Code-based cryptography: A superficial introduction

Master's Thesis in Informatics
Department of Informatics
University of Bergen

Candidate: Håkon Gjeraker Østerhus
Supervisor: Øyvind Ytrehus

September 2018

Abstract

This thesis provides an overview of some known post-quantum cryptography schemes with special attention given to code-based schemes. It also discusses some challenges that may arise trying to implement a McEliece-like cryptosystem.

The last part of the thesis examines the code-based submissions to NIST's post-quantum cryptography standardization process.

Abstrakt (Norwegian)

Denne oppgaven gir en oversikt over noen kjente postkvantekryptosystemer med spesielt fokus på kodebaserte systemer. Den diskuterer også noen utfordringer som kan oppstå når et McEliece lignende kryptosystem blir forsøkt implementert.

Den siste delen av oppgaven tar for seg de kodebaserte innleveringene til NISTs postkvantekryptografistandardiseringsprosess.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to cryptography

Communication occurs everywhere, in everything from human conversations to computers transmitting binary data over a communication link. Even yawning while reading this thesis can be regarded as a form for communication, signaling to the world that one would rather like to spend the time watching movies or browsing social medias, which again are forms of communication. Common to all forms is that information is being transferred from one party to another. "Another" may be a little misleading here because, in fact, a very common form for communication is from one party to itself; for instance, writing a list of things one needs to remember, and then recollect them by reading the list later.

Communication can be regarded as a system of components: a source of information, a channel or medium conveying the information, and receivers of the information. In this section the set of receivers is referred to as the audience of the communication. Sometimes the audience of the communication medium is larger than what is desired and may contain unknown receivers. The source would like to confide only to a subset of the audience. If the source trust everyone in its audience to a sufficient degree, it may first persuade the unwanted receivers to leave or simply overlook the secret information. However, in many scenarios, such as when the audience contains unknown receivers, the source naturally shouldn't trust all of them. Some receivers can betray and, depending on the circumstances, history spanning several millennia strongly indicates they will most definitely betray. The need for more sophisticated methods of transmitting secret messages are needed which is the study of cryptography.

What cryptographers have noticed is that the receivers may interpret the messages differently. For some, the message makes perfect sense and all the information that the message was meant to carry were extracted successfully by the receiver. Others are only able to understand some parts of the message and, in some cases, there are receivers who cannot distinguish the message from random noise and consequently aren't able to extract any sensible information from the message at all. As a completely made up example, Norwegians is a population known for their love of traveling abroad and also, maybe somewhat less known, is their love for secret gossiping. A few years ago, the neighboring countries Sweden and Denmark were by far the most popular destinations among

Norwegians, but since the Swedish and Danish languages have so much in common with Norwegian, the Norwegians soon realized that their secret gossiping weren't very secret after all. Norwegians eventually started traveling further, maybe with the goal of reaching areas whose language had less similarities with Norwegian.

In general, if some parties share some knowledge which are unknown for the other parties, this knowledge can be used to transfer information that only makes sense for these parties. This shared knowledge model is these days commonly referred to as symmetric cryptography and the shared secret knowledge is called the symmetric key. Furthermore, there is an encryption function taking the symmetric key as well as the information one wants to encrypt and maps it to a ciphered message. Lastly there is a decryption function taking the symmetric key in addition to the ciphered message which provides the deciphered information. Of course, it is absolutely possible to regard the encryption and decryption function themselves as part of the shared secret as well, which is indeed the case for some systems today.

Sometimes one wants to communicate with parties where no shared secrets are initially present. For this purpose, the relatively recent field of public key cryptography, also called asymmetric cryptography, is studied. A Public Key Cryptosystem (PKC) also has an encryption and decryption function, just as in the symmetric case. However, in public key cryptography, the keys used for encryption and decryption differ. One of the keys is considered public and is therefore called the public key. The other key is private and similarly called the private key. The idea is that if one party wants to send a message to another party. A public key is first requested from the recipient. If the recipient hasn't already done so a key pair is generated. Anyhow, the public key is transmitted to the other party fully visible to everyone in the audience. Now the public key is used to encrypt the information at the source and then transmitted as ciphered through the medium. The encryption partially applied with the public key is a "trapdoor function", not in the sense that it is theoretically impossible to find an inverse, but doing so should take so much time and power that it is practically infeasible. Therefore, only the party with the corresponding private key is able to decrypt the ciphered message.

There are several potential problems here; like for instance the possibility of an adversary injecting its own public key instead of the one that was requested. As a consequence, public key schemes require reliable methods for authenticating parties and their public keys. Another problem is that most PKCs are slow and consumes much more resources than a typical symmetric system. Hence, the use of PKCs in a lengthy communication is usually limited to the first step of agreeing upon the symmetric key that is going to be used for the rest of the session.

Additionally, a prominent use case of public key cryptography is for signing messages, i.e. guaranteeing that the message received is indeed from the proclaimed source. In this case the private key is used to produce a signature of the message, often in combination with a hash-function, and the public key is used to validate the signature. Unfortunately, not all asymmetric cryptosystems are well suited to be used in a signature scheme.

1.2 Properties of PKCs

All PKCs depends on trapdoor functions. A trapdoor function is a function taking a key and one or more additional arguments, with the restriction that the partial application of the key will return an injective function that is believed to be one-way, unless some additional secret information called the trapdoor is known. Furthermore, a one-way function is a function that is easy to compute on every input, but inverting the image of a random input is hard. Whether or not one-way functions actually exist is an unsolved question and therefore any security deductions regarding PKCs are under the assumption that such one-way functions exist.

Some encryption schemes are said to be probabilistic. This means that the encryption function, in addition to the message, takes one or more secret random numbers as arguments, and these random numbers affect the resulting ciphertext. Encrypting the same plaintext multiple times should then, with very high probability, give different ciphertexts. As a consequence, the number of valid ciphertexts for any probabilistic encryption is much larger than the number of valid plaintexts.

1.2.1 Some hard problems giving rise to trapdoor functions

Discrete logarithm The Discrete Logarithm Problem (DLP) can be formulated as follows: Given a prime modulus p, a primitive $g \in \mathbb{Z}_p^\star$, and some $g^x \in \mathbb{Z}_p^\star$, finding an integer x such that $x = \log_g g^x$ is hard to compute, especially if p and x is large.

Now assume one is given an integer a and $g, g^b, m \in \mathbb{Z}_p^\star$ for some unknown integer b, then it is easy to compute $\left(g^a, m \cdot g^{ab}\right)$, but given only $g, g^a g^b \in \mathbb{Z}_p^\star$ for unknown integers a and b it is hard to find m. If one knows either a or b, then g^{ab} can easily be found as either $\left(g^a\right)^b$ or $\left(g^b\right)^a$, the inverse of which can be used to find m. Notice that if DLP was easily solvable, one could find a and b from g^a and g^b. This is the trapdoor function used in the ElGamal cryptosystem.

Integer factorization Let n be the product of two large primes p and q. Factoring n into its prime components p and q is hard. This is called the integer factorization problem (IFP)

Euler's totient function can be computed efficiently for n as $\Phi(n) = (p-1)(q-1)$ if p and q is known. If $d \cdot e \equiv 1 \mod \Phi(n)$ for some integers d and e then, for any integer m, the following holds $(m^e)^d \equiv m \mod n$[24]. If one knew e, n and some m then one could easily compute $m^e \mod n$, however to get back to m the knowledge of d is needed which again requires p and q which are hard to find from n.

This is the trapdoor function used in the RSA cryptosystem[24]. The trapdoor itself does not take any random input, and so if e is the public key, then a message m is encrypted to the same ciphertext every time it occurs.

1.2.2 Security

The most important feature of a cryptosystem should obviously be its security. What level of security that is required can vary, and in some cases a lower

security level is adequate.

The french cryptographer Auguste Kerckhoffs wrote six basic principles for cryptosystems in his paper published in 1883 [14]. While the four last principles are strongly influence by that time's operational limitations and thus not especially applicable today, the first two are still highly relevant.

The first principle: "Le système doit être matériellement, sinon mathématiquement, indéchiffrable" says that the system must be substantially, if not mathematically undecipherable. When encrypting a message using a public key, one would like that only the owner of the associated private key is able to decrypt the ciphertext. This property is called confidentiality of the system and can be challenged in several ways.

First, an adversary can try to guess exactly what the private key looks like and create a perfect copy which can then be used to decrypt the ciphertext. Depending on how large the private key space is, the chance of guessing an exact match can be incredibly low. In some cryptosystems there may be multiple keys that can be used to decrypt the ciphertext. Another possibility is to iterate over all possible keys trying to find a sufficient key, which is commonly called a brute force attack.

It is hard to find the correct key if one has no way of determining if a given key is right or not. This can be done by trying to decrypt the ciphertext using the key in question and see if the result is sensible. However, this does not rule out the possibility that the key may be a false positive. If an attacker also knows something about the context of the communication, then some words in the message space may give more sense than others giving a stronger indication of which key was correct or not. Trying the key on multiple ciphertext blocks can also help eliminating false positives.

Using the public key, one can encrypt a known plaintext and then iterate over all private keys to find one that decrypts the generated ciphertext into the known plaintext. This key can then be used decrypt the ciphertext corresponding to the unknown plaintext.

If the key space is larger than the message space and it is known that the system is not probabilistic, then it can be more efficient to instead try to guess what plaintext was sent and then generate the corresponding ciphertext to check if it matches. This attack is weaker than the previous because it does not reveal the secret key, and hence does not provide decryptions for previous or future encryptions. On the other hand, sometimes one bullet is enough. This form of attack can be prevented by introducing some level of randomness in the encryption process, such that two encryptions of the same plaintext have a low probability of being equal.

Kerchoffs' second principle, also known as *the* Kerchoffs' principle: "Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi" says that the system must not require secrecy and that it safely can fall into the hands of the enemy. In the introduction it was mentioned that in some cases the workings of a cryptosystem itself are kept secret, however, following Kerckhoffs' principle it is rare for cryptanalysts to regard this secrecy as adding any value to the security of the system.

Attack goals Usually the attacks which are able to obtain the secret key is considered the strongest, but they can also be rather difficult to accomplish.

Other, less ambitious, attacks may try to only decrypt a single ciphertext block, alter message content or even just distinguishing ciphertext blocks.

Passive and active attacks Attacks can be placed in two main categories: passive and active. Attacks that in some way change the communication channel between the communicating parties, modifies messages under transfer, creates new, redirects or blocks messages are called active attacks, while eavesdropping and other non-intrusive methods are called passive attacks. The task of the cryptosystem is not to prevent such attacks from happening, but rather limit the amount of information an attacker is able to obtain, i.e. maintain the confidentiality of the messages.

Attack models Attack models is a way to categorize different forms of attacks after which kind of capabilities the adversary is given. The following list covers some common attack models.

- The Ciphertext-Only Attack (COA) model assumes only access to some ciphertext blocks. A notorious example of a COA is a brute force attack with complexity of $O\left(2^{|K_{\text{priv}}|}\right)$.
- The Known-Plaintext-Attack (KPA) model assumes access to pairs of related ciphertext and plaintext blocks. A variant of this model is the Partial-known-Plaintext Attack (PPA) where only parts of the related plaintext blocks are known.
- The plain Chosen-Plaintext Attack (CPA) model assumes everything from KPA, but in addition the attacker is allowed to request a group of plaintext blocks to be encrypted in one batch only once. Here, a batch means that all the plaintext blocks must be submitted to the encryption oracle as a group and is also returned as such. Thus an attacker can not adapt the requests to the oracle in response to previous requests. It is also expected that the batch requested is of a reasonable size.
- The adaptive Chosen-Plaintext Attack (CPA2) model is very similar to CPA with the only difference being that multiple batches can be submitted to the encryption oracle. As a consequence the attacker can adapt the requests by observing the results of previous requests.
- In addition to the assumption in of CPA2, the Chosen-Ciphertext Attack (CCA) model gives the attacker the possibility to request decryptions of a group of ciphertext in a single batch from a decryption oracle. If an attack tries to find the corresponding plaintext of some ciphertext, then the request to the decryption oracle must be done before the ciphertext in question is known.
- The Adaptive Chosen-Chipertext Attack (CCA2) model lets the attacker submit multiple batches to the decryption oracle. This way the attacker can adapt the requests in response to the results of previous ones. Obviously, if an attack tries to find the corresponding plaintext of some ciphertext, then the decryption of this ciphertext cannot be requested from the oracle. A stricter variant of this attack model called the Reaction Attack

Model	Access			
	Ciphertext	Plaintext	Encryption	Decryption
COA	Some	None	None	None
PPA	Some	Partial	None	None
KPA	Some	Some	None	None
CPA	Some	Some	Single batch	None
CPA2	Some	Some	Multiple batches	None
RA	Some	Some	Multiple batches	Only reactions
CCA	Some	Some	Multiple batches	Single batch
CCA1	Some	Some	Multiple batches	Limited batches
CCA2	Some	Some	Multiple batches	Multiple batches

Table 1.1: Attack models and the capabilities

(RA) model lets the attacker only observe the reactions of the decryption oracle, but not the decrypted result itself. Another variant called the Lunchtime Attack (CCA1) model only allows the attacker to submit new requests to the decryption oracle up to a certain point after which the decryption oracle can no longer be used.

- A Side-Channel Attack (SCA) assumes that the attacker has the ability to gain information about the operation of the encryption device through the use of "side-channels". This may involve physical access to the device or just being in close enough proximity to get any information about its operation.

In all the attack models it is assumed that the encryption and decryption algorithms, in addition to the public key, are publicly known (by Kerckhoffs' principle). Because of this an attacker will always have access to an encryption oracle and therefore it is reasonable to assume that an attacker at least has the capabilities of the CPA2 model when attacking PKCs. Table 1.1 summarizes some of the attack models and the capabilities of the attacker.

Perfect secrecy Perfect secrecy means that any valid ciphertext reveals absolutely no information about the plaintext unless one has the decryption key. Even if an attacker had unlimited computing power it is not possible to break the system because there is not enough information to verify if a key trial was correct or not. The chances of guessing the correct message should be no larger than $\frac{1}{|M|}$, where M is set of all possible messages. A PKC cannot have perfect secrecy because an attacker can always can try to encrypt every possible message with the public key to find the one corresponding to the ciphertext. In cases when a probabilistic scheme is used, then the attacker can just repeat the procedure for every possible random input as well. The chances of finding the correct message is then 1, which means that the ciphertext is leaking enough information to uniquely identify the corresponding message without any knowledge of the private key.

Semantically security The ciphertext of a plaintext may leak some information about the plaintext, but this information is infeasible to extract. This

can be considered a weaker, but much more realistic requirement than that of perfect secrecy. It has been shown that semantically security and ciphertext indistinguishability as described below are equivalent from a security perspective.

Ciphertext indistinguishability The concept of indistinguishability is most often explained in terms of a game that must be won within some polynomially bounded time frame. The player must produce two different plaintext blocks and give them to a gamemaster who picks one of the blocks at random, encrypts it and returns the corresponding ciphertext block. If the player is able to guess which plaintext the returned ciphertext belongs to with probability notably better than 1/2, then the game is won and the ciphertext blocks of the cryptosystem is distinguishable. Otherwise, the cryptosystem is said to have the property of indistinguishability. However, this property may heavily depend on the capabilities of the player, i.e. which attack model is used, and therefore it is most common to say that a cryptosystem is distinguishable under some given attack model. If, for instance, the the CCA model is used then one can say the cryptosystem is indistinguishable under CCA or just IND-CCA for short.

Non-malleability If an attacker can transform a ciphertext into another ciphertext which decrypts successfully to a different plaintext than the original, then the cryptosystem is malleable. Otherwise, the cryptosystem is said to have the property of non-malleability. Just like with ciphertext indistinguishability this property depends on which capabilities the adversary is given, and therefore it is common to say that a cryptosystem is non-malleable under some specific attack model. The abbreviation for non-malleability is NM and its often combined with the abbreviation for the attack model like NM-CCA for non-malleability under the chosen ciphertext attack model.

One-wayness One-wayness or preimage resistance requires that acquiring the plaintext for any given ciphertext is hard which is the most basic property of any one-way function. One-wayness can also depend on the attack model and just as with non-malleability and ciphertext indistinguishability there is as short notation. For example, claiming that a cryptosystem has one-wayness under chosen plaintext attack may be denoted by OW-CPA. Those who have studied cryptographic hash function may also be familiar with the related concepts of second preimage resistance and collision resistance. These are, however, irrelevant or trivially satisfied for one-way encryption functions because, in contrast to hash functions, all encryption functions are injective.

1.2.3 Message expansion

PKCs have a tendency to produce ciphertexts that are longer than the plaintext from which the ciphertext was derived. This generally undesired feature is commonly called message expansion or ciphertext expansion. While this feature is not particularly interesting from a security perspective, a large message expansion can be a major drawback in practical usage of the system. One can argue that message expansion is less of a problem these days because of the high channel capacity available using modern technologies, and then again, what message expansion is acceptable greatly depends on the use case of the system.

1.2.4 Key sizes

Because the public key of a asymmetric cryptosystem needs to be transmitted over the communication channel it is inconvenient if the key is very large. A large public and private key also puts a memory strain on the device using the cryptosystem. On the other hand, the keys should not be too short either as brute force attacks may become practically feasible either today or in a few years.

1.2.5 Complexity

Encrypting and decrypting messages involve a lot of computational effort and may consume a substantial amount of memory, which can be problematic for devices with limited resources available.

The computational effort and memory consumption can depend on the size of the input parameters, but also some instances of the input parameters of this fixed size may require more resources than others. The worst-case complexity is the complexity when the worst possible set of parameters are used. Furthermore, the average complexity is the complexity for average parameters. In some cases, it is tedious to exactly determine the number of operations needs to be performed or exactly the number of bits which needs to be stored on the system for any given input; also, the exact numbers may not be very relevant when complexity is large. Therefore, asymptotic complexity estimates can be provided instead.

1.3 Quantum computers and the consequences for PKCs

In recent years development of quantum computers have received massive attention from various tech companies and intelligence agencies. A huge amount of resources is being spent, pursuing the goal of being the first to get their hands on a device capable of doing large scale general quantum computations. As with all project receiving enormous investments, it's not only fulfilled out of pure scientific interest, but also as a tool for gaining power.

Several algorithms intended for quantum computers have been developed which are able to reduce the time complexity of several well-known problems to a level that is practically feasible. The most outstanding example is probably Shor's algorithm [25] which are able to solve IFP in polynomial time running on a quantum computer and it can also be used to solve DLP efficiently.

The Hidden Subgroup Problem (HSP) states that given a group G which has a subgroup $H < G$, a set X and a function $f : G \to X$ that acts as a distinguisher for the left cosets of $H < G$, it is hard to find a generating set for H. That f is a distinguisher means that $f(g_1) = f(g_2)$ if and only if $g_1H = g_2H$, i.e. g_1 and g_2 generates the same left coset of $H < G$. For certain abelian groups G it is known that HSP can be solved efficiently with a quantum computer using quantum Fourier transformations. Shor's algorithm reduces IFP to an instance of HSP over a cyclic group $\mathbb{Z}_N$ and DLP to an instance of HSP over an abelian group $\mathbb{Z}_N \times \mathbb{Z}_N$ which both can be solved on a quantum computer using quantum Fourier sampling.

As IFP and DLP are common trapdoors used in asymmetric cryptosystems the consequences can be devastating if they suddenly are getting feasible to solve. Some organizations have had the capability of storing massive amounts of encrypted data for several years in hopes of decrypting them successfully in the future. Having one or more parties with access to a large-scale quantum computer in a world where communication relies solely upon IFP and DLP, makes it impossible to trust services like banking, e-mail and electronic IDs.

Another important quantum procedure is Grover's algorithm [13] which given a black-box function (i.e. one can evaluate it, but not see its inner operations) and an output of this function finds an input which produces the given output with high probability. On a classical computer the best way to solve this problem is to try evaluating the function with inputs from the domain until one input matches the given output. In the worst case one has to try all possible inputs before finding the correct match, but on average the correct match is found after trying half of the domain. With a quantum computer Grover's algorithm can be used to solve this problem with an asymptotic worst-case complexity of just the square root of the size of the domain.

It's probably helpful to get a better understanding of why quantum algorithms have the potential of being faster than their classical counterpart while running on a quantum computer. Therefore, in this section a short introduction to quantum computers is given.

In classical computing, certain algorithms can take advantage of several processor cores to execute instructions in parallel and, depending on the algorithm, may obtain at most a linear speedup (with slope ≤ 1) in the number of cores. This means if the algorithm has exponential complexity, then one need an exponential increase in the number of processor cores to be able to solve it efficiently. In practice this doesn't work.

Unlike classical system the amount of quantum parallelism of a quantum system grows exponentially by the size of the system[23]. Whether this quantum parallelism can actually be used to speed up computations or whether it is real at all is still debated. The biggest problem with quantum parallelism is that in contrast to classical computing one can only measure the result of exactly one of the parallel threads and which one is measured is probabilistic. Performing the measurement will also destroy the result of all other threads.

Quantum computers uses quantum bits (or qubit for short) instead of classical bits. While a classical bit has two states 0 and 1, the qubit can be in a linear superposition of basis states. The basis states are $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and thus a linear superposition of such states can be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$. Here α and β are complex numbers with the additional constraint that $|\alpha|^2 + |\beta|^2 = 1$. One can think of a quantum state as being a point on the surface of the unit sphere. Measuring a qubit means that its state will immediately "collapse" into one of the basis states with probability $|\alpha|^2$ for $|0\rangle$ and $|\beta|^2$ for $|1\rangle$. Just as with classical bits qubits can be combined into sequences. The combination of two qubits $|\psi\rangle_A = \alpha_A|0\rangle + \beta_A|1\rangle$

and $|\psi\rangle_B = \alpha_B |0\rangle + \beta_B |1\rangle$ is $|\psi\rangle_{AB} = \begin{bmatrix} \alpha_A\alpha_B \\ \alpha_A\beta_B \\ \beta_A\alpha_B \\ \beta_A\beta_B \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ and can be measured to any of these four states $|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $|01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $|10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, and $|11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$. The operation used to combine the qubit is more generally called a tensor product denoted with the symbol $\otimes$ and a combination of n qubits can then be expressed as $|\psi\rangle_{ABC...} = \psi_0 \oplus \psi_1 \oplus ... \oplus \psi_{n-1}$. Notice that n qubits can be in a superposition of 2^n different states at once.

1.4 Alternative cryptosystem variants

There are several PKCs in development being based upon problems not believed to be practically solvable with a quantum computer, many suffering problems of their own like large keys sizes, poor plaintext to ciphertext ratio, and slow encryption or decryption. These include lattice-based, multivariate, hash-based, and code-based systems. Here we will give a very shallow introduction to some multivariate and lattice-based problems and then in the following chapters the focus will be on code-based systems.

1.4.1 Multivariate systems

Multivariate PKCs have the property that their public key is a set of multivariate polynomials: $P(x_1, \ldots, x_n) = (p_1(x_1, \ldots, x_n), p_2(x_1, \ldots, x_n), \ldots, p_m(x_1, \ldots, x_n))$. Given some plaintext $X = (x_1, \ldots, x_n)$, the ciphertext is a polynomial evaluation. $P(X) = (p_1(x_1, \ldots, x_n), \ldots, p_m(x_1, \ldots, x_n)) = (c_1, \ldots, c_m)$ The private key is then any trapdoor for computing P^{-1}. Multivariate systems can also be used for signatures in the following way. Let $h_1, \ldots, h_m$ be some information that should be signed. Then compute the signature using the private key $(x_1, \ldots, x_n) = P^{-1}(h_1, \ldots, h_m)$ The signature can be verified using the public key $(h_1, \ldots, h_m) = P(x_1, \ldots, x_m)$. Solving a set of m randomly chosen (nonlinear) equations with n variables is known to be NP-complete.

1.4.2 Lattice-based systems

Consider any set of linearly independent vectors with entries from the real numbers. The additive group of all linear combinations of these vectors with integer coefficients is a lattice. There are several different one-way trapdoors associated with lattices and as expected lattice-based cryptosystems relies on some of these hard lattice problems.

The shortest vector problem (SVP) tries to find the shortest non-zero vector with respect to some norm (usually the euclidean norm). At first this problem

may not seem very difficult at all, and for some sets of basis vectors it is in fact easily solvable like when short nearly orthogonal vectors are used.

Another problem called the closest vector problem (CVP) here a vector not necessarily in the lattice is given and the challenge is to find the closest vector to it in the lattice.

1.5 Introduction to coding theory

Information can be represented in several different forms and some forms are better suited for some applications than others. Finding good ways to represent the information for a given application and making the transition between the different representations is the science coding theory.

Example 1. Communication with errors

Assume Alice and Bob is communicating. Alice is a woman of few words and so the only two possible words she will use are "yes" and "no". Bob knows this, but suffers from a bothersome hearing loss and so he has a $(1-p) = 0.30$ chance of hearing the wrong word. He is never satisfied before he is at least $\tau = 0.95$ certain that he catches the right words and therefore he asks Alice to repeat her message until he is satisfied. On average, Alice will need to repeat her message $n = 17$ times before Bob is satisfied where n is the minimum that meets the following relation:

$$\tau \leq \sum_{\lfloor \frac{n}{2} \rfloor < k \leq n} \binom{n}{k} p^k (1-p)^{n-k}$$

Alice quickly gets tired of answering Bob's repeat requests and therefore she decides to encode her messages so that each word is repeated n times immediately.

1.5.1 Information value of a message

Intuitively information is what one gets from reading the newspaper, watching TV, listening to friends or more generally interpreting sources of data. For something to be considered information it must, at least partially, resolve the uncertainty of one or more random variables.

In the example above, even though Alice is creating longer messages by repeating each word, the total information the messages contains does not increase because the additional redundant message contents is fully dependent on already provided information.

1.5.2 Channel models

Often, it can be beneficial to assume a model of the channel in question which captures and illustrates the behavior of the channel. A widely used such model is the Binary Symmetric Channel (BSC). In the BSC model, every message bit has a fixed chance ε of flipping. Example 1 can be modeled with the BSC model.

1.5.3 Cope with noisy channels

Transmitting a message over a channel can introduce multiple errors or erasures of data bits. With all the errors, the receiver may not be able to tell what the original message was supposed to be. To avoid this redundant information is added to the original data before transmitting. The receiver is then hopefully able to use this redundant information to check if any errors occurred, or sometimes even correct the any errors that where detected. The transformation of data from one form to another form that is more robust against faults introduced by the channel is the science of coding theory. Ultimately one would like to correct as many bits possible using few redundant data bits.

The above is primary use cases for coding theory, but it can also be used for cryptographic applications as will be explained later.

Let A be a source alphabet and $A^\star$ be all sequences of As. A code $C : M \to A^\star$ is an injective mapping from a set M to codewords of $A^\star$, and each codword is just a sequence of codeword symbols from an alphabet A. An encoder for C is just an algorithm performing the mapping C, while a decoder for C is an algorithm that first maps an element of $A^\star$to a codeword of C (or fails) and then proceeds to find the preimage of the found codeword under C. Transmission of a codeword over a channel may introduce errors such that it is no longer a proper codeword. A Maximum Likelihood (ML) decoder always finds the codeword that most likely was sent . Whether or not a decoder is ML depends on the properties of the channel. A minimum distance decoder always finds the codeword that most closely resembles the received word, i.e. the codeword with the lowest edit distance. A special case of edit distance for fixed length binary sequences is the Hamming distance. The minimum distance decoders stays minimum distance decoders even if the properties of the channel changes. If A has a zero element, the weight of a codeword in C is the number of non-zero entries. The minimum distance of C is the minimum distance between any two distinct codewords.

If $C : A^k \to A^n$ where k and n are integers, then C is called a block code. The code rate of a block code C is given as $R(C) = \log|C| / n \log|A|$ and as such is a measure of how much redundant information that is included in the codewords. Usually the basis used for the logarithms is 2 which means that the unit of $R(C)$ is bits. If the minimum distance of a block code C is d then this gives an upper bound of $(d-1)/2$ for the number of correctable errors and $d-1$ for the number of detectable errors.

1.5.4 Linear block codes

A $[n, k]$ linear block code C is an injective mapping from $\mathbb{F}_q^k$ to a linear subspace of a vector space $\mathbb{F}_q^n$ with dimension k, where $\mathbb{F}_q$ is a finite field of q elements. The size of C is q^k. The linear code can be generated by all linear combinations of k linearly independent codewords. Let G be the $(k \times n)$ matrix having these linear indendent vectors as rows. G is called a generator matrix for C and each $m \in \mathbb{F}_q^k$ generates a unique linear combination in $\mathbb{F}_q^n$ when multiplied with G. The codewords of C can also be defined in terms of a $((n-k) \times n)$ check matrix H. In this case a word c is a codeword if and only if it satisfies all the check rows of H, this means that the product cH^T must equal the 0-vector.

A generator matrix for a linear code generates the same set of codewords even

after any combination of elementary row operations (row swaps, row scalings with nonzero scalar and additions with other rows). However, the mapping between messages and codewords is changed under such operations, and in that regard the code is changed. All invertible matrices can be written as a product of elementary row operations. If G is a generator matrix and A is an invertible matrix then AG generates the same set of codewords as G. If A also happens to be a permutation matrix the weights of the vectors v, vA and $\left(Av^T\right)^T$ are equal. Assume there is a known efficient decoder which can decode t errors for the code generated by G. A message m is encoded as mGP where P is a permutation matrix. Under transmission some errors occurs so that the received word is $mGP+e$ where e is a vector of weight as most t. To find the original message the receiver (which happens to know P) can observe this relation $(mGP+e)A^{-1} = mG+eP^{-1}$. eP^{-1} has the same weight as e and therefore the efficient decoder can be used to decode $mG+eP^{-1}$ to m. For another example, assume that A is invertible, but not necessarily a permutation, and $mAG+e$ is received. Because AG generates the same set of codewords as G the efficient decoder is applicable and will return mA from which m is found as mAA^{-1}.

A $(k \times n)$ generator matrix is said to be on systematic or standard form if the leftmost $(k \times k)$ block is the identity matrix. A generator matrix on systematic form greatly simplifies the decoding procedure as the k first bit of any codeword is the original message. The code generated by G in sytematic form is also said to be systematic in its first k coordinate positions.

If G is on standard form $[I_k|P]$ then a parity check matrix for G is given as $H = \left[-P^T|I_{n-k}\right]$. Note that neither the parity check nor the generator matrix is unique and a given parity check matrix may not be as suitable for certain decoders as others.

1.5.5 Binary Goppa code

Let $g(x)$be an irreducible polynomial over $\mathbb{F}_{2^m}$ of degree t without multiple roots and let L be a sequence of n distinct elements in from $\mathbb{F}_{2^m}$ that aren't roots of $g(x)$. Let V be the $(n \times t)$ Vandermonde matrix of the sequence L, that is $V_{i,j} = L_i^j$ where $i \in \{0, ..., n-1\}$ and $j \in \{0, ..., t-1\}$. Let D be a $(n \times n)$ diagonal matrix with entry $D_{i,i} = g(L_i)^{-1}$. A $(t \times n)$ parity check matrix for the Binary Goppa code defined by $g(x)$ and L is given as $H = V^T D$ or simply $H_{i,j} = L_j^i \cdot g(L_j)^{-1}$ where $i \in \{0, ..., t-1\}$ and $j \in \{0, ..., n-1\}$. From here one can find the $((n-t) \times n)$ generator matrix. The most widely used decoder is Patterson's algorithm which can correct t errors.

1.5.6 Quasi-cyclic codes

A quasi-cyclic (QC) code is a linear block code which can be generated by a block-circulant generator matrix G. A block-circulant matrix can be partitioned into equally sized square blocks such that each block has a circulant structure. The number of blocks that partitions a row in G is called index n_0 of the QC code. The order r of the circulant square blocks is also called the order the QC code. The length of each codeword is in that regard $n_0 r$. The number of circulant blocks partitioning any column of G will be referred to as the number k_0. Furthermore the dimension of C is $k_0 r$. The notation:(n_0, k_0)-QC code, specifies explicitly the values of n_0 and k_0.

Circulant blocks of order r in $\mathbb{F}_q^{r\times r}$ is ring-isomorphic to the quotient polynomial ring $R = \mathbb{F}_q[X]/(X^r - 1)$. An isomorphism maps the first row $(a_0, ..., a_{r-1})$ of any circulant matrix to the polynomial $a_0 + a_1X + ... + a_{r-1}X^{r-1} \in R$. The product of two polynomials $A = a_0 + a_1X + ... + a_{r-1}X^{r-1}$ and $B = b_0 + b_1X + ... + b_{r-1}X^{r-1}$ in R is given by the following formula $A \cdot B = a_0 \cdot b_0 + a_1b_{r-1}X + ... + a_{r-1}b_1X^{r-1}$. Furthermore, addition is given by $A + B = (a_0 + b_0) + (a_1 + b_1)X + ... + (a_{r-1} + b_{r-1})X^{r-1}$. Because of this the circulant blocks can be represented as polynomials in R. Using the alternative representation for the circulant blocks the generator matrix of a (n_0, k_0)-QC code can be viewed as an $(k_0 \times n_0)$ matrix over the polynomials in R.

1.5.7 Regular binary MDPC codes

A regular Moderate Density Parity Check (MDPC) code is a binary linear block code with a $((n-k)\times n)$ parity check matrix H of moderately low density, but constant row weight. Moderately low density means that the weight of the rows grows sublinear in n, typically $O(\log n)$ or $O(\sqrt{n})$. By contrast, a Low Density Parity Check (LDPC) code has row weights that are constant in n.

1.5.7.1 Decoding MDPC codes

For decoding a binary MDPC code one can among others use a simple Bit Flipping (BF) algorithm or a much more complex belief propagation method[15]. Belief propagation has shown itself very efficient for decoding LDPC codes, however with MDPC codes one can expect a much higher frequency of short cycles in H's Tanner graph. Belief propagation method doesn't deal well with graphs containing many short cycles. One of the greatest advantage of belief propagation is its capability of making use of "soft" information, i.e. instead of inputs bits being in either of two states it can carry a probability of being in either state. For the cryptographic applications of MDPC codes that will be discussed later only hard errors are considered.

The Tanner graph of H is a bipartite graph and can be constructed from H by creating a $n-k$ parity check nodes $\{p_0, ..., p_{(n-k)-1}\}$ and n variable nodes $\{v_0, ..., v_{n-1}\}$. p_i is connected to v_j if and only if the bit at $H_{i,j}$ is 1. Assume that a parity check is satisfied if the sum of its input is 0. A single flip BF algorithm goes as follows. First initialize the values of the variable nodes with the initial (hard) beliefs given by the input word. For each of the parity check nodes, mark them as either satisfied or unsatisfied depending on whether the sum of all connected nodes is 0 or not. If all parity checks are satisfied the algorithm outputs the current beliefs on the variable nodes, otherwise flip value of the variable node that is connected to most unsatisfied parity check nodes and reevaluate the check nodes.

Chapter 2

Code-based systems

The problem that code-based cryptosystems are ultimately based upon is the hardness of decoding a random linear code. This problem cannot be solved in polynomial time. However, there exist non-random and quasi-random linear codes that admit polynomial-time decoding algorithms.

McEliece-like and Niederreiter-like systems are the two main categories of code based systems. At the core McEliece-like systems always publish some kind of encoder for some error-correcting code together with error correction capability of the decoder while the decoder itself is kept private. Encryption is then done by encoding the message and then applying some amount of random errors not surpassing the error correction capability of the encoder. Decryption is done by decoding the received message using the secret decoder.

2.1 Trapdoor based on linear block codes

In 1978, Berlekamp, McEliece and van Tilborg showed the problem of decoding a general linear code and the problem of finding weights of a general linear code is NP-Complete[8]. For a binary linear code C with a generator matrix G and a check matrix H, assume that the word $c = mG + e$ was received over a channel that uniformly distributes a number of errors over the codeword. The syndrome is computed as $s = cH^T = (mG + e)\,H^T = mGH^T + eH^T = eH^T$. This means that to find e one can find a solution to the system $s = eH^T$ with e having either some expected weight or having the smallest weight possible. What Berlekamp, McElice and van Tiborg showed was that finding such a solution e is hard for a general linear code.

Now, if one knows about an efficient decoder for a linear code and distinguishing it from a random code is NP, then it can be used together with general decoding problem to form a trapdoor. Here the efficient decoder is the additional information one needs to solve the problem easily.

2.2 Noisy channel coding versus code-based cryptography coding

When considering transmissions of n-bits codewords over a BSC with error probability ε one expects the probability of any number of errors to follow a binomial distribution with mean εn. The error correcting code should have an error correction capability higher than the mean so that it is able to all errors up to a reasonable threshold.

When error correcting codes are used as part of a code-based cryptosystems, a channel that distributes a fixed or less than a fixed number t of errors uniformly over the codeword is used. That is, after n bits have passed through the channel at most t of those bits where flipped while the others remained untouched. In contrast to the binary symmetric channel, this channel depends on previous outcomes. Because there is a known upper bound for the number of errors that can occur it is possible to use a decoder with an error correction capability very close to t.

2.3 McEliece cryptosystem

The McEliece cryptosystem was developed by Robert J. McEliece in 1978[17] and its security is based on the hardness of decoding a general linear code and the difficulty of distinguishing a Goppa code from a random linear code. Up until recently the McEliece cryptosystem did not get a lot of attention, partly because of the very large key size of the original proposal. The original system uses binary irreducible Goppa codes, which is defined by an irreducible polynomial $g(x)$ of degree t over a finite field $\mathbb{F}_{2^m}$without multiple zeroes and a sequence L of n distinct elements, also from $\mathbb{F}_{2^m}$, that are not roots of $g(x)$. The minimum distance is $2t + 1$ and it can correct t errors. There is also a generalized version where any linear t error correcting code can be used instead of a binary Goppa code.

Algorithm 2.1 McEliece Key Generation

1. Decide desirable values for codeword length$n$ and error correction capability t.
2. Find a binary Goppa code with a generator matrix $G \in \mathbb{F}_2^{k\times n}$ and having a decoder that are able to correct t errors.
3. Create a random permutation matrix $P \in \mathbb{F}_2^{n\times n}$.
4. Create an invertible scrambler matrix $S \in \mathbb{F}_2^{k\times k}$.
5. Compute SGP
6. Public key is (SGP, t)
7. Private key is (S, G, P)

Algorithm 2.2 McEliece Encryption

Given message $m \in \mathbb{F}_2^k$ and public key (SGP, t)

1. Compute $mSGP$
2. Generate binary error string e of weight t
3. Return $mSGP + e$

Algorithm 2.3 McEliece Decryption

Given ciphertext $mSGP + e \in \mathbb{F}_2^n$ and private key (S, G, P)

1. Compute $mSG + eP^{-1} = (mSGP + e)\, P^{-1}$
2. Decode $mSG + eP^{-1}$ to get mS using the efficient decoder.
3. Return $m = mSS^{-1}$

The purpose of the scrambler and permutation matrix is to hide the structure of the private generator matrix G. If an attacker is able to find G the task of finding an efficient decoder is well studied and not too hard therefore hiding G is very important.

While the process of encryption and decryption can be done very efficiently the main drawbacks of the MEC is its huge key sizes compared to the security level it provides and message big message expansion.

McEliece originally suggested to use $n = 1024$, $k = 524$ and $t = 50$ for 80 bits security, but that was back in 1978 and these parameters is now insufficient. Bernstein, Lange and Peters' improvement to Stern's attack reduced the security of MEC with the original parameters to about 58 bits.

More recently PQCRYPTO has recommended to use $n = 6960$, $k = 5413$ and $t = 119$ for 2^{128} bits post-quantum security[5]. This gives a public key of $n \cdot k = 37674480$ bits.

2.4 Niederreiter cryptosystem

Very similar to the McEliece cryptosystem is the Niederreiter cryptosystem[22]. Instead of operating in the codeword domain, the Niederreier cryptosystem operates in the syndrome domain. It first encodes the message to vectors of length n. It is yet unbroken when Goppa codes are used.

The Niederreiter cryptosystem has an advantage over the McEliece cryptosystem in that the ciphertext messages can be shorter.

Algorithm 2.4 Niederreiter key generation

1. Select a linear code which are able to correct t errors, has a known efficient decoder and generator matrix $G \in \mathbb{F}_2^{k \times n}$.
2. Generate a parity check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ for G.
3. Create a random permutation matrix $P \in \mathbb{F}_2^{n \times n}$.
4. Create an invertible scrambler matrix $S \in \mathbb{F}_2^{(n-k) \times (n-k)}$.
5. Compute $SHP \in \mathbb{F}_2^{(n-k) \times n}$.
6. Public key is (SHP, t)
7. Private key is (S, H, P)

Algorithm 2.5 Niederreiter encryption

Given plaintext $m \in \mathbb{F}_2^k$ and public key SHP

1. m is encoded as a binary string with length n and weight t becoming $f(m) = e$.
2. Return chipertext $SHPe^T$

Algorithm 2.6 Niederreiter decryption

Given ciphertext $SHPe^T \in \mathbb{F}_2^{(n-k)}$ and private key (S, H, P)

1. Compute $HPe^T = S^{-1}\left(SHPe^T\right)$
2. Use the efficient decoder to find Pe^T from HPe^T
3. Compute $e = \left(P^{-1}\left(Pe^T\right)\right)^T$
4. Return plaintext $m = f^{-1}(e)$

Chapter 3

Implementing a McEliece-like system

In this part various implementation details that can be helpful when implementing a McEliece-like system are discussed. This is not supposed to be a complete guide and the steps shown are not necessarily the best ways to accomplish a given task.

3.1 Constructing, representing and computing with finite fields

Finite fields are widely used in both cryptography and coding theory, therefore it may not come as a surprise that they have found their use in McEliece-like cryptosystems. Finite fields have many beneficial properties making them appropriate in several situations, especially where multiplicative inverses are required.

3.1.1 About fields and finite fields

A set F and two binary operations $+ : F \times F \to F$ and $\cdot : F \times F \to F$ called addition and multiplication is a field if the following conditions are satisfied:

- $\forall a, b, c \in F : (a + (b + c) = (a + b) + c)$, Addition is associative
- $\exists 0 \in F : (\forall a \in F : (0 + a = a))$, Exists an additive identity element 0
- $\forall a \in F : (\exists - a \in F : (a + (-a) = 0))$, Exists a negative (additive inverse) of every element
- $\forall a, b \in F : (a + b = b + a)$, Addition is commutative
- $\forall a, b, c \in F : a \cdot (b \cdot c) = (a \cdot b) \cdot c$, Multiplication is associative
- $\exists 1 \in F/\{0\} : (\forall a \in F : (1 \cdot a = a))$, Exists a multiplicative identity element 1 different from the additive identity element

- $\forall a \in F : \big(\exists a^{-1} \in F : \big(a \cdot a^{-1} = 1\big)\big)$, Exists a multiplicative inverse for every element except 0
- $\forall a, b \in F : (a \cdot b = b \cdot a)$, Multiplication is commutative
- $\forall a, b, c \in F : (a \cdot (b + c) = (a \cdot b) + (a \cdot c))$, Multiplication distributes over addition

Additionally if the set F is finite it is called a finite field. The number of elements in a field is called its order. For finite fields, this order is always a prime power. Furthermore for every prime power q there exists fields with order q and they are all isomorphic, which means they are structurally identical up to relabeling of their elements.

To simplify the syntax involved when adding an additive inverses or multiplying with multiplicative inverses two new binary operations is defined. First, subtraction$- : F \times F \to F$ performs the mapping $a - b = a + (-b)$. In the same way, division $/ : F \times F \to F$ performs the mapping $a/b = a \cdot b^{-1}$.

3.1.1.1 Primitive fields

Prime fields are finite fields of prime order p and are convenient in the sense that they are isomorphic to $\mathbb{Z}_p$ and therefore can be easier to compute with. All finite fields that are not a primitive field is an extension of a primitive field.

3.1.1.2 Additional warning regarding integer types in hardware

Humans, with some exceptions, are not flawless. For the author, abstract algebra (and mathematics in general) doesn't come easy, and therefore some warnings of possible mistakes have been included, even if obvious for some.

The following is a warning regarding the hardware supported integer types found in most computers. As finite fields of order 2^8, 2^{16}, 2^{32}, 2^{64} exists it is easy to think (at least for the author it was) that the 8-, 16-, 32-, and 64-bit types together with addition and multiplication modulo 2^8, 2^{16}, 2^{32}, and 2^{64} respectively are finite fields. However, none of these are finite fields as they lack multiplicative inverses for several numbers. For instance the element 2 does not have a multiplicative inverse in any of them. In fact, no $\mathbb{Z}_q$ is a field unless q is a prime, in which case it's always a field.

3.1.2 Comfort of a programming language

Hardware details are complicated, and to avoid diving into a mess of compatibility issues the abstractions and simplifications provided by programming languages is helpful. Here it is assumed that the cost of these abstractions does not account for more than some small constant factors.

Assume the following primitives types are provided: "u8", "u16", "u32" and "u64", whose names indicated their sizes in bits. Addition and multiplication with these types can be configured to behave equivalently to the corresponding operations in $\mathbb{Z}_8$, $\mathbb{Z}_{16}$, $\mathbb{Z}_{32}$ and $\mathbb{Z}_{64}$ respectively, but division has a different semantic. Also there exists a remainder operation which for these unsigned types is the same as a modulo operation on an integer. For the rest of the section the primitive data types may be referred to as containers, as this is their

primary use case. To avoid confusion between field operations and operations on the containers provided by the programming language, the field addition, multiplication, subtraction and division are appended a subscript with the order of the field they applies to. As all fields of a given order are isomorphic there should not be any ambiguity of the meaning of this notation.

3.1.3 Representing primitive fields

Field elements needs to be represented using one of the primitive data types or some complex data structure emulating larger or smaller containers. If elements of a primitive field of order p should be stored, then at least $\lceil \log_2(p) \rceil$ bits are required from the container type. For reasons that will become clear in the following section it can be advantageous to require twice as many bits, i.e. $\lceil 2 \cdot \log_2(p) \rceil$ bits, for storing intermediate results of multiplication with these elements.

With such a requirement the largest primitive field that can be represented with a 32-bit container is one of order $p = 4.294.967.291$, but to store the result of the multiplication before the modulo reduction takes place a 64-bit container should be available as well. If fields of larger order are required it is possible to emulate larger containers.

Storage	Largest prime order
8 bit	251
16 bit	65 521
32 bit	4 294 967 291
64 bit	18 446 744 073 709 551 557

Table 3.1: Largest prime order field for containers

Currently the containers themselves are not aware of which field the element they represent belongs to. It is possible to make them self-aware in the sense that they are part of a structure containing or with a reference to this meta-data. In this thesis however, a slightly different approach is used where a structure containing all necessary field meta-data is created independently and operations on the field element containers is always done from the context of this structure. The main reason for this decision was to save some memory and avoid unnecessary data duplication. Elements of the same field are commonly aggregated into more complex types like matrices, and data duplication can significantly impact their total storage size.

3.1.4 Computing with primitive fields

Field addition and multiplication can be regarded as a two step computational processes. The first one is an additive step or multiplicative step, which is then followed by a reduction step. There is also other methods that takes

3.1.4.1 Addition

The first step of a field addition is to do a regular integer addition, but for this to work one must make sure that no overflow takes place, i.e. that the result can fit within the container. To keep the sum of two n-bit numbers at most $n + 1$ bits are needed.

After the addition step the result needs to be reduced modulo the order of the field. The operands of the addition is never lager than $p - 1$ and therefore

the largest achievable result is $2(p-1) = 2p-2$. So if the sum is larger or equal to p, the reduction is performed by subtracting p only once.

3.1.4.2 Multiplication

Just as with addition an overflow of the container must be prevented. To keep the product of two n-bit numbers at most $2n$ bits are needed. An integer division operation can be used to find what multiple of p must be subtracted to reduce the result, but a more straightforward and possibly better solution is to just use the remainder operation, which for positive integers should be equivalent to a modulo operation.

3.1.4.3 Subtraction

When doing integer subtraction, if the left hand side is strictly less than the right an overflow will happen. To prevent this p can be added to the left hand operand before the subtraction takes place in this case. Reduction is unnecessary as the result will be less than p anyway.

3.1.4.4 Division

Division is the same as multiplying the left hand operand with the inverse of the right hand operand. The inverse can be found by using the extended Euclidean algorithm. Note that all elements except zero have multiplicative inverse. The procedure is shown through the following example.

Example 2. Finding the multiplicative inverse of 8004 in $GF(65521)$ using the extended Euclidean algorithm and then using this number to solve 5000/8004.

i	r_i	q_i	t_i
0	65521	—	0
1	8004	$r_0/r_1 = 8$	1
2	$r_0 - q_1 \cdot r_1 = 1489$	$r_1/r_2 = 5$	$t_0 - q_1 \cdot t_1 = 65513$
3	$r_1 - q_2 \cdot r_2 = 559$	$r_2/r_3 = 2$	$t_1 - q_2 \cdot t_2 = 41$
4	$r_2 - q_3 \cdot r_3 = 371$	$r_3/r_4 = 1$	$t_2 - q_3 \cdot t_3 = 65431$
5	$r_3 - q_4 \cdot r_4 = 188$	$r_4/r_5 = 1$	$t_3 - q_4 \cdot t_4 = 131$
6	$r_4 - q_5 \cdot r_5 = 183$	$r_5/r_6 = 1$	$t_4 - q_5 \cdot t_5 = 65300$
7	$r_5 - q_6 \cdot r_6 = 5$	$r_6/r_7 = 36$	$t_5 - q_6 \cdot t_6 = 352$
8	$r_6 - q_7 \cdot r_7 = 3$	$r_7/r_8 = 1$	$t_6 - q_7 \cdot t_7 = 52628$
9	$r_7 - q_8 \cdot r_8 = 2$	$r_8/r_9 = 1$	$t_7 - q_8 \cdot t_8 = 13245$
10	$r_8 - q_9 \cdot r_9 = 1$	$r_9/r_{10} = 2$	$t_8 - q_9 \cdot t_9 = 39383$

The last non-zero remainder is 1, indicates that 8004 and 65521 are relatively prime.

$8004 \cdot 39383 = 1$ so $8004^{-1} = 39383$

$5000/8004 = 5000 \cdot 39383 = 24395$

As seen from the example, the process of finding a multiplicative inverse can be expensive and therefore division using this approach will also be relatively slow compared to the other field operations. For small fields it may be possible to store the inverse of every element and then look them up in a table when needed.

3.1.4.5 Fields of order 2

For fields of order two, the process of addition and multiplication can be simplified. Addition in the field behaves the same way as a "bit-wise exclusive-or" operation. Similarly multiplication can be carried out as a "bit-wise and" operation.

Another thing to notice is that addition and subtraction are identical operations.

3.2 Dealing with linear transformations

McEliece-like cryptosystems being code-based usually implies that a lot of linear transformations, more specifically matrix and vector computations, will be needed. In this section various issues considering matrices, vectors and operations between them will be discussed. This includes different ways of storing matrices of varying layouts and how to compute matrix products.

3.2.1 Schoolbook matrix multiplication

There is a number of sophisticated algorithms that can improve the performance of a matrix multiplication, but these often have restrictions how the matrices are laid out in memory. Many of the matrices that are dealt with in this thesis have a lot of structure that can be used as an advantage for more efficient specialized matrix multiplication. In these cases it is easier to base matrix multiplication on the schoolbook algorithm and evolve it from there. Here is the algorithm for the schoolbook matrix multiplication:

Algorithm 3.1 Schoolbook matrix multiplication

```
fn multiply_matrices(A: Matrix<r,s>, B: Matrix<s,t>) -> Matrix<r,t> {
        let C: Matrix<r,t> = Matrix<r,t>::zero
        for all (i,j,k) in cartesian_product((0..r),(0..t),(0..s)) {
                C[i][j] += A[i,k] * B[k,j]
        }
        return C
}
```

3.2.2 Composite matrices

To better take advantage of the fact that matrices can consist of sub-matrices with different structure one can consider composite matrices.

Example 3. Systematic matrices

Consider the following $m \times n$ matrix $A = [I|B]$ where I is the identity matrix and B is some unstructured matrix. A naive storage approach for A would store every individual entry into an array costing mn times the size of each entry . The leftmost $m \times m$ sub-matrix of A is the identity matrix and a better approach for storing this matrix is often to just store the rightmost $m \times m - n$ entries. In many cases some matrices will be systematic by assumption when used in

code-based cryptosystems and therefore no additional meta-information needs to be stored.

Representing composite matrices compactly doesn't help very much if one cannot do computations with them. The most common computation that will be needed is a vector matrix product. Assume the following composite $m \times n$ matrix is given along with a row vector v.

$$A = \begin{bmatrix} A_{0,0} & \cdots & A_{0,n} \\ \vdots & \ddots & \vdots \\ A_{m,0} & \cdots & A_{m,n} \end{bmatrix}$$

$$v = (v_0, ..., v_{m-1})$$

Also let r_i be the row range that sub-matrix $A_{i,j}$ occupies in A and v_{r_i} be the sub-vector of v only containing the elements within the the range r_i. The product can be computed as follows:

$$vA = (v_{r_0} A_{0,0} + \cdots + v_{r_m} A_{m,0}) \,|\, \cdots \,|\, (v_{r_0} A_{0,n} + \cdots + v_{r_m} A_{m.n})$$

This way, if for example $A_{0,0}$ is the identity matrix, then the this part $v_{r_0} A_{0,0}$ can be computed very efficiently as simply v_{r_0}.

3.2.3 Very sparse matrices

If a matrix has a very few number of non-zero elements it can be more efficient to just store the coordinates of these elements together with their values instead of storing the whole matrix.

Multiplying two sparse matrices can be done as follows:

Algorithm 3.2 Multiplication of sparse matrices

```
fn multiply_sparse_matrices(A: Matrix<r,s>, B: Matrix<s,t>) -> Matrix<r,t> {
        let C: Matrix<r,t> = Collection<(int,int,T)>
        for all (i,k,x) in A {
                for all (k,j,y) in B {
                        C[i][l] += A[i,s] * B[s,j]
                }
        }
}
```

3.2.4 Circulant square blocks

Circulant blocks can be fully determined by just the first row or column. Let A be an $m \times n$ circulant matrix block and $A_{(i,j)}$ be the element at row i and column j, then $A_{(i,j)} = A_{(i-j \mod m,0)} = A_{(0,j-i \mod n)}$.

Multiplying two circulant matrices is done by multiplying the first row of the left hand matrix with the first column of the right hand matrix. The result is also circulant matrix.

3.2.5 Generating quasi cyclic matrices

There is a need to generate invertible circulant matrices. Parameters should be the size n, the cyclic shift , and the weight w of the rows.

Algorithm 3.3 Construct QC-MDPC matrix, then check for invertibility

---

1. Generate a vector of length n containing only zeroes. Pick w non-repeating indices independently and set the correponding values to ones. The vector and n_0 now fully describes a quasi cyclic matrix.

2. Check for invertability by checking if the matrix is row equivalent to the identity matrix using gauss-jordan elimination. If the matrix is invertible we are done, otherwise we redo the process.

Constructing QC-MDPC matric, then check for invertibility This method can generate all possible invertible quasi-cyclic matrices. On the other hand the extra check for invertability is releatively costly and so increases the computational complexity. There is also a problem of having to redo the process when the matrix was singular which is also costly, however it has been shown that the chances of getting a singluar matrix is low.

Using the methods by Fabsic et al. Recently Fabsic et al.[11] showed that for some values of n, generating a invertible circulant matrix with some prescribed number of ones is easy. They identified the following special cases:

- n is prime, the order of 2 in $\mathbb{Z}_n$ is $n-1$ and w is odd
- n is a power of 2 and w is odd

For these cases they have provided algorithms to solve the problem.

3.2.6 Permutations

While a permutation function can be represented as a matrix this is not a very efficient way representation. A permutation matrix is characterized by having exactly one 1 on each row and column while the rest is 0, which means that only n of n^2 elements is of interest. An alternative way of representing a permutation is as a vector $\boldsymbol{p} = (p_0, ..., p_{n-1})$ where p_i is an integer between 0 and $n-1$ and indicates that the element at position i should be moved to position p_i. A permutation can be performed in place

Chapter 4

Modifications and conversions

4.1 Loidreau's Modification

Loidreau's Modification[16] makes it more difficult to break the OW-CPA property of the McEliece cryptosystem. It does so by applying a linear transformation that maps the code into itself. The way error vectors is generated is changes so that instead of choosing an error vector of decodable weight it chooses an error vector such that after applying the linear transformation it has decodable weight. Another good property of Loidreau's modification is that it does not increase the size of the public key.

4.2 Conversions

As explained earlier it is often beneficial for the generator matrix of a code to be in systematic form which makes it easier to extract the message after errors have been removed. When a systematic generator matrix G is used to encrypt a message $\boldsymbol{m}$ the resulting encoded string with errors $\boldsymbol{z}$ will look like the following $\boldsymbol{m}G + \boldsymbol{z} = (\boldsymbol{m}||\boldsymbol{p}) + \boldsymbol{z}$ where $\boldsymbol{p}$ is redundant information which allows decoding. Because $\text{Len}(\boldsymbol{m}||\boldsymbol{p})$ is much larger than $\text{Weight}(\boldsymbol{z})$, the overall content of the plaintext is directly visible in the ciphertext. To prevent that the message is transmitted in almost clear text it is important to apply a conversion.

Another important challenge the conversion needs to tackle is to hide the structure of the underlying code so that it is difficult to extract information about the private decoder by looking at the public key.

Overall, the goal of a conversion is to transform a cryptosystem with some security properties into a different system with better properties, preferably IND-CCA2.

The original MEC uses a permutation and a row scramble matrix to

Several generic conversions for the McEliece cryptosystem have been discussed by Kobara and Imai, and they also came up with some conversions that are designed specially with the McEliece cryptosystem in mind. This section mostly gives an overview of their work, with some additional notes.

All of the discussed conversions depends upon a bijective function $\text{Conv}(\bar{z})$ which converts an integer $\bar{z} \in Z_N$, where $N = C(n, t)$ into a corresponding error vector z. And its inverse is represented as $\text{Conv}^{-1}(z)$. This correspondence

between an integer $0 \leq \bar{z} < N = C(n,t)$ and a unique t-combination is often refereed to as the combinatorial number system of degree t. Let $c_t > \cdots > c_2 > c_1$ be the positions in the error vector which are 1. The corresponding integer is $\bar{z} = C(c_t,t) + \cdots + C(c_2,2) + C(c_1,1)$. Finding the combination $c_t > \cdots > c_2 > c_1$ from an integer $\bar{z}$ can be done using a greedy algorithm which always subtracts the largest possible value $C(c_i,i)$ from $\bar{z}$ such that the result is greater or equal to 0. The process is repeated until the result is zero.

Also note that the discussed conversions assumes MEC variants with input and output both strings of symbols over the binary field.

4.2.1 Insufficient conversions

As noted by Kobara and Imai, the Optimal Asymmetric Encryption Padding (OAEP) by Bellar and Rogaway does not work as intended with the McEliece-like it requires the underlying primitive to be a One-Way Trapdoor Permutation which McEliece-like systems are not.

Fujisaki-Okamot's Simple Conversion is also not suitable as the McEliece cryptosystem is distinguishable under chosen ciphertext attacks.

4.2.2 Pointcheval's generic conversion

The McEliece primitive can be categorized as a partial trapdoor one-way function. The requirement for an encrypton function to be a PTOWF is that it should be infeasible to get back the plaintext from the ciphertext unless some extra secret information is provided for which case it becomes easy to get back the plaintext.

Pointcheval's generic conversion transform any PTOWF into a PKC which is IND-CCA2.

4.2.3 Fujisaki-Okamoto's generic conversion

This conversion converts one-way encryption functions into a PKC which is IND-CCA2.

4.2.4 Kobara-Imai's conversions

The generic conversion obviously work for MEC, but in terms of overhead data it is possible to do much better by creating a conversion specially tailored towards MEC, which have been done by Kobara and Imai.

All of Kobara-Imai's conversions have depends on the parameters of n, k and t of the MEC in addition to the length of some constant sequence $Len(Const)$ and the length of some randomly generated sequence $Len(r)$, all of which is publicly known.

Chapter 5

Attacks against code-based systems and countermeasures

5.1 Generalized Information-Set-Decoding Attacks

This attack amounts to finding a set of k coordinates which are error free in the received word. Here k is the length of the plaintext block.

Consider a standard McEliece encryption: $c = m\bar{G} + e$ where $\bar{G}$ is SGP.

Now, let c_k be the vector only containing k selected positions in c. $\bar{G}_k$ is the corresponding selected columns from $\bar{G}$ and e_k the positions in e.

e is very sparse so there is a chance of picking k error free coordinates. In these cases e_k is zero so $c_k = m\bar{G}_k + e_k = m\bar{G}_k$. By some luck $\bar{G}_k$ is invertible which means one can find m as $\left(m\bar{G}_k\right)\bar{G}_k^{-1}$. This variant is the specialized Information-Set-Decoding. The Generalized Information-Set-Decoding attack allows finding m by guessing a value e_k which may not be zero.

5.1.1 Countermeasure

Apply Loidreau's modification or increase the size of the parameters.

5.2 Reaction Attacks

A reaction attack is made possible by observing and taking advantage of the different reactions by a legit recipient of some ciphertext.

5.2.1 A message recovery attack

Assume a receiver who has the private key reacts to incoming messages in two different ways. If the errors where uncorrectable or the plaintext was meaningless a repeat request is returned. Otherwise, an acknowledgment or nothing is returned. If Alice want to send a message to Bob, an adversary can copy the ciphertext during transmission. Then one or a few bits can be flipped and sent to Bob. If the bits flipped was not part of the error vector the total weight of the error will increase and so Bob will not be able to correct the errors. Hence, a repeat request should be expected. On the other hand, if the bits flipped

was partially a part of the error vector in such a way that the total number of errors does not exceed the amount expected by Bob then one should expect an acknowledgment. By repeating the procedure a number of times for the same ciphertext one can identify the error vector. Then one can use the generalized information set decoding attack to find the plaintext.

5.2.1.1 Countermeasure

Apply conversion by Kobara and Imai.

5.2.2 GJS attack: a key recovery attack using decoding errors

Another type of reaction attack was recently published by Qian Guo, Thomas Johansson and Paul Stankovski. Their attack against the QC-MDPC system can recover the secret key of the system.

For their attack they use the notion of a distance spectrum. For a bit pattern $\boldsymbol{x} = [x_0, ..., x_{k-1}]$ with some length k the distance spectrum is defined as $D(x) = \left\{d : 1 \leq d \leq \left\lfloor \frac{k}{2} \right\rfloor, \exists a, b \in \mathbb{Z}, x_a = 1, x_{a+d \mod k} = 1\right\}$. The same distance d can appear multiple times and in the pattern and therefore they also introduce the multiplicity $\mu(d)$. The idea behind the attack is that if one is able to find the distance spectrum of the private key h_0 it is possible to find the h_0 itself.

Reconstruction follows a simple procedure. Assume d_0 is the smallest distance in $D(h_0)$. Set the zeroth and d_0-th position to 1. Then test the bit at position $2d_0$. If the distances from this bit to the previously set bits all appears in the $D(h_0)$ set the bit, otherwise test the bit at position $2d_0 + 1$. Also remember that the total length of h_0 is k.

To obtain the distance spectrum $D(h_0)$ one can observe the decoding error probabilities for different error patterns. Their observation is that if an error pattern with at least one pair with distance d is used then the decoding error probability when $d \in D(h_0)$ is smaller than if $d \notin D(h_0)$.

While their attack was made specifically for the QC-MDPC system, there is no reason to believe that other code-based systems with non-zero decoding failure probability cannot be attacked by a similar type of attack.

5.2.2.1 Circumvention

Low decoding error probability for valid error patterns As suggested by the attack authors one possible countermeasure is to keep the decoding error probability very small which makes the attack harder to utilize, here they suggest a decoding error probability of 2^{-80} for 80-bit security. As a result a more modest value of t may need to be chosen which can potentially open up for other types of attacks.

Also obviously it is important that if a decoder is able to successfully correct more than t errors for a received cipher text it should respond with a decoding error anyway. This prevents an attacker from experimenting with adding more errors than intended to increase the decoding error ratio.

Using key pairs for only a limited number of transmissions By changing the keys after a limited number of transmissions there will not be enough

trials for the attack to give a good estimate for decoding error probabilities for various error patterns. This may open up for various denial of service attacks as a receiver can be spammed with random messages and thus forcing the receiver to update the keys sooner than expected.

Make the main observation less noticeable A different approach to circumvent the attack is to, in some way, make the main observation less significant so that the error probability when $d \in D(h_0)$ is no longer noticeably smaller. At key creation one can find the error correction failure rates for error patterns with ds compared to the rest. Let the average difference be Δ. Now determine a random partitioning of all error patterns containing a distance in $D(h_0)$ into two subset A and B where B contains about Δ percent of the total. If the decoding of a received word reveals an error pattern with a distance in $D(h_0)$ then lookup the error pattern to see if its a part of A or B. If its a part of B then return a decoding error.

Consequences are more complex key generation and decryption. Also finding a good way of efficiently representing the partitioning can be difficult.

Introducing fake ds If one is willing to reduce t slightly, at key creation one can create a sufficiently large set of fake ds which will be biased with less decoding errors at the decryption step. Whether or not an error pattern over some message should return an error or not must be determined by a deterministic function and not some random function as this will allow the attacker to send the same error pattern twice to check if it contains fake ds. The trickiest part is of course to make the fake ds indistinguishable from the real ones.

This countermeasure has a slight impact on general decoding capability t and adds some extra complexity to the decryption and the key generation step.

Chapter 6

MDPC and QC-MDPC McEliece

As mention earlier one of the main issues with the McEliece cryptosystem is the huge public key required to obtain a decent security level. Numerous modifications to the original system have been proposed trying to reduce the key size, many of which have been challenged with severe security vulnerabilities.

One of the more promising proposals is based upon replacing the Goppa codes used in the original system with Low Density Parity Check (LDPC) codes. LDPC codes is a class of linear forward error correcting block codes first discovered by Robert G. Gallager in 1960. They are known for having an error correction capability very close to the Shannon limit. An LDPC code can be uniquely specified as the null space of a parity-check matrix H. For a linear block code to be in the class of regular LDPC codes, the parity check matrix must satisfy the 4 additional properties. First, the weight of each row is constant. Second, the weight of each column is constant. Third, any two columns cannot have more than one 1 in common. Finally, it is required that both the row weight and column weight are small compared to the length of the code and number of parity checks respectively. There is also a less strict class of LDPC codes without the second and third requirement.

Some PKCs using LDPC codes is vulnerable to KPA attacks aiming at finding low weight codewords in the dual code of the public code. These low weight codewords can then be used to construct a LDPC matrix which reveals an efficient decoder.

A very similar family of codes is Moderately Density Parity Check Codes (MDPC). In contrast to the LDPC codes the number of non-zero entries in each row of the parity check matrix grows with the size of the matrix, usually by a growth rate of $O(log(n))$. As with LDPC codes there is also a variant of the McEliece cryptosystem which uses MDPC codes as suggested by Misoczki et al.. An advantage of using an MDPC or LDPC code is that they have little to none apparent algebraic structure. The scrambler and permutation matrix in the original McEliece system can therefore become redundant as their purpose was solely to hide the algebraic structure of the underlying code. Also compared to the LDPC codes there is a larger amount of valid parity check matrices, and thus increasing the size of the private key space. Decoders for MDPC codes

Algorithm 6.1 Classical multiplication

Input: Two polynomials $a(x) = \sum_{i=0}^{d-1} \alpha_i x^i$ and $b(x) = \sum_{i=0}^{d-1} \beta_i x^i$

1. Partition the polynomials into n blocks all of which having exactly ω terms with a possible exception of the last term which are allowed to have $\omega + \delta$ terms where δ is some integer less than or equal to zero.
2. Compute $a(x) = \sum_{i=0}^{d-1} \alpha_i x^i = \left(\sum_{i=0}^{\omega-1} \alpha_i x^i\right) + \left(\sum_{i=\omega}^{2\omega-1} \alpha_i x^i\right) + \cdots + \left(\sum_{i=(n-2)\omega}^{(n-1)\omega-1} \alpha_i x^i\right) + \left(\sum_{i=(n-1)\omega}^{n\omega-1+\delta_\alpha} \alpha_i x^i\right)$

generally have a lower error-correction capability than decoders for LDPC codes and this is probably one of the greatest disadvantages of using such codes in a McEliece-like cryptosystem. This implies that fewer errors can be introduced in the encryption phase.

While LDPC and MDPC codes are good because of their lack of algebraic structure they do only partially solve the problem of large key-sizes. Since the density of the parity check matrix is low, only storing the coordinates of the non-zero entries can be more efficient, especially for very large matrices. The public key is however not necessarily of low density. Multiple proposals suggest to use quasi-cyclic variants of LDPC and MDPC codes. The quasi-cyclic structure provides opportunities to reduce the size of the private parity check matrix and public generator matrix drastically. In the suggested QC-MDPC system by Misoczki et al., the parity check matrix consists of circulant block, each of which can be represented by only the first row.

Performing matrix multiplication for very large matrices are in general relatively expensive with a computational complexity of $\mathrm{O}(n^{2.807355})$ for the best known practical algorithm. However, in the quasi-cyclic case the matrices involved in the computations will be block circulant which allows for much faster computations. When doing matrix multiplication for general matrices one is limited by an absolute lower bound for the complexity of $\mathrm{O}(n^2)$ because one must look at each of the n^2 elements of each of the two matrices at least once. In the circulant case, however, the matrices is fully determined by the first row and therefore one can expect multiplication to be performed with a complexity closer to $\mathrm{O}(n)$.

Consider a classic matrix multiplication algorithm where each entry a in the resulting matrix is the dot product of a corresponding row in the left operand and a corresponding column in the right operand. If both the right and left operand are circulant the result will also be a circulant matrix.

A useful fact that is stated in a lot of papers is that there exists an isomorphism between the algebra of circulant matrices in $\mathbb{F}_2^{p\times p}$ and the ring of polynomials in $\mathbb{F}_2[x]/(x^p+1)$. This means that the circulant matrix multiplications can be reduced to the problem of multiplying two polynomials which are much less complex.

A different proposal is to let the generator matrix be composed of circulant blocks. This means that the size of the public key can be reduced.

Using Moderate Density Parity Check codes for the McEliece cryptosystem was suggested by Misoczki et al. [21]. An advantage of using quasi cyclic codes

Algorithm 6.2 QC-MDPC Key Generation

Given n and n_0 where $gcd(n, n_0) = 1$

1. Construct $H = [H_0, H_1, \ldots, H_{n_0-1}]$ where H_i is a $p \times p$ circulant block, by picking a random vector of length n with weight w.
2. Public key is $G = \left[\begin{array}{c|c} & \left(H_{n_0-1}^{-1} \cdot H_0\right)^T \\ & \left(H_{n_0-1}^{-1} \cdot H_1\right)^T \\ I & \vdots \\ & \left(H_{n_0-1}^{-1} \cdot H_{n_0-2}\right)^T \end{array}\right]$
3. Private key is H

Algorithm 6.3 QC-MDPC Encryption

Let $m \in \mathbb{F}_2^{n-r}$ be the plaintext to be encrypted and (G, t) the public key

1. Generate an error vector e with weight t
2. Compute and return $mG + e$

is that the public generator matrix can be represented by only the first row and thereby reducing the size of the public key drastically.

6.1 Relation between private and public key

From how the QC-MDPC keys are generated it is not immediately clear that it is (or if it is) hard to retrieve the original parity check matrix from the public generator matrix. The public generator has the following form:

$$G = \left[\begin{array}{c|c} & C_0^T \\ & C_1^T \\ I & \vdots \\ & C_{n_0-2}^T \end{array}\right]$$

Where C_i is a circulant and probably dense matrix. Each C_i are constructed from $\left(H_{n_0-1}^{-1} \cdot H_i\right)$. Here $H_{n_0-1}^{-1}$ is a common factor in all C_i, so if found, the reconstruction of H can easily be accomplished. For this problem it may be easier to regard the circulant matrices as polynomials. The goal is to find a common factor for all C_is. If $n_0 - 1$ is large, this common factor is with high probability $H_{n_0-1}^{-1}$. Also, the other factors H_i should be of moderately low

Algorithm 6.4 QC-MDC Decryption

Let $mG + e$ be the received ciphertext and H the private key

1. Find mG by applying the efficient decoder
2. Find and return m by extracting the first $n - r$ positions of mG

weight, which can possibly make the search a little easier.

6.2 Suggested parameters

Misoczki et al suggested the following parameters for their system:

Table 6.1: Misoczki et al. suggested parameters

Level security	n_0	n	r	w	t	QC-MDPC key-size
80	2	9602	4801	90	84	4801b
80	3	10779	3593	153	53	7186b
80	4	12316	3079	220	42	9237b
128	2	19714	9857	142	134	9857b
128	3	22299	7433	243	85	14866b
128	4	27212	6803	340	68	20409b
256	2	65542	32771	274	264	32771b
256	3	67593	22531	465	167	45062b
256	4	81932	20483	644	137	61449b

Operating on non-compressed matrices over $\mathbb{F}_2$ of the suggested sizes would yield a very large memory footprint. Since$\mathbb{F}_2$ only contains two distinct elements one can represent each element by a single bit. However, in most computers the smallest primitive that can be accessed and manipulated directly is a byte of 8-consecutive bits. To manipulate each bit directly it is possible to use a whole byte to represent a single bit. For $n = 81932$ and $r = 20483$ the H matrix will need 1.6 GB which is unacceptable for many applications. A different approach is to utilize bitwise operations which most processors provides. This allows indirect manipulation and access of individual bits within a byte and the storage required will be reduced to about 200 MB.

From w it is evident that the matrices are fairly sparse. This allows a different kind of representation where only the indices of the set bits are stored. Using 32-bit integers one can represent indices up to 2^{32} which is enough to give each entry of the largest matrix a unique index. This gives a memory usage of about 50 MB.

Another great feature of the matrix is its quasi cyclic structure which means only the first row needs to be stored. With a bit set representation this gives a memory usage of just 2.5kB.

Chapter 7

The NIST standardization process

The National Institute of Standards and Technology (NIST) in the U.S. has initiated a process to evaluate and standardize one or more post-quantum cryptosystems and by the time of this writing the submissions of round one are evaluated. The standardization is expected to be completed between 2022 and 2024.

The motivation for starting the process this early is, according to NIST, a combination of two factors. First, there have been major progress in the development of quantum computers. Second, the standardization process will require a lot of effort, the transition between the currently used cryptosystems and the quantum resistant replacements is expected to take time and the standardization and transition should be done well before any large-scale quantum computers are build.

This chapter provides a rough overview of some of the code based submissions. Most of the submission are so called Key Encapsulation Mechanisms (KEM) which is encryption techniques for sharing a symmetric key using public key algorithm. Ignoring signature schemes, the main usage for PKCs today is for exchanging symmetric keys. This is probably the major reason why submitters have chosen to focused on KEMs in precedence of more general purpose PKCs. KEMs can be easier to secure as only a very limited number of messages needs to be transmitted for the key exchange to take place, which makes various attack based on statistics collection difficult. Additionally, KEMs usually do not require very large plaintext message blocks. For AES, which is a very popular symmetric key cryptosystem, a very conservative key length option is 256-bits, and this is believed to remain conservative even in a post-quantum world. Because the plaintext block length of the KEMs can be very short, a large message expansion factor is less worrisome.

7.1 Security strength categories

NIST requires that systems, given some parameters specified by the system designers, is classified by five categories:

- Level 1: Complexity of attacks must be comparable or greater than that required by a key search on a block cipher with 128-bit keys.
- Level 2: Complexity of attacks must be comparable or greater than that required by a collision search on a 256-bit hash function.
- Level 3: Complexity of attacks must be comparable or greater than that required by a key search on a block cipher with 192-bit keys.
- Level 4: Complexity of attacks must be comparable or greater than that required by a collision search on a 384-bit hash function.
- Level 5: Complexity of attacks must be comparable or greater than that required by a key search on a block cipher with 256-bit keys.

For a system to be in any of the levels it must satisfy the complexity requirement with respect to any metric that NIST deems relevant. This can include number of execution cycles, memory usage and more.

7.2 Proposals

This section contains a quick overview over the code based proposals. It is not a complete overview and does not contain any signature schemes.

BIG QUAKE This proposal by Bardet et al. uses quasi-cyclic Goppa codes in a Niederreiter-like PKC shown to be IND-CPA, and is then converted to a IND-CCA2 KEM through a generic conversion[10].

The BIKE suite The Bit Flipping Key Encapsulation (BIKE) is a collection of KEMs proposed by Aragon et al. [2]. The systems proposed are BIKE-1, BIKE-2 and BIKE-3 which are all based upon quasi-cyclic codes and uses bit flipping decoders. They have also shown that the systems are IND-CPA.

As the PKC key-pair is only used for a single key-exchange attacks like GJS cannot be used.

BIKE-1 makes a trade-of between having a larger public key in exchange of a faster key generation. Since BIKE is a key encapsulation protocol it is important that the key generation procedure is fast as a new key pair must be made for every new connection.

BIKE-2 can make use of a batch key generation technique described in the proposal to significantly increase the efficiency of the key-generation step while keeping a public key in systematic form.

Classical McEliece As the Classical McEliece system has withstood any significant attacks since it was presented by R. J. McEliece in 1978, isn't it only natural to include it as one of the submissions? Well, despite its name, this submission by Bernstein et al. presents a key encapsulation scheme designed to be IND-CCA2 based on Niederreiter's dual version of the McEliece's encryption using binary Goppa codes[9].

DAGS DAGS is a KEM by Banegas et al. that is IND-CCA and utilizes structured algebraic codes.

HQC and RQC Melchor et al. proposed HQC[19] which is a KEM obtained through a conversion from a IND-CPA PKC. The conversion provides IND-CCA2 security. HQC uses Tensor Product codes.

In a separate submission the same authors (except one) have also proposed somewhat similar KEM called RQC[20]. Like RQC it achieves IND-CCA2 security through a conversion, but in in contrast this KEM is focuses on Gabidulin codes.

LAKE and LOCKER In a proposal by Aragon et al., a variation of low rank parity check codes is used for their KEM called LAKE[3] which is IND-CPA secure. In a separate submission by the same authors they introduces a very similar KEM called LOCKER[4] which is IND-CCA2 secure.

LEDAkem and LEDApkc Baldi et al. have submitted both a key encapsulation scheme and a public key cryptosystem. LEDAkem[6] is uses QC-LDPC with a new decoding algorithm and used ephemeral keys to prevent attacks like GJS. LEDAkem is IND-CPA secure. In the supporting documentation for the LEDAkem system, table 3.2 shows that the size of the shared secret is much larger than the encapsulated secret. This is inconsistent with the provided KAT.rsp file and also it does not make sense for the shared secret to get smaller after encapsulation. The table does make sense if the column headings are swapped, which was assumed when the data for the comparison below was collected. The parameter sets with $n_0 = 4$ was chosen in the comparison as the authors claims that is the most advantageous choice.

LEDApkc[7] is, like LEDAkem, based on QC-LDPC codes, but can transfer much larger messages. It is protected against GJS attack by giving each keypair a secure lifetime so that they must be renewed after a fixed number of decoding failures is encountered. It uses a conversion to achieve IND-CCA2 security.

Lepton Lepton[28] is a IND-CCA secure key encapsulation mechanism introduced by Yu et al. based on low-noise instances of the learning parity with noise problem which is a special case of the decoding problem for random linear codes.

McNie McNie[12] by Galvez et al. is a hybrid version of the McEliece and the Niederreiter cryptosystem using low rank parity check codes. It obtains IND-CCA2 security through a conversion.

NTS-KEM Albretch et al.'s submission proposes the NTS-KEM[1], which obviously from its name is a KEM. The system can be seen as a variant of the McEliece or Niederreiter PKC, but is in contrast a KEM. It achieves IND-CCA security through a conversion.

Ouroboros-R Melchor et al. have presented a IND-CPA secure KEM based on QC low rank parity check codes called Ouroboros-R[18].

QC-MDPC KEM QC-MDPC KEM[27] is a KEM from Yamanda based on a QC-MDPC McEliece variant and obtains IND-CPA security. It can be used with ephemeral keys to prevent attacks like GJS.

RLCE-KEM The RLCE-KEM[26] by Wang is a KEM which is specified over all efficiently decodable linear codes with generalized Reed-Solomon codes being recommended. By using a conversion, it is claimed to be IND-CCA2 secure.

7.3 Comparison

The data for this comparison is collected from the corresponding submission papers and / or estimated from the provided KAT.rsp files. In some cases where data was not explicitly given by neither, but clear instruction on how to calculate was stated, the values have been found using these instructions. Some numbers have been converted to a common format for easier comparison. Message expansion is calulated from the ciphertext and plaintext length.

The KAT.rsp is used to verify that the submissions work as expected. For given seeds they contains the expected generated public key (pk), expected generated private key (sk), expected ciphertext (ct) and expected shared secret (ss), all provided as hexadecimal numbers. In most cases the number of hexadecimal pairs in each number match the number of bytes stated by the authors to be the byte length of the value in the submission papers. However, this is not true for all submissions, like for instance is the case with the BIKE suite.

Some of the submissions also uses a session key, ephemeral key or alike. These have not been specifically accounted for.

While most of the submissions do contain timings for encryption / encapsulation, decryption / decapsulation and key generation, the measurements have been performed on different machines with varying capabilities and in that regard they are not comparable. A possibility could have been to run benchmarks on the same device to compare them, but then the implementations may not be likewise optimized for the given device. Nevertheless, it would not give much meaningful insight as devices are different and some systems may be more suited for some devices than others.

Some submissions includes multiple parameter sets for each security level, in these cases a parameter set was chosen pseudorandomly unless otherwise stated either above or in the table. If no parameter set was provided for a given security level then it is left blank.

System		Level 1	Level 2-3	Level 4-5
BIG QUAKE	Private key	118176b	246880b	334432b
	Public key	203856b	673056b	1198400b
	Ciphertext	1608b	3264b	3936b
	Plaintext	256b	256b	256b
	Expansion	6.28	12.75	15.38
BIKE-1	Private key	2130b	2296b	4384b
	Public key	20326b	43786b	65498b
	Ciphertext	20326b	43786b	65498b
	Plaintext	256b	256b	256b
	Expansion	79.40	171.04	255.85
BIKE-2	Private key	2130b	3296b	4384b
	Public key	10163b	21893b	32749b
	Ciphertext	10163b	21893b	32749b
	Plaintext	256b	256b	256b

	Expansion	39.70	85.52	127.93
BIKE-3	Private key	2010b	3168b	4522b
	Public key	22054b	43366b	72262b
	Ciphertext	22054b	43366b	72262b
	Plaintext	256b	256b	256b
	Expansion	86.54	169.40	282.27
Classic McEliece	Private key			111288b
	Public key			8378552b
	Ciphertext			1808b
	Plaintext			256b
	Expansion			7.50
DAGS	Private key	3461120b	10272768b	17542176b
	Public key	54080b	67584b	92928b
	Ciphertext	4416b	7552b	12928b
	Plaintext	512b	512b	512b
	Expansion	8.62	14.75	25.25
HQC	Private key	320b	320b	320b
	Public key	22552b	40920b	59336
	Ciphertext	44976b	81712b	118544b
	Plaintext	512b	512b	512b
	Expansion	87.85	159.59	231.53
LAKE	Private key	320b	320b	320b
	Public key	3384b	5088b	6632b
	Ciphertext	3384b	5088b	6632b
	Plaintext	512b	512b	512b
	Expansion	6.61	9.94	12.95
LEDAkem	Private key	192b	256b	320b
	Public key	51264b	105216b	181632b
	Ciphertext	17088b	35072b	60544b
	Plaintext	256b	384b	512b
	Expansion	66.75	91.33	118.25
LEDApkc	Private key	192b	256b	320b
	Public key	27840b	57600b	99072b
	Ciphertext	55680b	115200b	198144b
	Plaintext	29520b	60464b	103176b
	Expansion	1.89	1.91	1.92
Lepton	Private key	320b	448b	592b
	Public key	8360b	16416b	16416b
	Ciphertext	15728b	23784b	31912b
	Plaintext	256b	256b	256b
	Expansion	61.44	92.91	124.66
LOCKER	Private key	8400b	11032b	11856b
	Public key	7979b	9991b	11021b
	Ciphertext	8491b	10503b	11533b
	Plaintext	512b	512b	512b
	Expansion	16.58	20.51	22.53
McNie	Private key	2720b	3720b	4672b

	Public key	2776b	3896b	5040b
	Ciphertext	3376b	4720b	6088b
	Plaintext	1720b	2688b	3456b
	Expansion	1.96	1.75	1.76
NTS-KEM	Private key	73728b	140192b	159120b
	Public key	2555904b	7438080b	11357632b
	Ciphertext	1024b	1296b	2024b
	Plaintext	256b	256b	256b
	Expansion	4.00	5.06	7.91
Ouroboros-R	Private key	9440b	11920b	17024b
	Public key	9440b	11920b	17024b
	Ciphertext	9440b	11920b	17024b
	Plaintext	512b	512b	512b
	Expansion	18.44	23.28	33.25
QC-MDPC KEM	Private key	1440b	2272b	4384b
	Public key	4801b	9857b	32771b
	Ciphertext	9858b	19970b	65798b
	Plaintext	256b	256b	256b
	Expansion	38.51	78.01	257.02
RLCE-KEM	Private key	1439568b	3520064b	8385408b
	Public key	947528b	2298968b	5936712b
	Ciphertext	6280b	9904b	16184b
	Plaintext	512b	512b	512b
	Expansion	12.27	19.34	31.61
RQC	Private key	1491b	2741b	3510b
	Public key	1491b	2741b	3510b
	Ciphertext	1555b	2805b	3574b
	Plaintext	64b	64b	64b
	Expansion	24.30	43.83	55.84

Some of the systems in the table above use seed expansion procedures. This means that even if for instance the private key for some system is small when stored away it needs to be expanded to actually be usable in computations.

Though, it is bad to exclude candidates by just looking at their key and message sizes there are some systems that clearly stands out as less suitable for certain applications than the rest. For example, DAGS, having a private key of almost a half megabyte for security level 1, can be challenging to deploy on certain embedded devices with very limited memory. On the other hand, a half megabyte is not a lot and its probably something most modern devices can afford, especially if it means less complex encryption and decryption procedures and lower power consumption. When it comes to message expansion the overall trend is that the KEMs have larger expansions than the more general PKCs. McNie and LEDApkc, which are both general PKCs, have message expansion of factor less than 2, while the KEMs lies everywhere from a factor of 4 to several hundreds.

Figure 7.1, 7.2 and 7.3 compares the systems according to their private key size, public key size and ciphertext size respectively. Note that the three diagrams are all using the logarithmic scale base 2 which may lead to the impression

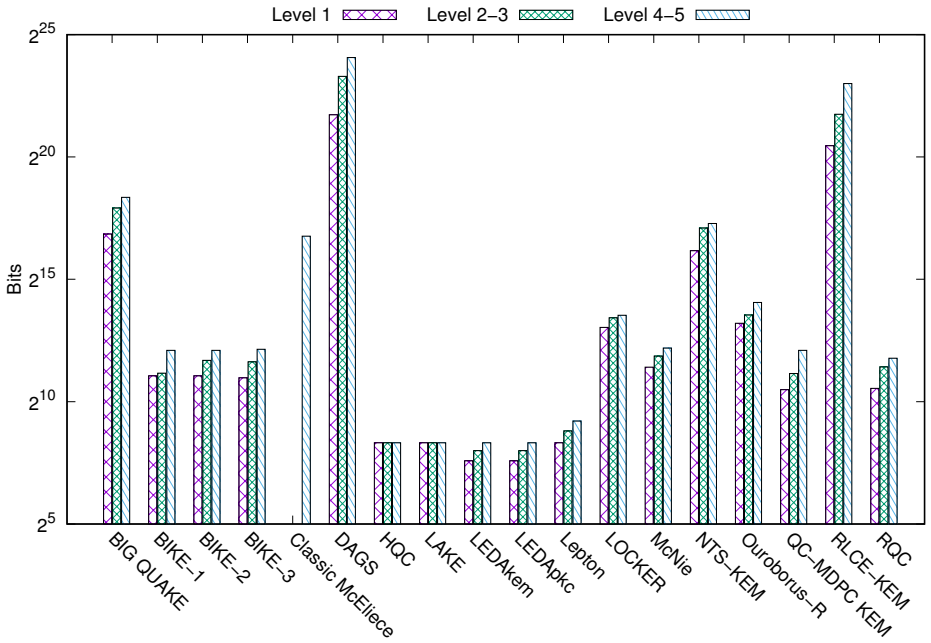

Figure 7.1: Private key sizes

that the systems are more alike than they actually are.

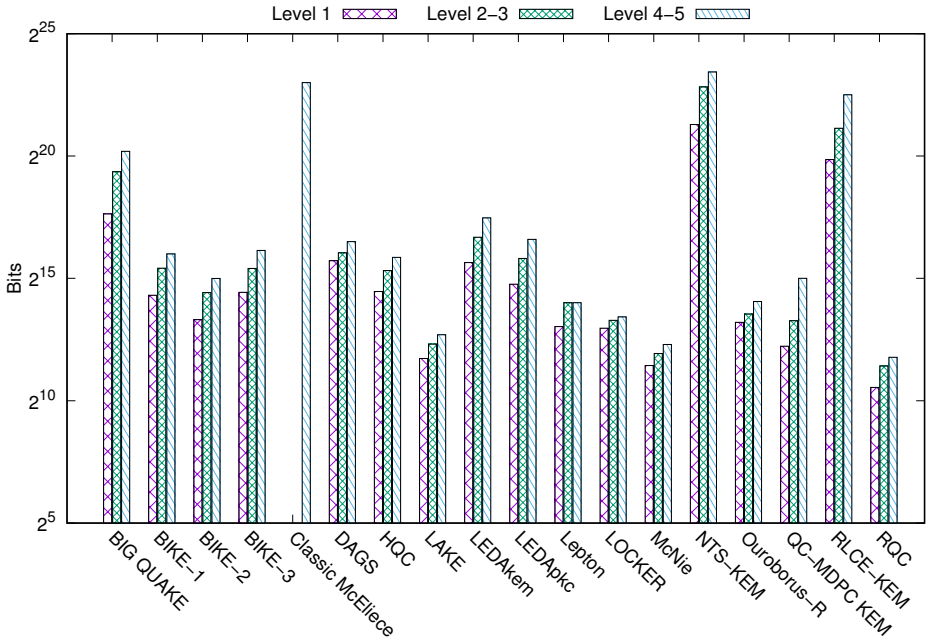

Figure 7.2: Public key sizes

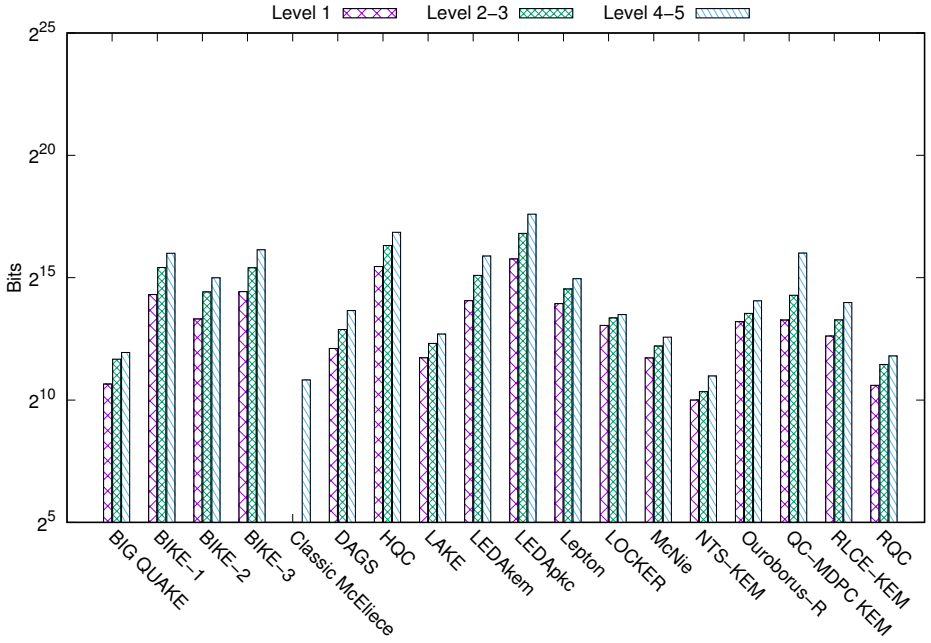

Figure 7.3: Chipertext sizes

Chapter 8

Afterthoughts and conclusion

The original goals of this project involved creating an overview of post-quantum cryptosystems and compare them with respect to security, key sizes, message expansion and complexity. It turned out to be a difficult task when one really doesn't understand the systems that are being compared. Some of the vocabulary used in associated papers were hard to grasp, and because of limited mathematical intuition and knowledge it was difficult to follow some of the deductions. In that regard this thesis is a "best effort" type of work, but what else was there to expect really?

The NIST Post-Quantum Cryptography standardization process helped enlighten many innovative post-quantum solutions. This thesis has aggregated some of the characteristics of the code-based PKCs candidates. Even if it wasn't particularly useful, it did to some degree provide perception of what one can expect in terms of public key, private key, ciphertext and plaintext sizes in a possibly code-based standard.

Also, one of the goals of this thesis was to actually implement a McEliece-like system. The implementation took a lot of time and ended up as a failure. In retrospect there are several reasons why the implementation work failed:

- Lack of understanding of the system that was being implemented.
- Inexperience with software projects of this kind and to some degree software development in general.
- Unawareness of tools that could have made the implementation work less extensive.
- No clear scope of the implementation work.

Nevertheless, even if the implementation didn't succeed, it provided some insight into various different implementation obstacles, a few of which wouldn't even have been necessary to overcome to complete the project. This thesis has explained, in various details, some of these obstacles and how one could have solved them.

8.1 Further work

While code-based cryptography is certainly interesting for post-quantum cryptography many of the system proposals with a few exceptions have had a relatively short lifespan before they have been broken. Starting to use one of these systems now is highly precarious. In the transition period before a large-scale quantum computer is built, hybrid schemes that is based on both well-established systems like RSA and ElGamal, and also a code based system could be interesting as they have the potential to fall back to the security of RSA or ElGamal if the code based system is broken.

Bibliography

[1] Martin Albrecht, Carlos Cid, Kenneth G. Paterson, Cen Jung Tjhai, and Martin Tomlinson. Nts-kem. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/NTS _KEM.zip 21.08.2018 08.33 UTC+01:00 DST.

[2] Nicolas Aragon, Paulo S. L. M. Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. Bike. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/BIKE.zip 21.08.2018 08.33 UTC+01:00 DST.

[3] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zemor. Lake. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/LAKE.zip 21.08.2018 08.33 UTC+01:00 DST.

[4] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zemor. Locker. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/LOCKER.zip 21.08.2018 08.33 UTC+01:00 DST.

[5] Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos, Johannes Buchmann, Wouter Castryck, Orr Dunkelman, Tim Güneysu, Shay Gueron, Andreas Hülsing, Tanja Lange, Mohamed Saied Emam Mohamed, Christian Rechberger, Peter Schwabe, Nicolas Sendrier, Frederik Vercauteren, and Bo-Yin Yang. Initial recommendations of long-term secure post-quantum systems, September 2015.

[6] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. Ledakem. NIST post-quantum

cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/LEDAkem.zip 21.08.2018 08.33 UTC+01:00 DST.

[7] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. Ledapkc. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/LEDApkc.zip 21.08.2018 08.33 UTC+01:00 DST.

[8] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractibility of certain coding problems. *IEEE Transactions on Information Theory*, 24:384–386, May 1978.

[9] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wen Wang. Classic mceliece. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/Classic _McEliece.zip 21.08.2018 08.33 UTC+01:00 DST.

[10] Alain Couvreur, Magali Bardet, Elise Barelli, Olivier Blazy, Rodolfo Canto-Torres, Philippe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich. Biq quake. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/BIG _QUAKE.zip 21.08.2018 08.33 UTC+01:00 DST.

[11] Tomás Fabsic, Otokar Grosek, Karol Nemoga, and Pavol Zajac. On generating invertible circulant binary matrices with prescribed number of ones. *Cryptogr. Commun.*, June 2017.

[12] Lucky Galvez, Jon-Lark Kim, Myeong Jae Kim, Young-Sik Kim, and Nari Lee. Mcnie. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/McNie.zip 21.08.2018 08.33 UTC+01:00 DST.

[13] Lov K. Grover. A Fast quantum mechanical algorithm for database search. 1996.

[14] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9, January 1883.

[15] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47, February 2001.

[16] Pierre Loidreau. Strengthening mceliece cryptosystem. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, pages 585–589, Berlin, Heidelberg, 2000. Springer-Verlag.

[17] Robert. J. McEliece. A public-key cryptosystem based on algebraic coding theory, 1978.

[18] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Adrien Hauteville, and Gilles Zemor. Ouroboros-r. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/Ouroboros _R.zip 21.08.2018 08.33 UTC+01:00 DST.

[19] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. Hpq. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/HQC.zip 21.08.2018 08.33 UTC+01:00 DST.

[20] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. rqc. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/RQC.zip 21.08.2018 08.33 UTC+01:00 DST.

[21] R. Misoczki, J. P. Tilich, N. Sendrier, and P. S. Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *Information Theory Proceedings*. IEEE International Symposium, 2013.

[22] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problemy Upravlenija i Teorii Informacii*, 15:159–166, 1986.

[23] Eleanor G. Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32:300–335, September 2000.

[24] Ron Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key. *Communications of the ACM*, 21:120–126, February 1978.

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

[26] Yongge Wang. Rlce-kem. NIST post-quantum cryptography standardization submission, November 2017. Downloaded

from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/RLCE.zip 21.08.2018 08.33 UTC+01:00 DST.

[27] Atsushi Yamada, Edward Eaton, Kassem Kalach, Philip Lafrance, and Alex Parent. Qc-mdpc kem. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/QC _MDPC _KEM.zip 21.08.2018 08.33 UTC+01:00 DST.

[28] Yu Yu and Jiang Zhang. Lepton. NIST post-quantum cryptography standardization submission, November 2017. Downloaded from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/Lepton.zip 21.08.2018 08.33 UTC+01:00 DST.