University of Bergen
Department of Informatics

Hidden parallelepiped in Hawk

Author: Eirik Djupvik Skjerve

Supervisors: Igor Aleksandrovich Semaev & Martin Feussner

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

November, 2024

Abstract

Some abstract here

Acknowledgements

Thank you to some people

Eirik D. Skjerve

Monday 18th November, 2024

Contents

Chapter 1

Introduction

1.1 Context and motivation

Digital signatures are an important part of secure communication today. The most used cryptographic scheme used for digital signatures today is DSA (Digital Signature Algorithm) or RSA (Rivest, Shamir, and Adleman) signatures (source). However, in 1994, Peter Shor developed Shor's algorithm, which, given a large enough quantum computer, is able to solve the hard problems DSA and RSA is based upon, namely the Discrete Logarithm Problem and Prime Factorization(source). Whether big enough quantum computers will emerge any time soon is debatable. However, measures against this potential looming threat has already begun. In 2016, NIST (National Institute of Standards and Technology) announced a standardization process for new standard schemes for KEMs (Key Encapsulation Methods) and digital signatures that have strong security against quantum computers (source). 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 (source).

Cryptographic schemes based on lattice problems are not an enirely new phenomenon, however. NTRU-Sign, published in 2003(source), is a digital signature scheme based on the hardness of the Closest Vector Problem (source). The original scheme was broken due to Phong. Q. Nguyen & Oded Regev in 2006 [2], who showed that by observing enough signatures generated with one secret key, one can retrieve the secret key. A newer digital signature scheme, Hawk (source), submitted to NIST's standardization process, is a scheme similar that of NTRU and GGH. The goal of this thesis is to try and adapt the Hidden Parallelepiped Problem attack to Hawk: [1].

1.2 Objectives

The objective for this thesis consists of two main parts:

- **Implementation of Hawk in Rust**. As the first part of the thesis I implement the Hawk digital signature scheme in the Rust programming language. Implementing a scheme on ones own is a good way to actually learn how it works. I chose to implement it in Rust for the sake of learning the programming language. Moreover, having ones own version makes it easier to experiment, adjust and modify to ones need. It would also be a challenge to understand and work with complicated source code someone else has written.
- **Cryptanalysis and experimentation**. As part two of the thesis I want to do cryptanalysis of Hawk. The goal is to use the "Learning a parallelepiped" attack [2] and adjusting it to try and break Hawk. This requires both theoretical and practical work, and experiments will 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 the *Learning a Parallelepiped* attack and implementations of these. In Chapter 4, a version HPP attack aimed at Hawk will be presented. Chapter 5 will show results, and Chapter 6 will discuss future work.

Chapter 2

Background

2.1 Asymmetric cryptography

2.1.1 Digital signatures

2.1.2 Hash-Then-Sign

2.1.3 GGH

2.2 Linear algebra & lattices

2.2.1 Lattices

2.3 Probability theory

2.3.1 Distributions

Chapter 3

Hawk and *Learning a parallelepiped*

3.1 Hawk

In the following Hawk, the digital signature scheme, will be presented, as in [1]

3.1.1 Simple Hawk

Present simple sketch of keygen, sign and ver, as well as an example in dimension 2

3.1.2 Key generation

3.1.3 Signature generation

3.1.4 Signature verification

3.2 Learning a parallelepiped

The paper *Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures* by Phong Q. Nguyen and Oded Regev from 2006 introduced a method for breaking digital signature schemes based on the GHH scheme [2]. Essentially, by observing enough *message, signature* pairs generated by a secret key one can deduce this secret key. The attack broke the NTRU-Sign scheme by observing as little as 400 signatures.

3.2.1 Solving the Hidden Parallelepiped Problem

First we define an idealized version of both the problem to solve and the solution as proposed in [2]. Then we discuss the problem and solution in a practical context.

Hidden Parallelepiped Problem (HPP): Let $V = [\mathbf{v}_1, ...\mathbf{v}_n]$ be a secret $n \times n$ matrix and $V \in \mathcal{GL}_n(\mathbb{R})$. Define by $\mathcal{P}(V) = \{\sum_{i=1}^{n} x_i \mathbf{v}_i : x_i \in [-1, 1]\}$ a secret n-dimensional parallelepiped defined by V. Given a polynomial (in n) number of vector samples uniformly distributed over $\mathcal{P}(V)$, recover the rows $\mathbf{v}_i$ of V.
We solve this problem in two main steps:

1. Transforming the hidden parallelepiped into a hidden hypercube:
2. Learning the hypercube:

Parallelepiped → Hypercube The first step of the attack is to convert our hidden parallelepiped $\mathcal{P}(V)$ into a hidden hypercube $\mathcal{P}(C)$ with orthogonal basis vectors. Essentially, one moves each sampled point in accordance to a transformation matrix L computed the following way:

- Approximate $G \approx V^T V$ using our samples $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_u\}$
- Compute L such that $LL^T = G^{-1}$
- Then $C = VL$
- By multiplying our samples $\mathcal{X}$ to the right by L, they are now uniformly distributed over the hidden hypercube $\mathcal{P}(C)$
- **(—SHOW CALCULATIONS—)**

Learning a Hypercube: The second step is to learn the hypercube. Given samples over $\mathcal{P}(C)$, we deduce the rows of the secret matrix C with the method described in Algorithm 1. After the rows are approximated, one can multiply the rows $\{\mathbf{c}_1, ..., \mathbf{c}_2\}$ by L^{-1} such that we have $\{\mathbf{v}_1, ..., \mathbf{v}_2\}$, and we are done.

Algorithm 1 Learning a Hypercube

```
Require: Descent parameter $\delta$, samples $\mathcal{X}$ uniformly distributed over $\mathcal{P}(C)$
Ensure: A row vector $\pm\mathbf{v}_i$ of $C$
  Choose uniformly at random $\mathbf{w}$ on the unit sphere of $\mathbb{R}^n$
  loop
      Compute $\mathbf{g}$, an approximation of $\nabla mom_4(\mathbf{w})$
      Let $\mathbf{w}_{new} = \mathbf{w} - \delta\mathbf{g}$
      Place $\mathbf{w}_{new}$ back on the unit sphere by dividing it by $\|\mathbf{w}_{new}\|$
      if $mom_4(\mathbf{w}_{new}) \geq mom_4(\mathbf{w})$ then          ▷ $mom_4$ are approximated by samples
          return w
      else
          Replace $\mathbf{w}$ with $\mathbf{w}_{\mathbf{new}}$ and continue loop
      end if
  end loop
```

3.2.2 HPP against NTRU

3.2.3 HPP against normally distributed samples

The key component and assumption of the *Learning a parallelepiped* attack is that the provided samples are distributed *uniformly* over $\mathcal{P}(V)$. Assume now that samples over $\mathcal{P}(V)$ is distributed normally.

Approximating V^tV Let $V \in \mathcal{GL}_n(\mathbb{R})$. Let $\mathbf{v}$ be chosen from a normal distribution over $\mathcal{P}(V)$. Then $\mathbb{E}[\mathbf{v}^t\mathbf{v}] = V^tV/\sigma^2$.

Proof. Let $\mathbf{v} = \mathbf{x}V$ where $\mathbf{x}$ is distributed over $\mathcal{N}(\mu = 0, \sigma)$, over the interval $[-\infty, \infty]$. Then $\mathbf{v}^t\mathbf{v} = V^t\mathbf{x}^t\mathbf{x}V$. Considering $\mathbb{E}[\mathbf{x}^t\mathbf{x}]$, we see that for $i \neq j$, $\mathbb{E}[x_ix_j] = \mathbb{E}[x_i]\mathbb{E}[x_j] = 0 \cdot 0 = 0$ due to independent random variables. For $\mathbb{E}[x_i^2] = \mathbb{V}[x_i]$ since $\mathbb{V}[x_i] = \mathbb{E}[x_i^2] - \mathbb{E}[x_i]^2 = \mathbb{E}[x_i^2] - 0 = \sigma^2$. Therefore, $\mathbb{E}[\mathbf{x}^t\mathbf{x}] = \frac{I_n}{\sigma^2}$, i.e. the identity matrix with diagonal entries divided by σ^2. Consequently, $\mathbf{v}^t\mathbf{v} = V^t\mathbb{E}[\mathbf{x}^t\mathbf{x}]V = V^t\frac{I_n}{\sigma^2}V = V^tV/\sigma^2$ and conversely $V^tV = \sigma^2 \cdot \mathbf{v}^t\mathbf{v}$. □

This means that we can in theory approximate the covariance matrix V^tV. However, it is clear that when the distribution is normal instead of uniform, one needs many more samples (why is this?). The question is also the following: how accurate does our approximation need to be?

Chapter 4

Implementation

Introduction to the implementation part of the thesis

4.1 Implementation of Hawk

Something something about the implementation of Hawk in Rust. Mentions of sampling, integer/float types, speed and comparison to Hawk team's C code?

4.2 Implementation of HPP

Something something about the implementation of HPP in Rust. Something about speedup/parallelization of gradient descent?

Chapter 5

Adapting HPP to Hawk

In this chapter we investigate the steps needed to possibly apply the Hidden Parallelepiped Problem to the Hawk digital signature scheme.

5.1 Covariance matrix secret key in Hawk

Nothing yet I'm afraid

5.2 Secret Key Recovery

Since $\mathbf{x}$ follows some distribution close to some normal distribution, we hope that enough vectors $\mathbf{w}$ will disclose some information about $\mathbf{B}^{-1}$. If we know $\mathbf{B}^{-1}$ we know $\mathbf{B}$. This is the goal.

Bibliography

[1] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, 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/`.

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

Appendix A

Generated code

Listing A.1: Source code of something

```
println!("Goodbye World");
```