Quantum Resistant Cryptography

Antal Nemes
MSc. Mathematics

MSc. Thesis

Supervisor: Viktória Villányi
Department of Operations Research

Institutional Consultant: Péter Sziklai
Department of Computer Science

Eötvös Loránd University, Budapest
Faculty of Science

Budapest, 2012

Abstract

Among the many cryptographic schemes there are two groups of algorithms that are widely used currently in applied cryptograpy: the algoritms which security is based on the integer factorization problem and those based on the discrete logarithm problem.

In the quantum computational model both these mathematical problems are solvable in polynomial time, which leads to security questions when these algorithms are applied.

In this thesis we will talk about the quantum computational model and discuss some cryptographic schemes, which are considered to be resistant in the quantum computational model.

A kvantum számítási modellben néhány olyan eddig nehéznek tartott problémára sikerült polinomiális algoritmust találni, melyekre titkosítási problémák alapulnak.

Így azon algoritmusok, melyek biztonságossága ezen problémák nehézségén alapulnak, megbízhatóságuk megkérdőjeleződött. Emiatt célszerű olyan titkosítási sémákat vizsgálni, melyek az alkalmazásokban elég hatékonyak, mindemellett ellenállnak a kvantum számítógépeknek is.

Ebben a szakdolgozatban bemutatjuk a kvantum számítási modellt, majd tárgyalunk néhány a kvantum számítógépeknek is ellenálló digitális aláírási sémát illetve titkosító algoritmust.

Acknowledgements

I would like to thank my supervisor, Viktória Villányi for introducing me to this exciting topic which hits many areas of science, such as quantum physics, functional analyzis, algorithm theory, complexity theory and applied cryptography. I met a lot of exciting and interesting theorems and concepts which motivated me to dive more and more into the topic. I also want to thank her the interesting articles she gave me, which was beneficial in writing the thesis, for her guidance, and for helping me refining my thesis.

Contents

Chapter 1

Introduction

The security of many cryptographic algorithms are based on mathematical problems that are difficult to solve. Two well-known problems among them are the integer factorization and the discrete logarithm problem. We will introduce both these problems in Chapter 2. The security of the RSA algorithm is based on the integer factorization problem, the DSA algorithm is related to the discrete logarithm problem.

These algorithms are considered to be secure since no one found efficient algorithms that solve neither the integer factorization nor the discrete logarithm problem, although they have been researched deeply for a long time. Moreover different computational complexity results strenghten the idea that it is not possible to solve them efficiently.

However, scientists invented a new way a computation based on the laws of physics, the quantum computational model. The computers in the quantum computational model are called quantum computers.

It is true that a classical algorithm can be simulated on quantum computers, therefore the quantum computational model extends the classical computational model. However, there are problems when quantum algorithms can be more efficient, see Section 3.2.4. There are other problems we have not found efficient algorithms to solve them in the "classical" computational model yet, but there are quantum algorithms which solve them: e.g both the integer factorization and the discrete logarithm problem can be solved in polynomial time in the quantum computational model. These results undermine the reliability of the corresponding cryptographic schemes.

An algorithm is quantum resistant if its security is not broken only by extending the classical computational model by quantum computers.

In Chapter 2 we will talk about integer factorization and the discrete logarithm

problem.

In Chapter 3 we discuss the quantum computational model. We start with an introduction to the necessary physics. Then we introduce the concepts of quantum computation and show a problem where the quantum computational model is stronger than the classical one. Then we introduce Grover's search algorithm, and finally discuss a result about quantum resistance.

In Chapter 4 we show digital signature schemes that are considered to be resistant to quantum computation. We will introduce the Lamport signature scheme and its enhancements, the Merkle signature scheme, the Merkle-Winternitz signature scheme and the Merkle hash trees.

In chapter 5 we show cryptographic schemes that are considered to be resistant to quantum computation. We will discuss two algorithms here, the GGH algorithm based on lattices, and the NTRU algorithm which works on polynomials. However, NTRU is also strongly connected to lattices, and the most efficient attacks against NTRU were related to lattice reductions. So the basic concepts of lattice theory also introduced in this chapter.

Chapter 2

Integer factorization, discrete logarithm

2.1 Integer factorization

The integer factorization problem is to find a (prime) factor of an integer n. The naive algorithm checks all the numbers below n whether they divide n. Let t denote the number of bits in the binary representation of n, then the running time of the naive algorithm is $\mathcal{O}(e^t)$. It is easy to see it is enough to check the numbers below $\sqrt{n}$, resulting $\mathcal{O}(e^{\mathcal{O}(t/2)}) = \mathcal{O}(e^{t/2})$, which is a speedup compared to the previous algorithm, however still exponential. Actually it is unknown that there is a classical algorithm solving the integer factorization problem in polynomial time.

The algorithm giving the best asymptotical result is the **general number field sieve** algorithm. Its complexity is $\mathcal{O}(e^{(\log^{1/3} n)(\log\log n)^{2/3}})$ [LJMP90].

However, Shor's algorithm solves the integer factorization problem in polynomial time in the quantum computational model [Sho97].

Suppose we want to find a factor a non-prime integer n. Note that if we can find one factor of n in $\mathcal{O}(p(\lceil \log n \rceil))$ polynomial time, then we can also find the factorization of n in polynomial time. Since n has $\mathcal{O}(\log n)$ polynomial factors at maximum, we have to repeat the factor finder algorithm for $\mathcal{O}(\log n)$ times, leading to an $\mathcal{O}(p(\log n)\log n)$ algorithm.

First we decide if n is a power of an integer. Suppose $n = m^k$ for some integer m. Since $k \leq \log_2 n$, it is enough to check polynomially many k-th root of n.

Therefore we can assume n is not a power of an integer, specially not a power of a prime.

The problem is first reduced to the problem of finding the order $\mathrm{o}(k)$ of a given integer k, i.e the smallest integer r so that $k^r \equiv 1 \mod n$, if exists.

The idea behind the reduction is the following: although it is difficult to find a factor of an integer, finding a common factor of two integers is easy using the Euclidean algorithm.

The reduction:

Choose an integer $1 < x < n$ randomly. If $\gcd(x, n) > 1$, then by the Euclidean algorithm we can find a divisor of n.

If $\gcd(x, n) = 1$, then with probability at least $\frac{1}{4}$ we have $\gcd(x^{o(x)/2} - 1, n) > 1$, so again we can find a divisor of n. If the greatest common divisor is still 1, then we choose another x.

Theorem 2.1.1. *Suppose n is not a power of a prime. Then at least probability $\frac{1}{4}$ the order of $x \in_R \mathbb{Z}_n^*$ is even and $x^{o(x)/2} \not\equiv -1 \mod n$.*

Theorem 2.1.2. *If $y^2 \equiv 1$ and $y \notin \{-1, 1\} \mod n$, then $\gcd(y - 1, n) > 1$.*

Shor's quantum algorithm solves the order finding problem in polynomial time, leading to a polynomial time algorithm for the integer factorization problem.

The algorithm was implemented first in 2001, when IMB succeeded in factoring the integer 15 to 3*5 by a 7-qubit quantum computer.

2.2 Discrete logarithm

Let G denote a group, and $\langle g \rangle$ denote the subgroup of G generated by g. Solving the equation $g^x \equiv a$ for fixed $a, g \in G$ is called the **discrete logarithm problem (DLP)**. The following notation is used: $x = \mathrm{ind}_g^G a$ or simply $\mathrm{ind}_g a$.

For cryptographic applications it is useful to choose g so that $\langle g \rangle$ to be a large subgroup of G. When there is a $g \in G$ so that $\langle g \rangle = G$, g is called the **primitive root** of G.

Not every group has primitive roots, but for p primes $\mathbb{Z}_p$ has primitive roots. In most cryptographic schemes we work under $\mathbb{Z}_p$ where p is a prime, therefore we assume from now that G has a primitive root. When G has a primitive root g, then the equation $g^x \equiv a$ can be solved for every $a \in G$.

Some discrete logarithm related cryptographic schemes do not directly depend on DLP but on an other problem called **Diffie-Hellman problem (DHP)**, when for given g, g^x, g^y we need to compute g^{xy}.

Clearly if we have an algorithm that solves the discrete logarithm problem, then we can solve the Diffie-Hellman problem too. The other direction of the relation has not been clarified yet.

Let $n = |G|$ and t denote the number of bits in n, $a \in G$ fixed. A naive algorithm to solve the discrete logarithm problem would be that we multiply g by itself till we reach a. This gives us an $\mathcal{O}(e^t)$ exponential algorithm. It is unknown if there is a polynomial algorithm that solves DLP in the classical computational model. The best asymptotical result for the problem is $\mathcal{O}(e^{(\log^{1/3} n)(\log\log n)^{2/3}})$, see [Mat03]. Note that we have the same asymptotics for the integer factorization.

However, as it is with the integer factorization problem, there is a polynomial time algorithm for DLP in the quantum computational model [Sho97].

Chapter 3

Quantum computation

In this chapter we will discuss the concepts of quantum computation. Physicists have found a very convenient abstract layer of quantum mechanics based on functional analysis and linear algebra. Instead of solving partial differential equantions, we will work with eigenvectors and eigenproblems of a Hilbert-space. By using this high level model, we can design quantum algorithms and talk about quantum information theory even if we are new to quantum mechanics. However, since seeing how physics had developed from classical mechanics to this layer can aid the understanding, we will start with quantum physics.

Then we introduce the Hilbert-space of the states of a quantum system, where the quantum computations are done. Then we show how to develop the quantum computational model in this space and show an example, when the quantum computational model is stronger than the ,,classical" computational model. Finally we introduce Grover's search algorithm, and discuss a result about quantum resistance.

3.1 Physics

3.1.1 Basic quantum physics

Our approach to the physical world has changed a lot over the past 400 years from classical mechanics to the concepts of modern physics. Scientists had difficulties in understanding how particles really work.

Some aspects of light could be explained when the light is threated as small particles, other aspects could be clarified if threated as a wave.

Since the concepts of being particle and being wave are far from each other in practical view, scientists first tried to eliminate one of these cases, but they could not really advance with it.

As they could not decide the real nature of the light, they started to think of it as it is both particle and wave. This is called the wave-particle dualism. It turned out in the 20th century that every particle can be treated as wave and vice versa, and there are formulae how to transform one concept to the other.

Finally scientists found a concept that generalized both particle and wave behaviour.

The states of a quantum system can be characterized by complex distribution functions $\varphi(\vec{r}, t)$ called **wave functions** so that

$$\int |\varphi(\vec{r}, t)|^2 d^3\vec{r} = 1$$

The strength of the characterization is that the important classical mechanical properties can be expressed by the expectation value of a proper operator according to this distribution. In [Ö00] the reader can find a table of formulae how these physical values can be expressed by the expectation value of a proper operator. For example the location can be expressed by the following:

$$\langle\vec{r}\rangle = \int \varphi^*(\vec{r}, t)\vec{r}\varphi(\vec{r}, t)d^3\vec{r} \quad \text{and} \quad \Delta r = \sqrt{\langle r^2\rangle - \langle\vec{r}^2\rangle},$$

where Δr is called **uncertainty**.

Although we can calculate these classical properties as expectation values using the wave function, the question remains how we can find the wave function itself. The answer is the **Schrödinger equation**, from which we can calculate (at least in theory) the wave function:

$$H\varphi = i\hbar\frac{\partial}{\partial t}\varphi.$$

Here H denotes the Hamilton operator, which expresses the total energy of the system.

To visualize a little bit why the introduction of wave functions generalizes both particle and wave behaviour, see the following example: ([Ö00] p. 8)

Suppose there is a particle trapped between two reflecting mirrors. So the particle moves periodically from left to right, from right to left, etc.

If you calculate the distribution of its location, you will get that the location of the particle between the two mirrors is uniform (as expected because of the symmetry). You can interpret the result as if it is a wave in real that vibrates between the mirrors, and the uniformity of the location means the wave itself stays in the space between the two mirrors, so it is a standing wave.

3.1.2 Hilbert-space

For special physical conditions you can deduct another form of the Schrödinger equation, the **time-independent Schrödinger equation**. If we want to solve this equation, in practice it is useful to assume the time and location can be separated in the wave function: $\varphi(\vec{r},t) = \varphi(\vec{r})\phi(t)$. In that case it is easy to solve the time part, and the location part leads to the eigenvalue problem $E\varphi = H\varphi$, where E is the expectation value of H related to the distribution φ. Solving this part leads us to the solution of the Schrödinger equation.

This motives us imagine the wave functions as vectors in the Hilbert-space of the complex valued functions. If we measure the eigenvalue, then we can find the state in which the particle resides by solving the eigenproblem.

We are not always interested in the explicit state of the particle, sometimes it is enough to be able to distinguish two (or more) possible states, when the measurement of the eigenvalues help. For example for a qubit (which is the quantum equivalent of a single bit), we are only interested in whether the system is in state zero or one.

This means we can focus on the eigenvalues rather than the states. If we want to use quantum mechanics to extend the classical algorithmic approach, it might be easier to calculate with vectors of a Hilbert-space, thinking about bases, subspaces and eigenvalues than solving partial differential equations.

To summarize: algebraic quantum mechanics threats the states as unit-long elements of the $\mathcal{H}$ Hilbert-space of the complex valued functions, where the scalar product is defined for $\varphi, \psi \in \mathcal{H}$ by the following:

$$\langle \varphi, \psi \rangle = \int \varphi^* \cdot \psi.$$

A nice property of this Hilbert space is that the eigenvectors of the H Hamilton operator are orthonormal, therefore they form an orthonormal base of the corresponding subspace.

3.1.3 Probabilitic properties

Consider a system containing a single particle. When you measure it, you will find that it is in the eigenstate φ' with eigenvalue E'. But generally, before you measure the system, the particle is not in a single eigenstate but in the superposition of several eigenstates of the H Hamilton operator. So its general state φ can be expressed by the following:

$$\varphi = \sum_{i=1}^{k} a_i \varphi_i$$

where φ_i are eigenstates of H, and by the normalization of a distribution function: $\sum a_i^2 = 1$. The square of the coefficients give a probability distribution. You can imagine that the particle is in the eigenstate φ_i with probability a_i^2. After measurement the superposition of states collapse, i.e. the coefficient of an eigenstate becomes 1, the other coefficients become 0.

This probabilistic feature also occurs to quantum bits. A quantum bit can be zero with some probability and one with some probability.

3.1.4 Bracket notation

Physicists use a special notation for algebraic quantum mechanics.
Let $\psi, \varphi \in \mathcal{H}$, and consider the scalar product $\langle \psi, \varphi \rangle$. Physicists split the formula in the middle: $\langle \psi |, | \varphi \rangle$, and use the notation $\langle \psi | \in \mathcal{H}^*, | \varphi \rangle \in \mathcal{H}$. So $\langle \psi |$ is an operator from the dual space, and $(\langle \psi |)(| \varphi \rangle) = \langle \psi, \varphi \rangle$. The $\langle \cdot |$ is called **bra**, the $| \cdot \rangle$ is called **ket**, and the scalar product is called **bracket**.

Riesz representation theorem states that every $A \in \mathcal{H}^*$ can be expressed as $A = \langle \psi, \cdot \rangle = \langle \psi |$ for proper $\psi \in \mathcal{H}$, which justify the notation bra.

So from now on we will use the notation $\langle \psi | = \langle \psi, \cdot \rangle \in \mathcal{H}^*$ for the vectors of the dual space, and $| \varphi \rangle \in \mathcal{H}$ for the vectors of $\mathcal{H}$.

3.2 Quantum computation

In this section we introduce the parts of a quantum computer, and define the concept of computation.

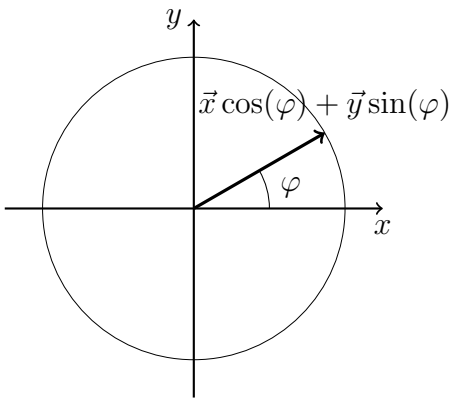
3.2.1 Qubits

In the classical computational model we store values and manipulate them. The logical base of these manipulations are the bits.

In the quantum computational model we also want to manipulate information. The logical base of computation here is called **qubit** (quantum bit).

Quantum bits are the superposition of two given eigenstates. So we choose two eigenstates $|0\rangle, |1\rangle \in \mathcal{H}$ from the Hilbert space. The possible states of the qubit is expressed as $\varphi = a \cdot |0\rangle + b \cdot |1\rangle$, where $a^2 + b^2 = 1$. That means the qubit is in state 0 with probability a^2 and in 1 with probability b^2.

It is often useful to imagine a qubit as a 1 long vector on the 2-dimensional plane.



General state of a single qubit system.

3.2.2 Registers

We can extend a system with more qubits. For $k = 2$ if the four eigenstates are denoted as $|00\rangle, |10\rangle, |01\rangle, |11\rangle$, then the system is in

$$\varphi = a_{00}|00\rangle + a_{10}|10\rangle + a_{01}|01\rangle + a_{11}|11\rangle$$

where $a_{00}^2 + a_{10}^2 + a_{01}^2 + a_{11}^2 = 1$. The probability that the system is in $|b_1 b_2\rangle$ is $(a_{b_1 b_2})^2$, where $b_1, b_2 \in \{0, 1\}$.

Remark 3.2.1. *Note that the state of the 2-qubit is more general than the product of two 1-qubit system. For instance we can initialize a qubit system to the state* $(1/\sqrt{2})(|00\rangle + |11\rangle)$*, but it is not a product of two 1-qubit system.*

Suppose $(1/\sqrt{2})(|00\rangle + |11\rangle) = (a \cdot |0\rangle + b \cdot |1\rangle)(c \cdot |0\rangle + d \cdot |1\rangle)$*. That means* $ac = 1/\sqrt{2} = bd$ *and* $bc = ad = 0$*. So* $0 = abcd = 1$*, which is a contradiction.*

An n-qubit **register** is an n-qubit system. Therefore the state of a register is a superposition of the product of the base states of the involved n qubits. So in general the state of an n-qubit register is the superposition of 2^n vectors, which coefficients can be *represented in* $\mathbb{C}^n$ by an n-long complex vector $\langle v_1, \ldots, v_l \rangle$ so that $\sum_{i=0}^{n} |v_i|^2 = 1$.

If we measure the state of a register, the register collapses to one of the base vectors $|b_1, \ldots, b_n\rangle$ (where $b_i \in \{0, 1\}$): the coefficient of this vector becomes 1, the other coefficients become 0.

3.2.3 Quantum operations

Quantum operations are operations which transform the state of a register to another state.

Definition 3.2.2 (Quantum operation)**.** *A $\mathcal{B} : \mathbb{C}^n \to \mathbb{C}^n$ function is called* ***quantum operation*** *if it*

(i) is linear,

(ii) preserves the norm.

The second assumption is reasonable, because if you use the operation on a state, then you want to acquire a state again, so if $\|v\| = 1$, then $\|\mathcal{B}(v)\| = 1$. This concludes that $\forall v \in \mathcal{H} : \|\mathcal{B}(v)\| = \|v\|$.

Note that a linear operation between finite dimensional vector spaces preserving the norm is invertible. Else there was a nonzero vector that is mapped to zero, so its norm is not preserved.

Even though norm-preserving is reasonable, reversibility causes some technical problems when designing algorithms. For instance copying is not reversible in general.

A $\mathcal{B}$ quantum operation is linear, so it can be represented by a complex B matrix. Since $\mathcal{B}$ preserves the norm, B is unitary. The reverse direction that a unitary matrix represents a linear, norm-preserver operation is trivial.

This concludes that we can work with quantum operations as unitary matrices.

Now we note some important quantum operations.

- **Flipping bits**. Flipping a quantum bit means to change its state from 0 to 1 or 1 to 0. Flipping the k-th qubit of a register can be solved by the operator that maps $|x_1, \ldots, x_{k-1}, x_k, x_{k+1}, \ldots, x_n\rangle \mapsto |x_1, \ldots, x_{k-1}, 1-x_k, x_{k+1}, \ldots, x_n\rangle$, $x_i \in \{0, 1\}$ $(1 \leq i \leq n)$. Since this operator is only a permutation over the base vectors, it is unitary.

- **Reordering qubits.** Again, since it is only a permutation over the base vectors, it is unitary.

- **Copying qubits.** Copying in general is not a reversible operation. However, if the target qubit is set to $|0\rangle$, we can achieve the same result by an invertible operation.

  Suppose we want to copy the q_1 qubit to the q_2 qubit, which is initialized to $|0\rangle$. Then the copying operation can be simulated by $|q_1 q_2\rangle \mapsto |q_1(q_1 \oplus q_2)\rangle$.

  This map is a quantum operation, and works exactly as copying when q_2 is $|0\rangle$. It is the algorithm designer's responsibility to provide that q_2 is really initialized to $|0\rangle$.

- **CNOT: Controlled not.** The operation defined above above negates the second qubit if and only if the first qubit is true. This feature also can be useful in quantum algorithms.

- **Rotating a qubit.** Since the rotation around the origo in the 2 dimensional plane is unitary, rotating a qubit is a quantum operation.

- **AND.** The AND operation is not invertible generally, similarly to the copy operation. However the same technique can be used when copying.

  The AND operation acts on three qubits: the q_1, q_2 input qubits and a target qubit q_3, which is initialized to $|0\rangle$. The AND operation can be expressed as follows:

  $$|b_1\rangle|b_2\rangle|b_3\rangle \mapsto |b_1\rangle|b_2\rangle|b_3 \oplus (b_1 \wedge b_2)\rangle.$$

  This operation is unitary, since it is a permutation operation.

- **Hadamard operation.** The Hadamard operation maps

  $$|b\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^b}{\sqrt{2}}|1\rangle, b \in \{0,1\}.$$

  Its matrix

  $$\begin{pmatrix} 1 & 1 \\ 1 & \text{-1} \end{pmatrix}$$

  is unitary. The Hadamard operation is useful when simulating random algorithms. A single qubit of $|0\rangle$ or $|1\rangle$ is transformed so that it is a quantum coin: $1/2 : |0\rangle, 1/2 : |1\rangle$.

  Note that if you apply the Hadamard operation to every qubit of an n-qubit register initialized to $|00\ldots0\rangle$ changes its state to the uniform distribution of the base states.

Definition 3.2.3 (Elementary quantum operation)**.** *A quantum operation is called* ***elementary*** *if it acts on three or less qubits of a quantum register.*

The concept is introduced because in theory it is easy to generate unitary operations, but in practice it might be difficult to build them. We more possibly build a quantum operation that works with a few bits.

Definition 3.2.4 (Quantum computation)**.** *A series of elementary operations applied to a quantum register is called* ***quantum computation****.*

3.2.4 The parity game

The parity game is a quantum experiment of John Bell to show the reality of a contemporary paradox, the Einstein-Podolsky-Rosen Paradox (EPR paradox). Here we are more interested in the game itself than in the paradox because it is an easy example when quantum computation is stronger than the classical one.

The parity game works as follows:

There are three players in the game, Alice, Bob and Carol, where Alice and Bob are in the same team. The game is set up so that Alice and Bob cannot communicate with each other except from an inital state, when they can agree on a strategy. Carol can communicate with both Alice and Bob.

First Carol to choose two random bits *uniformly* noted as $x, y \in_R \{0,1\}$. Then she sends x to Alice and y to Bob. After having received the bits, Alice and Bob to choose an own bit a and b respectively, and respond it to Carol.

Alice and Bob win if and only if $a \oplus b = x \wedge y$, or in details:

- if $x = 0$ or $y = 0$, then Alice and Bob win if and only if they send back the same bits, i.e $a = b$.
- if $x = 1$ and $y = 1$, then Alice and Bob win if and only if they send back different bits, i.e $a \neq b$.

Theorem 3.2.5. *In the classical computational model the maximum winning probability Alice and Bob can reach is* 0.75.

Proof. There are four deterministic strategies from which Alice and Bob can choose: $f : \{0,1\} \to \{0,1\}$. We will show that these deterministic strategies lead to 0.75 winning probability at most. Therefore any mixed strategy leads to 0.75 probability at most.

Suppose the strategy of Alice is constant 1. Then if $y = 0$, Bob should choose 1. If $y = 1$, then he has to answer with x, which is $\in_R \{0,1\}$. So the maximum probability to win is $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$. The same probability comes out when Alice's strategy is constant 0.

This also proves that they can reach 0.75.
Suppose the strategy of Alice is always to send back x, i.e $a = x$. We have to find the best strategy for Bob.

- Case $y = 1$. If Bob chooses 0, they surely win:
  - if $x = 0$, then they should choose the same bits, and $a = x = 0 = b$.

– if $x = 1$, then they should choose different, and $a = x = 1 \neq 0 = b$.

- Case $y = 0$. Now they should answer with different bits. If $x = 0$, he should choose $b = 1$, if $x = 1$ he should choose $b = 0$. So they can win only with $1/2$ at maximum in this subcase.

The total probability for this strategy is therefore 0.75.

The same probability comes out when Alice sends back $1 - x$. □

Now we show that if Alice and Bob may use quantum bits, there is a strategy by which they can win with 0.8 probability.

Remark 3.2.6. *We have to note that adding qubits to the system does not affect the rule that Alice and Bob cannot communicate with each other.*

Suppose we have a two-qubit system initialized to the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. *That means if we measure the first qubit, not only the state of the first qubit is collapsed to* $|0\rangle$ *or* $|1\rangle$ *but also the state of second qubit collapses. Moreover they will show the same bit information.*

At first glance it seems that the two qubits communicate with each other, but in fact they just act based on their initial state, there is no communication between them. That was the EPR paradox which Bell wanted to prove by experiment.

Strategy: Suppose Alice and Bob have a 2-qubit system initialized to the $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, and Alice has the first qubit, Bob has the second one.

If Alice receives 1 from Carol, then she rotates her qubit by $+\pi/8$, otherwise she leaves it alone. If Bob receives 1 from Carol, then he rotates his qubit by $-\pi/8$, otherwise he leaves it alone.

In the end they measure the state of their qubits and send back the result to Carol.

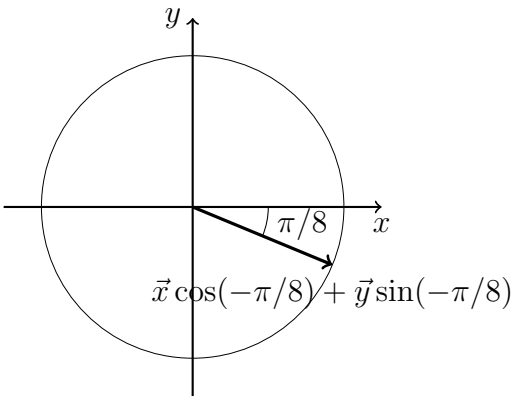
Theorem 3.2.7. *Choosing this strategy Alice and Bob win with probability more than 0.8.*

Proof.

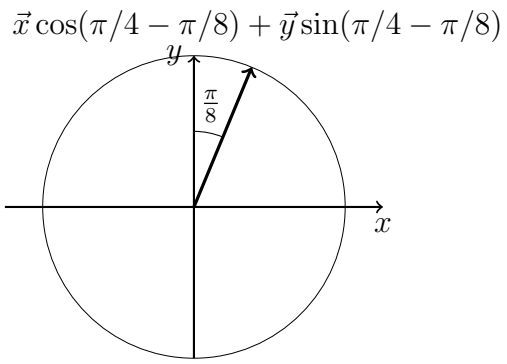
1. Suppose $x = y = 0$, so they have to send back the same bits. In that case they leave their qubits as they are, so the 2-qubit system stays in the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. That means they win with 1 probability, since the two qubits collapse to the same state.

2. Suppose $x = 0, y = 1$. Again, they have to send back the same bits. According to the strategy, Alice leaves her qubit as it is. With $\frac{1}{2}$ probability she gets 0 when the system collapses to $|00\rangle$, so Bob's qubit collapses to state $|0\rangle$. Then it is rotated by $-\pi/8$. In that case their winning probability is $\cos^2(-\pi/8)$.

   With $\frac{1}{2}$ probability Alice gets 1. Then the system collapses to $|11\rangle$, so Bob's qubit is set to $|1\rangle$, then rotated by $-\pi/8$. To win, Bob needs to send back 1. Its probability is $\sin^2(\pi/2 - \pi/8) = \cos^2(\pi/8)$. So the total probability is $\cos^2(\pi/8) > 0.85$.



When $a = 0$.



When $a = 1$.

3. Case $x = 1, y = 0$ is similar to the case above.

4. Case $x = 1, y = 1$. Now both players rotate their qubits, so we calculate the state of the 2-qubit system after the rotations.

   The calculation is done by simply calculating the rotated state of the eigenstates $|00\rangle = \big(1{\cdot}|0\rangle{+}0{\cdot}|1\rangle\big)\big(1{\cdot}|0\rangle{+}0{\cdot}|1\rangle\big)$ and $|11\rangle = \big(0{\cdot}|0\rangle{+}1{\cdot}|1\rangle\big)\big(0{\cdot}|0\rangle{+}1{\cdot}|1\rangle\big)$, then take the superposition of the result:

$$\frac{1}{\sqrt{2}} \cdot \overbrace{\Big(\cos(\pi/8)\cdot|0\rangle + \sin(\pi/8)\cdot|1\rangle\Big)\cdot\Big(\cos(\pi/8)\cdot|0\rangle - \sin(\pi/8)\cdot|1\rangle\Big)}^{\mathrm{rot}(|00\rangle)} \;+$$

$$\frac{1}{\sqrt{2}} \overbrace{\Big(-\sin(\pi/8)\cdot|0\rangle + \cos(\pi/8)\cdot|1\rangle\Big)\cdot\Big(\sin(\pi/8)\cdot|0\rangle + \cos(\pi/8)\cdot|1\rangle\Big)}^{\mathrm{rot}(|11\rangle)} \;=$$

$$\frac{1}{\sqrt{2}} \cdot \Big(\cos^2(\pi/8) - \sin^2(\pi/8)\Big)\cdot|00\rangle - \frac{1}{\sqrt{2}}\cdot 2\sin(\pi/8)\cos(\pi/8)\cdot|01\rangle \;+$$

$$\frac{1}{\sqrt{2}} \cdot 2\sin(\pi/8)\cos(\pi/8)\cdot|10\rangle + \frac{1}{\sqrt{2}}\cdot\Big(\cos^2(\pi/8) - \sin^2(\pi/8)\Big)\cdot|11\rangle$$

   Since $\cos^2(\pi/8) - \sin^2(\pi/8) = \cos(\pi/4) = \sin(\pi/4) = 2\sin(\pi/8)\cos(\pi/8)$, the system can be in the states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ each with probability $1/4$. That means Alice and Bob win with $\frac{1}{2}$ probability.

Now we can calculate the total winning probability:

$$\frac{1}{4} \cdot 1 + \frac{1}{4} \cdot 0.85 + \frac{1}{4} \cdot 0.85 + \frac{1}{4} \cdot \frac{1}{2} = 0.8$$

□

3.3 Quantum computational complexity

3.3.1 Running time

In this section we talk about running time and the complexity class of the polynomial quantum algorithms (BQP). Then we examine how is BQP related to P and BPP.

Suppose we want to compute an $f : \{0,1\}^* \to \{0,1\}$ in $T(n)$ time, where $\{0,1\}^*$ means the finite words over $\{0,1\}$.

The function f should map from $\{0,1\}^*$, since a function can operate on any long numbers in general. However, a given quantum computation works with a fixed register, therefore works with fixed number of bits. That means we have to build quantum computations for every input size. The number of elementary quantum operations of these quantum computations should be $T(n)$ at maximum as we want an algorithm running in $T(n)$.

Therefore we have to design $T(n)$-long quantum computations for every $n \in \mathbb{N}$, so there should be a Turing machine $\mathcal{A}$ that prints out the description of the elementary quantum operations of the quantum computations for $n \in \mathbb{N}$. Of course $\mathcal{A}$ needs to be polynomial to make sense.

Finally, since the whole quantum computation is probabilistic, we only demand the printed quantum computations to realize the function f with probability at least $2/3$.

The description of an elementary operation is its complex matrix. However a complex matrix cannot be presented by finite words, so we write out only the most significant $\mathcal{O}(\log T(n))$ bits of the complex numbers.

Putting this together ([AB09]):

Definition 3.3.1 (Running time)**.** *Let $f : \{0,1\}^* \to \{0,1\}$, and $T : \mathbb{N} \to \mathbb{N}$ be some functions. We say that f is computable in quantum $T(n)$ time if there is a polynomial time classical Turing Machine that on input $(1^n, 1^{T(n)})$ for any $n \in \mathbb{N}$ outputs the descriptions of quantum gates $F_1, \ldots, F_T$, such that for every $x \in \{0,1\}^n$ we can compute $f(x)$ by the following process with probability at least $\frac{2}{3}$:*

1. *Initialize an m qubit quantum register to the state $|x0^{n-m}\rangle$, i.e x is padded with zeros, where $m \leq T(n)$.*

2. *Apply one after the other $T(n)$ elementary quantum operations $F_1, \ldots, F_T$ to the register.*

3. *Measure the register and let Y denote the obtained value.*

4. *Return Y_1, i.e the value of the first qubit of the register.*

Definition 3.3.2. *A Boolean function $f : \{0,1\}^* \to \{0,1\}$ is in **BQP** if there is a polynomial $p : \mathbb{N} \to \mathbb{N}$, such that f is computable in quantum $p(n)$ time.*

3.3.2 P, BPP, BQP

Theorem 3.3.3. *If $f : \{0,1\}^n \to \{0,1\}^m$ is computable by a classical boolean circuit of size S, builded from AND, OR, and NOT gates, where the in-degrees of the nodes are at most 3, then there is a sequence of $2S+m+n$ quantum operations computing the mapping*

$$|x\rangle|0^{2m+s}\rangle \mapsto |x\rangle|0^m\rangle|f(x)\rangle|0^S\rangle.$$

Since every classical boolean circuit can be reconstructed so that the in-degrees are at most 3 by only polynomially increasing the number of gates, the in-degree restriction is only technical. As a corollary we can simulate boolean circuits by quantum computations in polynomial time. Since every Turing machine running in $T(n)$ time has an equivalent Boolean circuit of size $\mathcal{O}(T(n)\log T(n))$, we have $P \subset BQP$.

Proof. In the register we use n bits for the input, m bits for the output, m bits for saving purposes, and the last S qubits as a scratchpad of the quantum operations. The scratchpad of a quantum operation is the qubit where the quantum operation writes its result.

Replace the S gates with their quantum equivalent noted in the previous section. Note that the AND and OR gates do not change their input. The NOT gate can be realized by first copying to the scratchpad than flipping the scratchpad bit, so it does not change its input either. It means that more quantum operations can read the same input.

Now we apply these quantum operations to the register. Then the register is mapped to:

$$|x\rangle|0^{2m+s}\rangle \mapsto |x\rangle|f(x)\rangle|0^m\rangle|z\rangle,$$

where z denotes the value of the value of the scratchpad in the end.

Now using the COPY operation, copy $f(x)$ to 0^m:

$$|x\rangle|f(x)\rangle|0^m\rangle|z\rangle \mapsto |x\rangle|f(x)\rangle|f(x\rangle|z\rangle.$$

Now apply the inverse of the S quantum gates in reverse order. As a result:

$$|x\rangle|f(x)\rangle|f(x\rangle|z\rangle \mapsto |x\rangle|0^m\rangle|f(x\rangle|0^S\rangle,$$

which gives the desired result. □

Corollary 3.3.4. *$P \subset BQP$*

Using the Hadamard operation we can simulate random bits: $|0\rangle$ is mapped to $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, so with $\frac{1}{2} - \frac{1}{2}$ probability we read $|0\rangle$ or $|1\rangle$. As a consequence:

Corollary 3.3.5. *$BPP \subset BQP$*

3.4 Grover's search algorithm

Grover's search algorithm is a quantum algorithm for finding a satisfying evaluation of a function $\varphi(x)$ given by a boolean circuit, when the solution of the formula is unique.

The general case, when $\varphi(x)$ may have more than one solution can be reduced to this case. In [VV86] Leslie Valiant and Vijay Vazirani showed that given a φ boolean circuit formula, there is a randomized algorithm that with $\Omega(1/n)$ probability transforms φ to a φ' formula with a unique solution so that it also satisfies φ. The general version of the problem is called the **search problem**.

The idea behind the search algorithm is to think of the evaluations as a superposition of n qubits. First we initialize the quantum register to the uniform distribution of the evaluations. One of these evaluations evaluate $f(x)$ to true, let it be y_0. In each step the the actual evaluation is rotated towards y_0. We repeat the process till we get close enough to y_0 so that when reading the register its value is y_0 with high probability.

Grover's search algorithm

Suppose $n \geq 3$. For $n = 1, 2$ we can try the two or four evaluations. Let $\varphi(x)$ denote a boolean circuit formula with y_0 unique solution. Let

$$u = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle$$

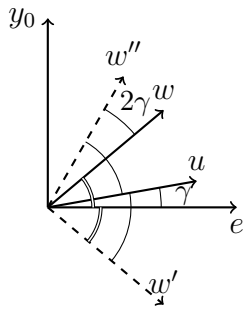
denote the uniform state of the quantum register (see Hadamard operation). Since $\langle u, y_0 \rangle = \frac{1}{2^{n/2}}$, the angle between u and y_0 is less than $\pi/2$. Let

$$e = \frac{1}{\sqrt{2^n - 1}} \sum_{x \neq y_0, x \in \{0,1\}^n} |x\rangle.$$

Note that e is on the plane spanned by u and y_0, and it is orthogonal to y_0.

The algorithm starts with $w = u$. Then in each step it rotates w toward $|y_0\rangle$ till the coefficient of y_0 in w is greater than $1/\sqrt{2}$, when the measurement collapses w to y_0 with $1/2$ probability.

The rotation is solved by two reflections. First we reflect w around e, resulting in w'. Then we reflect w' around u. After the two reflection w gets closer to $|y_0\rangle$ by 2γ, where γ is the angle between u and e.



Vector w after the two reflections.

Now we show how the reflections can be done by quantum computations.

Reflecting around e.

First we note the formula of the reflection. Suppose we want to reflect the vector $\vec{p}$ to vector $\vec{q}$. The vector $\vec{p}$ can be uniquely written as $\vec{p} = \alpha\vec{q} + \vec{q}^{\perp}$, where $\vec{q}^{\perp}$ is orthogonal to $\vec{q}$. Then the reflected vector is $\vec{p} = \alpha\vec{q} - \vec{q}^{\perp}$.

Using this formula the reflection of w around e is

$$\sum_{x \neq y_0} w_x |x\rangle - w_{y_0} |y_0\rangle.$$

We will calculate the reflection in three steps.

(i) In the first transformation we setup an indicator quantum bit that marks the satisfying evaluation. The transformation uses the boolean circuit φ itself, which is correct since φ is polynomial. Technically we should introduce a

scratchpad memory for the calculation of φ, but we leave it out for sake of simplicity as it does not make any difference:

$$|xz\rangle \mapsto |x\big(z \oplus (\varphi(x))\big)\rangle,$$

where x is n-long, and z is a single qubit initialized to $|0\rangle$. Note that it is exactly to copy the value of $\varphi(x)$ to z. Therefore the map:

- If $x \neq y_0$, then $|x0\rangle \mapsto |x0\rangle$.
- If $x = y_0$, then $|x0\rangle \mapsto |x1\rangle$.

Here we used that $\varphi(x)$ has unique solution, so it is only a permutation operation, so it is unitary. Otherwise the transformation would not be even invertible.

(ii) The second transformation is the following:

- $|x0\rangle \mapsto |x0\rangle$.
- $|x1\rangle \mapsto -|x1\rangle$.

This operation is also unitary.

(iii) Now we apply the first transformation again:

- If $x \neq y_0$, then $|x0\rangle \mapsto |x0\rangle$.
- If $x = y_0$, then $|x1\rangle \mapsto |x0\rangle$.

The resulting transformation: The vector $|xz\rangle$ is mapped itself for $x \neq y_0$ but $|y_0 0\rangle$ is mapped to $-|y_0 0\rangle$, which gives the reflection around $|e\rangle$ after truncating z, since the transformation does not alter the altitude of the coefficients.

Reflect around u.

We reduce the reflection around u to the reflection around $|00\ldots0\rangle$, which can be done the same way as reflecting around e by replacing φ to the function $\chi_{00\ldots0}$, i.e the indicator of evaluation $(00\ldots0)$. Let v_0 denote $v_0 = |00\ldots0\rangle$. Let $\mathcal{H}$ denote the Hadamard operation applied to every qubit in the input. By its definition we can see $\mathcal{H} = \mathcal{H}^{-1}$. Since $\mathcal{H}(y_0) = u$, we get $\mathcal{H}(u) = y_0$,

First we transform the space by $\mathcal{H}$, which maps u to y_0. After this we reflect the space to y_0 as mentioned above, then we apply $\mathcal{H}^{-1} = \mathcal{H}$. Finally w is reflected around u.

To see this, suppose $w = \alpha u + \beta u^{\perp}$, where $u^{\perp}$ is one-long vector and orthogonal to u. Applying $\mathcal{H}$ we get $w' = \alpha v_0 + \beta \mathcal{H}(u^{\perp}) = \alpha v_0 + \beta v_0^{\perp}$, where the first equation is

true because of the linearity, the second because of the angle-preservation (unitary). Applying T we get $w'' = \alpha v_0 - \beta v_0^{\perp}$, then applying $\mathcal{H}^{-1}$ we get $w''' = \alpha v_0 - \beta v_0^{\perp}$, which is exactly the rotated vector of w around u.

In summary, we can solve the reflection around u by applying $\mathcal{H}$ to w, then reflect to $|00\dots0\rangle$, then apply again $\mathcal{H}$.

Iteration number.

These reflections rotate the vector w 2γ closer to $|y_0\rangle$. So in $\mathcal{O}(1/\gamma)$ steps we get nearer to $|y_0\rangle$ than 2γ. Now we estimate the value of γ.

Note that e, y_0, u are in the same plane. The angle between u and y_0 is less than $\pi/2$, $y_0 \perp e$, and the angle between u and e is $\gamma \leq \frac{\pi}{2}$, meaning the angle between u and y_0 is $\pi/2 - \gamma$. Therefore

$$\frac{1}{2^{n/2}} = \langle u, y_0 \rangle = \cos\left(\frac{\pi}{2} - \gamma\right) = \sin(\gamma).$$

We know that for $0 \leq x \leq \pi/2 : (2/\pi) \cdot x \leq \sin(x) \leq x$. Putting γ into this we get $\gamma \geq 1/(2^{n/2})$ and $\gamma \leq \pi/(2 \cdot 2^{n/2})$

That means after the $\mathcal{O}(1/\gamma) = \mathcal{O}(2^{n/2})$ iterations the angle between y_0 and w is less than $2 \cdot \frac{\pi}{2 \cdot 2^{n/2}}$, so the probability to collapse to y_0 is $\cos^2\left(2 \cdot \frac{\pi}{2 \cdot 2^{n/2}}\right) \geq \cos^2(\pi/4) = 1/2$. This gives the desired result.

3.5 Quantum resistance

Grover's algorithm solves the search problem in $\mathcal{O}(2^{n/2})$ running time. Now we discuss a result which shows that this is the best possible in a certain sense.

Definition 3.5.1. *We call an $\mathcal{F} : A \mapsto B$ subroutine as a **random oracle**, if in one step for any input from A outputs an element of B uniformly in a consistent way: for a given $a \in A$, $\mathcal{F}$ outputs always the same $b = \mathcal{F}(a)$ whenever we invoke $\mathcal{F}$. So a random oracle can be imagined as a machine that for an unasked input chooses an output randomly, however stores the value, so later instead of generating again it outputs the stored value.*

The computational model where we can invoke a random oracle is called the **random oracle model**. The model without the hyphotesis of the existence of a random oracle is called the **standard model**. The following result in [BBBV97] shows that the asymptotics of the Grover's search algorithm is the best possible in the random oracle model.

Theorem 3.5.2. *For every $T(n) \in o(2^{n/2})$, relative to a random oracle chosen uniformly from the random oracles, with 1 probability, every NP problem cannot be solved in quantum-$T(n)$ time.*

The introduction of a random oracle to a computational model imports such "blackbox" functions that have no special properties because of the high-level randomness. These functions are get known only by evaluations. So in the random oracle model we can generate such formulae that cannot be solved by its specific properties.

In this sense the theorem above states that the quantum search algorithms which are purely based on the blackbox property of the formulae, cannot solve the search problem in $o(2^{n/2})$. It means that the search problem remains resistant in the quantum computational model.

This does not mean that we are sure there is no polynomial algorithm that solves the search problem, since the random oracle hypothesis is a strong tool.

Remark 3.5.3. *Note that the formula of the satisfiability problem (SAT) is also a boolean circuit formula. Therefore we believe that we cannot exponentially speedup the SAT problem only by quantum computers. This means the cryptographic schemes which security depends on NPC problems remain resistant.*

Remark 3.5.4. *Another consequence, since many hash functions we use can be imagined as boolean circuits, they cannot be reverted only by quantum computers.*

Therefore the cryptographic schemes which security purely depends on the pre-image property of the hash function believed to remain resistant.

Chapter 4

Resistant digital signatures

Digital signature schemes have a wide range of application areas. Browsers use them to verify the authenticity of web services, Intrusion Detection Systems (IDS) to check if a hacker changed system files, archiving and synchronizing algorithms to minimalize the amount of transmitted data or to verify their integrity, etc.

Since we use digital signature schemes in many situations, it is important that these schemes to be resistant to quantum computers in order to prevent potential corruption or forgery. In this thesis we will talk about signature schemes that are based on the Lamport signature scheme.

4.1 Lamport signature

Lamport signature scheme [Lam79],[Vil11] is a digital signature scheme. Since its reliability depends only on the properties of the hash function, it is believed to be resistant to quantum computers, as we mentioned in Remark 3.5.4.

The Lamport signature scheme is the following: Let f be a one-way function. Let n denote the length of the message digest n.

First we choose $2n$ random numbers (below a large p prime), these series will be the private keys.

$$S^0 = \left(S_1^0, S_2^0, \ldots, S_n^0\right),\ S^1 = \left(S_1^1, S_2^1, \ldots, S_n^1\right)$$

The public key will be the hash of the two lists:

$$f(S_1^0), f(S_2^0), \ldots, f(S_n^0),\ f(S_1^0), f(S_2^1), \ldots, f(S_n^1)$$

The signature K will be n-long, defined by the following: $0 \leq i < n$,

$$K_i = \begin{cases} S_i^0 & : M_i = 0 \\ S_i^1 & : M_i = 1 \end{cases}$$

So the S^0 and S^1 arrays are merged according to the message.

The verification of the signature is simple: we apply the f function to the signature and check if we got the values from the public keys.

If Mallory wants to compromise the signature, for instace he wants to change the i-th digit from 0 to 1 then he has to find an x so that $f(x) = f(a_i^1)$, and replace a_i^0 with x.

This shows that the security is based on the one-way property of the hash-function f.

One backdraw of the scheme is that the signature leaks out information about the private key, so we have to initialize new public and private key pairs each time we want to sign a message. A lot of key generation can lead to efficiency problems, moreover we have to keep track of every generated key. Another problem is that the size of the private key size is large. In the following section we will introduce two improvements suggested by Ralph Merkle for these problems.

4.2 Merkle one-time signature scheme

4.2.1 Merkle signature

Merkle one-time signature scheme is an improved version of the Lamport signature scheme: it reduces the size of the private key by about a factor of 2.

The idea behind the Merkle signature scheme is that in the Lamport signature scheme it is enough to sign the zeros and the number of zeros in the message, which can be done by adding $\lfloor \log n \rfloor + 1$ bits to the end of the message.

In the Merkle signature scheme we generate $l = n + \lfloor \log n \rfloor + 1$ random numbers below a large prime:

$$S = \big(S_1, S_2, \ldots, S_l\big)$$

S will be the private key. The public key will be the hash of the private key:

$$f(S_1), f(S_2), \ldots, f(S_l)$$

Let M' be the message digest extended by the number of zeros in binary representation.

The signature K will be l-long, signing M', defined by the following: $1 \leq i \leq l$

$$K_i = \begin{cases} S_i & : M'_i = 0 \\ f(S_i) & : M'_i = 1 \end{cases}$$

Suppose the attacker can change the ith digit from one to zero. Then he can find an x so that $f(x) = f(S_i)$, which contradicts to the pre-image resistance of the hash function.

Suppose the attacker wants to change the ith digit from zero to one in the message, then he will contradict to the checksum of the number of zeros. Since changing some zeros to one in the counter only increases its value, he cannot withdraw zeros from the message.

As a consequence the security of the Merkle signature scheme depends only on the pre-image resistance of the hash function.

Since the size of the private key has decreased from $2n$ to $n + \lfloor \log n \rfloor + 1$, the size of the private key is reduced by about a factor of two.

4.2.2 Merkle-Winternitz signature scheme

The size of the private keys can be further reduced in trade of performance by generalizing the technique above.

Above we signed every bits, which can be imagined as signing 1-long blocks. Instead of signing 1-long blocks we can sign k-long blocks by generating private random numbers for the block-value $\underbrace{00\ldots0}_{k}$ in each block. The signature of a block-value B_i will be $f^{B_i}(S_i)$, where $f^m = \underbrace{f \circ f \circ \cdots \circ f}_{m}$. The public key will be $f^{2^k-1}(S_i)$.

The verification process is again applying the f function to the blocks till we receive the public key, and checking if we receive the public key after the right iteration.

Here again an attacker cannot decrease the value of a block because of the pre-image resistance of the hash function. However there is a possibility again to increase the value of a block, so we have to add a checksum to the end of the message, similarly to the Merkle signature scheme: $\sum_{i=1}^{l}(2^k - 1 - B_i)$, where l is the number of blocks and the value of the ith block is B_i. We have to sign the checksum along with the message.

Since the time of signature and verification increases exponentially by k but the size of the private key is reduced approximately linearly, the optimal value of k depending on the hash function is small, usually 2,3,4.

4.2.3 Merke hash tree

Another problem with both Lamport and Merkle signature scheme that we have to use different public keys for different signatures.

The Merkle hash tree is a verification technique in order to prove that we used a public key from a given set. So instead of publishing every single public key, we

have to publish a proof that we used a legal public key.

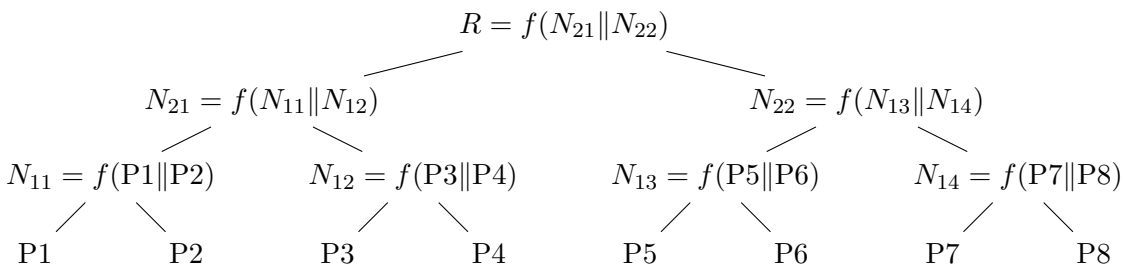
If we have generated 2^h public private keys in advance, then these keys can be accumulated into a tree so that finally it is enough to distribute a single key, the key of the root node.

The Merkle hash tree is a h-level binary tree, containing keys as follows: First generate 2^h public key/private key pairs for the Merkle signature scheme. These public keys will be the leaves of the Merkle hash tree. The key of a given non-leaf node is the hash of the concatenation of the keys stored by its two children. We only publish the value of the root node.

When we want to sign a message, we sign it with a generated private-public key, but we also prove that the signature is in the Merke hash tree by revealing the following nodes of the tree:

Suppose the unique path from the ith public key to the root node is l. We will validate l : we reveal the node values of l, and reveal the value of the sibling of $v \in l$.

The verification of l is the following: from the parent of the ith public key on every level we concatenate the values of the two children of the given node, generate its hash and check if we get the same node value as it was given in the signature.



However, for large h it is difficult to generate 2^n keys or store the hash tree. An idea to overcome the key generation problem is to use a seed based random number generator, and instead of storing all public key values we can store the seed. So the public key can be generated during the signature process, and not in advance.

The verification process can be also modified so that it is not necessary to store the whole tree, the signature of l can be calculated runtime. The detailed signature scheme can be found in [BDK$^+$].

By these technical modifications the authors achieved to sign messages with a $h = 80$ tall Merkle hash tree.

Chapter 5

Resistant encryption schemes

5.1 Lattice based encryption

5.1.1 Lattices

There are two equivalent definitions for lattices, a costructive and a structural definition.

Definition 5.1.1 (Generated lattice)**.** *Let $\{b_1, \ldots, b_k\} \subset \mathbb{R}^n$ be linearly independent vectors. The set $\mathcal{L} = \{\sum_{i=1}^{k} n_i b_i | n_1, \ldots, n_k \in \mathbb{Z}\}$ is called the **lattice generated by** $\{b_1, \ldots, b_k\}$. If $B \in \mathbb{R}^{n \times k}$ denotes the matrix which columns are b_i, then the generated lattice can be expressed as the following:*

$$\mathcal{L} = \{Bv : v \in \mathbb{Z}^k\}.$$

*B is called a **generator matrix** or **basis** of $\mathcal{L}$.*

Definition 5.1.2 (Lattice)**.** *A lattice is a discrete, additive subgroup of $\mathbb{R}^n$.*

Theorem 5.1.3. *$\mathcal{L}$ is a generated lattice $\Leftrightarrow$ $\mathcal{L}$ is a lattice.*

Proof. $\Rightarrow$ A generated lattice is an additive subgroup of $\mathbb{R}^n$, since multiplying by B is a linear transformation.

To prove that a generated lattice is discrete, we have to show that any bounded subset of $\mathbb{R}^n$ contains only finitely many lattice points.

Let B be a generator matrix of $\mathcal{L}$, and append vectors as columns to B till it becomes invertible. Denote the new matrix as Q, Q^{-1} its inverse matrix, and let $\alpha = \max\{|(Q^{-1})_{i,j}| : 1 \leq i, j \leq n\}$. Let $\mathcal{L}'$ denote the lattice generated by Q.

For a $z = Q\lambda \in \mathcal{L}'$ we have $\lambda = Q^{-1}z$, therefore $|\lambda_i| \leq \alpha n \|z\|$. For any bounded subset of $\mathbb{R}^n$ the norm $\|z\|$ is bounded, so the integer coefficients of λ is below

a dimension-dependent constant. Therefore the bounded subset can only contain finitely many lattice points from $\mathcal{L}'$, so from $\mathcal{L}$ too.

$\Leftarrow$ Suppose $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^n$. Let $g_1, \ldots, g_k$ denote a basis of the subspace generated by $\mathcal{L}$. Let

$$X = \{x \in \mathbb{R} : x = \sum_{i=1}^{k} \lambda_i g_i, 0 \leq \lambda_i < 1, \lambda_i \in \mathbb{R}\}.$$

Since X is bounded, $X \cap \mathcal{L}$ is finite. Choose the basis $g_1, \ldots, g_k$ so that their integer combination $\mathcal{L}_g$ is a maximal subset of $\mathcal{L}$: there is no $G' = (g'_1, \ldots, g'_k)$ so that $L_g \subsetneqq \{G'v : v \in \mathbb{Z}^k\}$.

Lemma 5.1.4. $X \cap \mathcal{L} = \{0\}$.

Proof. By contradiction suppose $\exists x^* \neq 0 : x^* \in X \cap \mathcal{L}$. Let

$$x^* = \sum_{i=1}^{k} \lambda_i g_i,$$

where λ is lexicographically minimal. There exists such x^* since $X \cap \mathcal{L}$ is finite. Let λ_i be the minimal nonzero coefficient in λ, and denote m the integer $1/\lambda_i \leq m < 1/\lambda_i + 1$. For this m we have $(m-1)\lambda_i < 1$ and $m\lambda_i \geq 1$. Let

$$z = (m\lambda_i - 1)g_i + \sum_{j=i+1}^{k} (m\lambda_j - \lfloor m\lambda_j \rfloor)g_j \in X.$$

From $(m-1)\lambda_i < 1$, we have $(m\lambda_i - 1) < \lambda_i$, so $z <_{\text{lex}} \lambda$, which means $z = 0$. So the coefficients of z is zero.

Putting this into x^*:

$$\begin{aligned} x^* &= \frac{1}{m} g_i + \sum_{j=i+1}^{k} \frac{\lfloor m\lambda_j \rfloor}{m} g_j \\ mx^* &= g_i + \sum_{j=i+1}^{k} (\lfloor m\lambda_j \rfloor) g_j \\ g_i &= mx^* - \sum_{j=i+1}^{k} (\lfloor m\lambda_j \rfloor) g_j \end{aligned}$$

Note that we cannot generate x^* as the integer combination of $g_1, \ldots, g_k$, since we can generate it by a non-integer combination, and by the independence of $g_1, \ldots, g_k$ the generating coefficients are unique.

That means if we substitute g_i by x^*, we can generate a strictly larger subset of $\mathcal{L}$, which is a contradiction. $\square$

We finish the proof by showing that $\mathcal{L}$ is the lattice generated by $g_1, \ldots, g_k$. Let $v = \sum_{i=1}^{k} \lambda_i g_i$. Since $g_1, \ldots, g_k$ generates the subspace spanned by $\mathcal{L}$, it is enough to show that $v \in \mathcal{L}$. The vector v can be written as

$$\begin{aligned} v &= \sum_{i=1}^{k} (\lfloor \lambda_i \rfloor) g_i \quad + \quad \sum_{i=1}^{k} (\lambda_i - \lfloor \lambda_i \rfloor) g_i \\ X \ni \quad \sum_{i=1}^{k} (\lambda_i - \lfloor \lambda_i \rfloor) g_i \quad &= \quad v - \sum_{i=1}^{k} (\lfloor \lambda_i \rfloor) g_i \quad \in \mathcal{L}. \end{aligned}$$

Therefore $\sum_{i=1}^{k} (\lambda_i - \lfloor \lambda_i \rfloor) g_i \in X \cap \mathcal{L} = \{0\}$. So $v = \sum_{i=1}^{k} (\lfloor \lambda_i \rfloor) g_i$ is a linear integer combination of $g_1, \ldots, g_k$. □

If you generate a lattice with two different bases, then the number of basis vectors in the bases are the same.

Suppose there exist two bases with different sizes. Then the elements of the larger basis can be expressed as the linear (integer) combination of the elements of the smaller basis, contradicting the linear independency.

So the dimension of the bases of a lattice is invariant, which leads to

Definition 5.1.5 (Dimension)**.** *If $\mathcal{B}$ generates the $\mathcal{L} \subset \mathbb{R}^n$ lattice, then the* ***dimension*** *of $\mathcal{L}$ is $\dim \mathcal{L} = |\mathcal{B}|$. If $\dim \mathcal{L} = n$, the $\mathcal{L}$ is called full-rank lattice.*

From now on we suppose that all lattices are full-rank lattices, so their bases are represented as invertible square matrices.

If $n > 1$, then every lattice has infinitely many bases. The following theorem shows a connection between those bases.

Theorem 5.1.6. *B and B' are bases of the same lattice $\Leftrightarrow \exists U : B' = B \cdot U$, where U is an integer matrix and $\det(U) = \pm 1$.*

Proof.

$\Leftarrow$ Let B be a basis of $\mathcal{L}$, let U be integer matrix with $\det(U) = \pm 1$. Denote $\mathcal{L}'$ the lattice generated by BU. We have to show that $\mathcal{L} = \mathcal{L}'$.

Suppose $v \in \mathcal{L}$, i.e. $\exists \lambda : B\lambda = v$. Since the determinant is ± 1 of the U integer matrix, its inverse matrix is also integer. That means the equation $U\mu = \lambda$ has an integer solution for μ. Since $(BU)\mu = B(U\mu) = B\lambda = v$, $v \in \mathcal{L}'$, therefore $\mathcal{L} \subseteq L'$.

Suppose $v \in \mathcal{L}'$. That means $\exists \mu$ so that $BU\mu = v$. With $\lambda = U\mu$, we get $B\lambda = v$, so $\mathcal{L}' \subseteq \mathcal{L}$.

$\Rightarrow$ Let B and B' be different bases of $\mathcal{L}$. Since the basis elements are in $\mathcal{L}$, there exist U, U' integer matrices so that $B = B'U'$, $B' = BU$. Joining the two equations:

$B(UU') = B$, therefore $1 = \det(UU') = \det(U)\det(U')$. As the determinant of an integer matrix is integer, $\det(U) = \pm 1$. □

As a corollary, not only the dimension of a lattice is invariant but also the determinant of the generator matrices.

Corollary 5.1.7. *If B and B' are bases of the same $\mathcal{L}$ lattice, then* $\det B = \det B'$. *The common value of the determinants is noted as* $\mathrm{vol}(\mathcal{L})$.

Although there are many different bases for the same lattice, it is useful to know a basis which has nearly orthogonal and short vectors.

One indices of quality of a basis is the weight:

Definition 5.1.8 (Weight)**.** *Let $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ be a basis of $\mathcal{L}$. Then the* ***weight*** *of $\mathcal{B}$ is:*

$$\|B\| = \prod_{i=1}^{n} \|b_i\|$$

Note that if the basis vectors are orthogonal, then the weight is equal to the determinant of the lattice.

If $\mathcal{B}$ and $\mathcal{B}'$ are bases of the same $\mathcal{L}$ and $\|B\| > \|B'\|$, then we say $\mathcal{B}'$ is *reduced relative to* $\mathcal{B}$.

Since the determinant of the vectors of a parallelepipedon is smaller than the product of their length, we get that $\mathrm{vol}(\mathcal{L})$ is a lower bound to the weight.

Theorem 5.1.9 (Hadamard)**.**

$$\|B\| \geq \mathrm{vol}(L),$$

There is equality if and only if the columns of B are orthogonal.

The following theorem of Minkowski states that there are no arbitrary large gaps in a lattice. Using this theorem we can give an upper bound to the shortest vector of a lattice.

Theorem 5.1.10. *Let $\mathcal{L}$ be a lattice so that* $\dim \mathcal{L} = n$. *Then every compact convex symmetric region of which volume is at least* $2^n \mathrm{vol}(\mathcal{L})$ *contains a nonzero lattice point.*

Proof. We prove the theorem in two parts, first for sets where $\mathrm{vol}(A) > 2^n \mathrm{vol}(\mathcal{L})$, then by compactness where $\mathrm{vol}(A) = 2^n \mathrm{vol}(\mathcal{L})$.

Suppose $A \subset \mathbb{R}$ is a compact, convex, symmetric set so that $\mathrm{vol}(A) > 2^n \mathrm{vol}(\mathcal{L})$. Let $g_1, \ldots, g_n$ be an arbitrary basis of $\mathcal{L}$, and

$$X = \{a_1 g_1 + \cdots + a_n g_n : 0 \leq a_i < 1, 1 \leq i \leq n\}.$$

If we shift A with all lattice vectors, it gives a cover of $\mathbb{R}^n$. Specially A is also covered, so $\forall a \in A$ can be written as $a = l + x$ where $l \in \mathcal{L}, x \in X$. We show that this decomposition is unique.

Suppose for $x = \sum x_i g_i \in X,\ x' = \sum x'_i g_i \in X,\ l = \sum l_i g_i \in \mathcal{L},\ l' = \sum l'_i g_i \in \mathcal{L}$ we have $l + x = l' + x'$. Since the basis vectors are linearly independent, we have $l_i + x_i = l'_i + x'_i$, meaning $x_i - x'_i$ is an integer, but $-1 < x_i - x'_i < 1$, so $x_i = x'_i$. Therefore $x = x', l = l'$. As a result we can define the following function:

$$g : A \to X, a \mapsto x.$$

Now we shrink A by 2. The volume of $(1/2)A = \{(1/2)a \in A\}$ is $\mathrm{vol}((1/2)A) = (1/2^n)\,\mathrm{vol}(\mathcal{A}) > \mathrm{vol}(\mathcal{L})$ by our assumption. As $\mathrm{vol}(\mathcal{L})$ is the volume of the parallelepipedon X, $\mathrm{vol}(\mathcal{L}) = \mathrm{vol}(X)$, so we get $\mathrm{vol}((1/2)A) > \mathrm{vol}(X)$. Therefore there are two points (a_1, a_2) in $\frac{1}{2}A$ which is mapped by $g|_{(1/2)A}$ to the same point $x \in X$: $\exists l_1, l_2 \in \mathcal{L}$ so that $a_1 = l_1 + x, a_2 = l_2 + x$.

After substraction: $0 \neq a_1 - a_2 = l_1 - l_2 \in \mathcal{L}$. Since $a_1, a_2 \in (1/2)A$ we have $2a_1, 2a_2 \in A$, and by simmetry $2a_1, -2a_2 \in A$. Since the mid-point of the segment $(2a_1, -2a_2)$ is $a_1 - a_2$ and A is convex, we have $a_1 - a_2 \in A$. That means $l_1 - l_2$ is a lattice point in A, proving the first part of the theorem.

Suppose now $\mathrm{vol}(A) = 2^n\,\mathrm{vol}(\mathcal{L})$. Then we generate nonzero l_k lattice points for regions $(1 + 1/k)A$ using the first part, since $(1 + 1/k)A$ is also compact, convex subset of $\mathbb{R}^n$, but its volume is $\mathrm{vol}((1 + (1/k))X) > 2^n\,\mathrm{vol}(\mathcal{L})$. Since the series $(l_i)_i \subset 2A$, which is bounded subset of $\mathbb{R}^n$, $(l_i)_i$ consists of only finitely many lattice points. Since by compactness $\bigcap_{k=1}^{n} (1 + 1/k)\, A = A$, there must be a nonzero lattice point in A. □

As a corollary of Minkowski theorem we have an upper bound for the shortest vector of $\mathcal{L}$.

Corollary 5.1.11. *If $\mathcal{L}$ is a lattice, then there is a $v \in \mathcal{L}$ so that*

$$\|v\| \leq c_n\,\mathrm{vol}(L)^{\frac{1}{n}},$$

where c_n is a dimension-dependent constant.

Proof. Let $R_r^n = \{x \in \mathbb{R}^n : \|x| \leq r\}$. It is known that

$$\mathrm{vol}(R_r^n) \sim \left(\frac{2\pi e}{n}\right)^{n/2} r^n,$$

therefore exists C_1, C_2 for every $n \in \mathbb{N}$:

$$C_1 \left(\frac{2\pi e}{n}\right)^{n/2} r^n \leq \mathrm{vol}(R_r^n) \leq C_2 \left(\frac{2\pi e}{n}\right)^{n/2} r^n$$

So choosing $r = \frac{1}{\sqrt[n]{C_1}}\sqrt{\frac{2n}{\pi e}}\,\mathrm{vol}(\mathcal{L})^{1/n}$, we have $\mathrm{vol}(R_r^n) \geq 2^n\,\mathrm{vol}(\mathcal{L})$. From Minkowski theorem there is a nonzero lattice point in R_r^n, so the length of the shortest nonzero vector is less than $\frac{1}{\sqrt[n]{C_1}}\sqrt{\frac{2n}{\pi e}}\,\mathrm{vol}(\mathcal{L})^{1/n} = c_n \cdot \mathrm{vol}(\mathcal{L})^{1/n}$.

□

5.1.2 Lattice based encryption

In this section we introduce two encryption schemes which are considered to be resistant to quantum computers. The first scheme is the Goldreich-Goldwasser-Halevi (GGH) encryption scheme, which is based on solving the closest vector problem. The second is the NTRU cryptosystem which works with polynomials, however it is strongly related to lattices. Its security is based on the shortest vector problem.

Problem 5.1.12 (Closest vector problem, CVP)**.** *Given an $\mathcal{L}$ lattice and $v_0 \in \mathbb{R}^n$, find a $v \in \mathcal{L}$ so that $\|v - v_0\|$ is minimal.*

Problem 5.1.13 (Shortest vector problem, SVP)**.** *Given an $\mathcal{L}$ lattice, find a nonzero $v \in \mathcal{L}$ so that $\|v\|$ is minimal.*

It is proven that the SVP is NP-hard, and the CVP is at least as hard as the SVP. As we mentioned in Remark 3.5.3, an NPC problem is believed to be quantum resistant. Since NP-hard problems are more difficult than the NPC problems, they are also believed to be quantum resistant.

Practically if we want to solve the SVP or the CVP, it is useful to find a basis so that has short and nearly orthogonal vectors. Changing a basis for a better basis is called lattice reduction.

A well-known lattice reduction algorithm is the Lenstra-Lenstra-Lovász (LLL) algorithm. It was the first algorithm that could find good lattice reduction in polynomial time. Although the LLL algorithm finds relatively long basis vectors, it is good enough to break some lattice based cryptographic systems.

5.1.3 GGH cryptographic scheme

Let $\mathcal{L}$ be a lattice. The private key is a basis $V = \{v_1, \ldots, v_n\}$ with small and nearly orthogonal basis vectors, the public key is a basis $W = \{w_1, \ldots, w_n\}$ with long and less orthogonal vectors. The idea behind the scheme is to choose the basis V so that we can solve the CVP problem with it, but with W we cannot.

Let m be an n-long binary vector, the text we want to cypher. To cypher the m

message, we calculate the vector e, which will be the cyphertext:

$$e = r + \sum_{i=1}^{n} m_i w_i,$$

where r is a small random perturbation vector.

During decryption we calculate the closest x lattice vector to e. That removes the noise, so we have

$$x = \sum_{i=1}^{n} m_i w_i.$$

Hence we can calculate the coefficients by solving this equation.

To calculate the shortest vector x we will use the private key. First we solve the equation

$$e = \sum_{i=1}^{n} y_i v_i,$$

where $y_i \in \mathbb{R}$, so the coefficients are real. Rounding the coefficients to the nearest integer we get

$$u = \sum_{i=1}^{n} \lfloor y_i \rceil v_i,$$

which is the closest vector if V is chosen well enough.

For small dimensions the LLL algorithm can reduce W so that it can break the scheme, for moderate dimensions there are variants for LLL that break the system. Therefore in order to achieve reasonably high security we have to use so high dimensional vectors that might be impractical.

The size of the keys in the cryptographic scheme is $\mathcal{O}(n^2)$.

5.2 Polynomial based encryption

5.2.1 NTRU cryptosystem

The cryptosystem works in the ring $R = \mathbb{Z}_r[X]/(x^N - 1)$, where N is a prime. This ring is a polynomial ring, where the coefficients are from $\mathbb{Z}_r$, with the rule $x^N \equiv 1$. Therefore any element of this ring can be reduced to a polynomial at most $N - 1$ degree.

In R we can solve addition and multiplication, and by the Euclidean algorithm the inverse calculation.

Key generation Fix an N prime, $p < q$ integers so that $\gcd(p, q) = 1$. Usually q is chosen as a power of 2, and p very small. Choose random $f, g \in R$ polynomials with

coefficients from $[-d, d]$, where d is chosen small enough (e.g. $(p+1)d^2 \leq (1/2)q$). Let the inverses be $F_q \equiv f^{-1} \mod q$ and $F_p \equiv f^{-1} \mod p$.

- Public key: $h = g \cdot F_q \mod q$.
- Private key: f.

Let the message we want to encrypt be an m polynomial, and choose a random small r polynomial for noise. Both these polynomials are with coefficients from $[-d, d]$. The encrypted text will be

$$e = p(r \cdot h) + m \mod q.$$

Decryption: Compute $a \equiv e \cdot f \mod q$ so that the coefficients of a are chosen from $[-(1/2)q, (1/2)q]$ instead of $(0, \ldots, p-1)$.

Statement 5.2.1. *$F_p \cdot a \mod p$ is equal to the plaintext m.*

Proof.

$$\begin{aligned} a &\equiv e \cdot f \mod q \\ &\equiv (p(r \cdot h) + m) \cdot f \mod q \\ &\equiv p(r \cdot h \cdot f) + m \cdot f \mod q \\ &\equiv (pr) \cdot (g \cdot F_q) \cdot f + m \cdot f \mod q \\ &\equiv p(r \cdot g) + m \cdot f \mod q. \end{aligned}$$

Since we chose the coefficients of r, g, m, f from $[-d, d]$, after computing the convolutions and the sum, the coefficients of $p(r \cdot g) + m \cdot f$ are in $[-(1/2)q, (1/2)q]$. So the equation is true not only in $\mathbb{Z}_q[x]$ but in $\mathbb{Z}[x]$.

$$a = p(r \cdot g) + m \cdot f.$$

Now if we multiply a by F_p, we get

$$\begin{aligned} F_p \cdot a &= F_p \cdot (p(r \cdot g) + m \cdot f) \mod p \\ &\equiv F_p \cdot m \cdot f \mod p \\ &\equiv m \mod p \end{aligned}$$

□

This algorithm is efficient, the encryption/decryption is in $\mathcal{O}(n \log n)$ using Fast-Fourier transformation for polynomial multiplication.

Now we discuss how we can use the CVP problem to acquire the private key f.

5.2.2 Connection to lattices

We will follow the deduction as in [HPS98]. Let denote the following matrix:

$$B = \left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & \alpha & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & \mathrm{q} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \mathrm{q} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & \mathrm{q} \end{array}\right)$$

Consider the lattice $\mathcal{L}$ generated by the rows of B, where h denotes the public key of the NTRU algorithm, q is the corresponding parameter of the cryptosystem, and a is a parameter of the lattice we will choose later.

Note that since $h = g \cdot F_q = g \cdot (f^{-1}) \mod q$, the vector $\mu = (\alpha f, g)$ is in the lattice. If we could somehow recover this vector from the lattice then we could acquire the private keys.

We will set the parameter α so that μ is the shortest vector in $\mathcal{L}$ with high probability. From the Gaussian heuristic we can estimate the length of the shortest vector $\lambda(\mathcal{L})$:

$$(\det B)^{\frac{1}{n}} \sqrt{\frac{n}{2\pi e}} \leq \lambda(\mathcal{L}) \leq (\det B)^{\frac{1}{n}} \sqrt{\frac{n}{2\pi e}},$$

where $n = 2N, \det(B) = q^N \alpha^N$. Putting this together we have that the length of the smallest vector in $\mathcal{L}$ is

$$s = \sqrt{\frac{N\alpha q}{2\pi e}}$$

So in order to let μ be the shortest vector in $\mathcal{L}$, we have to maximalize $s/\|\mu\|_2$ in α.

$$\left(\frac{s}{\|\mu\|}\right)^2 = \frac{\alpha}{\alpha^2 \|f\|_2^2 + \|g\|_2^2} = (\alpha \|f\|_2^2 + \alpha^{-1} \|g\|_2^2)^{-1}$$

The latter formula is maximalized when $\alpha = \|g\|_2 / \|f\|_2$. Finally we estimate $\|g\|_2, \|f\|_2$ using the following:

Proposition 5.2.2. *For any $\varepsilon > 0$ there are γ_1, γ_2 constants depending on ε, N so that for randomly chosen polynomials $\gamma_1 \|f\|_2 \|g\|_2 \leq \|f \cdot g\|_\infty \leq \gamma_2 \|f\|_2 \|g\|_2$.*

Practical experiments show that γ_2/γ_1 is close to 1, therefore we can estimate $\|g\|_2, / \|f\|_2$ from h. Putting the estimation into α we have that μ is with high probability the shortest vector in $\mathcal{L}$.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, October 1997.

[BDK+] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, Camille Vuillaume, and Technische Universität Darmstadt. Merkle signatures with virtually unlimited signature capacity.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. *ANTS*, pages 267–288, 1998.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International, October 1979.

[LJMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. *STOC*, pages 564–572, 1990.

[Mat03] D. V. Matyukhin. On asymptotic complexity of computing discrete logarithms over $gf(p)$. *Discrete Mathematics and Applications*, 13, 2003.

[Ö00] Bernhard Ömer. Quantum programming in qcl. *Master thesis, computing science*, 2000.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[Vil11] V. I. Villányi. Signature schemes in single and multi-user settings. *ProQuest, UMI Dissertation Publishing*, 2011.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. Np is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

NYILATKOZAT

Név:
ELTE Természettudományi Kar, szak:
ETR azonosító:
Szakdolgozat címe:

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 20 _____

a hallgató aláírása