CS 198 Codebreaking
Spring 2023 Ryan Cottone, Imran Khaliq-Baporia

Note 9

Post-Quantum Cryptography

So far, we have covered an extensive survey of the field of modern cryptography, which chiefly relies on the mathematical hardness of certain problems like the discrete logarithm for security. This is especially true for asymmetric cryptography like RSA and Diffie-Hellman. Quantum algorithms are a major threat to these systems, because they give far better time complexity than the best know classical algorithms. When, not just if, these quantum computers become powerful enough in practice, the vast majority of modern cryptography is dead in the water.

Shor's Algorithm

The most famous quantum algorithm relevant to cryptography is **Shor's Algorithm**, which factors integers in $O((\log N)^2(\log\log N))$ time. Recall that the best classical algorithms are sub-exponential (much worse than poly-logarithmic like Shor's). Since the security of RSA relies on factoring, this algorithm immediatly compromises the security of RSA. Shor's algorithm can also solve the discrete logarithm problem in around the same time.

Grover's Algorithm

Another popular algorithm is known as **Grover's Algorithm**, which provides for generic function inversion in $O(\sqrt{n})$ time. Recall that normal inversion for an unknown functions requires guessing all possible inputs – with n inputs, this takes $O(n)$ time. Therefore, Grover's algorithm can invert hash functions and symmetric ciphers much quicker than classical ones. Remember that the strength of AES-128 is considered to be 128 bits: $O(2^{128})$ searches required. Grover's algorithm would reduce that to $O(\sqrt{2^{128}}) = O(2^{64})$ searches, which is in the realm of feasibility. In this sense, it halves the bits of security for pretty much all symmetric schemes.

However, this is easily countered by increasing the bits – the underlying complexity is still exponential, so none of the schemes are considered broken. Therefore post-quantum cryptography is primarily concerned with **asymmetric encryption**.

Lattices

The current most promising post-quantum candidate is *lattice cryptography*, which is built upon mathematical lattices. A **lattice** is the set of integer linear combinations of basis vectors:

$$L = \{a_0 v_0 + a_1 v_1 + \ldots + a_n v_0 n : a \in \mathbb{Z}, b \in \mathbb{Z}^n\}$$

.

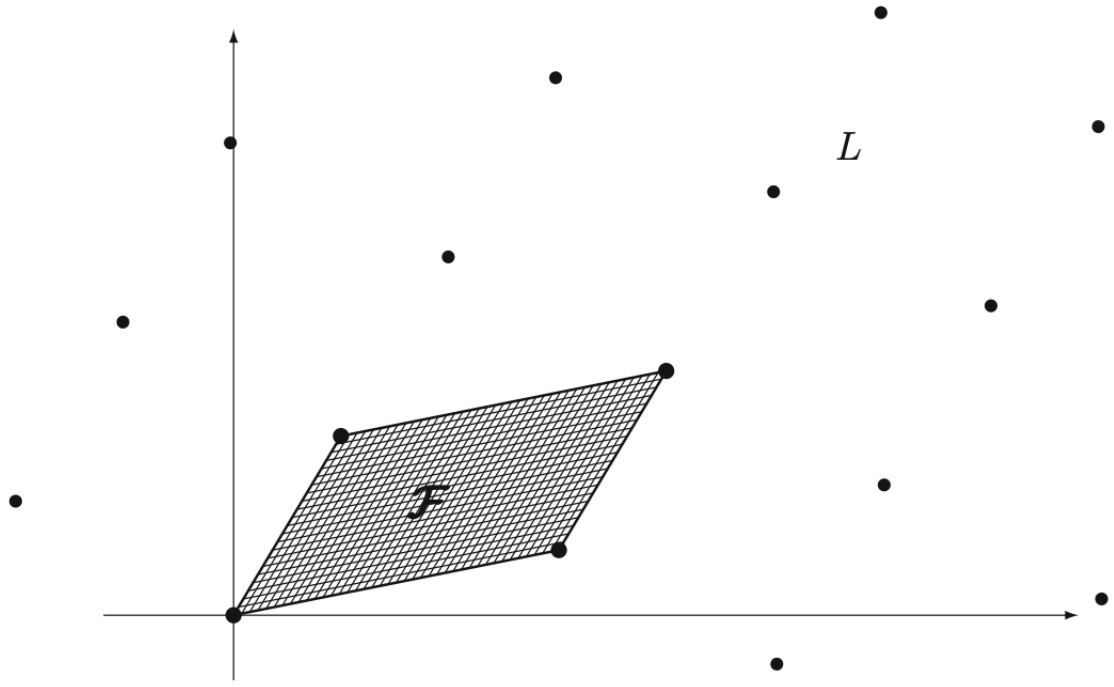

Photo credit to Hoffstein et. al. textbook.

Note that this is a discrete set, unlike normal vector spaces. A vector is said to be in the lattice if it can be expressed as a integerlinear combination of these basis vectors. Note that this basis is **not unique**, and can be efficiently transformed to other basis through any integer-valued matrix with determinant ± 1.

The **fundamental domain** of a lattice is defined as the set

$$F(B) = \{t_1 b_1 + t_2 b_2 + \ldots + t_n b_n : 0 \leq t_i < 1\}$$

i.e. all (not necessarily integer) linear combinations from 0 up to 1. This region turns out to be very important in the study of lattices, mainly because **the volume of a fundamental domain is invariant of the basis**. The volume of this fundamental domain is defined as the determinant of the lattice: $\det L = \text{Vol}(F)$. This leads us to **Hadamard's inequality**, which states that

$$\det L = \text{Vol}(F) \leq \|v_1\| \cdot \|v_2\| \cdots \|v_n\|$$

When the basis vectors v are orthogonal, this becomes an equality. Try to convince yourself why less orthogonal basis have a higher right-hand side than left-hand side in this inequality.

The key reason to use lattices for cryptography is the existence of **good basis** and **bad basis**. A good basis is one in which the basis vectors are highly orthogonal, and a bad basis is one where the basis vectors are not very orthogonal.

To quantify this, we define the **Hadamard ratio** as:

$$H(B) = \left(\frac{\det L}{\prod_i^n \|v_i\|} \right)^{\frac{1}{n}}$$

.

Note that a larger Hadamard ratio means the basis is more orthogonal. Why does this matter? It turns out that solving certain problems on lattices is far easier when working in a good basis than a bad basis.

Lattice Problems

There are mathematical problems on lattices that are considered hard to solve in the worst case, even for quantum computers. They have a lot of applications to mathematical schemes, especially in cryptography.

The **shortest vector problem**, or SVP, asks for the vector in the lattice with the smallest **norm** (usually ℓ_2, aka Euclidean distance). The exact version of this problem is known to be **NP-Hard** (under randomized reductions). The approximation version, which asks for vector $v' \in L$ such that $\|v'\| = \gamma\lambda(L)$ (where $\lambda(L)$ is the real shortest vector in L). The approximation version, often denoted SVP_γ or apprSVP, asks for a short vector with a norm at most some fixed constant multiple greater than the real one. Interesting enough, this problem is also considered NP-Hard for small γ, but polynomially solvable for $\gamma \geq 2^N$.

The **closest vector problem**, or CVP, asks for a vector in the lattice with the smallest distance from another given vector. Formally, we want v' such that $\|v' - v\|$ is minimized for the input v. The CVP is known to be NP-Hard unconditionally. The approximation version, often denoted CVP_γ or apprCVP, asks for some vector v' given v such that $\|v' - v\| \leq \gamma(a - v)$, where a is the optimal closest vector. CVP_γ is generally hard for subexponential γ.

Note that CVP is considered at least as hard as SVP, since a reduction exists from SVP to CVP in polynomial time. **Exercise: Work out the reduction from apprSVP to apprCVP. Note that we cannot pass in 0 as the target vector to CVP, since the oracle would just return 0 itself.**

There are other problems in lattices that are considered hard, such as the shortest independent vector problem, but we will focus on these two for now.

Babai's Algorithm

How do we build cryptographic systems, assuming these problems are hard? We generally require the existence of a trapdoor function, easy to compute knowing a key but hard to do otherwise. This is where the good/bad basis idea comes into play. **Babai's algorithm** is a rather straightforward approximation algorithm for the CVP. Given the input vector $v = a_0b_0 + a_1b_1 + \ldots + a_nb_n$, we round all the coefficients to the nearest integer and return $v' = \lfloor a_0 \rceil b_0 + \lfloor a_1 \rceil b_1 + \ldots + \lfloor a_n \rceil b_n$. Evidently this algorithm is fast, but has really poor approximation results in higher dimensions and for less orthogonal basis vectors.

However, if the basis B is very orthogonal, Babai's algorithm is pretty good at recovering an approximate closest vector. We can use this to our advantage!

Lattice Cryptography

We will show an example public-key encryption scheme known as the GGH scheme.

We generate a **good basis** for a lattice L, denoted B_{good}. This will be our **private key**. We then transform it using a integral matrix M with determinant equal to 1 to get $B_{bad} = M(B_{good})$, our **public key**. Both of these basis describe the same lattice.

To send a message m, Bob first generates some short error vector r (often using Gaussian noise). This is to hide the actual message m, which is a lattice vector. Bob sends $m + r$ to Alice.

Remember that Bob only has access to the public (bad) basis, but Alice knows the good basis. Alice can then run Babai's algorithm on $m + r$ in the good basis to recover m' (a representation of m in the good basis), and then transform it back into the original form via B_{good}^{-1}.

Another popular lattice cryptography scheme is the NTRU cryptosystem, whose details we will sadly leave out of scope. Lattice cryptography is a very interesting field with way more details than included in this note.