An introduction to lattice trapdoor

Ganyu Xu (g66xu)

Fall, 2023

Acknowledgement

Thank you to Vinod Vaikuntanathan[10] and Daniele Micciancio[6], for publishing the lecture notes from MIT's CS294 and UCSD's CSE206A respectively, as well as Chris Peikert[9] for his survey *A decade of lattice cryptography*. The content from these lecture notes and survey inspired this project and provided a wealth of technical details that this project is based on.

1 Introduction

Trapdoor functions are the foundations of public-key cryptography. The most famous and popular trapdoor functions include the RSA trapdoor (it is easy to compute m^e from m and e but difficult to recover m from m^e and e) and discrete log (it is easy to compute g^x from g, x but hard to recover x from g^x, g). While these two trapdoors have been proved very successful over their decades of application, they are unfortunately both based on the hardness of integer factorization. Trapdoor functions based on other classes of hard problems have been of research interest since the early days of public-key cryptography, and they have gained increased attention efficient quantum factorization algorithm threatens the security guarantee of RSA and Diffie-Hellman schemes.

This project surveys a particular class of cryptographic trapdoor functions whose construction is based on lattices and whose security is based on the hardness of lattice problems such as the shortest vector problem (SVP) and the closest vector problem (CVP). In section 2, we will state these hard problems and discuss solutions including the nearest plane algorithm and the LLL basis reduction algorithm. In section 3, we will introduce the Goldreich-Golwasser-Halevi trapdoor construction and its applications.

1.1 Notations and preliminaries

Troughout this survey, vectors are denoted with bolded letters $\mathbf{v}$ while matrices are denoted by capital letters B. A basis $B \in \mathbb{Z}^{n \times b}$ is assumed to be full-rank integer matrix unless explicitly stated. The lattice generated by B is denoted by $\mathcal{L}(B) = \{B\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$. The linear span of B is denoted by $\operatorname{span}(B) = \{B\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n\}$.

Given a basis B, the Gram-Schmidt orthogonalization algorithm returns a second basis B^* that spans the same linear space as B, but whose base vectors are pairwise orthogonal:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^*, \text{where } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$$

We define a projection of $\mathbf{x} \in \operatorname{span}(B)$ onto the set of orthogonalized basis $\{\mathbf{b}_j\}_{j=i}^n$ by the function π_i:

$$\pi_i(\mathbf{x}) = \sum_{j \geq i} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*$$

It is easy to see that for $i = 1$, $\pi_i(\mathbf{x}) = \mathbf{x}$ is the identity function. Also, we can show that $\pi_i(\mathbf{b}_i) = \mathbf{b}_i^*$:

$$\begin{aligned}
\pi_i(\mathbf{b}_i) &= \sum_{j \geq i} \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^* \\
&= \sum_{j \geq i} \mu_{i,j} \mathbf{b}_j^* \\
&= \sum_{1 \leq j \leq n} \mu_{i,j} \mathbf{b}_j^* - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^* \\
&= \pi_1(\mathbf{b}_i) - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^* \\
&= \mathbf{b}_i - \sum_{j<i} \mu_{i,j} \mathbf{b}_j^* \\
&= \mathbf{b}_i^*
\end{aligned}$$

2 Hard lattice problems and best known solutions

We begin the discussion of hard lattice problems by defining two such problems. Unless otherwise specified, the Euclidean norm is used.

Definition 2.1. *Given a lattice $\mathcal{L}(B)$ spanned by basis B and a norm $\|\cdot\| \to \mathbb{R}$, the* ***shortest vector problem*** *asks to find the shortest non-zero vector $\mathbf{v} \in \mathcal{L}(B)$*

The norm of the shortest non-zero vector is called the **minimum distance** and is denoted by $\lambda(\mathcal{L})$. Where the lattice is unambiguous we simply denote the minimum distance by λ.

Definition 2.2. *Given a lattice $\mathcal{L}(B)$, a norm, and some target vector $\mathbf{t}$ in the same vector space as the lattice, the* ***closest vector problem*** *asks to find a lattice point $\mathbf{v} \in \mathcal{L}$ that minimizes $\|\mathbf{t} - \mathbf{v}\|$*

There are two main parameters that affect the difficulty of SVP and CVP. The first is the number of dimensions n: the higher the number of dimension, the harder it is to find the shortest vector and/or the closest vector. The second is the orthogonality and/or the length of the basis. Intuitively, orthogonality and length are inversely related because the determinant of the lattice is invariant with respect to the choice of basis: higher orthogonality automatically implies shorter lengths. The more skewed the choice of basis, the more difficult it is to solve SVP/CVP.

2.1 Minimum distance and orthogonalized basis

The minimum distance of a lattice is bounded below by the shortest orthogonalized basis:

Theorem 2.1. *Given lattice $\mathcal{L}(B)$ with basis $B = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ and its Gram-Schmidt orthogonalization $B^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$,*

$$\lambda \geq \min_{1 \leq i \leq n} \|\mathbf{b}_i^*\|$$

To prove this lower bound, we split the lattice $\mathcal{L}(B)$ into two partitions: the subset of lattice points generated by the sub-basis $B' = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{n-1}]$ and the subset of lattice points whose has non-zero coefficient for the last base vector $\mathbf{b}_n$:

$$\mathcal{L}(B) = \mathcal{L}(B') \cup \{B\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n, x_n \neq 0\}$$

Then, the minimum distance of $\mathcal{L}(B)$ is the minimum between the minimum distance of each subset. While we don't know the exact minimum distance of the second partition, we do know that such minimum distance must not be less than $\|\mathbf{b}_n^*\|$:

$$\begin{aligned}
\|B\mathbf{x}\|^2 &= \|\sum_{i=1}^{n} \mathbf{b}_i x_i\|^2 \\
&= \|\sum_{i=1}^{n-1} \mathbf{b}_i x_i + \mathbf{b}_n x_n\|^2 \\
&= \|\sum_{i=1}^{n-1} \mathbf{b}_i x_i + (\mathbf{b}_n^* + \sum_{j<n} \mu_{j,n} \mathbf{b}_j^*) x_n\|^2 \\
&= \|\sum_{i=1}^{n-1} \mathbf{b}_i x_i + (\sum_{j<n} \mu_{j,n} \mathbf{b}_j^*) x_n + \mathbf{b}_n^* x_n\|^2 \\
&= \|\sum_{i=1}^{n-1} \mathbf{b}_i x_i + (\sum_{j<n} \mu_{j,n} \mathbf{b}_j^*) x_n\|^2 + \|\mathbf{b}_n^* x_n\|^2 \\
&\geq \|\mathbf{b}_n^* x_n\|^2 \\
&\geq \|\mathbf{b}_n^*\|^2
\end{aligned}$$

On the other hand, the minimum distance of the first partition is simply the minimum distance of the sub-lattice. Therefore we've obtained a recursive relationship that can be repeatedly applied n times:

$$\begin{aligned}
\lambda(\mathcal{L}(B)) &\geq \min(\lambda(\mathcal{L}([\mathbf{b}_i]_{i=1}^{n-1})), \|\mathbf{b}_n^*\|) \\
&\geq \min(\min(\lambda(\mathcal{L}([\mathbf{b}_i]_{i=1}^{n-2})), \|\mathbf{b}_{n-1}^*\|), \|\mathbf{b}_n^*\|) \\
&\ldots \\
&\geq \min_{1 \leq i \leq n} (\|\mathbf{b}_i^*\|)
\end{aligned}$$

■

2.2 Reduced lattice basis

At the time of this survey, the best algorithm for solving the (approximate) shortest vector problem is the LLL lattice basis reduction algorithm[4], attributed to Arjen Lenstra, Hendrik Lenstra, and László Lovász. The reduction algorithm transforms an input basis into an LLL-reduced form parameterized by a real number $\frac{1}{4} < \delta \leq 1$, where the first base vector of the reduced basis is an approximation of the shortest vector. Details of the actual algorithm for obtaining a reduced basis will be discussed in 2.5. Meanwhile, it is helpful to state the definition of the reduced basis and discuss its relationship to the shortest vector.

Definition 2.3. *A basis B is δ-LLL reduced if two conditions are satisfied*

1. *For all $j > i$, $|\mu_{j,i}| \leq \frac{1}{2}$*
2. *For all $1 \leq i \leq n-1$, $\delta\|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2$*

The first condition will be discussed in details in 2.4. For now, we focus on the second condition, which gives us a description of how a reduced basis can be used to approximate the shortest vector.

Theorem 2.2. *If B is δ-LLL reduced basis, then*

$$\|\mathbf{b}_1\| \leq \alpha^{\frac{n-1}{2}} \lambda$$

where $\alpha = \frac{1}{\delta - \frac{1}{4}}$

Proof. First observe the RHS of the condition:

$$\begin{aligned}
\|\pi_i(\mathbf{b}_{i+1})\|^2 &= \|\sum_{j \geq i} \mu_{i+1,j}\mathbf{b}_j^*\|^2 \\
&= \|\mu_{i+1,i}\mathbf{b}_i^* + \sum_{j \geq i+1} \mu_{i+1,j}\mathbf{b}_j^*\|^2 \\
&= \|\mu_{i+1,i}\mathbf{b}_i^* + \pi_{i+1}(\mathbf{b}_{i+1})\|^2 \\
&= \|\mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*\|^2 \\
&= \|\mu_{i+1,i}\mathbf{b}_i^*\|^2 + \|\mathbf{b}_{i+1}^*\|^2
\end{aligned}$$

Substituting the equation back into the condition:

$$\delta\|\pi_i(\mathbf{b}_i)\|^2 \leq \|\mu_{i+1,i}\mathbf{b}_i^*\|^2 + \|\mathbf{b}_{i+1}^*\|^2$$

Which transforms into:

$$\|\mathbf{b}_i^*\|^2 \leq \frac{1}{(\delta - \mu_{i+1,i}^2)}\|\mathbf{b}_{i+1}^*\|^2$$

By the first clause of the reduced basis we know $\mu_{i+1,i}^2 \leq \frac{1}{4}$, which means that $\frac{1}{\delta - \mu_{i+1,i}^2} \leq \frac{1}{\delta - \frac{1}{4}}$. Denote $\alpha = \frac{1}{\delta - \frac{1}{4}}$, then $\alpha > \frac{4}{3}$, and we have the following inequality:

$$\|\mathbf{b}_i^*\|^2 \leq \alpha\|\mathbf{b}_{i+1}^*\|^2$$

Note that because $\mathbf{b}_1 = \mathbf{b}_1^*$, we can recusively evaluate the inequality above can obtain the following closed inequality:

$$\|\mathbf{b}_1\|^2 \leq \alpha^{i-1}\|\mathbf{b}_i^*\|^2 \leq \alpha^{n-1}\|\mathbf{b}_i^*\|^2$$

The inequality above states that $\mathbf{b}_1$ is not longer than $\alpha^{n-1}\|\mathbf{b}_i^*\|$ for all i, so it must be not longer than the minimum among $\alpha^{n-1}\|\mathbf{b}_i^*\|$:

$$\|\mathbf{b}_1\|^2 \leq \min_{1\leq i\leq n} \alpha^{n-1}\|\mathbf{b}_i^*\|^2 = \alpha^{n-1} \min_{1\leq i\leq n} \|\mathbf{b}_i^*\|^2$$

From the previou section we've derived that $\lambda \geq \min_{1\leq i\leq n} \|\mathbf{b}_i^*\|$, which we can plug into the inequality above:

$$\|\mathbf{b}_1\| \leq \alpha^{\frac{n-1}{2}}\lambda$$

□

In other words, the first base vector of a LLL-reduced basis is an approximation of the true shortest vector within an exponential factor.

2.3 Fundamental regions

A fundamental region S is a subset of the (real) linear span of the basis such that it tiles the linear span of the basis and each tile contains exactly one lattice point. If we have a fundamental region S defined, then each point in the linear span of the basis can be uniquely decomposed into the sum of a lattice point and a point in the fundamental region:

$$\text{span}(B) = \{S + B\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$$

A few notable examples include the fundamental parallelpiped

$$\mathcal{P}(B) = \{B\mathbf{x} \mid \mathbf{x} \in [0, 1)^n\}$$

and the centered fundamental parallelpiped

$$\mathcal{C}(B) = \{B\mathbf{x} \mid \mathbf{x} \in [-\frac{1}{2}, \frac{1}{2})^n\}$$

Computing the decomposition using the (centered) fundamental parallelpiped is easy. First compute $B^{-1}\mathbf{t}$ (over $\mathbb{R}$), then perform some kind of rounding (flooring or nearest integer) depending on whether the fundamental parallelpiped is centered or or not. $B\lfloor B^{-1}\mathbf{T}\rceil$ is the lattice point whose corresponding fundamental region contains the target. As we will discuss in section 3, when the basis is short and orthogonal, this rounding algorithm indeed provides a good approximate solution to the closest vector problem. However, when the basis is long and skewed, these fundamental regions become terrible approximation.

A better approximation for the closest vector problem can be found using a fundamental region defined by the centered paralellpiped spanned by the orthogonalized basis:

$$\mathcal{C}(B^*) = \{B^*\mathbf{x} \mid \mathbf{x} \in [-\frac{1}{2}, \frac{1}{2})^n\}$$

Because B^* is an orthogonal basis, it's easy to see that the sphere centered at some lattice point $\mathbf{v} \in \mathcal{L}$ and whose radius is $\frac{1}{2} \min \|\mathbf{b}_i^*\|$ is entirely contained in the (shifted) fundamental region sphere $\subset \{\mathcal{C}(B^*) + \mathbf{v}\}$. If a target point $\mathbf{t}$ falls in such a sphere, then the corresponding lattice point is the closest lattice point to that target point.

2.4 Babai's nearest plane algorithm

Babai's nearest plane algorithm, attributed to László Babai, is a recursive algorithm that can approximate the closest vector under a given basis to a target vector. More specifically, it returns a vector point $\mathbf{v}$ such that, if target vector is projected onto the orthogonalized basis, the projection is contained in $\mathbf{v} + \mathcal{C}(B^*)$. If the target vector is in the linear span of the basis, then the target vector itself is contained in $\mathbf{v} + \mathcal{C}(B^*)$.

Algorithm 1 NearestPlane

```
1: if $B$ is empty then
2:     return $\mathbf{0}$
3: end if
4: $B^* \leftarrow \text{GramSchmidt}(B)$
5: $c \leftarrow \lfloor \frac{\langle \mathbf{t}, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} \rceil$, where $\mathbf{b}_n^*$ is the last base vector in $B^*$
6: return $c\mathbf{b}_n + \text{NearestPlane}([\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{n-1}], \mathbf{t} - c\mathbf{b}_n)$
```

Theorem 2.3. *Given basis B and target $\mathbf{t}$, denote the output of* NearestPlane *by* $\mathbf{v} \leftarrow \text{NearestPlane}(B, \mathbf{t})$*, then for all* $1 \le i \le n$*:*

$$\frac{\langle \mathbf{t} - \mathbf{v}, \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle} \in [-\frac{1}{2}, \frac{1}{2})$$

Intuitively, the inequality above states that the deviation of $\mathbf{t}$ from $\mathbf{v}$ is between $-\frac{1}{2}\mathbf{b}_i^*$ and $\frac{1}{2}\mathbf{b}_i^*$ in any of the chosen direction $\mathbf{b}_i^*$, which means that $\mathbf{t} - \mathbf{v}$ is indeed contained in the orthogonalized fundamental parallelpiped.

Proof. We will prove by induction. The base case correpsonds to B being an empty basis, and so this statement is vacuously true.

In the inductive case, we denote the output of $\text{NearestPlane}(B', \mathbf{t} - c\mathbf{b}_n)$ by $\mathbf{v}'$. Assuming that the algorithm produces the desired result for the sublattice $\mathcal{L}(B')$:

$$\forall 1 \le i \le (n-1), \frac{\langle (\mathbf{t} - c\mathbf{b}_n) - \mathbf{v}', \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle} \in [-\frac{1}{2}, \frac{1}{2})$$

However, notice in step 5 of the algorithm $c\mathbf{b}_n + \mathbf{v}'$ is the output of the current iteration of the algorithm: $\mathbf{v} = c\mathbf{b}_n + \mathbf{v}'$, therefore:

$$\begin{aligned}(\mathbf{t} - c\mathbf{b}_n) - \mathbf{v}' &= \mathbf{t} - (c\mathbf{b}_n + \mathbf{v}') \\ &= \mathbf{t} - \mathbf{v}\end{aligned}$$

which means that

$$\forall 1 \le i \le (n-1), \frac{\langle \mathbf{t} - \mathbf{v}, \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle} \in [-\frac{1}{2}, \frac{1}{2})$$

It remains show that this relationship also holds for $i = n$.
To prove that the relationship holds for $i = n$, first observe how c is computed:

$$c \leftarrow \lfloor \frac{\langle \mathbf{t}, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} \rceil$$

which is equivalent to saying that

$$\frac{\langle \mathbf{t}, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} - c \in [-\frac{1}{2}, \frac{1}{2})$$

Also recall the Gram-Schmidt orthogonalization process:

$$\mathbf{b}_n^* + \sum_{i<n} \mu_{n,i} \mathbf{b}_i^* = \mathbf{b}_n$$

Therefore

$$\begin{aligned} \langle \mathbf{b}_n, \mathbf{b}_n^* \rangle &= \langle \mathbf{b}_n^* + \sum_{i<n} \mu_{n,i} \mathbf{b}_i^*, \mathbf{b}_n^* \rangle \\ &= \langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle \end{aligned}$$

All non $\mathbf{b}_n^*$ terms can be cleared because they are orthogonal to $\mathbf{b}_n^*$ and their inner product is 0. This equality means that:

$$\frac{\langle \mathbf{b}_n, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} = 1$$

Because $\mathbf{v}'$ is a sub-lattice point, it is a linear combination of $\{\mathbf{b}_j\}_{j<n}$, which is orthogonal to $\mathbf{b}_n^*$, so $\mathbf{v}'$ is also orthogonal to $\mathbf{b}_n^*$.
Finally we can put everything together:

$$\begin{aligned} \frac{\langle \mathbf{t}, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} - c &= \frac{\langle \mathbf{t}, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} - c\frac{\langle \mathbf{b}_n, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} \\ &= \frac{\langle \mathbf{t} - c\mathbf{b}_n, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} \\ &= \frac{\langle \mathbf{t} - c\mathbf{b}_n - \mathbf{v}', \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} \\ &= \frac{\langle \mathbf{t} - \mathbf{v}, \mathbf{b}_n^* \rangle}{\langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle} \in [-\frac{1}{2}, \frac{1}{2}) \end{aligned}$$

Hence we've proved the relationship for $i = n$. □

Babai's nearest plane algorithm is an essential component to the LLL lattice basis reduction algorithm. Specifically, the nearest plane algorithm is used to reduce the size of the basis vector so the first condition of the reduced basis (shown below) can be satisfied:

$$\forall j < i, |\mu_{i,j}| \leq \frac{1}{2}$$

The size reduction algorithm, which we will denote by SizeReduce, takes a basis B and returns a size-reduced basis B' such that the condition above is true.

Algorithm 2 SizeReduce

```
for $i \in \{2, 2, \ldots, n\}$ do
    $\mathbf{v} \in \mathcal{L}([\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}]) \leftarrow \text{NearestPlane}([\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}], \mathbf{b}_i - \mathbf{b}_i^*)$
    $\mathbf{b}_i \leftarrow \mathbf{b}_i - \mathbf{v}$
end for
```

Recall the result of the nearest plane algorithm we know that:

$$\forall j < i, \frac{\langle \mathbf{b}_i - \mathbf{b}_i^* - \mathbf{v}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \in [-\frac{1}{2}, \frac{1}{2})$$

Notice in the equation above, we have $\mathbf{b}_i^* \perp \mathbf{b}_j^*$ because $j < i$. We also have $\mathbf{b}_i - \mathbf{v}$ being the new value for $\mathbf{v}_i$ after the substitution. This means that after the substitution:

$$\forall j < i, \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \in [-\frac{1}{2}, \frac{1}{2})$$

The LHS of the equation above is exactly $\mu_{i,j}$. Thus we have satisfied the first condition of δ-LLL reduced basis. In addition, because $\mathbf{v} \in \mathcal{L}([\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}])$, the substitution $\mathbf{b}_i \leftarrow \mathbf{b}_i - \mathbf{v}$ is equivalent to adding onto the substituted basis a linear combination of other base vectors, which does not change the lattice.

2.5 The LLL basis reduction algorithm

Applying the SizeReduce algorithm to a basis transforms it to satisfy the first condition of being δ-LLL reduced. However, after the transformation, the second condition might not be satisfied everywhere, and we need to apply additional transformation to satisfy the second condition:

$$\delta \|\pi_i(\mathbf{b}_i)\|^2 \leq \|\pi_i(\mathbf{b}_{i+1})\|^2$$

It turns out that such transformation is rather simple: if there is some i such that the condition above does not hold, then swapping $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ will make the pair satisfy the condition. To see that it works, denote the swapped basis vectors by $\mathbf{b}_i' = \mathbf{b}_{i+1}$, $\mathbf{b}_{i+1}' = \mathbf{b}_i$. First observe that the function $\pi_i : \mathbf{x} \mapsto \sum_{j \geq i} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*$ is not changed after the swap because it is still projecting $\mathbf{x}$ onto the same set of orthogonal basis, and the set of orthogonal basis is unchanged by the swap.

If the condition does not hold, then $\delta \|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2$, and we have

$$\begin{aligned}\delta\|\pi_i(\mathbf{b}_i')\|^2 &= \delta\|\pi_i(\mathbf{b}_{i+1})\|^2\\ &< \delta\cdot\delta\|\pi_i(\mathbf{b}_i)\|^2\\ &\leq \|\pi_i(\mathbf{b}_i)\|^2\\ &= \|\pi_i(\mathbf{b}_{i+1}')\|^2\end{aligned}$$

We know that swapping columns preserves the lattice, so we can define a second algorithm ColumnSwap that takes a basis B and transforms it into a second basis of the same lattice that satisfies the second condition of δ-LLL reduced basis:

Algorithm 3 ColumnSwap

```
for $i \in \{1, 2, \ldots, n-1\}$ do
    if $\delta\|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2$ then
        $\mathbf{b}_i \leftarrow \mathbf{b}_{i+1}$
        $\mathbf{b}_{i+1} \leftarrow \mathbf{b}_i$
    end if
end for
```

It's easy to see that ColumnSwap always terminates in n iterations, and upon termination, the transformed basis satisfies the second condition of being δ-LLL reduced. Unforutnately, now the first condition might not hold everywhere, so we need to apply SizeReduce again.

Indeed, the LLL basis reduction algorithm repeatedly alternates between applying the two steps SizeReduce and ColumnSwap until the basis becomes δ-LLL reduced. It's trivially true that if the algorithm terminates, the output is guaranteed to satisfy both conditions of being δ-LLL reduced. It remains to show that, at least for $\delta < 1$, this algorithm terminates in polynomial time.

Algorithm 4 LLLReduce

```
while $B$ is not $\delta$-LLL reduced do
    $B \leftarrow$ SizeReduce($B$)
    $B \leftarrow$ ColumnSwap($B$)
end while
```

Theorem 2.4. *For $\delta < 1$, reducing a basis to δ-LLL reduced form takes polynomial time*

Proof. To prove that LLLReduce will terminate in polynomial time, we define a positive integer quantity associated with a basis and show that each iteration of SizeReduce and ColumnSwap reduce this quantity by δ.

Recall that although determinant is not defined for non-square matrix, it is defined for (sub)lattice generated by sub-basis $B_k = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k] \in \mathbb{Z}^{n\times k}$ where $1 \leq k \leq n$:

$$\det(\mathcal{L}(B_k)) = \prod_{i=1}^{k} \|\mathbf{b}_i^*\| = \sqrt{B_k^\intercal B_k}$$

From the definition above we can see that the square of the determinant of a lattice generated by integer basis is an integer.

We define the "potential" of a full-rank basis $B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$ by the product of determinants of sub-lattices generated by sub-basis $B_1, B_2, \ldots, B_n$:

$$\mathcal{D} = \prod_{k=1}^{n} \det(\mathcal{L}(B_k))^2$$

$\mathcal{D}$, being a product of integers, is itself an integer.

First observe that in SizeReduce, the basis vector is changed by subtracting a linear combination of basis vectors before the changed basis vector. This means that after SizeReduce, each of B_k for $1 \le k \le n$ still generates the same lattice, and the determinant of that lattice remains unchanged. In other words, SizeReduce does not change the value of $\mathcal{D}$.

Second, for $k < i$, swapping column $\mathbf{b}_i$ with $\mathbf{b}_{i+1}$ does not affect the lattice generated by the partial basis B_k because $B_k = \{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}$ does not contain the swapped columns anyways. On the other hand, for $k > i$, swapping column $\mathbf{b}_i$ with $\mathbf{b}_{i+1}$ also does not affect the lattice generated by the partial basis B_k because it contains both of the swapped column, and swapping column preserves the lattice.

Therefore the only change to $\mathcal{D}$ when swapping $\mathbf{b}_i$ with $\mathbf{b}_{i+1}$ comes from the factor $\det(\mathcal{L}(B_i))$. Denote the potential of the basis B after the swap by $\mathcal{D}'$ then:

$$\frac{\mathcal{D}}{\mathcal{D}'} = \frac{\det(\mathcal{L}([\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_i]))^2}{\det(\mathcal{L}([\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}]))^2}$$

Recall that the determinant of a lattice is the product of the orthogonalized basis vectors. The first $i-1$ terms are identical between the two basis so they all cancel out. The i-th term on the numerator is exactly $\mathbf{b}_i^*$, while the i-th term on the denominator is as follows:

$$\begin{aligned}
\text{last orthogonalized vector in the denominator} &= \mathbf{b}_{i+1} - \sum_{j<i} \mu_{i+1,j}\mathbf{b}_j^* \\
&= \mathbf{b}_{i+1} - (\sum_{1\le j\le n} \mu_{i+1,j}\mathbf{b}_j^* - \sum_{j\ge i} \mu_{i+1,j}\mathbf{b}_j^*) \\
&= \mathbf{b}_{i+1} - (\mathbf{b}_{i+1} - \sum_{j\ge i} \mu_{i+1,j}\mathbf{b}_j^*) \\
&= \sum_{j\ge i} \mu_{i+1,j}\mathbf{b}_j^* \\
&= \pi_i(\mathbf{b}_{i+1})
\end{aligned}$$

Therefore we have:

$$\frac{\mathcal{D}}{\mathcal{D}'} = \frac{\det(\mathcal{L}([\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_i]))^2}{\det(\mathcal{L}([\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}, \mathbf{b}_{i+1}]))^2} = \frac{\|\mathbf{b}_i^*\|^2}{\|\pi_i(\mathbf{b}_{i+1})\|^2} = \frac{\|\pi_i(\mathbf{b}_i)\|^2}{\|\pi_i(\mathbf{b}_{i+1})\|^2}$$

Because we only swap when $\delta\|\pi_i(\mathbf{b}_i)\|^2 > \|\pi_i(\mathbf{b}_{i+1})\|^2$, the quotient above satisfies:

$$\frac{\mathcal{D}}{\mathcal{D}'} = \frac{\|\pi_i(\mathbf{b}_i)\|^2}{\|\pi_i(\mathbf{b}_{i+1})\|^2} > \frac{1}{\delta}$$

Which implies $\mathcal{D}' < \delta\mathcal{D}$.

Notice that in the relationship above, the second condition of being δ-LLL reduced is violated if and only if $\mathcal{D}' < \delta\mathcal{D}$. This means that when $\mathcal{D}' \geq \delta\mathcal{D}$, the second condition will hold. Indeed, when the potential converges, the algorithm terminates, and since each iteration reduces the potential by δ, we can deduce that the algorithm will terminate in $O(\log_{\frac{1}{\delta}} \mathcal{D})$ time. □

3 The Goldreich-Goldwasser-Halevi trapdoor

One class of constructing lattice trapdoor uses a pair of public and secret basis for the same lattice. Since the two basis generate the same lattice, they are equally good at mapping integer coordinates into a lattice point. On the other hand, the secret basis is very "good" and can be used to efficiently find the closest lattice point, but the public basis is very "bad" at recovering the closest lattice point.

One of the earliest instances of such class of lattice trapdoor is proposed by Goldreich, Goldwasser, and Halevi in their 1997 paper *"Public-key cryptosystem from lattice reduction problem"*[2]. At a high level, the trapdoor is parameterized by three items: a "bad" basis B, a "good" basis R, and an error bound σ. In the forward direction, the function maps a pair of lattice coordinate $\mathbf{v} \in \mathbb{Z}^n$ and a small error vector $\mathbf{e} \leftarrow \{-\sigma, \sigma\}^n$ to $\mathbf{x} = B\mathbf{v} + \mathbf{e} \in \mathbb{R}^n$. If the parameters are generated correctly, then the closest lattice point in $\mathcal{L}(B)$ is exactly $B\mathbf{v}$.

Inverting the function involves recovering the integer coordinate $\mathbf{v}$ (or the error vector $\mathbf{e}$, since recovering one of them automatically gives you the other). However, the inversion is exactly the closest vector problem (CVP), and should be hard if B is a sufficiently bad basis. On the other hand, since R is a good basis, finding the closest vector point should be "easy" if we have R.

From here, GGH '97 proposed a public-key cryptosystem as well as a digital signature scheme that uses such a trapdoor construction, and the security of the two schemes naturally rest on the hardness of the underlying hard lattice problem.

3.1 The trapdoor scheme

The GGH trapdoor scheme contains four components:

1. **Parameter generation** The main security parameter in this scheme is the number of dimensions n of the lattice. In the original paper, the authors claimed $n \approx 150$ is sufficient for making inverting the function without the private basis hard and $n \approx 250$ should be a safe choice for the foreseeable future.

2. **Key generation** First generate the "good" basis $R \in \mathbb{R}^{n \times n}$, then apply some (unimodular) transformation to R to obtain the "bad" basis B. The error bound $\sigma > 0 \in \mathbb{R}$ is dependent on the choice of R and the choice of "probability of inversion error" $\epsilon >= 0$, which will be discussed in a later section.

3. **Forward evaluation** $f_B(\mathbf{v}, \mathbf{e}) = B\mathbf{v} + \mathbf{e}$, where $\mathbf{v} \in \{-n, \ldots, n\}^n$ and $\mathbf{e} \in \{-\sigma, \sigma\}$. According to the authors, the choice of bounds for values of $\mathbf{v}$ is arbitrary and not a significant contributor to the overall security of the scheme.

4. **Inversion** Denote the output by $\mathbf{x} = f_B(\mathbf{v}, \mathbf{e})$, first attempt to recover the integer coordinate $\mathbf{v} \leftarrow B^{-1}R\lfloor R^{-1}\mathbf{x}\rceil$. From here it is trivial to recompute the lattice point $B\mathbf{v}$ and recover the error term $\mathbf{e} = \mathbf{x} - B\mathbf{v}$.

3.2 Correctness of trapdoor inversion

Without the error term, the function $f_B : \mathbf{v} \mapsto B\mathbf{v}$ is trivially invertible with either choice of the basis. However, with a non-zero the error term, the quality of the basis makes a substantial difference in how much error can be added before the points can no longer be recovered.

Observe the calculation used for recovering the integer coordinate $\mathbf{v}$:

$$\begin{aligned}\lfloor R^{-1}\mathbf{x}\rceil &= B^{-1}R\lfloor R^{-1}(B\mathbf{v} + \mathbf{e})\rceil \\ &= B^{-1}R\lfloor R^{-1}B\mathbf{v} + R^{-1}\mathbf{e}\rceil\end{aligned}$$

Since R, B are related by a unimodular matrix and $\mathbf{v}$ is an integer vector, $R^{-1}B\mathbf{v}$ is an integer vector and can be moved out of the "rounding" operator:

$$\begin{aligned}\lfloor R^{-1}\mathbf{x}\rceil &= B^{-1}RR^{-1}B\mathbf{v} + B^{-1}R\lfloor R^{-1}\mathbf{e}\rceil \\ &= \mathbf{v} + B^{-1}R\lfloor R^{-1}\mathbf{e}\rceil\end{aligned}$$

Since $B^{-1}R$ is also a unimodular matrix, we can conclude that the equation above is successful at recovering the original coordinate if and only if $R^{-1}\mathbf{e} = \mathbf{0}$.

To guarantee that inversion error never happens, we can bound the error term $\sigma > 0$ by $\frac{1}{2\rho}$, where ρ is maximal L_1 norm among the rows of R^{-1}. This bound is excessively conservative, however, and we might want to relax the bound to enhance the security of the trapdoor scheme (larger error terms makes it harder to invert the function using only the public basis). The authors provided one such relaxation based on the Hoeffding inequality. This relaxation is stated as follows

$$P(\text{inversion error}) \leq 2n \cdot \exp(-\frac{1}{8\sigma^2\gamma^2})$$

Where $\gamma = \sqrt{n} \cdot \max(L_\infty \text{ norm of rows of } R^{-1})$. Simple reorganization of inequality shows that to bound the inversion error by ϵ, the error term will be bounded by $\sigma \leq (\gamma\sqrt{8 \ln 2n/\epsilon})^{-1}$.

3.3 Generating the pair of basis

The authors described two ways of generating the private basis. The first way is to sample each coordinate of $R \in \mathbb{R}^{n\times n}$ from a uniform distribution on $\{-l, -l+1, \ldots, l-1, l\}$, where according to the authors, the value of the bound l has negligible impact on the quality of the generated basis (the authors chose ±4 in their implementation). A second method is to first generate a square lattice $L(kI)$ for some positive integer k, then add a small amount of noise $R' \in \{-l, \ldots, l\}^{n\times n}$. With this method of sampling the private basis, it is important to balance the choice of values between k and l, where a larger k value gives a more orthogonal basis, but also weakens the security of the trapdoor function by making it easier to reduce the public basis into a short, orthogonal basis using basis reduction algorithm.

The authors also described two methods for generating the public basis B from the private basis. The first method is to directly generate random unimodular matrix T and set $B = TR$, then repeat until satisfactory. A second method is to repeatedly apply column mixing, where at each mixing a column of R is chosen, and a linear combination of all other columns is added to the chosen column.

Mathematically, the two methods are equivalent. However, in implemenetation, we would like the values of B to be smaller for space efficiency while maintaining sufficient security so that the function cannot be easily inverted using the public basis alone and so that B cannot be easily reduced using basis reduction algorithm. The authors preferred column mixing for requiring less computation and for producing public basis B with smaller values.

Unfortunately, there is no known rigorous description of how skewed B needs to be for the trapdoor function to be secure. The authors relied on experimental methods and determined that for $n \approx 100$, $2n$ steps of column mixing is enough to render LLL basis reduction ineffective at meaningfully improving the quality of the public basis B.

In 2001, Daniele Micciancio proposed to generate the public basis by computing the Hermite normal form of the private basis[5]. From a high level, Hermite normal form provides an optimal security guarantee because it is an invariant of the lattice, so a public basis computed from the Hermite normal form is guaranteed to give exactly zero information about the private basis. We will not discuss the details of Hermite normal in this survey.

3.4 Security of the trapdoor

It is easy to see that if an adversary can invert the trapdoor function without with only the public basis, then the adversary has found the closest vector $\mathbf{v} \in \mathcal{L}$ to the target $\mathbf{x}$. In other words, the adversary can solve the (approximate) closest vector problem.

As discussed in section 2, a polynomial-time LLL basis reduction algorithm only provides an approximation within an exponential factor; while it is possible to achieve approximation within a polynomial factor, the basis reduction algorithm is no longer guaranteed to run in polynomial time. In addition, the complexity of both LLL basis reduction and Babai's nearest plane algorithm scales exponentially with the security parameter n, and in experimental settings, the authors found that for $n \geq 100$ the workload for these algorithms starts becoming infeasible. The authors speculated that $n = 250$ should provide sufficient security in future production usage.

3.5 Public key cryptosystem

The authors of the GGH paper presented a public key cryptosystem that makes direct usage of the trapdoor [2]. The main security parameter is n the number of dimension of the lattice. The secret key is the short, orthogonal basis R generated by applying a small amount of noise to a truly orthogonal basis. The public key is the long basis B generated by applying column mixing to R. The encryption function is exactly the evaluation of the trapdoor, and the decryption function is the inversion of the trapdoor.

What remains is the question of "where to encode the message", for which the authors discussed a few options:

1. Use a generic encoding that takes advantage of the hard-core bits of the one-way function, although this encoding is inefficient (it only encodes $\log n$ bits at a time), and it does not take advantage of any specific features of the trapdoor construction itself
2. Directly encode the message into a lattice point $\mathbf{v}$. However, this scheme is insecure because an adversary can compute $B^{-1}\mathbf{c} = \mathbf{v} + B^{-1}\mathbf{e}$, where $B^{-1}\mathbf{e}$ might not be big enough to obscure all information about $\mathbf{v}$, which means this adversary can obtain partial information about the message
3. Encode the message into the least significant bits of each entry of $\mathbf{v}$ and choose all other bits randomly. The authors argued that with appropriate choice of probability distribution for the error term $\mathbf{e}$, no polynomial-time adversaries will be able to distinguish the parity of the entries of the ciphertext from truly random, thus achieving IND-CPA security.

Later Micciancio also proposed to encode the message in the error term $\mathbf{e}$, and instead chooses the lattice point $\mathbf{v}$ at random [5]. However, this scheme is not trivial because special care is needed for security requirement, and we also have the problem of a lattice being a countably infinite set, making defining probability distribution a non-trivial problem.

3.6 Digital signature

The GGH trapdoor is also suitable for a digital signature: the message is an arbitrary point in the linear span of the good basis, and the signature is the (approximate) closest lattice point, which can be efficiently computed using the good basis R. The verifier checks that the signature is valid by first using the public basis to verify that the signature is a lattice point, then computing the norm between the signature and the message to check that the distance is sufficiently small.

4 Cryptanalysis of GGH signatures

While the GGH trapdoor is based on CVP, a known hard lattice problem, the public-key cryptosystem and the signature schemes proposed by GGH in their original form are not given formal security proofs. Indeed, in 1999 the proposed public-key cryptosystem was cryptanalyzed by Phong Q. Nguyen [7], who successfully solved the GGH encryption challenges posed by the original authors. Later, Nguyen and Oded Regev published cryptanalysis of the GGH signature scheme [8], which exploited the signature scheme's design flaws that caused message-signature pairs to leak information about the secret key.

In this section we will discuss the cryptanalysis of the GGH signature scheme by first summarizing the key results from Nguyen and Regev, then provide an implementation that simulate the key-recovery attack.

4.1 Learning a hidden parallelepiped

As Nguyen and Regev pointed out, the main weakness of the GGH signature scheme lies in the fact that messages are hashed uniformly into (a finite subset) of $\mathbb{Z}^n$ and each signature is exactly the unique lattice point $\mathbf{v} \in \mathcal{L}(R)$ (where R is the secret basis) whose centered fundamental parallelepiped $\mathbf{v} + \mathcal{C}(R) = \{\mathbf{v} + R\mathbf{x} \mid \mathbf{x} \in [-\frac{1}{2}, \frac{1}{2})\}$ contains the hashed message. This means that for each message-signature pair $(\mathbf{m}, \sigma)$, their difference $\mathbf{m} - \sigma$ is a uniform sample of the centered fundamental parallelepiped $\mathcal{C}(R)$. The cryptanalysis of the GGH signature scheme is thus an algorithm that approximates the basis spanning $\mathcal{C}(R)$, which Nguyen and Regev termed the *hidden parallelepiped problem* (HPP).

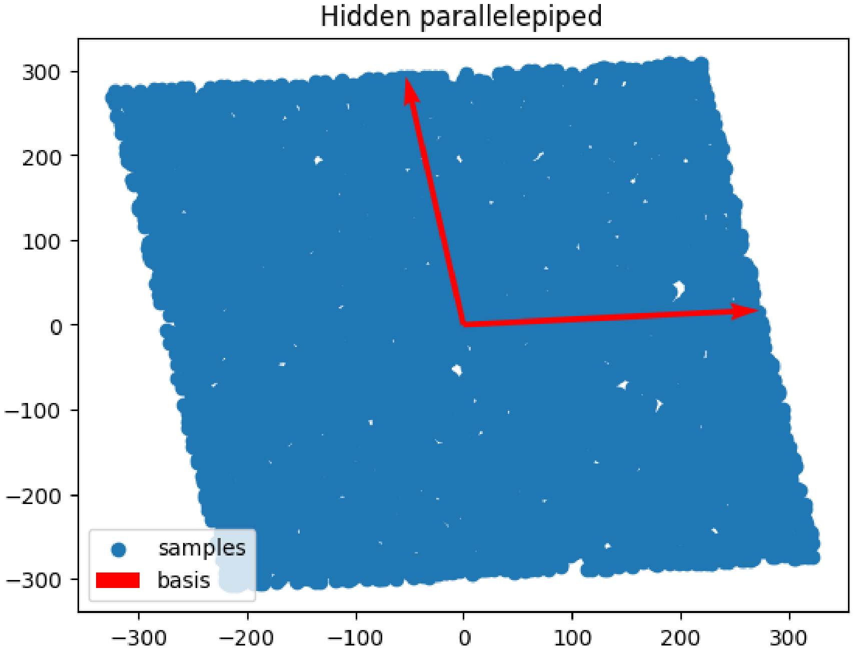

Figure 1: message-signature pairs reveal the parallelepiped spanned by the secret basis

Definition 4.1. *let $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$ be a full-rank matrix in $\mathbb{R}^{n \times n}$ and let $\mathcal{P}(V) = \{V\mathbf{x} \mid \mathbf{x} \in [-1, 1]^n\}$ denote the parallelepiped spanned by V. The **hidden parallelepiped problem** asks to approximate the vectors of V using a polynomial-bound number of uniform samples in the parallelepiped*

The main algorithm for solving HPP first approximates the Gram matrix of the basis of the parallelepiped, which can then be used to approximate a linear transformation that maps the parallelepiped to a hypercube (which is a parallelepiped with an orthonormal basis). The basis of the hypercube can be approximated using a gradient descent, and the basis of the parallelpiped can be computed by inverting the hypercube transformation.

Learning the basis of a hypercube can be efficiently done through a gradient descent on the fourth moment of a uniform distribution on the hypercube:

Definition 4.2. *Given basis $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbf{r}^{n \times n}$, define the k-th moment of a uniform distribution on $\mathcal{P}(V)$ over a vector $\mathbf{w} \in \mathbb{R}^n$ as*

$$\mathrm{mom}_{V,k}(\mathbf{w}) = \exp[\langle \mathbf{u}, \mathbf{w} \rangle^k]$$

Where $\mathbf{u}$ *is uniform sample from* $\mathcal{P}(V)$

Gradient descent on the fourth moment of the hypercube is an appropriate algorithm due to the following lemma:

Lemma 4.1. *Given basis* $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n \times n}$ *of a unit hypercube, there exists a unique global minimum achieved at* $\pm\mathbf{v}_1, \pm\mathbf{v}_2, \ldots, \pm\mathbf{v}_n$ *and no other local minimum*

Algorithm 5 Learning a hidden parallelepiped

```
Approximate $G' \approx V^\intercal V$
Compute the Cholesky factor $L$ of $G^{-1}$: $G^{-1} = LL^\intercal$
Transform the parallelepiped samples into hypercube samples $\mathbf{x} \in \mathcal{P}(V) \mapsto \mathbf{x}L$
Approximate the basis of the hypercube $C' = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n]$
Transform the hypercube basis back to parallelepiped basis $\mathbf{v} \leftarrow \mathbf{c}L^{-1}$
```

Algorithm 6 Learning a hidden hypercube

```
Choose some initial unit vector for $\mathbf{w}$
while True do
    Approximate the gradient $\nabla\,\mathrm{mom}_{V,4}(\mathbf{w})$ of the fourth moment at $\mathbf{w}$
    Denote $\mathbf{w}' \leftarrow \mathbf{w} - \delta\nabla\,\mathrm{mom}_{V,4}(\mathbf{w})$, then normalize $\mathbf{w}'$
    if $\mathrm{mom}_{V,4}(\mathbf{w}') > \mathrm{mom}_{V,4}(\mathbf{w})$ then
        return $\mathbf{w}$
    else
        Replace $\mathbf{w}$ with $\mathbf{w}'$
    end if
end while
```

4.2 Experimental results

The GGH signature scheme and the HPP learning algorithm were implemented using Python. The matrix arithmetics, including dot products, matrix inversion, and Cholesky factorization, are all computed using numpy 1.26.2. A demo of the cryptanalysis is available on GitHub [1].

In this implementation, the secret basis is generated using the formula $R \leftarrow kI + R'$, where R' is uniformly sampled from $\{-l, -l+1, \ldots, l-1, l\}^{n \times n}$, $l = 100$, and $k = \lfloor \sqrt{n} \cdot l \rceil$. Messages are uniformly sampled from $\mathbf{m} \leftarrow \{-99999, \ldots, 99999\}^n$, and signatures are computed using simple rounding algorithm $\sigma \leftarrow R\lfloor R^{-1}\mathbf{m} \rceil$. Note that we can also use nearest plane to compute the signature, although with that we will be learning the basis of the orthogonal parallelepiped spanned by B^*.

[1] https://github.com/xuganyu96/lattice-crypto-notes/blob/main/gghattack/demo.ipynb

Following the recommendation of Nguyen and Regev, the rate of the gradient descent is set to $\delta = 0.7$. For initial positions of $\mathbf{w}$, we iterated through the columns of the identity matrix I_n, and we found that they each descended into a distinct basis. In other words, running gradient descent on each of these initial positions allowed us to approximate all secret basis vectors.

While this implementation did not recover the exact value of the secret basis, the approximated secret basis is numerically very close to the true secret basis. We meansured the quality of the basis by computing the determinant of $B^{-1}B'$ where B' is the approximated basis: the closer this determinant is to 1, the better B' is an approximation of B.

On an M1 MacBook Air, approximating the secret basis at dimension $n = 80$ with 200000 pairs of message-signature pair could be done in only a few minutes. I could not reliably approximate secret basis of higher dimensions due to integer overflow limitations of numpy, though using C++ with appropriate big integers, the original authors were able to recover secret basis with more than 400 dimensions using 200000 samples.

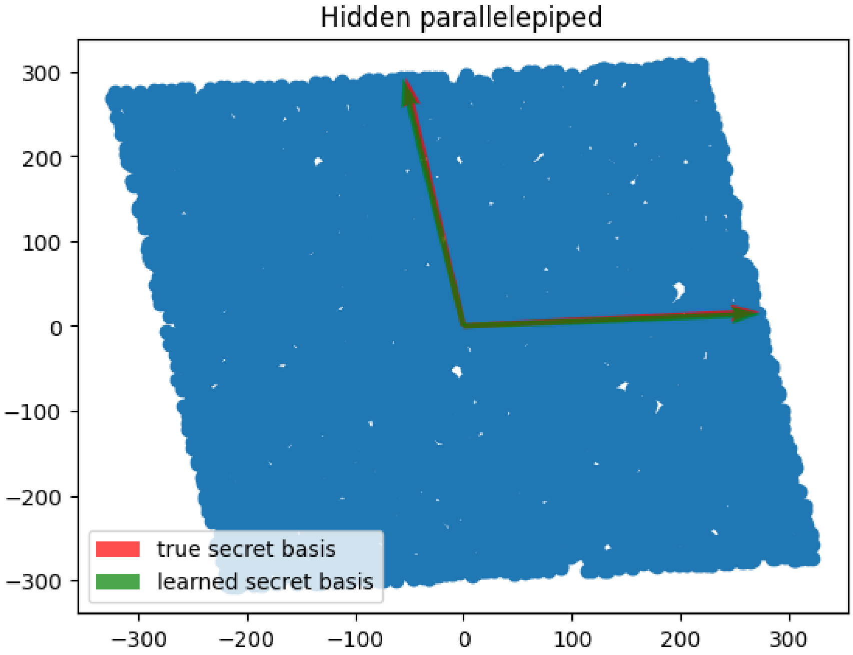

Figure 2: Recovering a strong approximation of the secret basis

5 Conclusion and final comments

In this project, we reviewed some fundamental results of lattices, analyzed the difficulty of hard lattice problems and the efficiency of their best known solutions. We then discussed a lattice trapdoor construction that takes advantage of an exponential gap between short basis and long basis in their ability to solve the closest vector problem, and applied the trapdoor to build a public key cryptosystem and a digital signature scheme.

Both the public-key cryptosystem and the signature scheme have been cryptanalyzed [7][8] in their original proposed form. However, the idea of using a pair of short and long basis to build trapdoors has been extended in later studies. Gentry, Peikert, and Vaikuntanathan developed this idea into the GPV framework[1] that later led to FALCON, a digital signature scheme that was submitted to NIST's post-quantum cryptography competition. This idea was also behind NTRUSign[3], whose post-quantum variant was also submitted to NIST's post-quantum cryptography competition.

References

[1] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[2] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 112–131. Springer, 1997.

[3] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Cryptographers' track at the RSA conference*, pages 122–140. Springer, 2003.

[4] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.

[5] Daniele Micciancio. Improving lattice based cryptosystems using the hermite normal form. In *International Cryptography and Lattices Conference*, pages 126–145. Springer, 2001.

[6] Daniele Micciancio. Cse206a: Lattices algorithms and applications, 2021. Accessed in November, 2023.

[7] Phong Nguyen. Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto'97. In *Annual International Cryptology Conference*, pages 288–304. Springer, 1999.

[8] Phong Q Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 271–288. Springer, 2006.

[9] Chris Peikert et al. A decade of lattice cryptography. *Foundations and trends® in theoretical computer science*, 10(4):283–424, 2016.

[10] Vinod Vaokuntanathan. Cs 294-168: Lattices, learning with errors, and post-quantum cryptography, 2020. Accessed in November, 2023.