Lattice cryptography: isomorphisms & dual attacks

Pulles, L.N.

Summary

This thesis is about lattices. Much of the cryptography that is being used today, can be broken by (large) quantum computers in the future! There are multiple computational problems involving lattices that are easy in dimension 2 or 3 but are hard in high dimensions. These problems are, as far as we know, difficult for *both* classical computers *and* quantum computers. Therefore, we can use lattices to build so-called *post-quantum cryptography*, which will remain secure even when large quantum computers are built.

A goal of cryptography is to make secure communication between two or more parties possible. There are two important cryptographic protocols: *encryption* and *signature schemes*. Encryption makes the content of messages incomprehensible for eavesdroppers. A signature scheme enables one to verify whether a message was really sent by some party. The two parties hold different keys in a signature scheme, just like in the real world: one party writes signatures with a special pen, while the other knows the precise colour composition of the ink in the pen. Encryption is similar: everyone can encrypt a message using a *public key*, while deciphering is only possible for one owning the *private key*.

Lattices can be used to generate such a public/private key pair. The private key is a description of the lattice, that consists of the shortest lattice points. This key is private because it is difficult in high dimensions, to even compute a somewhat short lattice point. The public key is an arbitrary description of the lattice, which usually consists of long lattice points. The fastest algorithm that upon input a public key, computes the private key associated to it, requires time and memory that grows exponentially as a function of the dimension.

Part II concerns the *dual attack*. This is an algorithm that, upon

input a description of a lattice and a point, outputs the lattice point that is the closest to the point while making use of the *dual lattice*. Every dual lattice point corresponds with a wave function (in fact, character) that is maximal at every lattice point. Using this wave function, one can decide whether a point is close to the lattice or not. By subdividing the lattice like a chess board into a black sublattice and a white coset, you can determine whether the black sublattice is closer to the point than the white coset. If this method is repeatedly sufficiently many times, one can determine which lattice point is closest to the point.

Chapter 3 shows that an assumption that was used in literature to analyse the so-called *dual-sieve attack* is incorrect, because it produces contradictions in multiple theoretic regimes, and experiments show that it leads to wrong predictions. Chapter 4 contains an alternative assumption for the dual-sieve attack. This assumption leads to accurate predictions for multiple important probability distributions of points.

Part III concerns the *lattice isomorphism problem* (LIP). Special lattices with good properties are convenient when building a *signature scheme*, because they make a signature scheme very efficient. Because every cryptology researcher knows a good description of such a lattice, the lattice has to be hidden differently. Using LIP we can hide the lattice by randomly rotating it. The specific rotation now becomes the private key.

Chapter 5 is dedicated to the construction of the signature scheme Hawk, which is based on LIP. To find the private key of Hawk, one needs to solve smLIP, a module variant of LIP for the simple lattice $\mathbb{Z}^n$, the lattice consisting of all integer vectors. Security of Hawk is based on the new *one more shortest vector problem* (omSVP). In Chapter 6 we study *automorphisms* of the module lattice: a rotation that maps the lattice to itself. Theorem 6.2 states that, using a nontrivial automorphism, it becomes much easier to find a short vector, and to solve smLIP. An automorphism sends a short lattice point to a (possibly different) short lattice point, and therefore may easily solve omSVP. If smLIP is a difficult computational problem, then it will be difficult to find a nontrivial automorphism, and thus, it will be difficult to solve omSVP by 'just' finding a nontrivial automorphism.