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

January, 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 Rivest, Shamir, Adleman (RSA) signature schemes are in peril due to the potential emergence of quantum computers which, theoretically, can be able to 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 the 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 for this attack was proposed, but ultimately broken again in 2012 due to a more advanced extension of the original HPP 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, signature- and keysizes. It has some notable structural similarities to that of NTRUsign, but is theoretically safe from the HPP attack. In practice, however, this might not necessarily be the case. This thesis will therefore investigate if a method based on solving the HPP can be aimed at Hawk to retrieve some information about the secret key.

1.2 Objectives

The objective for 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 on ones own is a good way to learn how it works. I chose to implement it in Rust for the sake of learning this programming language as a 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 followed.
- **Cryptanalysis and experimentation**. The second part of this thesis is cryptanalysis of Hawk. The goal is to use the *Learning a parallelepiped* attack [4] 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 implmentation, and the *Learning a Parallelepiped* attack. In Chapter 4 the cryptanalysis of Hawk is presented. The final chapter will discuss 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

```
package main

import "fmt"

func main() {
    fmt.Println("hello world")
}
```

1.3.2 Figures

Example of a centred figure

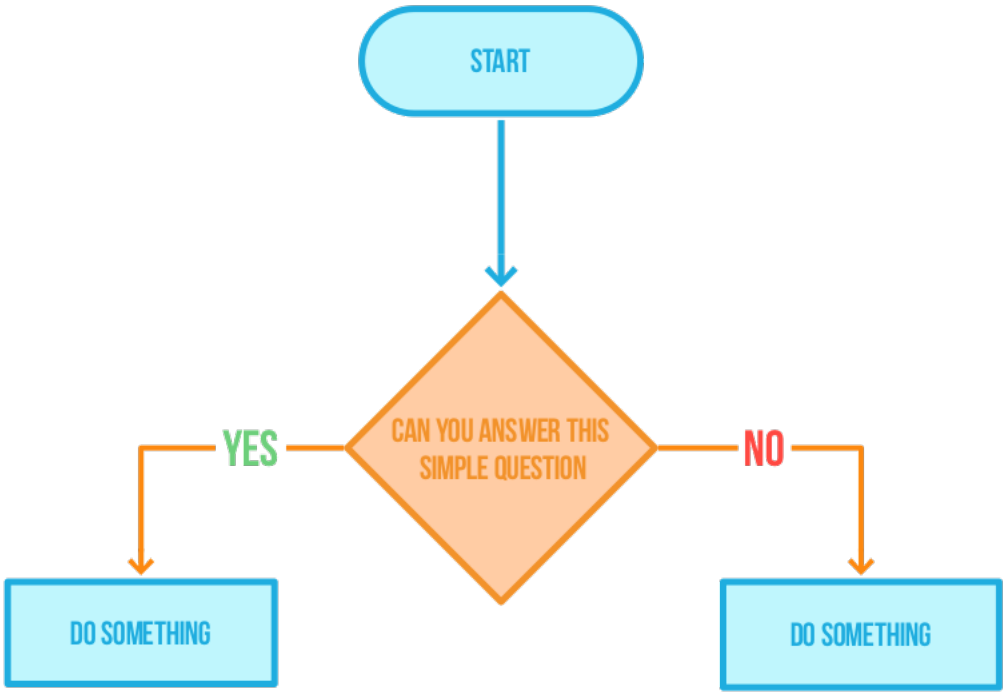

Figure 1.1: Caption for flowchart

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

Chapter 3

Hawk and HPP

3.1 Hawk

3.2 Implementation of Hawk

3.3 HPP

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 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.

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 $\hat{\sigma}^2 = \frac{1}{t}\sum_{i=1}^{t}(x_i - \hat{\mu})^2$. 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");
```