POST-QUANTUM CRYPTOGRAPHY

MANU ISAACS

Contents

1. Intro

On a classical computer, the cryptosystems we use today are safe. But in 1994, Peter Shor changed the world forever by discovering an algorithm for solving the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP) and even the Elliptic Curve Discrete Logarithm Problem (ECDLP) in polynomial time on a quantum computer. Many of the cryptosystems we use today, like RSA and (Elliptic Curve) Diffie-Hellman depend on these problems for their security.

In this paper, we explore some of the ideas being developed in the relatively new field of Post-Quantum Cryptography (PQC). PQC anticipates Q-day, the day when quantum computers can solve IFP and (EC)DLP for cryptographically large numbers. When this day arrives, the world should be ready for it - we should all be using so-called *quantum safe cryptosystems*, which appear to be resistant to quantum computers.

In fact, there is an argument to switch to quantum safe cryptosystems as soon as possible,

Date: August 17, 2024.

because of the so-called *store now, decrypt later* attack. One can store encrypted information that is believed to be useful even after a long time, and then decrypt it on a quantum computer on Q-day. This is arguably the most pressing argument for the importance of PQC.

2. What Still Works?

As it turns out, there are many cryptosystems which appear to hold their own against quantum computers. Some examples are:

(1) AES (for $b \leq 128$),
(2) McEliece with code length $b^{1+o(1)}$,
(3) Merkle signatures with "strong" $b^{1+o(1)}$-bit hash,
(4) HFEv- with $b^{1+o(1)}$ polynomials,
(5) NTRU with $b^{1+o(1)}$ bits, etc.

according to [1]. In general, quantum computers appear to have a minimal effect on symmetric key cryptography and hash-based cryptography. This is because the fastest quantum algorithm for inverting a general function, called Grover's algorithm, provides only a square root speedup. This means that we achieve the same security simply by doubling the key length.

3. Challenges with PQC

But if this is true, why isn't post-quantum cryptography solved? There are 3 main reasons [1]: efficiency, confidence, usability.

- **Efficiency:** Many post-quantum cryptographic schemes are computationally more expensive than current ones. This poses a challenge for widespread adoption because the internet infrastructure isn't yet ready to handle the increased load.
- **Confidence:** While current post-quantum schemes appear resistant to known quantum attacks, there's always the possibility of new quantum algorithms emerging that could break them.
- **Usability:** Post-quantum cryptography introduces new complexities. Ensuring secure and practical implementations requires careful consideration of issues like padding and parameter choices. The same thing happened with RSA - the original RSA paper wasn't secure and developments like padding needed to be made.

However, systems like lattice based cryptography and multivariate-quadratic cryptography are new and promising proposals. In the remainder of the paper, we will discuss lattice-based cryptography.

4. Lattices and Lattice Problems

4.1. Lattice Basics.

What is a lattice? You might be familiar with the lattice of integer points on the Cartesian plane $\mathcal{L} = \mathbb{Z}^2$. More generally, we can define an n-dimensional lattice as follows.

Definition 4.1. For a given set of linearly independent basis vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n\} \in \mathbb{Z}^n$, define the lattice $\mathcal{L}$ as

$$\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z} \right\}.$$

Defining the $n \times n$ matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n]$, we can more concisely write

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{B}x : x \in \mathbb{Z}^n\}.$$

Example. The familiar lattice $\mathbb{Z}^2$ can be generated by the basis matrix

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

among others.

Next we will prove some basic results to do with lattices.

Definition 4.2. A matrix $\mathbf{U}$ is *unimodular* if it is an integer square matrix with determinant ± 1.

Theorem 4.3. *For all matrices* $\mathbf{B}$ *and unimodular matrices* $\boldsymbol{U}$*, (figure out how to make the U not be italicized here)*

$$\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}\boldsymbol{U}).$$

Proof. We will prove that $\mathcal{L}(\mathbf{B}) \subset \mathcal{L}(\mathbf{BU})$. If $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, then we can write $\mathbf{v} = \mathbf{Bx}$. Notice that because $\mathbf{U}^{-1}$ is unimodular, $\mathbf{x}' = \mathbf{U}^{-1}\mathbf{x} \in \mathbb{Z}^n$, so $\mathbf{v} = \mathbf{BUx}'$ hence $\mathbf{v} \in \mathcal{L}(\mathbf{BU})$. A similar argument shows $\mathcal{L}(\mathbf{BU}) \in \mathcal{L}(\mathbf{B})$ so we are done. ■

In fact, we can make an even stronger statement:

Theorem 4.4. *Bases* $\mathbf{B}_1$ *and* $\mathbf{B}_2$ *generate the same lattice if and only if there exists some unimodular matrix* $\boldsymbol{U}$ *such that* $\mathbf{B}_1 = \boldsymbol{U}\mathbf{B}_2$.

Proof. The "if" part of this statement is proven by 4.3. For the "only if" part, we can prove the contrapositive. Begin by assuming that $\mathcal{L}(\mathbf{B}_1) \neq \mathcal{L}(\mathbf{B}_2)$. Then, WLOG we can say there is some $\mathbf{x} \in \mathcal{L}(\mathbf{B}_1)$ such that $\mathbf{x} \notin \mathcal{L}(\mathbf{B}_2)$. We let $\mathbf{U} = \mathbf{B}_1\mathbf{B}_2^{-1}$, where we aim to show $\mathbf{U}$ is not unimodular. For some vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, we have

$$\begin{aligned} \mathbf{x} = \mathbf{B}_1\mathbf{v}_1 &\neq \mathbf{B}_2\mathbf{v}_2 \ \forall \mathbf{v}_2 \in \mathbb{Z}^n \\ \mathbf{B}_1\mathbf{B}_2^{-1}\mathbf{v}_1 &\neq \mathbf{v}_2 \ \forall \mathbf{v}_2 \in \mathbb{Z}^n \\ \mathbf{U}\mathbf{v}_1 &\notin \mathbb{Z}^n \end{aligned}$$

but this is impossible if $\mathbf{U}$ is unimodular, so we are done. ■

Example. Consider the basis

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

which corresponds to the basis vectors $\mathbf{v}_{11} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{v}_{12} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$. Let's try multiplying the basis by the unimodular matrix

$$\mathbf{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Then, we have

$$\mathbf{B}_2 = \mathbf{U}\mathbf{B}_1 = \begin{bmatrix} 1 & 4 \\ 2 & 6 \end{bmatrix}.$$

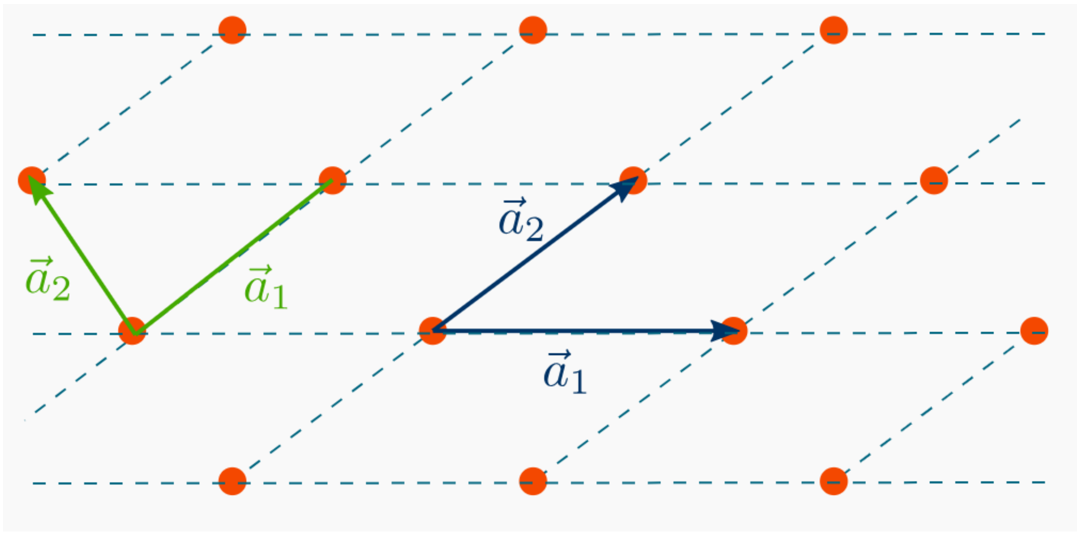


Figure 1. Two bases which generate the same matrix - $\mathbf{U} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}$.

Which corresponds to the basis vectors $\mathbf{v}_{21} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $\mathbf{v}_{22} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$. Notice that $\mathbf{v}_{21} = \mathbf{v}_{11}$ and $\mathbf{v}_{22} = \mathbf{v}_{11} + \mathbf{v}_{12}$. Drawing these four vectors gives a good intuitive understanding for why they generate the same lattice.

Example. See figure 1.

Let's define the determinant of a lattice:

Definition 4.5. The determinant of a lattice $\mathcal{L}$ is

$$d(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$$

There are a few important things to notice: firstly, note that the determinant of a lattice is independent of the choice of basis by 4.4 and the fact that all unimodular matrices $\mathbf{U}$ satisfy $|\det(\mathbf{U})| = 1$. Second, notice that the determinant of a lattice can be said to be equal to its inverse point density, because it is the absolute volume of the parallelopiped created by the basis vectors of $\mathcal{L}$.

4.2. **Hard Lattice Problems.**

In conventional cryptography, the trapdoor hard-to-invert problem is often the IFP or (EC)DLP, both of which can be broken by a quantum computer using Shor's algorithm. The following three problems are considered hard problems and provide the basis for Lattice-based cryptography.

All of these problems involve a notion of a norm of a vector. While different norms can be used, the following norm is most common:

Definition 4.6. Define the *norm*, *size*, or *magnitude* of a vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$ as $|\mathbf{v}| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$. This norm is called the Euclidean norm.

Remark 4.7. When we use words like "shortest" and "closest" we mean the vector with minimum norm.

Definition 4.8. Shortest Vector Problem (SVP): Given a lattice basis $\mathbf{B}$, compute the shortest nonzero vector in $\mathcal{L}(\mathbf{B})$.

While this problem seems easy in two dimensions, it appears to get exponentially harder in n, the number of dimensions.

Definition 4.9. Closest Vector Problem (CVP): Given a lattice basis $\mathbf{B}$ and vector $\mathbf{t}$, compute the vector closest to $\mathbf{v}$ in $\mathcal{L}(\mathbf{B})$.

While this problem isn't precisely a generalization of the SVP (because of the nonzero condition in the SVP), it is possible to relate the two problems. Specifically, if one has an oracle for solving the SVP, they can solve the CVP.

Definition 4.10. Shortest Independent Vectors Problem (SIVP): Given a lattice basis $\mathbf{B}$, find the n linearly independent lattice vectors $S = \begin{bmatrix}\mathbf{s}_1, \mathbf{s}_2, \dots \mathbf{s}_n\end{bmatrix}$ such that $\max_i \mathbf{s}_i$ is minimal.

It is more usual to use the approximate versions of these problems. They depend on some approximation factor γ sometimes written as $\gamma(n)$ because of its dependence on the dimension n:

Definition 4.11. The SVP_γ problem asks to find a vector of magnitude at most γ times the magnitude of the smallest nonzero vector in $\mathcal{L}$.

The definitions for CVP_γ and SIVP_γ are identical. The following conjecture [2] is what makes these lattice-based problems so useful in cryptography:

Conjecture 4.12. *There is no polynomial time algorithm (in the dimension n) to solve any approximate lattice problem, where the approximation factor γ is polynomial in n, on a classical or quantum computer.*

This conjecture has some evidence to loosely back it up; there has been lots of relatively recent improvement on the problem of factoring: the continued fractions factoring algorithm in 1980, the quadratic sieve factoring algorithm in 1990, and the number field sieve in 1996 [3]. In contrast, there has been no significant improvement on this problem since the 1980s [2].

5. The LLL Lattice Reduction Algorithm

How do we solve Lattice problems? One method is a lattice reduction algorithm. Lattice reduction algorithms aim to reduce the length of the basis, which is defined as follows:

Definition 5.1. The *length* of a basis $\mathbf{B} = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_n\end{bmatrix}$ is $\max_i |\mathbf{b}_i|$.

A small basis is the solution to the SIVP, which can also be efficiently used to solve the SVP and CVP. The LLL lattice reduction algorithm [4] is the first polynomial time Lattice reduction algorithm. It solves SVP and other lattice problems with $\gamma(n) = 2^{O(n)}$ in $O(n^6)$ bit operations [2]. To understand it, we'll first need some ideas from linear algebra.

5.1. Linear Algebra Preliminaries.

Definition 5.2. Given $\mathbf{v}, \mathbf{w} \in \mathbb{Z}^n$, let the *standard inner product* or *dot product* of $\mathbf{v} = \begin{bmatrix}v_1 & v_2 & \dots & v_n\end{bmatrix}$ and $\mathbf{w} = \begin{bmatrix}w_1 & w_2 & \dots & w_n\end{bmatrix}$, denoted by $\mathbf{v} \cdot \mathbf{w}$, be

$$\mathbf{v} \cdot \mathbf{w} = \sum_i v_i w_i.$$

Remark 5.3. Notice that $\mathbf{v} \cdot \mathbf{v} = |\mathbf{v}|^2$.

Definition 5.4. We say vectors $\mathbf{v}$ and $\mathbf{w}$ are orthogonal if $\mathbf{v} \cdot \mathbf{w} = 0$. A set of vectors $\mathbf{v}_i$ is orthogonal if $\mathbf{v}_i \cdot \mathbf{v}_j = 0$ for $i \neq j$.

How do we project a vector $\mathbf{v}$ onto a vector $\mathbf{w}$? Intuitively, the projection $\mathbf{proj}_{\mathbf{w}}(\mathbf{v})$ should be parallel to $\mathbf{w}$ and $\mathbf{v} - \mathbf{proj}_{\mathbf{w}}(\mathbf{v})$ should be orthogonal to $\mathbf{w}$. It is easy to check that the following definition satisfies these two conditions.

Definition 5.5. The projection of $\mathbf{v}$ onto $\mathbf{w}$ is

$$\mathbf{proj}_{\mathbf{w}}(\mathbf{v}) = \frac{\mathbf{v} \cdot \mathbf{w}}{\mathbf{w} \cdot \mathbf{w}}\mathbf{w}.$$

5.2. **Gram–Schmidt Orthogonalization.** The Gram-Schmidt orthogonalization process, or just Gram-Schmidt process, converts a set of basis vectors $\mathbf{v}_i$ to an orthonormal basis $\mathbf{u}_i$ which has the same span over $\mathbb{R}^n$ (but not $\mathbb{Z}^n$ in general). In English, we can describe it as follows:

(1) Project $\mathbf{v}_i$ onto the orthogonal subspace U generated by $\mathbf{u}_1, \mathbf{u}_2 \dots \mathbf{u}_{i-1}$.
(2) Let $\mathbf{u}_i$ be the difference between $\mathbf{v}_i$ and this projection so that $\mathbf{u}_i$ is orthogonal to $\mathbf{u}_1, \mathbf{u}_2 \dots \mathbf{u}_{i-1}$.

Notice that this process generates an orthogonal subspace by induction on i. Algebraically, the algorithm can be written as

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1 \\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathbf{proj}_{\mathbf{u}_1}\mathbf{v}_2 \\
&\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \mathbf{proj}_{\mathbf{u}_j}(\mathbf{v}_k).
\end{aligned}$$

Equivalently, in the language of [4]:

Definition 5.6. For a given basis $\{\mathbf{b}_i\}$ define the Gram-Schmidt orthogonalization as $\mathbf{b}_i^*$

$$\mathbf{b}_i^* = \mathbf{b}_i - \textstyle\sum_{j=1}^{i-1} \mu_{ij}\mathbf{b}_j^*, \tag{5.1}$$

$$\mu_{ij} = \frac{\mathbf{b}_i \cdot \mathbf{b}_j^*}{\mathbf{b}_j^* \cdot \mathbf{b}_j^*}. \tag{5.2}$$

So $\mathbf{b}_i^*$ is an orthogonal basis of $\mathbb{R}^n$.

5.3. **The Algorithm.** For this algorithm, we'll need a definition of what exactly we mean by a reduced basis.

Definition 5.7. We call a basis $\{\mathbf{b}_i\}$ *reduced* if

$$|\mu_{ij}| \leq 1/2 \quad \text{for} \quad j < i \tag{5.3}$$

$$|\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*|^2 \geq \frac{3}{4}|\mathbf{b}_{i-1}^*|^2 \quad \text{for} \quad 1 < i \leq n \tag{5.4}$$

where the constants are arbitrarily chosen.

What makes this basis reduced? Intuitively, the first equation means that $\mathbf{b}_i$ is *close* to perpendicular to $\mathbf{b}_j^*$ for $j \neq i$. The second equation essentially ensures that one vector isn't too much smaller than the next one.

More rigorously, we prove the following facts about a reduced basis:

Theorem 5.8. *For a reduced basis $\{\mathbf{b}_i\}$ of lattice $\mathcal{L}$ and corresponding Gram-Schmidt orthogonalization $\mathbf{b}_i^*$ given by 5.6,*

$$|\mathbf{b}_j|^2 \leq 2^{i-1}|\mathbf{b}_i^*|^2 \quad \textit{for} \quad j \leq i, \tag{5.5}$$

$$d(\mathcal{L}) \leq \prod |b_i| \leq 2^{n(n-1)/4} d(\mathcal{L}), \tag{5.6}$$

$$|b_1| \leq 2^{(n-1)/4} d(\mathcal{L})^{1/n}. \tag{5.7}$$

Proof. In equation 5.4, notice that $\mathbf{b}_i^*$ and $\mathbf{b}_{i-1}^*$ are perpendicular, so we can apply the Pythagorean theorem:

$$|\mathbf{b}_i^*|^2 + \mu_{i,i-1}^2 |\mathbf{b}_{i-1}^*|^2 \geq \frac{3}{4}|\mathbf{b}_{i-1}^*|^2.$$

Plugging in equation 5.3 we get

$$|\mathbf{b}_i^*|^2 \geq \frac{1}{2}|\mathbf{b}_{i-1}^*|^2.$$

Inducting on i, we have

$$|\mathbf{b}_j^*|^2 \leq 2^{i-j}|\mathbf{b}_i^*|^2 \quad \text{for} \quad j \leq i. \tag{5.8}$$

Shifting our focus to equation 5.1, we can again use the fact that the $\mathbf{b}_i^*$ are orthogonal to write, by the Pythagorean theorem,

$$|\mathbf{b}_i|^2 = |\mathbf{b}_i^*|^2 + \sum_{j=1}^{n} \mu_{ij}^2 |\mathbf{b}_j^*|^2.$$

Plugging in the bounds 5.3 and 5.8 to the RHS, we have

$$|\mathbf{b}_i|^2 \leq |\mathbf{b}_i^*|^2 + \sum_{j=1}^{n} \frac{1}{4} 2^{i-j} |\mathbf{b}_i^*|^2.$$

which can be simplified using geometric series to

$$|\mathbf{b}_i|^2 \leq 2^{i-1}|\mathbf{b}_i^*|^2. \tag{5.9}$$

This is almost 5.5. We just need to incorporate equation 5.8:

$$|\mathbf{b}_j|^2 \leq 2^{j-1}|\mathbf{b}_j^*|^2 \leq 2^{i-1}|\mathbf{b}_i^*|^2$$

so we have proven equation 5.5.

Using definition 4.5 and the fact that $\mathbf{b}_i^*$ is orthogonal, we have

$$d(\mathcal{L}) = \det\left(\begin{bmatrix} \mathbf{b}_1^* & \mathbf{b}_2^* & \dots & \mathbf{b}_n^* \end{bmatrix}\right) = \prod_{i=1}^{n} \mathbf{b}_i^*$$

Because a projection of a vector cannot be of greater magnitude than the original vector, we can write $|\mathbf{b}_i^*| \leq |\mathbf{b}_i|$, and adding equation 5.9, we have

$$|\mathbf{b}_i^*| \leq |\mathbf{b}_i| \leq 2^{i-1}|\mathbf{b}_i^*|^2.$$

Taking the product over i gives equation 5.6. Finally, setting $j = 1$ in 5.5 and taking the product over i gives equation 5.7, so we are done. ∎

Now, we will show that if we can find a reduced basis (by our definition), we've solved SVP_γ for $\gamma(n) = O(2^{n/2})$.

Theorem 5.9. *Let a lattice $\mathcal{L}$ have a reduced basis $\{\mathbf{b}_i\}$. Then, for all nonzero $x \in \mathcal{L}$,*

$$|\mathbf{b}_1|^2 \leq 2^{n-1}|x|^2.$$

Remark 5.10. By taking the square root of this equation, we see why $\gamma(n) = O(2^{n/2})$.

Proof. We can write

$$x = \sum r_i \mathbf{b}_i = \sum r_i^* \mathbf{b}_i^*$$

for $r_i \in \mathbb{Z}$ and $r_i^* \in \mathbb{R}$. Because x is nonzero, not all r_i can be zero. Thus, we can let i be the largest index such that $r_i \neq 0$, from which $r_i^* = r_i$, by definition 5.6 of Gram-Schmidt orthogonalization. So for this value of i,

$$|x|^2 \geq r_i^* |\mathbf{b}_i^*|^2 \geq |\mathbf{b}_i^*|^2.$$

But by 5.5, we have $|\mathbf{b}_1|^2 \leq 2^{i-1}|\mathbf{b}_i^*|^2 \leq 2^{i-1}|\mathbf{b}_i^*|^2$, so combining these two inequalities, we are done. ∎

So now we know that producing a reduced basis is enough to solve an exponential approximation of the SVP. But how does the algorithm itself work? At a high level, we start with a reduced basis of dimension 2, and add elements inductively until we achieve a basis of dimension n. Each time we add an element, we have to modify the basis to ensure that it is reduced again. Figure 2 shows the pseudocode for this algorithm.

6. Lattice-Based Cryptography

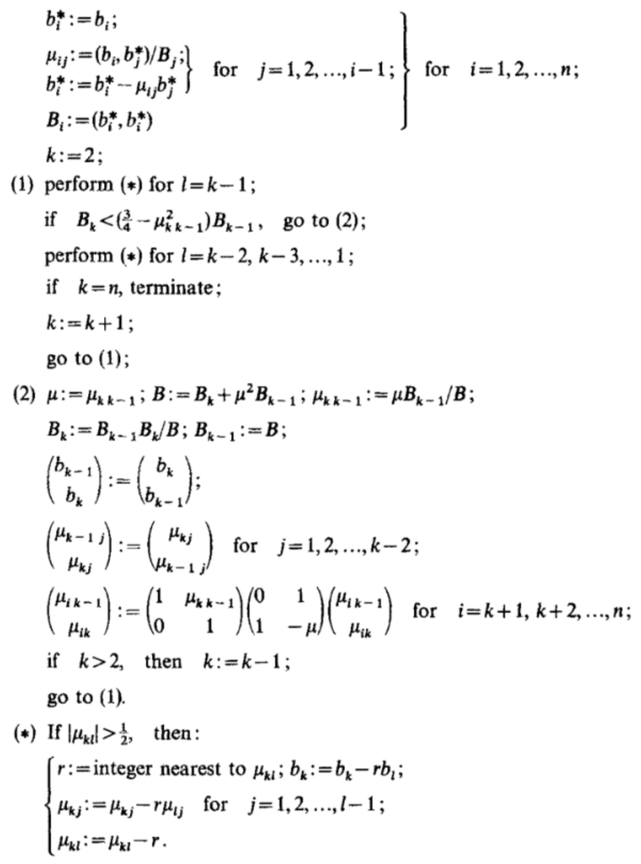
Now that we know some difficult problems in Lattice-Based Cryptography and how to solve them (admittedly, only with exponentially bad accuracy), let's take a look at some cryptosystems. Lattice based cryptosystems can be divided into two categories: provably secure but inefficient and efficient but not provably secure [2]. Lets take a look at a cryptosystem which falls into the first category.

6.1. A Provably Secure but Inefficient Cryptosystem - Ajtai and Dwork.

In a breakthrough paper by Ajtai and Dwork [5], a cryptosystem was developed which is provably at least as secure as SVP. That is, in order to break the cryptosystem, a hacker *must* first solve the SVP. What follows is a high level outline of the cryptosystem, then we will dig into some of the details.

Suppose that Bob wants to send a message to Alice. The following steps outline how this would happen.

1. Alice creates her private key, a collection of hyperplanes in n dimensional space along with a lattice of a specific type.
2. Alice publishes a public key, a method for how to encrypt 0's and 1's.
3. Bob encrypts his messages bit by bit. If he wants to encrypt a 1 he picks a lattice point randomly and uniformly from a large cube. If he wants to encrypt a 0 he picks a lattice point close to one of the hyperplanes (in a way described by the public key).



```
    $b_i^* := b_i$;                                                          ⎫
    $\mu_{ij} := (b_i, b_j^*)/B_j$;   ⎫                                      ⎪
    $b_i^* := b_i^* - \mu_{ij} b_j^*$ ⎭ for $j = 1, 2, \ldots, i-1$;         ⎬ for $i = 1, 2, \ldots, n$;
    $B_i := (b_i^*, b_i^*)$                                                  ⎭
    $k := 2$;
(1) perform (*) for $l = k-1$;
    if $B_k < (\frac{3}{4} - \mu_{k\,k-1}^2) B_{k-1}$, go to (2);
    perform (*) for $l = k-2, k-3, \ldots, 1$;
    if $k = n$, terminate;
    $k := k+1$;
    go to (1);
(2) $\mu := \mu_{k\,k-1}$; $B := B_k + \mu^2 B_{k-1}$; $\mu_{k\,k-1} := \mu B_{k-1}/B$;
    $B_k := B_{k-1} B_k / B$; $B_{k-1} := B$;
    $\begin{pmatrix} b_{k-1} \\ b_k \end{pmatrix} := \begin{pmatrix} b_k \\ b_{k-1} \end{pmatrix}$;
    $\begin{pmatrix} \mu_{k-1\,j} \\ \mu_{kj} \end{pmatrix} := \begin{pmatrix} \mu_{kj} \\ \mu_{k-1\,j} \end{pmatrix}$ for $j = 1, 2, \ldots, k-2$;
    $\begin{pmatrix} \mu_{i\,k-1} \\ \mu_{ik} \end{pmatrix} := \begin{pmatrix} 1 & \mu_{k\,k-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu \end{pmatrix} \begin{pmatrix} \mu_{i\,k-1} \\ \mu_{ik} \end{pmatrix}$ for $i = k+1, k+2, \ldots, n$;
    if $k > 2$, then $k := k-1$;
    go to (1).
(*) If $|\mu_{kl}| > \frac{1}{2}$, then:
    ⎧ $r :=$ integer nearest to $\mu_{kl}$; $b_k := b_k - r b_l$;
    ⎨ $\mu_{kj} := \mu_{kj} - r\mu_{lj}$ for $j = 1, 2, \ldots, l-1$;
    ⎩ $\mu_{kl} := \mu_{kl} - r$.
```

Figure 2. Pseudocode for the LLL Algorithm

(4) Alice computes the distance from Bob's point is to any hyperplane using her private key and determines if it should be decrypted as a 0 or a 1.

There's a lot to unpack here. Is this secure (i.e. can some eavesdropper Eve gain information based on the public information)? Can Alice ever decrypt messages incorrectly?

Let's tackle these questions one by one. The following theorem answers the question of security:

Theorem 6.1. *An eavesdropper Eve being able to decrypt messages implies her ability to solve the SVP (in the worst case or in a random case, depending on the exact variant of the cryptosystem.*

Proof. See [5]. ■

This means that if one believes conjecture 4.12, then one believes this cryptosystem is secure.

To answer the second question, it turns out that there is a small chance that a 1 is decrypted incorrectly as a 0, and this chance tends to 0 as $n \to \infty$.

Some more specific questions come to mind - what kinds of lattices $\mathcal{L}$ can we use? What is a hyperplane? How does Bob know how to encrypt his messages without knowing the hyperplanes? What exactly is "close"? For a much more detailed explanation answering all of these questions, see [5].

6.2. **An Efficient but not Secure Cryptosystem - GGH.** While GGH is no longer a practically secure cryptosystem, it is still worth looking at because of its simplicity.

GGH is an asymmetric key cryptosystem developed by Goldreich, Goldwasser, and Halevi.

- **Public key:** A "bad" basis $\mathbf{B}_1$ for a lattice $\mathcal{L}$
- **Private key:** A "good" basis $\mathbf{B}_2$ for the same lattice $\mathcal{L}$
- **Encryption:** Pick a vector $\mathbf{v} \in \mathcal{L}$ and encode your message as some noise vector $\mathbf{r}$. Then, $\mathbf{v} + \mathbf{r}$ is the ciphertext
- **Decryption:** Solve the CVP for the vector $\mathbf{v} + \mathbf{r}$. This is easy if one has access to the private key, which is a "good" basis, but hard if one only has access to the public key. This gives $\mathbf{v}$, from which we can compute the plaintext $(\mathbf{v} + \mathbf{r}) - \mathbf{v} = \mathbf{r}$.

What exactly do we mean by good and bad? A good basis is a reduced basis, while a bad basis is a random basis, which is hard to reduce (this is lattice reduction). The key is that it's easy to start with a reduced basis and generate some bad basis to create the keys.

6.3. **Lattice-Based Hash Functions.** Before moving on to an efficient Lattice-Based Cryptosystem which is believed to be secure, let's take a small detour and explore the world of hash functions. A cryptographic hash function is a function f with the following properties:

- The probability of a particular n-bit output result $f(x)$ (called the hash value) for a random input x is 2^{-n}
- Solving for x given $y = f(x)$ (a pre-image) is computationally unfeasible, assuming all input strings are equally likely. How long it takes to perform such an inversion is called security strength.
- Finding some distinct x, y such that $f(x) = f(y)$ (a collision) is also computationally unfeasible. This is called collision resistance.

We can build a simple hash functions using lattices. For example, consider the following hash function:

- **Parameters:** n,m,q,d
- **Key:** $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$
- **Hash Function:** $f_A : \{0, 1, \ldots d - 1\}^m \to \mathbb{Z}_q^n$ given $f_A(\mathbf{v}) = \mathbf{A}\mathbf{v}$

This has function has the advantage that finding collisions is provably as hard as solving lattice problems in the worst case. However, this hash function is inefficient in practice. One way around this problem is to use circulant matrices.

Definition 6.2. A circulant matrix $\mathbf{A}$ has columns which are circular shifts of each other. Specifically, letting

$$\mathbf{T} = \left[\begin{array}{c|c} \mathbf{0}^T & 1 \\ \hline \mathbf{I} & \mathbf{0} \end{array}\right],$$

we write $\mathbf{A} = \mathbf{T}^*\mathbf{v} = \begin{bmatrix}\mathbf{v} & \mathbf{v}\mathbf{T} & \mathbf{v}\mathbf{T}^2 & \ldots & \mathbf{v}\mathbf{T}^{n-1}\end{bmatrix}$.

Using a circulant matrix in the previous hash function allows us to reduce space from $O(n^2)$ to $O(n)$ because A is only defined by one vector. It also allows us to reduce the complexity of

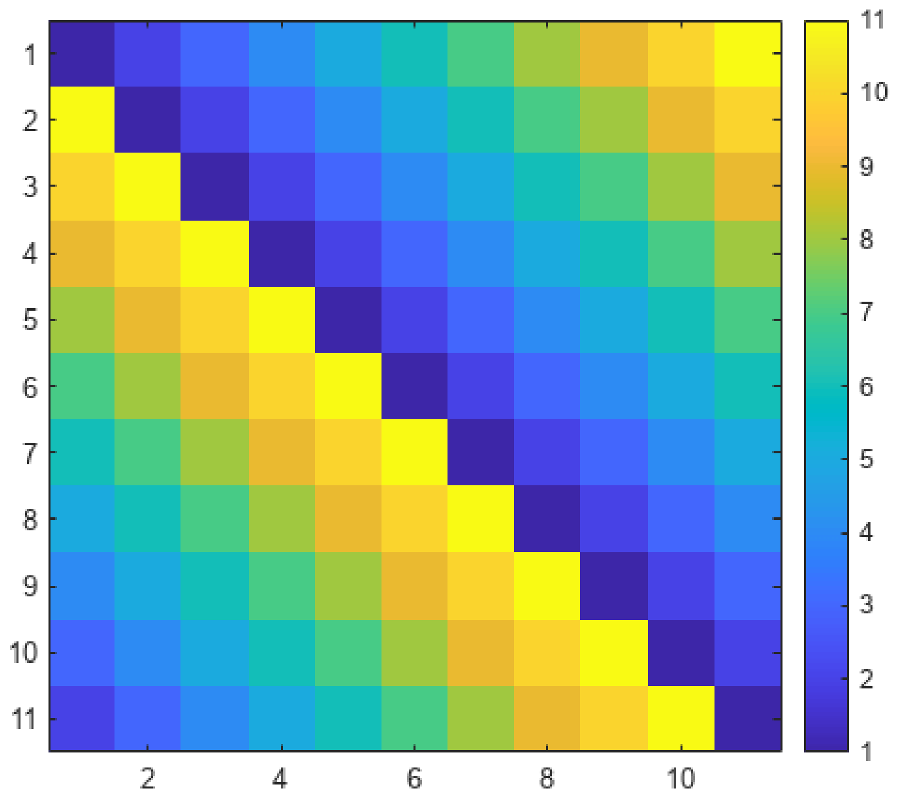


Figure 3. A circulant matrix in color

the matrix-vector product significantly using FFT. However, the downside of using circulant matrices is that we invalidate our proofs of security. Similarly to the cryptosystem, we see the issue of balancing efficiency and security.

6.4. **An Efficient but not Provably Secure Cryptosytem - NRTU.** NTRU is a Lattice-based cryptosystem proposed by Hoffstein, Pipher and Silverman. Originally, the cryptosystem was proposed using the language of rings, but we will describe it using lattices generated by circulant matrices and lattices. Because of its use of circulant matrices, it is efficient. However, it isn't (yet) provably secure. This is because using circulant matrices introduces structure to the lattices, invalidating proofs of security.

- **Parameters**: Prime n, modulus q, and integer bound d_f, small integer p
- **Private key**: Vectors $f \in e_1 + \{p, 0, -p\}^n$ and $g \in \{p, 0, -p\}^n$, such that each of $f - e_1$ and g contains exactly $d_f + 1$ positive entries and d_f negative ones, and the matrix $[T^* f]$ is invertible modulo q
- **Public key**: The vector $h = [T^* f]^{-1} g \bmod q \in \mathbb{Z}_n^q$
- **Encryption**: The message is encoded as a vector $m \in \{1, 0, -1\}^n$, and uses as randomness a vector $r \in \{1, 0, -1\}^n$, each containing exactly $d_f + 1$ positive entries and d_f negative ones. The encryption function outputs $c = m + [T^* h] r \bmod q$

- **Decryption**: On input ciphertext $c \in \mathbb{Z}_n^q$, output $(([T * f]c) \bmod q) \bmod p$, where reduction modulo q and p produces vectors with coordinates in $[-\frac{q}{2}, +\frac{q}{2}]$ and $[-\frac{p}{2}, \frac{p}{2}]$ respectively

Notice the use of circulant matrices $[T^* f]$ and $[T^* h]$. To see exactly why this cryptosystem works, see the original paper [6].

7. Conclusion

We discussed Post-Quantum Cryptography (PQC) and why it's an important field to study. After introducing the basics of lattices, some lattice problems and algorithms to solve them, we looked at a few lattice-based cryptosystems and hash functions. A common theme throughout lattice-based cryptography (and indeed PQC in general) is that finding an efficient AND provably secure post quantum cryptosystem is a very difficult problem.

References

[1] Daniel J. Bernstein. *Introduction to post-quantum cryptography*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[2] Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] Carl Pomerance and Paul Erdös. A tale of two sieves. 1998.

[4] Lenstra, H.W. jr., Lenstra, A.K., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[5] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 284–293, New York, NY, USA, 1997. Association for Computing Machinery.

[6] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

Email address: `manu.isaacs@gmail.com`