathrm{X} = \mathrm{x}] = \frac{\rho_\sigma(x)}{\sum\limits_{y \in 2\mathbb{Z}+c} \rho_\sigma(y)}$$

For $z \geq 0$ we define the function $P_c(z) = \Pr[|X| \geq z]$ when $X \sim \mathcal{D}_{2\mathbb{Z}+c,\sigma}$. Using this function for $c \in \{0, 1\}$, one computes two Cumulative Distribution Tables (CDT) T_0 and T_1 such that $T_0[k] = P_0(2+2k)$ and $T_1[k] = P_1(3+2k)$ where $T_c[k]$ denotes the k-th index in the table. In practice, to avoid using floating point numbers, the entries in the table are scaled by 2^{78} such that entries are integers with a high enough precision. For this overview and later theoretical analysis, however, we only consider the unscaled version. Using the tables, we can define a procedure sample given by the following algorithm:

Algorithm 3 sample

```
Require: parameter $c \in \{0, 1\}$
q ← uniformly random from $[-1, 1] \in \mathbb{Q}$
z ← 0
v ← 0
while $T_c[z] \neq 0$ do
    if $|q| \leq T_c[z]$ then
        $v \leftarrow v + 1$
    $z \leftarrow z + 1$
    $v \leftarrow 2v + c$
    if $q < 0$ then
        $v \leftarrow -v$
return $v$
```

This algorithm only samples one point given parameter c. We can extend this to generate vectors of length n where each entry is distributed according to $\mathcal{D}_{2\mathbb{Z}+c,\sigma}$ by the following procedure:

Algorithm 4 Sample vector of length n according to $\mathcal{D}_{2\mathbb{Z}+c,\sigma}$

```
Require: Binary vector $\mathbf{t}$ with entries uniformly distributed from $\{0, 1\}$
$\mathbf{x}$ ← empty vector of length $2n$
for $i = 0, 1, ..., 2n - 1$ do
    $\mathbf{x}[i] \leftarrow \mathsf{sample}(\mathbf{t}[i])$
return $\mathbf{x}$
```

In this manner, one can sample a lattice point $\mathbf{x}$ which is relatively close to a target lattice point $\mathbf{t}$, which will be used in the signature generation step.

3.2.6 Hawk signature generation

To generate a Hawk signature of a message $\mathbf{m}$, one computes a hash digest $\mathbf{h}$ of the message, and samples a point $\mathbf{x}$ from $\mathcal{D}_{2\mathbb{Z}^{2n}+\mathbf{t},\sigma_{sign}}$ that is close to $\mathbf{h}$ in the private lattice generated by $\mathsf{rot}(\mathbf{B})$. By transforming the point $\mathbf{x}$ back to the lattice $\mathbb{Z}^{2n}$ via $\mathbf{B}^{-1}$ one has a signature that can be verified by anyone that has access to the public key $\mathbf{Q}$. Below is the (simplified) procedure formulated in Algorithm 5:

Algorithm 5 Simplified Hawk Signature Generation

```
Require: Message $\mathbf{m}$, private key $\mathbf{B}$
$M \leftarrow H(\mathbf{m})$    ▷ H is a hash function
$\mathbf{h} \leftarrow H(M||\mathsf{salt})$    ▷ $\mathsf{salt}$ is a randomly generated value and || denotes concatenation
$\mathbf{t} \leftarrow \mathbf{Bh} \mod 2$    ▷ Reduction $\mod 2$ is done entrywise
$\mathbf{x} \leftarrow \mathsf{sample}(\mathbf{t})$
if $||\mathbf{x}||^2 \geq 4 \cdot \sigma^2_{\text{sign}} \cdot 2n$ then
    restart
else
    $\mathbf{w} \leftarrow \mathbf{B}^{-1}\mathbf{x}$
    $\mathbf{s} \leftarrow \frac{1}{2}(\mathbf{h} - \mathbf{w})$
    return signature $\mathbf{s}$, salt $\mathsf{salt}$
```

3.2.7 Hawk signature verification

To verify a Hawk signature, one simply recomputes the vector $\mathbf{h}$ from $\mathbf{m}$ and salt and in turn the vector $\mathbf{w}$, and check if the $\mathbf{Q}$-norm of $\mathbf{w}$, $||\mathbf{w}||^2_{\mathbf{Q}}$ is not too high. A signature with $\mathbf{Q}$-norm of appropriately short length will be considered valid. A summary of the simplified procedure is given in Algorithm 6:

Algorithm 6 Simplified Hawk Signature Verification

```
Require: Signature $\mathbf{s}$, message $\mathbf{m}$, salt $\mathsf{salt}$, public key $\mathbf{Q}$
$M \leftarrow H(\mathbf{m})$
$\mathbf{h} \leftarrow H(M||\mathsf{salt})$
$\mathbf{w} \leftarrow \mathbf{h} - 2\mathbf{s}$
if $||\mathbf{w}||^2_{\mathbf{Q}} \geq 4 \cdot \sigma^2_{\text{verify}} \cdot 2n$ then
    return false
else return true
```

We now briefly show why verification works:
As defined,

$$||\mathbf{w}||_{\mathbf{Q}} = \sqrt{\mathbf{w}^*\mathbf{Q}\mathbf{w}} \implies ||\mathbf{w}||^2_{\mathbf{Q}} = |\mathbf{w}^*\mathbf{Q}\mathbf{w}|$$

$$\mathbf{w}^*\mathbf{Q}\mathbf{w} = \mathbf{w}^*\mathbf{B}^*\mathbf{B}\mathbf{w} = (\mathbf{B}\mathbf{w})^*(\mathbf{B}\mathbf{w}) = ||\mathbf{B}\mathbf{w}||^2 = ||\mathbf{x}||^2$$

$$\text{since } \mathbf{w} = \mathbf{B}^{-1}\mathbf{x} \implies ||\mathbf{B}\mathbf{w}||^2 = ||\mathbf{B}\mathbf{B}^{-1}\mathbf{x}||^2 = ||\mathbf{x}||^2$$

This shows that by knowing $\mathbf{Q}$, one can measure the length of vectors in the secret lattice generated by $\mathbf{B}$. To find vectors close to a target vector $\mathbf{t}$ in the private lattice, however, requires knowledge about $\mathbf{B}$.

3.2.8 Hawk security

3.3 Implementation of Hawk

3.3.1 Smaller degree Hawk

For cryptanalysis and experiments, it can often be useful to have a ”toy” version of the crypto-system at hand that has small parameters. Attacks against real life schemes may take days, even with good hardware and carefully implemented code. With smaller parameters, however, one can more easily check if a method for cryptanalysis is a viable candidate for further experimentation. Conversely, if a specific method appears useless facing a small-parametered scheme, it will most certainly also be useless against the real life version of the scheme.

Find an anecdote here

With this in mind, we need a small parameter version of Hawk. Key generation is simple for any degree which is a power of 2, since the NTRU-solve method works on any power of 2. Signature generation is also trivial to achieve. We don’t consider signature verification for smaller parameters, since we are only interested in observng $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$

3.4 HPP

In this section we present the *Learning a Parallelepiped* attack as described in [7]. Note that in this section bold lowercase $\mathbf{v}$ denotes a *row*-vector to follow the same notation as in the original paper.

3.4.1 Setup and idealized version

Let $\mathbf{V} \in \mathcal{GL}(\mathbb{R})$ be a secret $n \times n$ unimodular matrix and $\mathcal{P}(\mathbf{V})$ be a fundamental parallelepiped, defined as the set $\{\mathbf{xV} : \mathbf{x} \in [-1, 1]^n\}$. Let $\mathbf{v} = \mathbf{xV}$. The problem to solve is informally described below and visualized in figure 3.1.

Problem 1 *Given enough vectors* $\mathbf{v} = \mathbf{xV}$ *(i.e. by observing a large enough subset of* $\mathcal{P}(\mathbf{V})$*), where both* $\mathbf{x}$ *and* $\mathbf{V}$ *is unknown, recover rows of* $\pm\mathbf{V}$

The following three steps are used to solve Problem 1:

1. Estimate the covariance matrix $\mathbf{V}^t\mathbf{V}$
2. Transform samples $\mathbf{v} \in \mathcal{P}(\mathbf{V})$ to $\mathbf{c} \in \mathcal{P}(\mathbf{C})$ where $\mathcal{P}(\mathbf{C})$ is a hypercube, i.e. $\mathbf{CC}^t = \mathbf{I}_n$
3. Do gradient descent to minimize the fourth moment of one-dimensional projections and reveal a row of $\pm\mathbf{C}$ which finally can be transformed into a row of $\pm\mathbf{V}$

In the following, each of these steps will be covered in detail.

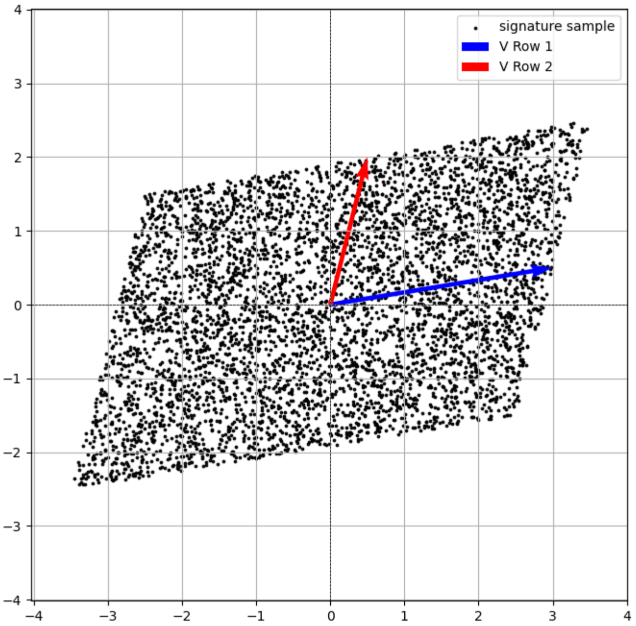

Figure 3.1: Hidden parallelepiped problem in dimension 2. Given only signature samples, recover rows of $\mathbf{V}$, here marked as vectors in red and blue

3.4.2 Covariance matrix estimation

Given enough samples on the form $\mathbf{v} = \mathbf{x}\mathbf{V}$, we want to estimate the covariance matrix $\mathbf{G} \approx \mathbf{V}^t\mathbf{V}$. This is achieved as $\mathbf{v}^t\mathbf{v} = (\mathbf{x}\mathbf{V})^t(\mathbf{x}\mathbf{V}) = \mathbf{V}^t\mathbf{x}^t\mathbf{x}\mathbf{V}$. Now, by taking the expectation of the inner term we get $\mathbb{E}[\mathbf{x}^t\mathbf{x}] = \mathbf{I}_n/3$ where $\mathbf{I}_n$ is the $n \times n$ identity matrix because of the following: Since all $x_i \in \mathbf{x}$ are distributed according to the uniform distribution over $[-1, 1]$ and each x_i are independent, we have that $\mathbb{E}[x_i] = 0$ and $\mathbb{E}[x_i x_j] = \mathbb{E}[x_i]\mathbb{E}[x_j] = 0$ when $i \neq j$. For the case when $i = j$, we have

$$\mathbb{E}[x_i x_j] = \mathbb{E}[x^2] = \int_a^b x^2 \frac{1}{b-a} dx = \int_{-1}^{1} x^2 \frac{1}{2} dx = \frac{1}{3}$$

Therefore, $\mathbb{E}[\mathbf{x}^t\mathbf{x}]$ is an $n \times n$ matrix with $\frac{1}{3}$ down the diagonal and is 0 otherwise, i.e. $\mathbf{I}_n/3$. Thus, as number of samples grow, $\mathbf{v}^t\mathbf{v} \to \mathbf{V}^t(\mathbf{I}_n/3)\mathbf{V}$, and therefore $\mathbf{v}^t\mathbf{v} \cdot 3 \to \mathbf{V}^t\mathbf{V}$. In conclusion: by taking the average of $\mathbf{v}^t\mathbf{v}$ for all collected samples, and multiplying the resulting $n \times n$ matrix with 3, one has a good approximation of the covariance matrix $\mathbf{V}^t\mathbf{V}$.

3.4.3 Hidden parallelepiped to hidden hypercube transformation

Given a good approximation $\mathbf{G} \approx \mathbf{V}^t\mathbf{V}$, the next step is to calculate a linear transformation $\mathbf{L}$ such that the following is true:

1. $\mathbf{C} = \mathbf{VL}$ is orthonormal, i.e. the rows are pairwise orthogonal and the norm of each row is 1. In other words, $\mathbf{CC}^t = \mathbf{I}$. Consequently, $\mathcal{P}(\mathbf{C})$ becomes a hypercube.
2. If $\mathbf{v}$ is uniformly distributed over $\mathcal{P}(\mathbf{V})$ then $\mathbf{c} = \mathbf{vL}$ is uniformly distributed over $\mathcal{P}(\mathbf{C})$.

This is achieved by taking the Cholesky decomposition of $\mathbf{G}^{-1} = \mathbf{LL}^t$ where $\mathbf{L}$ is a lower-triangular matrix. To compute Cholesky decomposition of $\mathbf{G}^{-1}$, we must first show that $\mathbf{G}$ is symmetric positive definite.

1. $\mathbf{G}$ is symmetric $\iff$ $\mathbf{G}^t = \mathbf{G}$ which is clear as $\mathbf{G}^t = (\mathbf{V}^t\mathbf{V})^t = \mathbf{V}^t(\mathbf{V}^t)^t = \mathbf{V}^t\mathbf{V} = \mathbf{G}$.
2. $\mathbf{G}$ is positive definite if for any non-zero column vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^t\mathbf{G}\mathbf{x} > 0$. We have that $\mathbf{x}^t\mathbf{G}\mathbf{x} = \mathbf{x}^t\mathbf{V}^t\mathbf{V}\mathbf{x} = (\mathbf{V}\mathbf{x})^t(\mathbf{V}\mathbf{x})$. Denote by $\mathbf{y} = \mathbf{V}\mathbf{x}$. Since $\mathbf{x} \neq \mathbf{0}$ and $\mathbf{V}$ is invertible (and therefore non-zero) it is clear that $\mathbf{y} \neq \mathbf{0}$ and $\mathbf{y}^t\mathbf{y} = ||\mathbf{y}||^2 > 0$.

Since $\mathbf{G}$ is symmetric positive definite we can do decomposition to get $\mathbf{L}$. Then:

1. if $\mathbf{C} = \mathbf{VL}$, then $\mathbf{CC}^t = \mathbf{VLL}^t\mathbf{V}^t = \mathbf{VG}^{-1}\mathbf{V}^t = \mathbf{V}(\mathbf{V}^t\mathbf{V})^{-1}\mathbf{V}^t = \mathbf{VV}^{-1}\mathbf{V}^{-t}\mathbf{V}^t = \mathbf{I}$
2. since entries in $\mathbf{v} = \mathbf{xV}$ is uniformly distributed over $\mathcal{P}(\mathbf{V})$, $\mathbf{c} = \mathbf{Cx}$ is uniformly distributed over $\mathcal{P}(\mathbf{C})$

By multiplying our samples $\mathbf{v}$ by $\mathbf{L}$ on the right, we transform them from the hidden parallelepiped to the hidden hypercube. Therefore, the problem to solve now can be called the "Hidden Hypercube Problem". If one finds the rows of $\pm\mathbf{C}$, one can simply multiply the result on the right by $\mathbf{L}^{-1}$ to obtain the solution for $\mathbf{V}$.

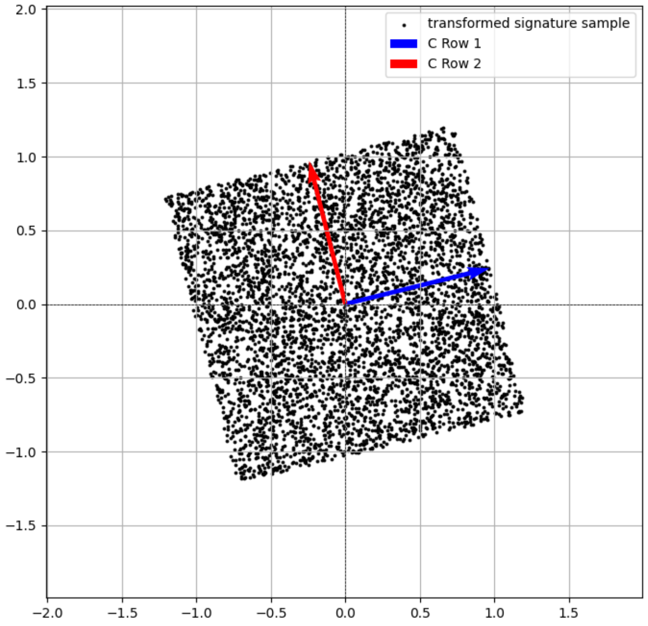

Figure 3.2: Hidden hypercube problem in dimension 2

3.4.4 Moments and Gradient Descent

The last major step in the attack is to measure and minimize the fourth moment of one-dimensional projections to disclose rows of $\pm\mathbf{C}$. Let $mom_{k,\mathbf{C}}(\mathbf{w})$ be defined as the k-th moment of $\mathcal{P}(\mathbf{C})$ projected onto $\mathbf{w}$, i.e. $\mathbb{E}[\langle \mathbf{c}, \mathbf{w}\rangle^k]$ where $\mathbf{c} = \mathbf{xC}$ for $\mathbf{x}$ uniformly distributed and $\mathbf{w} \in \mathbb{R}^n$. Looking at the term $\langle \mathbf{c}, \mathbf{w}\rangle$, we have $\langle \mathbf{xC}, \mathbf{w}\rangle = \langle \sum_{i=1}^n x_i c_i, \mathbf{w}\rangle$ where c_i is the i-th row of $\mathbf{C}$. Since x_i is a scalar, we can move it out of the dot-product brackets as $\sum_{i=1}^n x_i\langle c_i, \mathbf{w}\rangle$. Evaluating this inside the expectation operator $\mathbb{E}[\ ]$ we have $\mathbb{E}[\sum_{i=1}^n x_i\langle c_i, \mathbf{w}\rangle^k]$. We consider the two cases when $k = 2$ and $k = 4$.

- $\mathbf{k = 2}$: $\mathbb{E}[(\sum_{i=1}^n x_i\langle c_i, \mathbf{w}\rangle)^2]$ expands to $\mathbb{E}[\sum_i^n \sum_j^n x_i x_j \langle c_i, \mathbf{w}\rangle\langle c_j, \mathbf{w}\rangle]$. As seen before in section 3.4.2, $\mathbb{E}[x_i x_j] = \frac{1}{3}$ when $i = j$ and 0 otherwise. Thus, we have the expression $mom_{2,C}(\mathbf{w}) = \frac{1}{3}\sum_i^n \langle c_i, \mathbf{w}\rangle^2$ which can also be written as $= \frac{1}{3}\mathbf{wC}^t\mathbf{Cw}^t$

Should show why this is

- $\mathbf{k = 4}$:

$$\mathbb{E}[(\sum_{i=1}^n x_i\langle c_i, \mathbf{w}\rangle)^4] = \mathbb{E}[\sum_i^n \sum_j^n \sum_k^n \sum_l^n x_i x_j x_k x_l \langle c_i, \mathbf{w}\rangle\langle c_j, \mathbf{w}\rangle\langle c_k, \mathbf{w}\rangle\langle c_l, \mathbf{w}\rangle]$$

There are three cases for the indices i, j, k and l:

1. **All equal:** If $i = j = k = l$, we simply have $\sum_i^n \mathbb{E}[x^4]\langle c_i, \mathbf{w}\rangle^4 = \frac{1}{5}\sum_i \langle c_i, \mathbf{w}\rangle^4$ due to the fact that $\mathbb{E}[x^4] = \int_{-1}^{1} x^4 \frac{1}{2} dx = \frac{1}{5}$
2. **None equal:** If $i \neq j \neq k \neq l$ the expression is zero due to $\mathbb{E}[x_i] = 0$ and all x_i, x_j, x_k, x_l are independent.
3. **Pairwise equal:** If either

   - $i = j \neq k = l$
   - $i = k \neq j = l$
   - $i = l \neq j = k$

   then we have $\sum_{i \neq j} \mathbb{E}[x_i^2 x_j^2]\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2 = \frac{1}{9}\sum_{i \neq j}\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2$. By putting the above together we get

   $$\frac{1}{5}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 + 3(\frac{1}{9}\sum_{i \neq j}\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2)$$

   and the final expression becomes

   $$mom_{4,\mathbf{C}}(\mathbf{w}) = \frac{1}{5}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 + \frac{1}{3}\sum_{i \neq j}\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2$$

Now, since $\mathbf{C}$ is orthonormal, and by restricting $\mathbf{w}$ to the unit sphere in $\mathbb{R}^n$, we can simplify the expressions further. The second moment becomes $mom_{2,C}(\mathbf{w}) = \frac{1}{3}\mathbf{w}\mathbf{C}^t\mathbf{C}\mathbf{w}^t = \frac{1}{3}\mathbf{w}\mathbf{I}\mathbf{w}^t = \frac{1}{3}\mathbf{w}\mathbf{w}^t = \frac{1}{3}\|\mathbf{w}\|^2 = \frac{1}{3}$.

By rewriting and expanding the fourth moment:

$$mom_{4,\mathbf{C}}(\mathbf{w}) = \frac{1}{5}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 + \frac{1}{3}\sum_{i \neq j}\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2$$

$$mom_{4,\mathbf{C}}(\mathbf{w}) = \frac{1}{5}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 + \frac{1}{3}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2 - \frac{1}{3}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^2 \langle c_i, \mathbf{w}\rangle^2$$

$$mom_{4,\mathbf{C}}(\mathbf{w}) = \frac{1}{5}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 + \frac{1}{3}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2 - \frac{1}{3}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4$$

$$mom_{4,\mathbf{C}}(\mathbf{w}) = \frac{1}{3}\|\mathbf{w}\|^4 - \frac{2}{15}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 = \frac{1}{3} - \frac{2}{15}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4$$

The key observation is that to minimize $mom_{4,\mathbf{C}(\mathbf{w})}$ one has to maximize the term $-\frac{2}{15}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4$. Since c_i and $\mathbf{w}$ are both on the unit circle, and all c_i are orthogonal to each other, the term is maximized whenever $\mathbf{w} = c_i$ since $\langle c_i, \pm c_i \rangle = \langle \mathbf{w}, \pm\mathbf{w} \rangle = 1$. Using this observation, we employ a gradient descent to minimize $mom_{4,\mathbf{C}}(\mathbf{w})$ w.r.t the input vector $\mathbf{w}$, and the global minima is in $\pm\mathbf{C}$. The expression when $\mathbf{w} = \pm c_i$ is

$$mom_{4,\mathbf{C}}(\pm c_i) = \frac{1}{3} - \frac{2}{15}\sum_{i=1}^{n}\langle c_i, \pm c_i\rangle^4 = \frac{1}{3} - \frac{2}{15} = \frac{1}{5}$$

To do a gradient descent we need to compute the gradient of the fourth moment, $\nabla mom_{4,\mathbf{C}}(\mathbf{w})$.

- The gradient of the first term, $\frac{1}{3}\|\mathbf{w}\|^4$, is computed as follows:
  We rewrite $\|\mathbf{w}\|^4$ as $(\mathbf{w}\mathbf{w}^T)^2$. Using the chain rule we have that
  $\nabla((\mathbf{w}\mathbf{w}^T)^2) = 2(\mathbf{w}\mathbf{w}^T) \cdot \frac{\partial}{\partial \mathbf{w}_j}(\mathbf{w}\mathbf{w}^T) = 2(\mathbf{w}\mathbf{w}^T) \cdot 2\mathbf{w}$.
  The gradient of the first term is then $\frac{1}{3} \cdot 2(\mathbf{w}\mathbf{w}^T) \cdot 2\mathbf{w} = \frac{4}{3}\|\mathbf{w}\|^2\mathbf{w}$.
- For the second term, $\frac{2}{15}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4$ we use the fact that the gradient is a linear operator:

  $$\nabla\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^4 = \sum_{i=1}^{n}\nabla(\langle c_i, \mathbf{w}\rangle^4)$$

  Looking at just the inner term, $\nabla(\langle c_i, \mathbf{w}\rangle^4) = 4\langle c_i, \mathbf{w}\rangle^3 \cdot \frac{\partial}{\partial w_j}(\langle c_i, \mathbf{w}\rangle) = 4\langle c_i, \mathbf{w}\rangle^3 \cdot c_i$
  The gradient of the second term is therefore $\dfrac{8}{15}\displaystyle\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^3 c_i$

Putting together the terms we get

$$\nabla mom_{4,\mathbf{C}}(\mathbf{w}) = \frac{4}{3}\|\mathbf{w}\|^2\mathbf{w} - \frac{8}{15}\sum_{i=1}^{n}\langle c_i, \mathbf{w}\rangle^3 c_i$$

Should prove that there are no other local minima somehow

In a practical attack, one simply approximates $mom_{4,\mathbf{C}}(\mathbf{w})$ as $\mathbb{E}[\langle \mathbf{c}, \mathbf{w}\rangle^4]$ by averaging over all available samples $\mathbf{c}$. Similarly, to approximate the gradient of the 4th moment, $\nabla mom_{4,\mathbf{C}}(\mathbf{w})$, again, using the linearity of the gradient operator, we can compute $\mathbb{E}[\nabla\langle \mathbf{c}, \mathbf{w}\rangle^4] = \mathbb{E}[4\langle \mathbf{c}, \mathbf{w}\rangle^3\mathbf{c}] = 4\mathbb{E}[\langle \mathbf{c}, \mathbf{w}\rangle^3\mathbf{c}]$ by averaging the expression over all available samples $\mathbf{c}$.

In Algorithm 7, a simple gradient descent for minimization of $mom_{4,\mathbf{C}}(\mathbf{w})$ is described. One such descent will return one a candidate row c_i of $\pm\mathbf{C}$ that is a minimum in the gradient landscape. By multiplying this row with $\mathbf{L}^{-1}$ on the right and rounding the

Algorithm 7 Gradient descent

```
Require: Parameter $\delta$, samples $\mathbf{c}$
1: Choose random vector $\mathbf{w}$ on the unit sphere
2: Compute an approximation $\mathbf{g} \leftarrow \nabla mom_{4,\mathbf{C}}(\mathbf{w})$
3: Set $\mathbf{w}_{new} \leftarrow \mathbf{w} - \delta\mathbf{g}$
4: Normalize $\mathbf{w}_{new}$ as $\frac{\mathbf{w}_{new}}{\|\mathbf{w}_{new}\|}$
5: Approximate $mom_{4,\mathbf{C}}(\mathbf{w})$ and $mom_{4,\mathbf{C}}(\mathbf{w}_{new})$
6: if $mom_{4,\mathbf{C}}(\mathbf{w}) \geq mom_{4,\mathbf{C}}(\mathbf{w}_{new})$ then
7:     Return $\mathbf{w}$
8: else
9:     Set $\mathbf{w} \leftarrow \mathbf{w}_{new}$ and go to step 2
```

entries one has a candidate row v_i in $\pm\mathbf{V}$. This procedure can be repeated until all rows of $\pm\mathbf{C}$ are found.

Depending on the specific signature scheme one attacks, knowing when all correct rows have been found can be a tricky task. For example, if all rows of $\mathbf{C}$ are independent of each other, one has to find the correct combination of the rows that constitutes $\mathbf{C}$. This requires at most $2 \cdot n!$ attempts, which quickly becomes an impossibly large number as n grows. In the case of NTRU-sign, however, the rows are not independent. In fact, finding any *one* single row of $\mathbf{C}$ reveals all the remaining rows, up to a polynomial number of different ordered permutations. Thus, one successful descent effectively breaks the scheme.

Discuss the attack in practice, i.e. the attack against discrete uniform distribution etc.

3.5 HPP against a non-uniform distribution

The success of the HPP attack is in part based on the assumption of an underlying uniform distribution of the vector $\mathbf{x}$. We now investigate how the method would work against a different underlying distribution.

Assume now that observed signatures are on the form $\mathbf{c} = \mathbf{xV}$ where $x_i \sim \mathcal{D}(0, \sigma)$, i.e. some distribution with expectation 0 and some standard deviance σ. For this theoretical work we assume the distribution $\mathcal{D}(0, \sigma)$ is continuous as in the idealized version of HPP in section 3.4.

3.6 HPP against normal distribution

Assume now that the signatures on the form $\mathbf{v} = \mathbf{x}\mathbf{V}$ where $x_i \sim \mathcal{N}(0, \sigma^2)$, the continuous normal distribution with $\mu = 0$ and std.dev. σ. After converting $\mathcal{P}(\mathbf{V})$ to $\mathcal{P}(\mathbf{C})$, the fourth moment of $\mathcal{P}(\mathbf{C})$ is constant over $\mathbf{w}$ on the unit circle.

To show this, we simply do the same calculations as in the previous section, with the difference that x follows a normal distribution. Considering

$$mom_{4,\mathbf{C}}(\mathbf{w}) = \mathbb{E}[(\sum_{i=1}^{n} x_i \langle c_i, \mathbf{w}\rangle)^4] = \mathbb{E}[\sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n}\sum_{l}^{n} x_i x_j x_k x_l \langle c_i, \mathbf{w}\rangle\langle c_j, \mathbf{w}\rangle\langle c_k, \mathbf{w}\rangle\langle c_l, \mathbf{w}\rangle]$$

for the three different cases:

- **All equal:** If $i = j = k = l$, we simply have $\sum_i^n \mathbb{E}[x_i^4]\langle c_i, \mathbf{w}\rangle^4 = 3\sigma^4 \sum_i \langle c_i, \mathbf{w}\rangle^4$ due to the well known fact that $\mathbb{E}[x^4] = 3\sigma^4$ for x distributed according to $\mathcal{N}(0, \sigma^2)$.
- **None equal:** Since $\mathbb{E}[x] = 0$ and x_i, x_j, x_k, x_l are independent, this results in zero.
- **Pairwise equal:** If either
  - $i = j \neq k = l$
  - $i = k \neq j = l$
  - $i = l \neq j = k$

  we have $\sum_{i \neq j} \mathbb{E}[x_i^2 x_j^2]\langle c_i, \mathbf{w}\rangle^2\langle c_j, \mathbf{w}\rangle^2 = \sigma^4 \sum_{i \neq j}\langle c_i, \mathbf{w}\rangle^2\langle c_j, \mathbf{w}\rangle^2$ since $\mathbb{E}[x^2] = \sigma^2$.

Putting together expressions like in section 3.4.4 we get

$$mom_{4,\mathbf{C}}(\mathbf{w}) = 3\sigma^4 \sum_i \langle c_i, \mathbf{w}\rangle^4 + 3(\sigma^4 \sum_{i \neq j} \langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2)$$

$$mom_{4,\mathbf{C}}(\mathbf{w}) = 3\sigma^4 \sum_i \langle c_i, \mathbf{w}\rangle^4 + 3\sigma^4 \sum_i \sum_j \langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2 - 3\sigma^4 \sum_i \langle c_i, \mathbf{w}\rangle^4$$

$$mom_{4,\mathbf{C}}(\mathbf{w}) = 3\sigma^4 \sum_i \sum_j \langle c_i, \mathbf{w}\rangle^2 \langle c_j, \mathbf{w}\rangle^2$$

$$mom_{4,\mathbf{C}}(\mathbf{w}) = 3\sigma^4 (\|\mathbf{w}\|^2)^2 = 3\sigma^4$$

Since $mom_{4,\mathbf{C}}(\mathbf{w})$ is constant for $\mathbf{w}$ on the unit sphere (depending only on σ), the term no longer depends on the secret matrix $\mathbf{C}$ and the attack will not work.

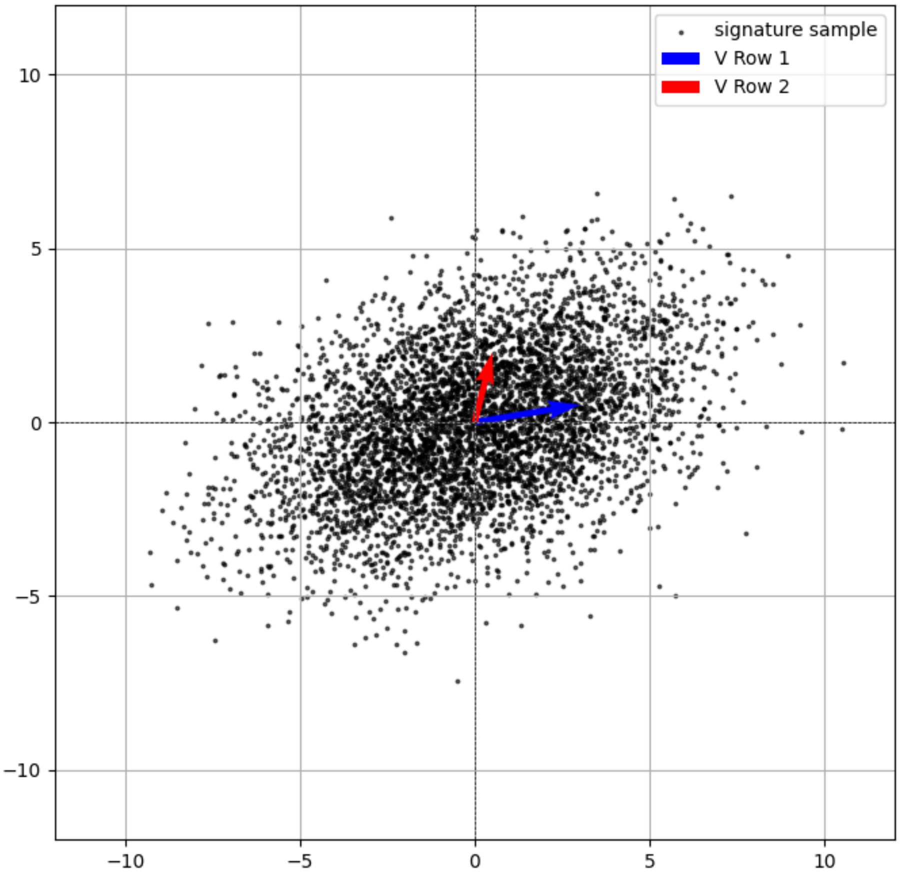

Figure 3.3: HPP in dimension 2 for normal distribution

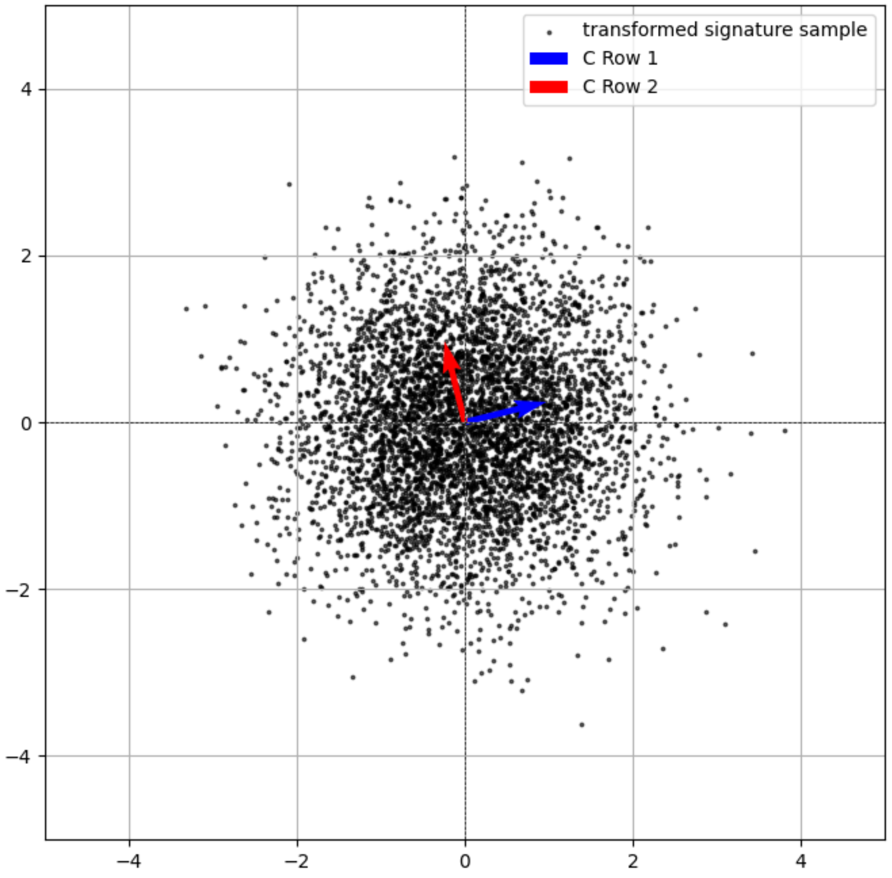

Figure 3.4: Hidden hypercube problem in dimension 2 for normal distribution

Chapter 4

Cryptanalysis of Hawk

4.1 Overview

In this chapter we introduce our cryptanalysis of Hawk by adapting the original HPP attack to the Hawk setting. As previously shown, a continuous normal distribution is immune against such an attack due to the constant 4-th moment. In Hawk, however, the distribution of entries in $\mathbf{x}$ is discrete, not continuous. We will show in section 4.2 that the Discrete Gaussian Distribution based on tables and the algorithm **??** is indeed not identical to a continuous normal distribution, and that an attack akin to the original HPP may still be applicable.

In section 4.3, we explain the procedure to transform the hidden parallelepiped to a hidden hypercube, and show why this is a trivial task for Hawk signatures. Section 4.4 will cover the main part of the attack, namely the gradient descent, which will in many ways be similar to the original attack. Lastly, section 4.5 will present the complete practical method and give details about its implementation.

4.2 Redefining the problem and solution

We now restate the Hidden Parallelepiped Problem in the Hawk setting. Recall in HPP that one observes row vectors $\mathbf{v} = \mathbf{x}\mathbf{V}$. In the Hawk setting, the observed signature is $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$ since we have moved to column-notation for vectors. The problem to solve is formulated in Problem 1

Problem 1 *Given signature samples on the form* $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$ *where only* $\mathbf{w}$ *is known,* $\mathbf{B}$ *is a Hawk private key and* $\mathbf{x}$ *is distributed according to the discrete Gaussian Distribution* $\mathcal{D}_{2\mathbb{Z}^{2n}+\mathbf{t},\sigma}$*, recover columns of* $\mathbf{B}$*.*

The general steps in Hawk setting will be similar to that of the original HPP:

- Compute Covariance Matrix $(\mathbf{B}^{-1})^T(\mathbf{B}^{-1})$
- Transform the samples $\mathbf{w} \in \mathcal{P}(\mathbf{B^{-1}})$ to $\mathbf{c} \in \mathcal{P}(\mathbf{C})$ where $\mathbf{C}$ is orthonormal
- Deduce columns of $\mathbf{C}$ doing gradient descent of minimizing the 4th moment of one-dimensional projections of $\mathcal{P}(\mathbf{C})$

Note that in this section we denote by $\mathbf{u}$ a vector on the unit sphere $\mathbb{R}^{2n}$ instead of $\mathbf{w}$ as in the previous section, to avoid confusion with the Hawk notation for $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$. Also note that henceforth we will consider only $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$ as a signature instead of $\mathbf{s} = \frac{1}{2}(\mathbf{h} - \mathbf{w})$ since $\mathbf{w}$ is easily recoverable given $\mathbf{s}$ and message $\mathbf{m}$. Lastly, we consider $\mathbf{B}$ as $\mathsf{rot}(\mathbf{B})$ and thus work with matrices in $\mathbb{Q}^{2n\times 2n}$ and column vectors in $\mathbb{Q}^{2n}$.

4.2.1 Covariance matrix and hypercube transformation

In the original HPP attack one has to estimate the matrix $\mathbf{G} \approx \mathbf{V}^t\mathbf{V}$. For Hawk, the signatures are on the form $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$ which means we would need to compute $\mathbf{G} = \mathbf{B}^{-1}\mathbf{B}^{-T}$ (recall the HPP paper uses row vectors while Hawk use columns vectors). However, the public key $\mathbf{Q} = \mathbf{B}^T\mathbf{B}$, enables us to skip this step. In the original attack one takes Cholesky decomposition of the inverse of the covariance matrix such that $\mathbf{G}^{-1} = \mathbf{L}\mathbf{L}^T$. For $\mathbf{G} = \mathbf{B}^{-1}\mathbf{B}^{-T}$, the inverse, $\mathbf{G}^{-1} = \mathbf{B}^T\mathbf{B} = \mathbf{Q}$. Therefore, we can simply take the Cholesky decomposition of $\mathbf{Q}$ to get $\mathbf{L}$ such that $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$. By multiplying our samples $\mathbf{w}$ by $\mathbf{L}^T$ on the left, we have transformed our samples to the hidden hypercube as in the original attack.
By taking $\mathbf{C} = \mathbf{L}^T\mathbf{B}^{-1}$, we have that

$$\mathbf{C}^T\mathbf{C} = (\mathbf{L}^T\mathbf{B}^{-1})^T(\mathbf{L}^T\mathbf{B}^{-1}) = \mathbf{B}^{-T}\mathbf{L}\mathbf{L}^T\mathbf{B}^{-1} = \mathbf{B}^{-T}\mathbf{Q}\mathbf{B}^{-1} = \mathbf{B}^{-T}\mathbf{B}^T\mathbf{B}\mathbf{B}^{-1} = \mathbf{I}_n$$

and

$$\mathbf{C}\mathbf{C}^T = (\mathbf{L}^T\mathbf{B}^{-1})(\mathbf{L}^T\mathbf{B}^{-1})^T = \mathbf{L}^T\mathbf{B}^{-1}\mathbf{B}^{-T}\mathbf{L} = \mathbf{L}^T\mathbf{Q}^{-1}\mathbf{L} = \mathbf{L}^T(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{L} = \mathbf{L}^T\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{L} = \mathbf{I}_n$$

Thus $\mathbf{C}$ is an orthonormal matrix. Since $\mathbf{w}$ is distributed according to $\widehat{\mathcal{D}}$ over $\mathcal{P}(\mathbf{B^{-1}})$, by taking $\mathbf{c} = \mathbf{L}^T\mathbf{w}$ we have $\mathbf{c} = \mathbf{L}^T\mathbf{B}^{-1}\mathbf{x} = \mathbf{Cx}$, $\mathbf{c}$ is distributed according to $\widehat{\mathcal{D}}$ over $\mathcal{P}(\mathbf{C})$. Lastly, we also want to normalize the distribution of entries in $\mathbf{x}$ to have variance 1. Therefore we consider instead $\mathbf{c} = \mathbf{Cz} = \dfrac{\mathbf{Cx}}{\sigma}$ where σ is the std.dev. of $\mathcal{D}_{2\mathbb{Z}+\mathbf{t},\sigma}$. We summarize this procedure in the following algorithm:

```
Algorithm 8 Hawk Hypercube Transformation
Require: Samples $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$ and public key $\mathbf{Q}$
  Compute $\mathbf{L}$ s.t. $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$          ▷ by Cholesky decomposition
  Compute $\mathbf{c} = \mathbf{L}^T \frac{\mathbf{w}}{\sigma}$          ▷ entrywise division by $\sigma$
  return $\mathbf{c}$ and $\mathbf{L}^{-T}$
```

4.2.2 Moments and gradient search

Assume now observed signatures on the form $\mathbf{c} = \mathbf{Cz}$ where $\mathbf{C}$ is orthonormal, and the normalized distribution entries in $\mathbf{z}$ follows have variance 1. By rewriting the terms from section 3.3.4 for this distribution, we have that

$$mom_{4,\mathbf{C}}(\mathbf{u}) = 3\|\mathbf{u}\|^4 + (\mu_4 - 3)\sum_{i=1}^{n}\langle c_i, \mathbf{u}\rangle^4$$

and

$$\nabla mom_{4,\mathbf{C}}(\mathbf{u}) = 12\|\mathbf{u}\|^2\mathbf{u} + 4(\mu_4 - 3)\sum_{i=1}^{n}\langle c_i, \mathbf{u}\rangle^3 c_i$$

where μ_4 is the 4th moment of $z_i \in \mathbf{z}$, i.e. $\mathbb{E}[z^4]$, and $\mathbf{u}$ is a vector on the unit sphere of $\mathbb{R}^{2n}$, i.e. $\|\mathbf{u}\| = 1$. This means that if the difference $(\mu_4 - 3)$ determines the applicability of the attack. As we showed in section 3.6, the attack does not work for a normal distribution, i.e. a distribution where $\mu_4 = 3$. However, if $\mu_4 \neq 3$, and the difference $(\mu_3 - 3)$ is significant enough, one might be able to employ the same minimization technique as in the original attack to reveal a column of ±$\mathbf{C}$.

Independent of what the underlying distribution is, $\langle c_i, \mathbf{u}\rangle^4 = 1$ if $\mathbf{u} = \pm c_i$ since $\langle c_i, c_i\rangle^4 = 1$, so the crux of the attack still lies in optimizing the $mom_{4,\mathbf{C}}(\mathbf{u})$ function w.r.t. the vector $\mathbf{w}$ to reveal columns of ±$\mathbf{C}$. Note that if $(\mu_4 - 3) < 0$ we have the same case as in the original attack, where minimization of the entire term entails maximization of $\sum_{i=1}^{n}\langle c_i, \mathbf{u}\rangle^4$, which gives us a row of ±$\mathbf{C}$. If $(\mu_4 - 3) > 0$, we need to maximize the entire term $3\|\mathbf{u}\|^4 + \sum_{i=1}^{n} \langle c_i, \mathbf{u} \rangle^4$, which is achieved by doing a gradient *ascent* instead of a gradient *descent*.

4.2.3 Gradient search overview

Now, given samples $\mathbf{c} \in \mathcal{P}(\mathbf{C})$, we run a gradient search to minimize or maximize the fourth moment of one-dimensional projections onto $\mathbf{u}$, as described above.

We evaluate the functions $mom_{4,\mathbf{C}}(\mathbf{u})$ as $\mathbb{E}[\langle \mathbf{c}, \mathbf{u} \rangle^4]$ and $\nabla mom_{4,\mathbf{C}}(\mathbf{u})$ as $\mathbb{E}[\nabla \langle \mathbf{c}, \mathbf{u} \rangle^4] = 4\mathbb{E}[\langle \mathbf{c}, \mathbf{u} \rangle^3 \mathbf{c}]$. These can be evaluated by precomputed samples $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t\}$, or by continuously generating one and one signature sample since we have access to the signature generation algorithm.

This might be redundant, as the method will be described in chapter on HPP

Algorithm 9 Gradient descent on $\mathcal{P}(\mathbf{C})$

```
Require: Samples on the form $\mathbf{c} = \mathbf{Cx}$, descent parameter $\delta$
 1: $\mathbf{u} \leftarrow$ random vector on unit sphere in $\mathbb{R}^{2n}$
 2: loop
 3:     $\mathbf{g} \leftarrow \nabla mom_{4,\mathbf{C}}(\mathbf{w})$
 4:     $\mathbf{u}_{new} = \mathbf{u} - \delta \mathbf{g}$
 5:     normalize $\mathbf{u}_{new}$ as $\frac{\mathbf{u}_{new}}{\|\mathbf{u}_{new}\|}$
 6:     if $mom_{4,\mathbf{C}}(\mathbf{u}_{new}) \geq mom_{4,\mathbf{C}(\mathbf{u})}$ then
 7:         return $\mathbf{u}$
 8:     else
 9:         $\mathbf{u} \leftarrow \mathbf{u}_{new}$
10:         go to step 3
```

For a gradient *ascent* one can flip the sign and inequality sign on lines 4 and 6, respectively.

4.3 Practical method

Below is a basic description of the approach.

Algorithm 10 Proposed basic version of attack

```
1: Collect signatures $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$
2: Using public key $\mathbf{Q}$, find $\mathbf{L}$ s.t. $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$
3: Transform samples s.t. $\mathbf{c} = \mathbf{L}^T\mathbf{w}$
4: Find columns of $\pm\mathbf{C}$ by doing gradient search over $\mathcal{P}(\mathbf{C})$
5: Multiply columns of $\pm\mathbf{C}$ by $\mathbf{L}^{-T}$ on the left and round the result to get columns in
   $\pm\mathbf{B}^{-1}$
```

After having transformed the samples $\mathbf{w} \in \mathcal{P}(\mathbf{B^{-1}})$ to $\mathbf{c} \in \mathcal{P}(\mathbf{C})$ we want to recover columns of $\pm\mathbf{C}$ and transform them back to columns of $\pm\mathbf{B}^{-1}$ by multiplying by $\mathbf{L}^{-T}$ on the left. Due to the special structure of Hawk private keys, finding one column of $\mathbf{B}$ automatically gives n columns. Unfortunately, since revealing a single column of $\mathbf{B}^{-1}$ reveals a "shift" of either the two polynomials G and g *or* F and f, this is not enough to disclose the entire matrix. If samples were on the form $\mathbf{w} = \mathbf{B}\mathbf{x}$, a single column would reveal f and g (or F and G), and one could simply reconstruct F and G (or f and g) by solving the NTRU-equation as in the key generation step of Hawk. Nevertheless, if one finds two columns of $\mathbf{B}^{-1}$, it is easy to check if they are shifts of each other. If they are not, one has found shifts of all four polynomials in the secret key, and by trying all combinations of shifts, of which there are $4n^2$ (accounting for negative and positive sign), one can easily verify if a candidate $\mathbf{B}'$ is valid by checking if $\mathbf{B}'$ is unimodular and if $\mathbf{B}'^T\mathbf{B}' = \mathbf{Q}$. If so, one is able to forge signatures, and the attack is done.

After each gradient ascent and/or descent returns a possible solution $\mathbf{z}'$, we multiply it to the left as $\mathbf{b}' = \mathbf{L}^{-T}\mathbf{z}'$ where $\mathbf{b}'$ is a possible column of $\pm\mathbf{B}^{-1}$ as $\mathbf{z}'$ is a possible column of $\pm\mathbf{C} = \mathbf{L}^T \cdot (\pm\mathbf{B}^{-1})$. Since in experiments we have access to the correct secret key, we simply check directly if $\mathbf{b}' \in \pm\mathbf{B}^{-1}$. In a real word attack however, one would, as described above, have to compute candidate $\mathbf{B}'$ and check if $\mathbf{B}'^T\mathbf{B}' = \mathbf{Q}$.

4.3.1 Measuring method

Having access to the correct $\mathbf{B}^{-1}$ we can do measures on how close a proposed solution $\mathbf{b}'$ is to one of the columns of $\mathbf{B}^{-1}$. We can for example take the difference in length of $|\mathbf{b}'|$ and each vector in $|\mathbf{B}^{-1}|$, i.e. $\mathsf{diff}_{\mathsf{min}} = \mathsf{min}\{||\mathbf{b}'| - |\mathbf{b}_i|| : \mathbf{b}_i \in \pm\mathbf{B}^{-1}\}$. A low value for this (this depends on the Hawk parameter n) would indicate a good guess, whereas a higher value indicates greater difference between the vectors, and thus a bad guess. Alternatively, one can count entrywise how many entries out of $2n$ that matches between $\mathbf{b}'$ and $\mathbf{b}_i$, or use some other suitable measurement.

Below is a more detailed version of the entire attack/experiment.

Algorithm 11 Proposed version of attack with measuring

```
Require: Hawk parameter $n$, number of samples $t$, Hawk key pair $\mathbf{B}$, $\mathbf{Q}$
1: Collect $t$ signatures $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$        ▷ Optional - can also generate signatures continuously
2: Using public key $\mathbf{Q}$, compute $\mathbf{L}$ s.t. $\mathbf{Q} = \mathbf{L}\mathbf{L}^t$
3: Transform samples s.t. $\mathbf{c} = \mathbf{L}^t\mathbf{w}$
4: loop
5:     Candidate $\mathbf{z}' \leftarrow \mathsf{gradient\ search}$        ▷ Do both ascent and descent
6:     Candidate $\mathbf{b}' \leftarrow \lfloor \mathbf{L}^{-T}\mathbf{z}' \rceil$        ▷ Entrywise rounding to nearest integer
7:     Check if $\mathbf{b}' \in \mathbf{B}^{-1}$
8:     Measure $\mathsf{diff}_{\mathsf{min}}(\mathbf{b}', \mathbf{B}^{-1})$
```

The loop from line 4 can be run several times to get a new random starting point $\mathbf{u}$ for the gradient search. Line 6 rounds the candidate *after* multiplying with $\mathbf{L}^{-T}$ to avoid rounding errors. Line 8 can also give which column of $\mathbf{B}^{-1}$ $\mathbf{b}'$ is closest to, so one can compare the vectors side by side.

4.4 Estimating μ_4

4.4.1 Estimating μ_4 via direct sampling

Consider the Discrete Gaussian Distribution as described in [5] and section 3.1.5. We can use our implementation of Hawk to sample many points from the distribution. Let $\widehat{\mathcal{D}}$ denote the practical discrete Gaussian distribution from sampled points. Let $\hat{\mu}$, $\hat{\sigma}^2$ be the expectation and variance of $\widehat{\mathcal{D}}$. Assume we sample t points from $\widehat{\mathcal{D}}$ as $X = \{x_1, x_2, ..., x_t\}$. We estimate $\hat{\mu}$ and $\hat{\sigma}^2$ simply as $\hat{\mu} = \dfrac{1}{t}\displaystyle\sum_{i=1}^{t} x_i$ and $\hat{\sigma}^2 = \dfrac{1}{t}\displaystyle\sum_{i=1}^{t}(x_i - \hat{\mu})^2$. For simplicity, we can also assume $\hat{\mu} = 0$ as claimed in [5]. To simplify later computations we also normalize our samples by computing $Z = \{z_1, z_2, ..., z_t\} = \{\frac{x_1}{\hat{\sigma}}, \frac{x_2}{\hat{\sigma}}, ..., \frac{x_t}{\hat{\sigma}}\}$ such that $\mathbb{V}[z_i] = 1$. Now $\hat{\mu}_4 = \mathbb{E}[z_i^4] = \dfrac{1}{t}\displaystyle\sum_{i=1}^{n} z_i^4$.

The problem with this approach is the requirement of many samples in order to determine $\hat{\mu}_4$ with high enough accuracy. By using Confidence Intervals and Central

Limit Theorem, we get a number of how many samples one should sample in order for $\hat{\mu}_4$ to be accurate.

$$n = (\frac{z_{\alpha/2} \cdot \hat{\sigma}}{E})^2$$

where E is the acceptable error and $z_{\alpha/2}$ is found in standard normal tables. Assume one wants to determine with confidence 95% that an estimate $\hat{\mu}_4$ differs from the real μ_4 by maximum 0.00001, or 1×10^{-5}. Then $n = (\frac{1.96 \cdot \hat{\sigma}}{10^{-5}})^2$

Show this value maybe??

4.4.2 Computing μ_4 analytically

The probability that one can discern $mom_{4,\mathbf{C}}(\mathbf{u})$ when $\mathbf{u} \in \pm\mathbf{C}$ and when $\mathbf{u} \notin \pm\mathbf{C}$ ultimately depends on the value of μ_4 and the number of signature samples one has available.

For implementation of Hawk, they provide tables which along with an algorithm determines the distribution $\widehat{\mathcal{D}}_{2\mathbb{Z}+c,\sigma}$. Based on the tables one can reconstruct a Probability Mass Function (PMF) $Pr(X = x)$ for $X \sim \widehat{\mathcal{D}}_{\mathbb{Z},\sigma}$. The PMF of $\widehat{\mathcal{D}}_{\mathbb{Z},\sigma}$ from tables is:

$$Pr(X = x) = \frac{1}{2} \cdot \begin{cases} 1 - T_c[0] & \text{, if } |x| = c \\ \frac{1}{2}(T_c[\frac{|x| - c}{2} - 1] - T_c[\frac{|x| - c}{2}]) & \text{, if } |x| > c \end{cases}$$

Here, $c = |x| \mod 2$ that determines which table (T_0 or T_1) is used.

Explain this stuff a bit better maybe

Given the PMF we can compute the moments of X as $\mathbb{E}[X^k] = \sum_x x^k \cdot Pr(X = x)$, as there is a finite number of entries in the tables. More specifically, for the 256 parameter set, there are 10 entries in T_0 and T_1. Therefore we sum from $x = -20$ to $x = 20$. When $|x| = 20$, $Pr(|X| \geq 20) = \frac{1}{4} \cdot (T_0[9] - T_0[10])$, where $T_0[9]$ is the last non-zero entry, and so all successive entries will yield 0. Similarly, for parameters 512 and 1024 we sum from $x = -26$ to $x = 26$, since there are 13 entries in T_0 and T_1.

By using the Decimal module in Python [1], a module designed for high precision when working with decimal numbers, we can be confident that the computed results are correct, and that we do not get e.g. floating point errors when summing, since we are dealing with very small numbers.

We compute μ_4 as $\mu_4 = \mathbb{E}[Z^4] = \mathbb{E}[(\frac{X}{\sigma})^4] = \frac{1}{\sigma^4}\mathbb{E}[X^4]$. In table 4.1 we show the value of μ_4 and $\mu_4 - 3$ rounded at 25 and 39 decimal points, respectively, for all three Hawk degrees.

Table 4.1: Measure of μ_4

Degree	$\mathbb{E}[Z^4]$	$\mathbb{E}[Z^4] - 3$
256	2.9999999999999790015752619	-2.0998424738064930232320331 $\cdot 10^{-14}$
512	2.9999999999999999999862684	-1.3731593731486976093589903 $\cdot 10^{-20}$
1024	2.999999999999999999987558	-1.2441396421557256288414318 $\cdot 10^{-20}$

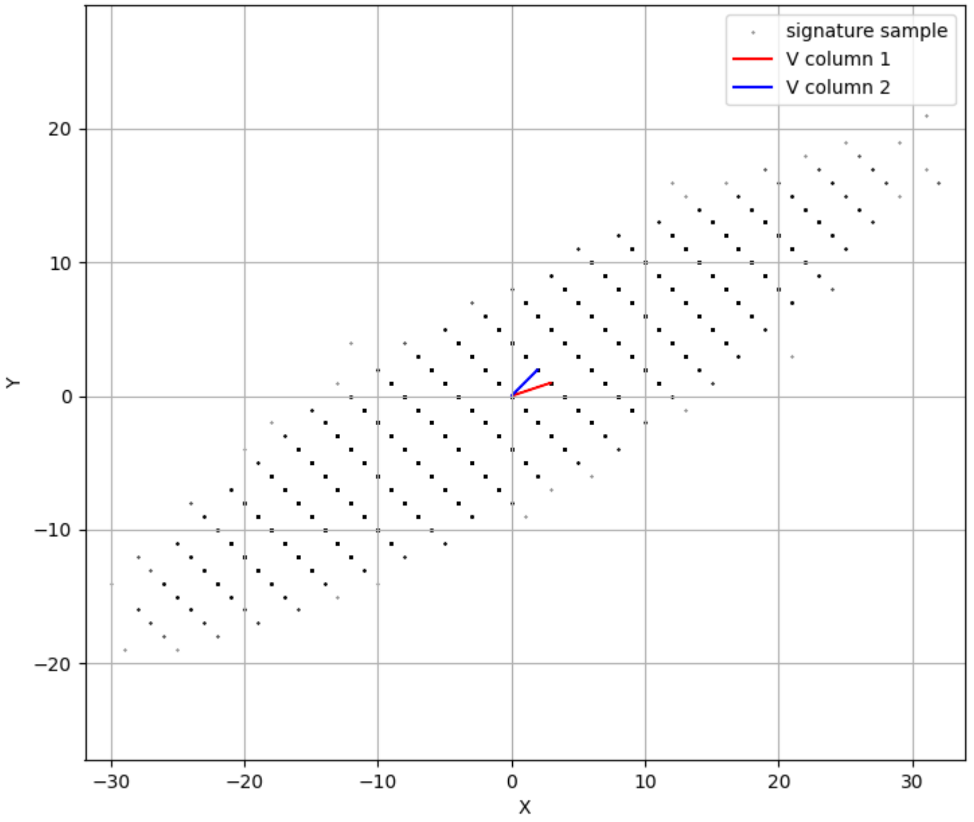

Figure 4.1: Hidden parallelepiped in dimension 2 for discrete Gaussian distribution ($\sigma = 2.02$)

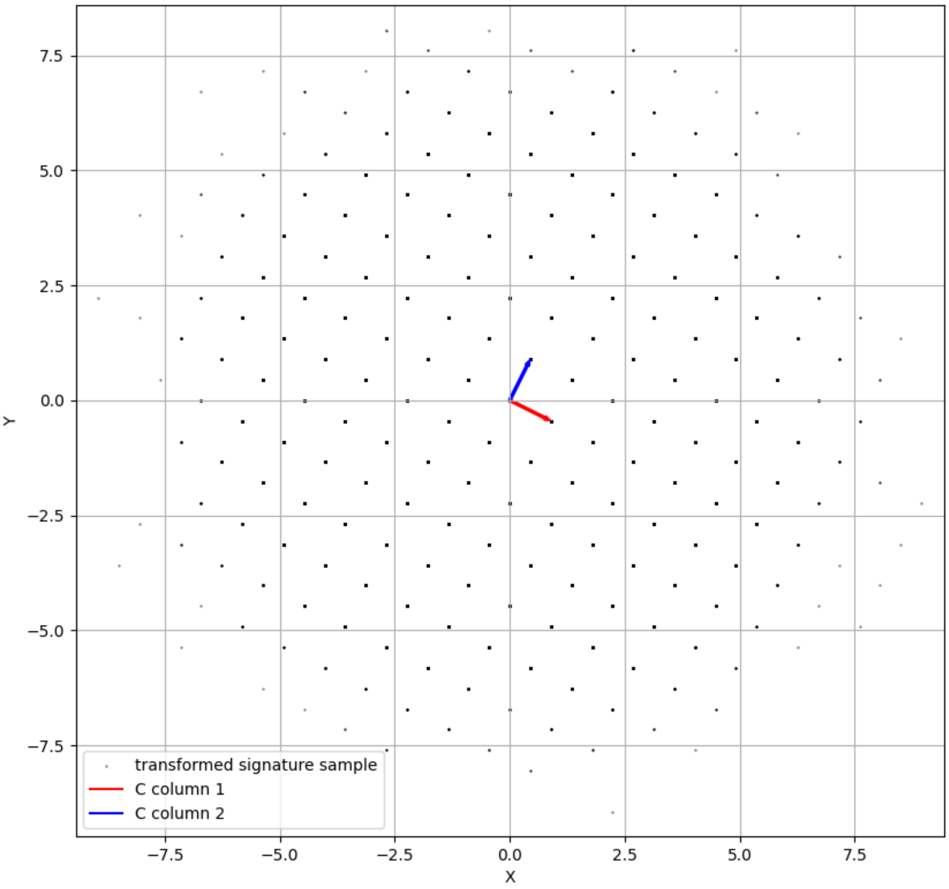

Figure 4.2: Hidden hypercube in dimension 2 for discrete Gaussian distribution ($\sigma = 2.02$)

4.5 Alternative method

We have established that the direct method of observing $\mathbf{w} = \mathbf{B}^{-1}\mathbf{x}$ will not lead to a successful attack. We now try an alternative method.

Consider in Hawk signature generation as described in section 3. The secret vector $\mathbf{x}$ is sampled from $\mathcal{D}_{2\mathbb{Z}^{2n}+\mathbf{t},\sigma}$. Since $\mathbf{t}$ is defined as $\mathbf{Bh} \mod 2$ (entrywise reduction $\mod 2$), we can rewrite $\mathbf{t}$ as $\mathbf{t} = \mathbf{Bh} + 2 \cdot \mathbf{d}$, for an integer vector $\mathbf{d}$ such that adding $2 \cdot \mathbf{d}$ is equivalent to reducing $\mod 2$.

Now, we also consider $\mathbf{x}$, which can be written as $\mathbf{x} = \mathbf{t} + 2 \cdot \mathbf{y}$ for some integer vector $\mathbf{y}$. Then we have the following:
$\mathbf{w} = \mathbf{B}^{-1}\mathbf{x} = \mathbf{B}^{-1}(\mathbf{t} + 2 \cdot \mathbf{y}) =$
$\mathbf{B}^{-1}(\mathbf{Bh} + 2 \cdot \mathbf{d} + 2 \cdot \mathbf{y}) =$
$\mathbf{h} + 2 \cdot \mathbf{B}^{-1}(\mathbf{d} + \mathbf{y})$
$\implies \dfrac{\mathbf{w} - \mathbf{h}}{2} = \mathbf{B}^{-1}(\mathbf{d} + \mathbf{y})$

Since both $\mathbf{w}$ and $\mathbf{h}$ is available to an attacker, we analyze the distribution of $\mathbf{x}_1 = (\mathbf{d}+\mathbf{y})$. Note that the distribution of $\mathbf{y}$ in this case will be related to the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^{2n},\sigma}$, which we have already analyzed.

The next step is then to analyze the distribution that $\mathbf{d}$ follows. Note that the vector $\mathbf{h}$, which is originally a hash digest of length $n/4$ of the message $\mathbf{m}$ and a salt salt, is converted into a binary polynomial of length $2n$. We assume $\mathbf{h}$ is uniformly distributed.

Firstly, we need to estimate (most probably experimentally by samples) the distribution that $\mathbf{d}$ follows. We do this experimentally by running the signature generation algorithm, modified in this case to return also $\mathbf{d} = \dfrac{\mathbf{t} - \mathbf{Bh}}{2}$. Note that $\mathbf{d}$ is not available information for an attacker in real life, and so we only use it now to do experiments on the distribution $\mathbf{d}$. From this we have m observed sample points. From such observed points d_j we compute $\mu = \dfrac{1}{m}\sum_{j=0}^{m} d_j$ and similarly for its variance $\sigma^2 = \dfrac{1}{m}\sum_{j=0}^{m}(d_j - \mu)^2$

For different secret keys, the values for μ and σ^2 vary quite a lot. Therefore it might be challenging to estimate. The hope now is that if we can estimate the distribution of $\mathbf{y}$, and then observe public signature samples on the form $\mathbf{w}_1 = \mathbf{B}^{-1}\mathbf{x}_1$, we can easily deduce the distribution of $\mathbf{d}$.

Now on to the distribution of $\mathbf{y}$. Trivially, based on the tables and definitions, $Pr(y = k) = Pr(x = 2k) + Pr(x = 2k + 1)$ since $\mathbf{y} = \dfrac{\mathbf{x} - \mathbf{t}}{2}$. From this we have that

- $\mathbb{E}[y] = -0.25$
- $\mathbb{E}[y^2] = 1.1451$
- $\mathbb{E}[y^4] = 3.918137$.

Now, assume now we observe many signatures on the form $\frac{\mathbf{w}-\mathbf{h}}{2} = \mathbf{B}^{-1}(\mathbf{d}+\mathbf{y})$. After computing $\mathbf{L}^T$ and computing $\mathbf{L}^T\frac{\mathbf{w}-\mathbf{h}}{2} = \mathbf{L}^T\mathbf{B}^{-1}(\mathbf{d}+\mathbf{y}) = \mathbf{C}(\mathbf{d}+\mathbf{y})$. Now we can measure the expectation $\mathbb{E}[\mathbf{d}_i+\mathbf{y}_i]$ and variance $\mathbb{E}[(\mathbf{d}_i+\mathbf{y}_i)^2]$ by averaging over all available points since $\mathbf{C}$ preserves the distribution.

Chapter 5

Results and discussion

5.1 Results

The method has unfortunately not proven to work, as no correct key has been found in any of the runs. It seems that regardless of number of signatures (above a certain point, e.g. one million), the method cannot give candidate solutions with better comparison than random guessing. Random guessing in this case is assuming one knows what type of distribution the columns of a secret key is. One knows the distribution that f, g follows, but F and G depends on f and g.

For reference, I ran a test using the closeness measure $\mathsf{diff}_{\mathsf{min}} = \mathsf{min}\{||\mathbf{b}'| - |\mathbf{b}_i|| : \mathbf{b}_i \in \pm\mathbf{B}^{-1}\}$ by fixing one private key ($\mathbf{B}^{-1}$), and generating random keys $\mathbf{B}^{'-1}$ (which will serve as random guesses), to check if the attack on average gives better results than random guessing. Table 1 shows the result of comparing a key with 100 random keys, and the result of 100 random starting points for the gradient search (both ascent and descent).

One thing to note is that Hawk does not specify parameters (such as width σ of $\widehat{\mathcal{D}}$) for lower values of n than 256. Therefore, when sampling signatures for $n = 32, 64$ and 128, I use the same tables, i.e. same σ for $\widehat{\mathcal{D}}$, as in Hawk 256.

Table 5.1: Closeness measure for Hawk attack

Type	$\mathsf{diff}_{\mathsf{min}}$	$\mathsf{diff}_{\mathsf{max}}$	**Avg** $\mathsf{diff}_{\mathsf{min}}$	**Avg** $\mathsf{diff}_{\mathsf{max}}$
Key comparison degree 32	6.25	15.81	7.74	11.22
Attack on Hawk 32 (1m samples)	7.14	16.24	8.50	12.87
Key comparison degree 64	10.77	26.51	13.96	22.18
Attack on Hawk 64 (1m samples)	13.49	26.00	16.25	22.59
Key comparison degree 128	25.33	47.26	30.00	41.60
Attack on Hawk 128(1m samples)	24.27	46.91	29.61	46.91
Key comparison degree 256	56.33	82.34	61.02	75.02
Attack on Hawk 256 (1m samples)	54.50	84.07	59.20	71.09
Attack on Hawk 256 (10m samples)	57.72	77.54	62.37	77.54

Lastly, I include values from sampling points from $\widehat{\mathcal{D}}$ for Hawk 256. 100 million vectors $\mathbf{x}$ were sampled, resulting in $2 \cdot 256 \cdot 100$ million independently sampled points from $\widehat{\mathcal{D}}$. First, I compute σ^2 and σ, then compute μ_4 of normalized samples on the form $z = \frac{x}{\sigma}$. We then get that

- $\sigma^2 = 4.080429335$
- $\sigma = 2.020007261$
- Normalized $\mu_4 = \mathbb{E}(z^4) = 3.000007624$
- $(\mu_4 - 3) = 0.000007624$

5.2 Conclusion

We have computed μ_4, i.e. the 4th moment of Z to be 2.999..., and that the difference $(\mu_4 - 3)$ is about $2.1 \cdot 10^{-14}$ for parameter 256. Let us now compute how many samples one would need to successfully distinguish $mom_{4,\mathbf{C}}(\mathbf{u})$ when $\mathbf{u} \in \pm\mathbf{C}$ and when $\mathbf{u} \notin \pm\mathbf{C}$. $mom_{4,\mathbf{C}}(\mathbf{u}) = \mathbb{E}[(\langle\mathbf{u}, \mathbf{Cx}\rangle^4] = 3 + (\mu_4 - 3)\sum_{i=1}^{n}\langle\mathbf{u}, \mathbf{c}_i\rangle^4$. If $\mathbf{u} = \mathbf{c}_i$, the entire expression is $3 + (\mu_4 - 3) \cdot 1 = \mu_4$ because $\langle\mathbf{c}_i, \mathbf{c}_i\rangle = 1$ and $\langle\mathbf{c}_i, \mathbf{c}_j\rangle = 0$ for $i \neq j$. Otherwise, the sum $(\sum_{i=1}^{n}\langle\mathbf{u}, \mathbf{c}_i\rangle^4) < 1$.

Say one wants to distinguish $mom_{4,\mathbf{C}}(\mathbf{c}_i)$ and $mom_{4,\mathbf{C}}(\mathbf{u})$ for $\mathbf{u} \notin \pm\mathbf{C}$. Let $y = \langle\mathbf{u}, \mathbf{Cx}\rangle$ and assume y follows a standard normal distribution with $\mu = 0$ and $\sigma^2 = 1$. Then we approximate the variance σ^2 and standard deviation σ for $\langle \mathbf{u}, \mathbf{Cx} \rangle^4$ as $\mathbb{E}[y^8] - \mathbb{E}[y^4]^2$. Then $\sigma^2 \approx 105 - 9 = 96$ and $\sigma = \sqrt{96} \approx 9.79796$.

Now, to estimate number of samples we use formula for confidence intervals and Central Limit Theorem. Let s denote required number of samples. Then

$$s \geq (\frac{Z_{\alpha/2} \cdot \sigma}{\mathsf{err}})^2$$

Now, we want err to be smaller than $2.1 \cdot 10^{-14}$ for parameter 256. If we set err to be 10^{-14}, and we want to determine a difference with, say, 75 % confidence, we get:

$$s \geq (\frac{0.68 \cdot 9.79796}{10^{-14}})^2 \approx 4.439 \cdot 10^{29}$$

where 0.68 is from standard normal tables for 75 %. This result is way beyond the limit of acceptable transcript size, which for parameter 256 is $2^{32} \approx 4.3 \cdot 10^9$ and for non-challenge parameters 512 and 1024 is $2^{64} \approx 2 \cdot 10^{19}$. One can therefore conclude that the attack will not be successful.

Acronyms

CVP Closest Vector Problem.

DSA Digital Signature Algorithm.

HPP Hidden Parallelepiped Problem.

KEMs Key Encapsulation Methods.

NIST National Institute of Standards and Technology.

RSA Rivest, Shamir, Adleman.

Bibliography

[1] decimal — decimal fixed-point and floating-point arithmetic, 2025.
URL: `https://docs.python.org/3/library/decimal.html`.

[2] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: cryptanalysis of ntrusign countermeasures. In *Proceedings of the 18th International Conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'12, page 433–450, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642349607. doi: 10.1007/978-3-642-34961-4_27.
URL: `https://doi.org/10.1007/978-3-642-34961-4_27`.

[3] Sushil Jajodia, Pierangela Samarati, and Moti Yung. *Encyclopedia of Cryptography, Security and Privacy*. Springer, Cham, 3 edition, 2024. ISBN 9783030715205.

[4] Jill Pipher Joseph H. Silverman-William Whyte Jeffrey Hoffstein, Nick Howgrave-Graham. Ntrusign: Digital signatures using the ntru lattice. Technical report, NTRU Cryptosystems, 2003.

[5] Léo Ducas Serge Fehr Yu-Hsuan Huang Thomas Pornin Eamonn W. Postlethwaite Thomas Prest Ludo N. Pulles Joppe W. Bos, Olivier Bronchain and Wessel van Woerden. Hawk. Technical report, NXP Semiconductors, Centrum Wiskunde & Informatica, Mathematical Institute at Leiden University, NCC Group, PQShield, Institut de Mathématiques de Bordeaux, September 2024.
URL: `https://hawk-sign.info/`.

[6] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, Boca Raton, FL, 3 edition, 2020.

[7] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures, 2009.

[8] Thomas Pornin and Thomas Prest. More efficient algorithms for the NTRU key generation using the field norm. Cryptology ePrint Archive, Paper 2019/015, 2019.
URL: `https://eprint.iacr.org/2019/015`.

[9] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, January 1983. ISSN 0001-0782. doi: 10.1145/357980.358017.
URL: `https://doi.org/10.1145/357980.358017`.

[10] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.
URL: `https://arxiv.org/abs/1609.04747`.

Appendix A

Generated code from Protocol buffers

Listing A.1: Source code of something

```java
System.out.println("Hello Mars");
```