University of Bergen
Department of Informatics

Learning a Parallelepiped attack against Hawk digital signature scheme

Author: Eirik Djupvik Skjerve
Supervisors: Igor Aleksandrovich Semaev & Martin Feussner

UNIVERSITY OF BERGEN
Faculty of Science and Technology

February, 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.

Acknowledgements

Acknowledgements here Eirik D. Skjerve

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Context and motivation

Digital signatures are an integral part of secure communication today. They enable a receiver of a digital message to mathematically verify the sender is who they say they are. 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 break the hard problems DSA and RSA-sign are based upon. Whether practical quantum computers with these powers will emerge any time soon is debatable. However, 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.

Cryptographic schemes based on lattice problems are not an entirely new phenomenon, however. NTRU-Sign [2], the signature counterpart of the NTRU crypto-system, is a digital signature scheme based on the hardness of the Closest Vector Problem (CVP), a well known lattice problem (source?). The original scheme was broken by Phong. Q. Nguyen & Oded Regev in 2006 [4]; not by solving the CVP, but by retrieving a secret key by observing enough signatures. In other words, 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) as one tries to *learn* a parallelepiped. Countermeasures for this attack was proposed, but ultimately broken again in 2012 due to a more advanced extension of the original attack [1].

Hawk [3] 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. This thesis will investigate if a method based on HPP can be aimed at Hawk to retrieve information about the secret key.

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 [3] in the Rust programming language. Implementing a scheme and its algorithms is a good way to more deeply learn how it works. I chose to implement it in Rust for the sake of learning the language as a personal bonus objective of the thesis. Moreover, having ones own version of an algorithm makes it easier to experiment, run simulations, adjust, and modify it 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 and reference implementation see https://github.com/hawk-sign.
  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.
- **Cryptanalysis and experimentation**. The second part of this thesis is cryptanalysis of Hawk. The goal is to use the *Learning a parallelepiped* attack and adjusting it to attack Hawk. This requires both theoretical and practical work, and experiments will, like the Hawk implementation itself, be implemented in Rust.

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.

1.3.1 Listings

You can do listings, like in Listing 1.1

Listing 1.1: Look at this cool listing. Find the rest in Appendix A.1

```
$ java -jar myAwesomeCode.jar
```

You can also do language highlighting for instance with Golang: And in line 6 of Listing 1.2 you can see that we can ref to lines in listings.

Listing 1.2: Hello world in Golang

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("hello world")
7 }
```

1.3.2 Figures

Example of a centred figure

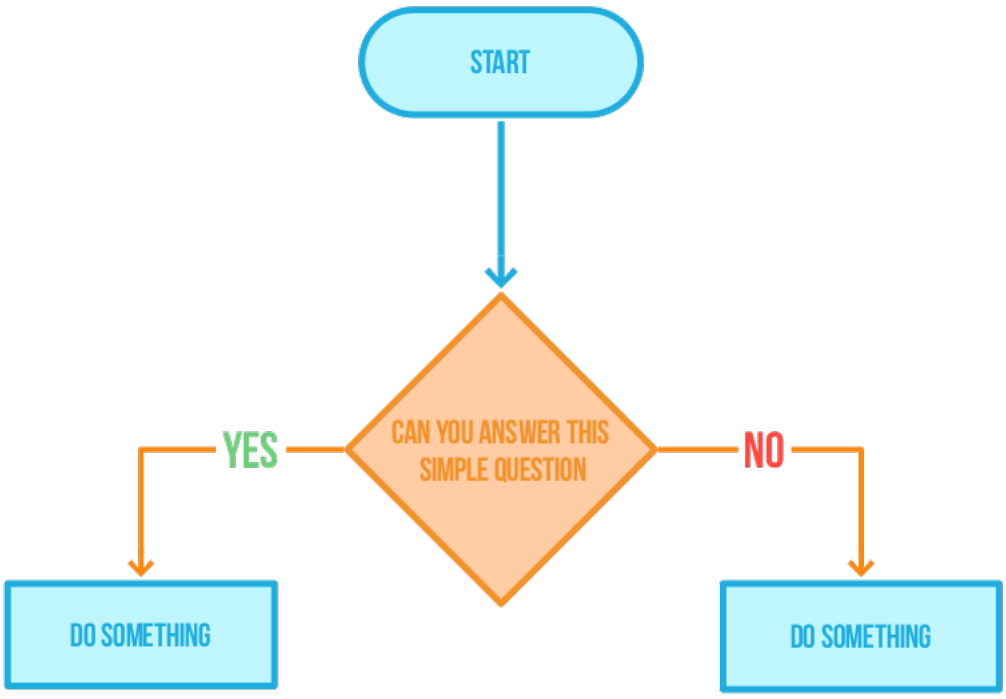

Figure 1.1: Caption for flowchart

Credit: Acme company makes everything `https://acme.com/`

1.3.3 Tables

We can also do tables. Protip: use `https://www.tablesgenerator.com/` for generating tables.

Table 1.1: Caption of table

Title1	Title2	Title3
data1	data2	data3

1.3.4 Git

Git is fun, use it!

Chapter 2

Background

2.1 Cryptology

2.1.1 Cryptography

2.1.2 Cryptanalysis

2.2 Digital Signatures

2.2.1 Hash-and-Sign

2.2.2 GGH

2.2.3 NTRU

2.3 Linear Algebra and Lattices

2.4 Probability Theory

2.5 Gradient Search

Chapter 3

Hawk and HPP

3.1 Hawk

3.2 Implementation of Hawk

3.3 HPP

In this section we present the *Learning a Parallelepiped* attack as described in [4], with some different notation to not confuse with Hawk notation.

3.3.1 Setup and idealized version

Let $V \in \mathcal{GL}(\mathbb{R})$ be a secret $n \times n$ unimodular matrix and $\mathcal{P}(V)$ be a fundamental parallelepiped, defined as $\{V\mathbf{x} : \mathbf{x} \in [-1,1]^n\}$. Let $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$ be t row vectors of length n with entries uniformly distributed over $[-1,1] \in \mathbb{Q}$ and $\mathbf{v} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_t\} = \{V\mathbf{x}_1, V\mathbf{x}_2, ..., V\mathbf{x}_t\}$ such that $\mathbf{v} \subset \mathcal{P}(V)$. By observing $\mathbf{v}$ for large enough t, one is able to retrieve the rows of $\pm V$ by the following steps:

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

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

3.3.2 Covariance matrix estimation

Given enough samples $\mathbf{v} = V\mathbf{x}$, we want to estimate the covariance matrix $G \approx V^tV$. This is achieved as $\mathbf{v}^t\mathbf{v} = (V\mathbf{x})^t(V\mathbf{x}) = V^t\mathbf{x}^t\mathbf{x}V$. Now, $\mathbb{E}[\mathbf{x}^t\mathbf{x}] = I/3$ where I is the identity matrix because of the following: Since all $x_i \in \mathbf{x}$ are independent, and $\mathbb{E}[x_i] = 0$, we have that $\mathbb{E}[x_ix_j] = \mathbb{E}[x_i]\mathbb{E}[x_j] = 0$ for $i \neq j$. For the case when $i = j$, we have

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

Therefore, $\mathbb{E}[\mathbf{x}^t\mathbf{x}]$ is $\frac{1}{3}$ down the diagonal and is 0 otherwise, i.e. $I/3$. Consequently, $\mathbf{v}^t\mathbf{v} = V^t(I/3)V$, and therefore $\mathbf{v}^t\mathbf{v} \cdot 3 = V^tV$. Clearly, the more samples $\mathbf{v}$ one has, the more accurate the approximation $G \approx V^tV$ is.

3.3.3 Hidden parallelepiped to hidden hypercube transformation

Given a good approximation G of V^tV, the next step is to calculate a linear transformation L such that the following is true:

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

This is achieved by taking the Cholesky decomposition of $G^{-1} = LL^t$. To compute Cholesky decomposition of G^{-1}, we must first show that G is symmetric positive definite. G is symmetric $\iff$ $G^t = G$ which is clear as $G^t = (V^tV)^t = V^t(V^t)^t = V^tV = G$. G is positive definite if for any non-zero column vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^tG\mathbf{x} > 0$. We have that $\mathbf{x}^tG\mathbf{x} = \mathbf{x}^tV^tV\mathbf{x} = (V\mathbf{x})^t(V\mathbf{x})$. Denote by $\mathbf{y} = V\mathbf{x}$. Since $\mathbf{x} \neq \mathbf{0}$ and V needs is invertible and therefore non-zero, it is clear that $\mathbf{y} \neq \mathbf{0}$ and $\mathbf{y}^t\mathbf{y} = ||\mathbf{y}||^2 > 0$.

From this, we easily show the following:

1. If $C = VL$, then $CC^t = VLL^tV^t = VG^{-1}V^t = V(V^tV)^{-1}V^t = VV^{-1}V^{-t}V^t = I$.
2. Since entries in $\mathbf{x}$ is uniformly distributed, then $\mathbf{c} = C\mathbf{x}$ is clearly uniformly distributed over $\mathcal{P}(C)$.

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

3.3.4 Moments and Gradient Descent

We now measure and minimize the fourth moment of one-dimensional projections to disclose rows of $\pm C$. Let $mom_{k,C}(\mathbf{w})$ be defined as the k-th moment of $\mathcal{P}(C)$ projected onto $\mathbf{w}$, i.e. $\mathbb{E}[\langle \mathbf{c}, \mathbf{w}\rangle^k]$ where $\mathbf{c} = \mathbf{x}C$ for uniformly distributed $\mathbf{x}$ and $\mathbf{w} \in \mathbb{R}^n$. Looking at the term $\langle \mathbf{c}, \mathbf{w}\rangle$, we have $\langle \mathbf{x}C, \mathbf{w}\rangle = \langle \sum_{i=1}^n x_ic_i, \mathbf{w}\rangle = \sum_{i=1}^n x_i\langle c_i, \mathbf{w}\rangle$. Moving this inside the $\mathbb{E}[\ ]$ we have $\mathbb{E}[\sum_{i=1}^n x_i\langle c_i, \mathbf{w}\rangle^k]$. We look at the two cases when $k = 2$ and $k = 4$.

- $\mathbf{k = 2}$: $\mathbb{E}[(\sum_{i=1}^{n} x_i \langle c_i, \mathbf{w}\rangle)^2] = \mathbb{E}[\sum_i^n \sum_j^n x_i x_j \langle c_i, \mathbf{w}\rangle \langle c_j, \mathbf{w}\rangle]$. As seen before in section 3.3.2, $\mathbb{E}[x_i x_j] = \frac{1}{3}$ when $i = j$ and 0 otherwise. Thus, we have the final 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 $= \frac{1}{3}\mathbf{w}C^tC\mathbf{w}^t$
- $\mathbf{k = 4}$:

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

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 and 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

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

Now, since 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 $mom_{2,C}(\mathbf{w}) = \frac{1}{3}\mathbf{w}C^tC\mathbf{w}^t = \frac{1}{3}\mathbf{w}I\mathbf{w}^t = \frac{1}{3}\mathbf{w}\mathbf{w}^t = \frac{1}{3}\|\mathbf{w}\|^2 = \frac{1}{3}$.
By rewriting and expanding the fourth moment:

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

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

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

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

3.4 HPP against the Normal Distribution

Lemma 1 (Lemma 1) *Assume signatures on the form* $\mathbf{v} = \mathbf{x}V$ *where* $x_i \sim \mathcal{N}(0, \sigma^2)$. *After converting* $\mathcal{P}(V)$ *to* $\mathcal{P}(C)$, *the fourth moment of* $\mathcal{P}(C)$ *is constant over some* $\mathbf{w}$ *on the unit circle.*

Proof 1 *Some proof here*

Chapter 4

Cryptanalysis of Hawk

In this chapter we perform the cryptanalysis of Hawk.

4.1 Overview

The original HPP attack can not work if the vector $\mathbf{x}$ multiplied with secret V has normally distributed entries as shown in section 3.4. In practical Hawk, however, the distribution of entries of $\mathbf{x}$ is discrete, not continuous. The Discrete Gaussian Distribution (DGD) as described in [3] and in section 2.4.... closely emulates that of its continuous normal counterpart. It is however not that straightforward to prove 1 for the discrete case. The sampling procedure in the signature generation step in Hawk (see section 3.2...) emulates sampling from DGD using cumulative distribution tables. Instead of showing theoretical and asymptotic results for the DGD, we instead use our implementation of Hawk to measure the properties the distribution of the practical sampler.

The belief is that the discretization of the distribution is enough to make 1 not hold. Consequently, one might be able to disclose a column of

4.2 HPP against practical Discrete Gaussian Distribution

Consider the Discrete Gaussian Distribution as described in [3] and in section 2.4... We use our implementation of Hawk to sample many points from the practical distribution.

Let $\mathcal{D}$ denote the theoretical discrete Gaussian distribution, and let $\widehat{\mathcal{D}}$ denote the practical discrete Gaussian distribution from sampled points. Let 0, σ^2 be the expectation and variance of $\mathcal{D}$, and $\hat{\mu}$, $\hat{\sigma}^2$ be the expectation and variance of $\widehat{\mathcal{D}}$. Assume we sample t points as $X = \{x_1, x_2, ..., x_t\}$. We estimate $\hat{\mu}$ and $\hat{\sigma}^2$ simply by $\hat{\mu} = \frac{1}{t}\sum_{i=1}^{t} x_i$ and $\hat{\sigma}^2 = \frac{1}{t}\sum_{i=1}^{t}(x_i - \hat{\mu})^2$. For simplicity, we can also assume $\hat{\mu} = \mu = 0$ as specified in [3]. To simplify later computations we can 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, assume

Glossary

Git Git is a Version Control System (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.

Acronyms

CVP Closest Vector Problem.

DGD Discrete Gaussian Distribution.

DSA Digital Signature Algorithm.

HPP Hidden Parallelepiped Problem.

KEMs Key Encapsulation Methods.

NIST National Institute of Standards and Technology.

RSA Rivest, Shamir, Adleman.

VCS Version Control System.

Bibliography

[1] 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.

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

[3] 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/.

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

Appendix A

Generated code from Protocol buffers

Listing A.1: Source code of something

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