Postquantum cryptography: An overview on the McEliece cryptosystem

Jannis Priesnitz
University of Applied Sciences Darmstadt
Department of Computer Science
Schöfferstraße 3
64295 Darmstadt
Email: jannis.priesnitz@stud.h-da.de

***Abstract*—This article gives an elaborated overview of the McEliece cryptosystem from today's perspective. Starting with some basics on post quantum computing, the reason why such an cryptoscheme is necessary is discussed. The cryptosystem itself is described in detail, supported by an example. Moreover the Niederreiter cryptoscheme as a famous variant is described. For an understanding about the insights of both schemes, some details of underlying coding theory are discussed. Finally, state of the art issues and attempts to solve them are presented.**

***Index Terms*—McEliece cryptosystem, Niederreiter cryptosystem, post quantum cryptography, coding theory**

I. Introduction

State of the art asymmetric cryptosystems in combination with usage of sufficient long keys are considered as "safe" on current computer systems. This changes immediately when quantum computers enter the scene. But why is this the case?

In this paper, I briefly present the main differences between normal computer systems and quantum computers and why state of the art cryptosystems like RSA and ECC are not safe in the quantum computation world. In addition to that, I present an extensive introduction to the McEliece Cryptosystem, explain the strengths and downsides of the algorithm and give some details on the development of codes which can be used for the system.

For decades, computer systems have been seen as digital circuits based on the rules of physics. A bit in these systems is seen as a value of voltage. If this value is above a certain level, the binary representation is one, otherwise it's zero. On quantum computers, information is handled in binary, too. The main difference between the bit representation and the quantum bit representation (so called qubit) is that in addition to the two states zero and one, there are several more states which one qubit can reach due to a superposition. This means that a qubit is not either in state one or in state zero but in theory can have an arbitrary number of different states. All these states can appear with an certain probability. Each of these states is able to compute one possibility of a NP hard problem.

Due to this fact, quantum computers are able to solve problems which are NP hard, in sense of complexity, theory much faster than traditional computers.

One of these computational problems, which are traditionally hard to solve, is the prime factorization problem which is in several variations the core of many public key algorithms such as the RSA cryptosystem. The second main category, the elliptic curves cryptography, is even more affected by this problem, as the logarithm of a finite field of an elliptic curve can be computed efficiently and can be broken in less time compared to the RSA algorithm because of lower keysizes. In conclusion, there is no established asymmetric cryptoscheme which is post quantum resistant. Therefore, completely different approaches are needed.

Outline

In section II, some background information regarding post quantum cryptography is given. With help of Shor's algorithm the principle of quantum cryptography is described more precisely. Section III takes a detailed look into the McEliece cryptosystem supported by an example in section VII. The Niederreiter cryptosystem, as famous variant of the McEliece cryptosystem, follows in section IV with an additional view on signing with it in section V. The underlying error correcting codes are described in section VI, supported by some basics about coding theory in same section. In section VIII, a slight overview on attacker models and current developments such as wild McEliece are given. Finally a brief conclusion is presented in section IX.

II. Post quantum cryptography

In order to reach post quantum proof cryptographic algorithms, some fundamentally different approaches than the established ones are required. Shor's algorithm in II reveals that cryptography based on integer factorization or the discrete logarithm problem is no longer an issue for quantum computers. Therefore, other algorithm types are needed which are briefly described in section II-0b.

Shor's Algorithm

In 1994, Peter Shor presented an algorithm which is able to factorize a composite number n into its prime factors on a quantum computer. This algorithm only needs $logn$ qubits and has a runtime of $O((log\, n)^3)$ for finding a non trivial factor of n.

Procedure Shor's algorithm is divided into a classical part which can be executed on a conventional computer and a quantum part which has to be executed on a quantum computer in order to perform the computation efficiently.

The basic idea is that the classical part reduces the problem while the quantum part finds the order of the group in which n is. In this section Shor's algorithm is shortly outlined to get an idea of the method of quantum computing and the principles an quantum proof algorithm has to follow.

a) Classical part The classical part of the algorithm mainly contains computation of the greatest common divisor of a randomly selected number lower than n and n itself. Then it is necessary to compute the order r of x which is where the quantum part emerges. The classical part is executed in a loop while the order r is odd or x to the power of $r/2$ is equivalent to $-1\, mod\, n$. If this is not the case the gcd of $x^{(r/2)} - 1$ n is computed.

b) Quantum part At first, a q is determined which is a power of 2 and lies between n^2 and $2 * n^2$. To prepare the input quantum register, the superposition of all states $a\, mod\, q$ is loaded. In this case a is smaller than n. The output quantum register is initialized with all states of $x^a(modn)$. A quantum Fourier transformation is computed on the input register which is treated as a black box in this work. The result values are gathered from the output register.

To sum up, Shor's algorithm with support of a quantum computer is able to find the period of a prime in polynomial time.

With this algorithm, all cryptography based on prime factorization can be broken by a quantum computer in polynomial time. Note that not the quantum fourier transformation, which takes only $O(1)$ operations, is the bottleneck but the fast exponentiation which takes $O(log\, n)$ with the currently best implementation[1]. Together with the pre- and postprocessing, the time complexity is at $O(log(n)^3)$.

Candidates for post quantum cryptography

In this section, a short overview over promising state of the art post quantum cryptoschemes based on [2] is given.

Lattice-based cryptography One of the most studied types of algorithms is the lattice based cryptography which exists in several variants. The Algorithm works on a lattice over a n-dimensional finite Euclidian field L with a strong periodicity property. A set of vectors provides the basis of L in the way that every element is uniquely represented. The cryptographic problem is to find the closest vector to a given lattice point e.g. by adding an error vector[2][3].

Multivariate cryptography Multivariate cryptography is based on a multivariate polynomials over a finite field F which are defined over both a ground and an extension field. In the case of solving these multivariate polynomial equation systems, they are NP-complete and due to this fact a candidate for post quantum cryptography. They are topic of studies up to now and are promising especially for signature schemes[2][4].

Hash-based cryptography Hash-based algorithms such as the Lamport- [5] and the Merkle [6] signature scheme are based on strong hash functions but have the disadvantage that only a limited number signatures can be created per key. The algorithm reduces the one time signature to an hash value using a hash function[2].

Code-based cryptography The forth group, the code based algorithms, are based on error-correcting codes. First investigations have been developed by Robert McEliece using random Goppa codes[7]. This paper deals with the properies of McEliece- and the related Niederreiter cryptosystem[2][8].

This raw overview of some state of the art algorithms show different approaches but in general, it can be said that a much higher effort has to be taken to achieve a strong system compared to the traditional ones.

III. The McEliece cryptosystem

In 1978, Robert McEliece suggested an asymmetric quantum resistant cryptosystem based on the theory of algebraic codes. He selected binary Goppa codes with the property of irreducibility as a base for the cryptosystem [9]. The chosen code C has a length of $n = 2^m$ and a dimension of $k >= n - tm$ where t denotes the degree of the polynomial over $GF(2^m)$. In this case, m indicates the size of the binary field. These codes are able to correct any pattern of t or fewer errors. For each of these codes, an irreducible polynomial of degree t exists. The main reason for McEliece to select this setup is that an fast algorithm to decode these codes [10] then exists. An illustrating example can be found in section VII. More details about the underlying coding theory can be found in section VI.

Key generation

For the key-generation, a n and t with the above mentioned properties are picked. Additionally, an irreducible polynomial of degree t over $GF(2^m)$ is selected randomly. The probability that this selection leads to an irreducible polynomial is $1/t$ and there is an efficient algorithm to prove the irreducibility[11]. In the next step, a generator matrix G sized $n \times k$ is produced. This can be transformed into canonical form.

Now the information of G has to be obscured. Therefore, a random dense $k \times k$ matrix S which is nonsingular and a random $n \times n$ permutation matrix P is selected. Both of them are multiplied to $G' = SGP$. Due to the matrix multiplication properties, the linear code generated by G' has the same rate and distance like G. G' is the public generator matrix and is sent to the encrypting entity.

The following encryption algorithm is published so that the encrypting entity can use it.

Encryption

First of all, the message m which is to be encrypted has to be divided into k-bit blocks. The public key encryption is performed by $x = uG' + z$ with u being one of such k-bit blocks. z denotes a randomly generated vector with length n and weight t [1].

The vector x is the encrypted message which is transmitted to the private key owner who can decrypt the message block as follows.

Decryption

The decryption of one block x starts with computing $x' = xP^{-1}$ with P^{-1} as inverse of the permutation matrix P.

With an error correcting algorithm for the code C, the codeword u' "next" to x', is computed. As already mentioned, with Patterson's algorithm, there exists an efficient method for computing the error correction, which is described in [12]. To get a plaintext message block, the calculation $u = u'S^{-1}$ is performed[9].

A closer look at the key pair

As described in the key generation, the public key is a single matrix $G' = SGP$, wheras the private key consists of the three matrices S, G and P. It is obvious that from $G' = G$ there is no security achievable. Some kind of "hiding" the original information of G is required. This is achieved by the matrices S and P. Of course, this hiding must be invertable in order to decrypt the message, so both matrices have to be invertable.

S is randomly chosen with the dimension of the amount of rows of G. By performing the matrix multiplication $S * G$, the information of G is "scrambled". It is important to note that the properties of the Goppa code still hold because it stays in the same equivalence class [13].

If G' remains in the same equivalence class this reduces the security of the system. To change the an additional computation has to be performed. For this the permutation P of dimension of the amount of columns of G is needed. P changes the equivalence class of SG so that an attacker can't get any information on the selected G.

Relating to the example in section VII: Leaving P during the key generation works fine. Bob is able to decode the message from Alice, but also an attacker is able to do so. For this P is needed and sequence of decoding is defined in this way.

Correctness

Assuming that P is a permutation matrix and random vector z with length n and weight t, it obvious and can easily be computed that zP^{-1} has a weight of t or less. As discussed, the computation is $x' = xP^{-1} = (mG' + z)P^{-1} = mSG + zP^{-1}$. The chosen Goppa code C is designed to correct up to t errors. On the other hand, mSG has a maximum distance from xP^{-1} of t. This leads to the fact that the correct code mS is determined by the algorithm. To obtain the message block m from mS, we can easily multiply the inverse $m = mSS^{-1}$[14].

[1]The weight of an vector is defined as Hamming weight.

Security properties

The security of the presented scheme refers on the one hand to the basics of the learning with errors principle. More precisely the hypothesis of *Learning Parity with Noise*[15] which is not discussed in this article. On the other hand, it refers to the hypothesis that the generator matrix G has to be indistinguishable from any other $k \times n$-matrix. This leads to the property of a trapdoor function.

IV. The Niederreiter cryptosystem in comparison to McEliece

The Niederreiter cryptosystem is highly comparable to the McEliece cryptosystem due to the fact that it is following the same basic idea. More details about the underlying coding theory can be found in section VI.

Niederreiter designed his $(n, k, 2t + 1)$ linear code C over a Galois field, too. In contrast to McEliece, the code size does not have to be a power of 2 instead of an arbitary integer $GF(q)$. Another difference to McEliece is the usage of a $(n - k) \times (n)$ parity check matrix H instead of the generator matrix G. A parity check matrix is able to determine the error positions in a word, the so called syndrome, and can be computed directly from the generator matrix [2]. Consequently, syndromes are used for the en- and decryption in contrast to error containing messages.

The nonsigular $(n-k)\times(n-k)$ matrix M is defined sightly different from the McElieces $(k \times k)$ matrix S because of the different dimensions of parity check matrix.

The original work of Niederreiter, the permutation matrix P, an arbitrary $n\times n$ matrix, is defined in the same way compared to McEliece. [3]

The private key is then defined as M, H and P, the public key consists of $H' = MHP$ which again is comparable to McEliece. Additionally, the hamming weight t is part of the public key. This has to be the case because encrypted messages are syndromes, whose number of bit with the value 1 has to be lower than the hamming distance in order to decode them. The messages y in the system of Niederreiter have to be n dimensional vectors over $GF(q)$ and they must have a hamming weight of t or less. This is an important fact, issuing the signature creation in section V. Encryption with Niederreiter is performed with $z = yH'^T$ which again is comparable to the encryption operation $x = uG' + z$ in McEliece except that an error vector is unnecessary because of the encoding with H' reveals a syndrome. The ciphertext consist of the syndrome and has only $n - k$ bit dimension compared to n bit in McEliece.
The decryption is computed firstly with $(yP^T)H^T = z(M^T)^{-1}$. Then H is eliminated by a syndrome decoding algorithm [17, P. 332ff] which leads to $(yP)^T$ which can easily be computed to the plaintext y. [8][18][19]

[2]Refer also to VI for more details.

[3]Newer works like [16, P. 69ff] did not consider a permutation. The necessarity of a permutation stays a topic of investigation.

V. Signing with Niederreiter

Besides en- and decryption signature building and verification is an common requirement to an asymmetric cryptoscheme. In state of the art algorithms the principal is quite simple: The message to be signed is *de*crypted with a given public key. The verifier *en*crypts the message with his private key and compares the result with the message.

In the case of McEliece, this is not so easy because there is no possibility to decrypt (= sign) a message before encrypting (= verify) it. More precisely in most cases the process of signing produces a syndrome whose error pattern is bigger than the error correcting property t. In fact, it is hard to create a ciphertext that fits to the error correcting properties of the encryption without using it.

Compared to en- and decryption, signing and verifying is much harder to realize with McEliece. Just in 2001, a digital signature scheme were presented by Courtois et. al.[20]. The problem with signing a given hash value n is that generally, it has a higher hamming weight than the decoding capacity t of the used code is. More generally, one can say that it is difficult to generate a random ciphertext without using the encyption algorithm.

Complete decoding One possible solution would be to use complete decoding. Thereby not only words within the radius of t can be decoded but all words situated in the code space. In other words, with complete decoding we can find an error pattern to any given syndrome as long as it is in the code space. This means that we have to add a δ with random columns from the parity check matrixto t. The decoding works exactly when all of the δ-columns fit to an error position because then the syndrome will fit to an word of weight t. Else, we have to iteratively add another δ to t and try again.

From these properties we can now construct a digital signature scheme: We have to select a δ which is small enough to get an usable key size but on the other hand has a good security.

For achieving a small δ, the code has to be selected carefully in the way that it has to have a high density of decodable syndromes. This makes sure that the δ is kept small because the probability of finding a fitting one is high. For building up a signature, the signer now takes a syndrome and hashes it together with the document. This is tried as long as he gets a decodable syndrome by modifying the document with every try (possibly with some kind of padding) . [20]

VI. Error correctiong Codes

As described in the recent sections, the McEliece cryptosystem is based on the properties of error correcting codes. In this section, a short introduction to the basics of error correcting codes is presented, followed by a closer look at linear block codes such as Goppa codes, which are suggested by McEliece and still are the best choice.

Facing the task of transferring information from sender to a receiver, four types of "encoding" can be done for different purposes:

- **Data compression:** *Reducing* the amount of data[4] by exploiting statistical redundancy in it.
- **Error correction:** *Adding* redundant data for detecting and correcting errors during transmission.
- **Cryptographic encoding:** *Changing* data in the way that only a legitimate receiver is able to read the information.
- **Line coding:** *Representing* binary information as analogue signal in the transmission medium.

Typically, these approaches are combined to reach an efficient, and secure communication.[21, 323ff.]

Principles of error correcting codes

As the name states, the basic purpose of an error correcting code is to detect and correct errors which occur during the transition of messages. For this additional information is added to the message by the sender following an certain encoding algorithm. Knowing this algorithm, the receiver is able to decode the original word, detect errors and correct them with help of the additional information and the corresponding decoding algorithm.

In this section a slight overview from fundamentals of coding theory to the binary goppa codes is presented to get an understanding of underlaying principles.

Overview on common block codes

In this section block codes are discussed. Systematic codes, as trivial approach, are discussed slightly followed by linear codes which are used in the McEliece cryptosystem. The hamming code serves as an illustrating example for a linear block code.

As an alternative to block codes, convolutional codes should be mentioned which are not described in this article.

Parity checks and cyclic redundancy checks

The easiest way of adding redundant information is to provide a parity check bit. For this, an one is added if the information word has an odd parity and a zero otherwise. With this method, it is possible to detect an error of one bit, but it is not possible to say where the error has happened.

Cyclic redundancy checks follow the same approach but is using a generator polynomial instead of a single bit. Therefore, a message is simply divided through a generator polynomial, which in binary can be seen as a bitwise XOR-operation. The rest of the division is added to the message. By dividing the sent message incl. the check value through the generator, the amount of errors can be revealed. Depending on the properties of the generator polynomial, the error positions can be determined and corrected.[22]

Linear codes

Linear codes are defined over a finite vector space F_q^n over a finite field $\mathbb{F}_q$. All codewords are elements of F_q^n, so the whole code is defining a subspace C.

[4]"Data" refers to the binary word which represent the information to be transferred.

Because of this property a base $g_1, ..., g_k$ exists which forms a generator matrix G for C. Additionally there is a parity check matrix H with the property that for every codeword c $Hc^T = 0$ holds. H can easily be computed from G with help of linear algebra by tranforming G into $G = (\mathbb{E}_k|P)$, where $\mathbb{E}_k$ denotes the k-dimensional indentity matrix. Finally H reveals as $H = (P^T|\mathbb{E}_{n-k})$. A linear code is defined by either G or H.

Linear codes are based on operations of the linear algebra and therefore are computational easy to en- and decode. As described in section III linear codes are defined as $[n, k, d]$-code where k defines the dimension of the linear subspace and following from this the rank of G, n the length of a message and revealing from this the rank of k is defined as $n - k$. d denotes the resulting Hamming distance which also is the minimal amount of linear independent columns in k.[22]

Encoding The encoding of a k-bit *message* x to an n-bit *code* is simply done by the multiplication $c = x * G$.

Decoding For getting back the original message a decoding algorithm is needed. It should be noted, that the received message c' could contain errors which can be corrected up to $\lfloor \frac{d-1}{2} \rfloor$ errors. This is in general not performed by inverting the encoding, but determining the codeword which is next "neighbour" c to received c'. This can be done with the help of the Hamming distance by computing the minimum distance of all codevectors $c = argmin\{hammingweight_{y \in C}(c' - y)\}$. The result can contain only one vector if a perfect code is used.

Besides the maximum likelihood decoding sketched above, syndrome decoding is a more efficient method for finding the related message word. A syndrome s_x of any vector x can be computed by $s_x^T = H * x^T$. The error vector results from $e = x - c$ and contains 1-entries at all position where an error occurred. Because of the property of linearity also $s_x^T = Hc^T + He^T$ holds true. This leads to the fact that all messages with the same error vector e are in the same subfield and have the same syndrome s_e^T.

To decode x to c', an error vector e' has to be determined which syndrome has minimal hamming weight and is identical to the syndrome of x. Therefore the coset of the subfield C has to be formed by subtracting one certain vector from one arbitrary codevector. All resulting vectors then form the coset, which can be seen as list, and the vector with minimal hamming weight is called the coset leader.

With this error vector $c' = x - e'$ is calculated.

Having c' and the matrix which connects an error vector to a syndrome, it is easy to read the syndrome from the error vector, swap 1-positions of the code word c' and finally compute the message $x = c' * G$.[23]

Properties of codes can be used for McEliece and Niederreiter

Goppa codes are the class of error correcting codes which are suggested for the McEliece cryptosystem. They are following the principles of linear block codes, which is described previously. This section highlights some special properties and backgrounds of codes which are used with the McEliece cryptosystem. Firstly some important basic properties and terminologies are presented, which have been use implicitly in the previous sections.

Binary Due to the fact that almost all transition is performed by computers and is in binary the explanations in following are focussed on the binary case too. Never the less all of them hold in the general case.

For generating these additional information a vector with binary indexes $V = [c1c2...cn]|ci \in 0, 1$ is created as set of $\mathbb{F}_2$. Further the addition is defined componend-wise modulo 2, which also can be seen as a XOR-operation. With this definition V covers all n-tuples in $\mathbb{F}_2$. Having this, a binary, linear $[n, k]$ code can be defined as subspace of $\mathbb{F}_2$ in which the n is the length and the k is the dimension. A codeword is represented as vector in the code.[9]

Hamming distance Another important variable for error correcting codes is the hamming weight or more precise the minimum weight d of a vector. The weight is simply defined as count of non-zero positions in the vector. Furthermore the minimum weight is defined as minimum is the smallest weight of a codeword under the condition that it is not zero. A $[n, k, d]$ code can correct $t = \lfloor \frac{d-1}{2} \rfloor$ errors.

Linearity Goppa Codes are linear codes. This is not a coincidence because non-linear codes do not provide an efficient decoding algorithm like Pattersons algorithm for linear codes does[12]. In [24] some remarkable results to achieve more efficiency are stated but also it is mentioned, that nonlinear codes are not able to achieve more comparable efficiency. However as most of the popular codes are linear ones this condition does not rule out so many codes.

Irreducibility To achieve a finite field with usable properties for a Goppa code the generator polynomial has to bew irreducible. This means that for a polynomial p over a finite field $GF(p^m)$ there exists no polynomial $p' \in GF(p^m)$ of lower degree which divides p.

Efficency To keep the keyspace as efficient as possible the factor between the matrix dimensions n, k and the error correction range t (respectively the minimum hamming distance d) should be as high as possible. The most important point is that Goppa Codes are a suspect of research for years so the probability of success is really low.

Insights of Goppa codes McEliece selected the code for his cryptosystem carefully because many other linear block codes the McEliece cryptosystem was tried with were broken in the past (e.g. [25]). Goppa codes have some properties which are usable for cryptographic applications. For example the lower bounds of the minimum distance is easy to compute, which helps constructing a secure code. Also only the entity knowing the generator polynomial has th possibility to perform efficient error correction, which makes attacking the scheme harder. The third point is that there is no efficient algorithm known, which performs the error correction whithout knowing the gernerator polynomial[7].

VII. Example of the McEliece cryptosystem

In section III, the theoretical aspects of the McEliece cryptsystem are described. To illustrate the algorithm more in detail an example is provided in this section. Note that the parameters are far too small and the code is based on much simpler Hamming codes.
Consider Alice and Bob as two entities who want to communicate encryptedly. For this, Bob chooses the following matrix G as generator:

$$G = \begin{Bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{Bmatrix}$$

Additionally, a matrix S to obscure the generator is chosen by Bob:

$$S = \begin{Bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{Bmatrix}$$

A permutation matrix P is also needed:

$$P = \begin{Bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

Now Bob calculates the public key G':

$$G' = SGP = \begin{Bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{Bmatrix}$$

G' is sent to Alice who wants to encrypt the message

$$x = \begin{Bmatrix} 1 & 1 & 0 & 1 \end{Bmatrix}$$

For this, she chooses a random error vector e:

$$e = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{Bmatrix}$$

This matrix is sent to Alice who calculates $y = x * G' + e$:

$$y = x * G' + e = \begin{Bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{Bmatrix}$$

This codeword is sent to Bob who decodes the word with P^{-1} which leads to:

$$y' = y * G^{-1} = \begin{Bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{Bmatrix}$$

Now the fast decoding algorithm enters the scene. Note that in this example the deconding is done by heart with the approach of maximum likelihood decoding. In section VI more details on decoding algorithms can be found.

In this case, Bob compares the result to the rows of G. The row with the lowest Hamming distance is the corresponding codeword which leads to:

$$y = x * G' + e = \begin{Bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & 1 & 0 \end{Bmatrix}$$

Because of the structure of G, which has an identity matrix on the left hand side, the first four digits are taken as error vector. The last step to decrypt the message is to multiply the error vector xS with S^{-1}:

$$x = y * S^{-1} = \begin{Bmatrix} 1 & 1 & 0 & 1 \end{Bmatrix}$$

So Bob is able to decrypt the message from Alice properly.

A. Different message

If Alice wants to encrypt the message $x = \begin{Bmatrix} 1 & \mathbf{0} & 0 & 1 \end{Bmatrix}$ instead of $\begin{Bmatrix} 1 & 1 & 0 & 1 \end{Bmatrix}$, the decrypted word which is sent to Bob is $y = \begin{Bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{Bmatrix}$. He performs $y' = y * G^{-1} = \begin{Bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{Bmatrix}$ which decodes as described to $\begin{Bmatrix} 0 & 1 & 0 & 0 \end{Bmatrix}$ and finally with $x = y * S^{-1}$ to $\begin{Bmatrix} 1 & 0 & 0 & 1 \end{Bmatrix}$. From these two examples we can see that flipping a single bit can lead to a code of another "domain" ($\begin{Bmatrix} 0 & 1 & 0 & 0 \end{Bmatrix}$ in this case) during encryption which is correctly decrypted because of the usage of similar codes. The behavior can be seen when the 1-digit of the error vector is changed. Note that an error vector of Hamming weight > 1 is not working properly because of the error correcting capabilities of the code.

VIII. Further topics

To get an elaborated overview over all facets of the McEliece cryptosystem, many more topics have to be issued which are not covered in detail in the article. In this section, some of the most important topics are breifly described.

Resistency against various attacker models

For a cryptoanalyst attacker, models are the first choice to design attacks on a given scheme. Two of the most common attacker model and resistance approaches are briefly presented in this section.

Chosen plain text resistance Chosen plaintext resistance means that an attacker owns two different plaintexts p_1 and p_2 and gets two belonging ciphertexts c_1 and c_2. He has no chance to gather information from the ciphertext which tells him that it belongs to plaintext p_1 or p_2. It is obvious that the McEliece cryptosystem in its basic idea is not resistant against those attacks. Obtaining a ciphertext c and knowing that it is either of the plaintext m_1 or m_2, an attacker can easily compute the weight of m_1 by $m_1 * G' \oplus c$ and check if it is the same compared to weight w transmitted within the public key [5].

An easy approach to avoid this and to gain IND-CPA is to simply add random sequence r the message e.g. in the way $[r|m]$. [26]

[5]Note that the authors assumed, that the McEliece cryptosystem is transferrring a Hamming weight as public key, too.

Chosen cipher text resistance Like the RSA cryptosystem in the original idea is not resistant against chosen cipher attacks.

This means if an attacker has access to an oracle which is able to decrypt a given ciphertext without knowing the key and the attacker is permitted to give all ciphertexts except the one he is asked to crack to the oracle, he is not able to gather any reasonable information from the oracle. This goes over several iterations and is called CCA2-attack in the literature. Resistance against this attack is described as IND-CCA2.

Like the RSA cryptosystem uses padding standards such as PSS for signing and OAEP for encrypting to achieve IND-CCA2, the McEliece cryptosystem has some advanced padding and randomization schemes as well. The common approach on this is to add random vectors and perform shift operations to the ciphertext during the encryption[27].

Practical security attacks

Every cryptosystem with certain parameters has to hold against attacks in order to be considered as safe. These attacks can be very different for every cipher suite. For the McEliece cryptosystem, there are two types of promising attacks, named structural and information set decoding attacks.

Structural attacks describes the approach of constructing a code for the information of the code generated by the public key G'. If this succeeds, the private key G is revealed. To fulfil such an attack, an equivalent code E to G' has to be found, which means that a permutation m exists which permutes G' to E'. Then E' generated by G' and the E generated by G are from the same class. Knowing this dependence, an attacker can compare a representant from each of such classes to E' to get an equivalent code. This is still a huge computational problem which is not solvable in sufficient time because the cardinality of such an equivalent matrix is small. [28]

Information set decoding is the most promising method of breaking McEliece so far. The idea behind this is to find a set of error free coordinates in a codeword. If the corresponding columns in the generator matrix form an invertible submatrix, the information from original message word can be obtained easily.

Assume there is a (n, k, t) code of a generator G and a ciphertext $c = uG+e \in \mathbb{F}_2^n$. An attacker now randomly selects a subset (the information set) $I \subset 1, ..., n$ of size k being linearly independent. Now a "masking" function δ is needed which projects c on I such as $\delta(c)_{i\in I} = (uG)_{i\in I} + e_{i\in I}$.

If this projection leads to an error free result (also $weight(e) = 0$), we can obtain the message word from the code by $\delta(r) * \delta(G)^{-1} = \delta(uG)^{-1} * \delta(G)^{-1} = u$. But this is not the common case and typically needs many iterations.

Note that information set decoding is not a total break of the system because it does not reveal the private key but only the message word corresponding to the codeword.

Recommended parameters

In [29], Niebuhr et al computed some key length based on the Model of Lenstra and Verheul. They proofed that the keylength has to grow with speedup of computer hardware and concluded state of the art keylenths of traditional algorithms which is ported to the McEliece Cryptosystem by Niebuhr et al. Based on the best known attack of Sendrier at al, which is able to break the original parameters ($n = 1024$, $k = 524$, $t = 50$) with a minimal binary work factor of $2^59, 9$ operations [30], they presented the following values as secure parameters until the year 2050: $n = 2804$, $k = 2048$ and $t = 66$. A public key has a size of about 189 kilobyte. This is almost 400 times more keyspace than a RSA key and almost 3000 times more than an eliptic curves key which promises safety until 2050, too, but only on conventional computers. The key size issue is decribed a little more into detail in section VIII.

Challenges and optimizations

From the practical point of view the, presented schemes are more sketches of one way trapdoor functions than fully filled crypto schemes. These trapdoors can be used in many different ways to generate a "standard" crypto scheme maybe as replacement for RSA- and ECC-cryptography, but this has to be done carefully. In this section, a short summary of the challenges, such as usability properties, computation time and key sizes is presented. These three topics are highly connected to each other.[2]

Confidence One of the most delicate tasks when putting an abstract algorithm into a real cryptographic application is to achieve confidence into it. Many pitfalls like wrong codes or parameters and weak padding algorithms have to be discovered by cryptoanalysts to increase the confidence level. Although both the RSA and the McElice where suggested over 35 years ago, the development on RSA was pushed forward much faster because of the adavantage of a lower keyspace. This development has to be done in the next years for McEliece to get over issues discoverd in the past and achieve an optimized and reliable system.

Efficiency In most cases the abstract term "efficiency" consists of time- and space efficiency. Daniel J. Bernstein made [2] a short comparison with RSA which will be summed up here briefly.

Time effciency is seen as the time needed to perform the key generation, encryption and decryption. More precisely, the different "operations" executed and their complexity are reviewed.

Compared to RSA, McEliece is quite time efficient. As described in section III, the main computations are matrix operations in the binary field, which can be broken down to highly efficient additions, multiplications, shifts etc. for which no special hardware requirements are needed[2].

Space efficiency refers to the storage needed to be provided in order to execute the algorithm. In our case, the bytes needed to transmit the public key to the encryption entity or signer is an important topic. [2]

The most important drawback is the key sizes to be transferred. With n denoted as code length, respectively key length, McEliece needs about $b^2 * (lgb)^2$ bits whereas the RSA only needs $(\approx 0, 16) * b^3/(lgb)^2$ bits. One might notice the higher exponent in the RSA formula which leads to an faster growing key space if the security level goes to ∞ but this point where the RSA key size is higher than the one of McEliece is far beyond practical relevant key sizes. For keys assumed as secure for today's proposal the McEliece key is size is about 10 times larger than RSA[6]. Refer also to section VIII for a closer look at keylenght with different parameters.[2]

Usability The last important point is to provide an usable interface for the cryptosystem. New crypto protocols can only be successful if many applications in different domains are supporting them. Therefore, a padding schemes, as addressed in section VIII, has to be provided and standardised to make sure that different software systems on different platforms can speak to each other. [2]

Wild McEliece

Due to the fact that the McEliece cryptosystem was published over 30 years ago and still is one of the most promising post quantum security algorithms, many variants came up. Many of the previously described issues could be solved by reducing the key space. In conclusion, this means optimising on the used codes. Many codes like Reed-Solomon codes, quasi dyadic codes, and variants of Goppa codes have been tried, seen as secure and been broken a little later.

The most promising optimization is the idea of using "wild" Goppa codes[31]. Wild Goppa codes are codes which are no longer over a field F_2 but on F_q with q as a small prime number. This approach was broken by generating the square code of the public key and revealing information about the original private key, because the square product lines are not equally distributed compared to the binary code. In [32], the wild McEliece idea is improved with an "incognito" variant. As the name says, the wild Goppa codes are hidden by multiplying an extra factor f to the code and by using only special codes described in [33]. This topic and the question if squaring technique is still considerable for the incognito version, is still a topic of research.

IX. Conclusion

The aim of this work was to present a basic introduction on postquantum cryptography focussing on the McElice cryptosystem. With basic information of the underlying Goppa codes a closer look at the algorithm was taken supported by an example computation. Moreover, special properties and disadvantages were highlighted.

In conclusion, the McEliece cryptosystem remains one of the most promising candidates for a post quantum cryptography and is still an huge topic of research as the literature references show. Nevertheless, there are grave obstacles, especially the large keysize and in relation to the practical usability.

[6]Assuming unoptimized but optimal algorithms without the attention of quantum computers.

References

[1] I. L. Markov and M. Saeedi, "Constant-optimized quantum circuits for modular multiplication and exponentiation," *arXiv preprint arXiv:1202.6614*, 2012.

[2] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-quantum cryptography*, pp. 1–14, Springer, 2009.

[3] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-quantum cryptography*, pp. 147–191, Springer, 2009.

[4] Wikipedia, "Multivariate cryptography — wikipedia, the free encyclopedia," 2017. [Online; accessed 10-March-2017].

[5] L. Lamport, "Constructing digital signatures from a one-way function," tech. rep., Technical Report CSL-98, SRI International Palo Alto, 1979.

[6] R. C. Merkle, R. Charles, *et al.*, "Secrecy, authentication, and public key systems," 1979.

[7] D. Engelbert, R. Overbeck, and A. Schmidt, "A summary of mceliece-type cryptosystems and their security.," *J. Mathematical Cryptology*, vol. 1, no. 2, pp. 151–199, 2007.

[8] N. Sendrier, "Niederreiter encryption scheme," in *Encyclopedia of cryptography and security*, pp. 842–843, Springer, 2011.

[9] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[10] R. McEliece, *The theory of information and coding*. Cambridge University Press, 2002.

[11] E. R. Berlekamp, "Algebraic coding theory," 1968.

[12] N. Patterson, "The algebraic decoding of goppa codes," *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 203–207, 1975.

[13] J. A. Ryan and K. Magamba, "Equivalent irreducible goppa codes,"

[14] Wikipedia, "Mceliece cryptosystem — wikipedia, the free encyclopedia," 2017. [Online; accessed 12-March-2017].

[15] K. Pietrzak, "Cryptography from learning parity with noise," in *International Conference on Current Trends in Theory and Practice of Computer Science*, pp. 99–114, Springer, 2012.

[16] M. Baldi, *QC-LDPC code-based cryptography*. Springer Science & Business, 2014.

[17] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Elsevier, 1977.

[18] Y. X. Li, R. H. Deng, and X. M. Wang, "On the equivalence of mceliece's and niederreiter's public-key cryptosystems," *IEEE Transactions on Information Theory*, vol. 40, no. 1, pp. 271–273, 1994.

[19] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *PROBLEMS OF CONTROL AND INFORMATION THEORY-PROBLEMY UPRAVLENIYA I TEORII INFORMATSII*, vol. 15, no. 2, pp. 159–166, 1986.

[20] N. T. Courtois, M. Finiasz, and N. Sendrier, "How to achieve a mceliece-based digital signature scheme," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 157–174, Springer, 2001.

[21] D. W. Hoffmann, *Einführung in die Informations-und Codierungstheorie*. Springer, 2014.

[22] D. Schönfeld, H. Klimant, and R. Piotraschke, *Informations-und Kodierungstheorie*. Springer-Verlag, 2012.

[23] M. Greferath, "An introduction to ring-linear coding theory," in *Gröbner Bases, Coding, and Cryptography*, pp. 219–238, Springer, 2009.

[24] F. Zeng, *Nonlinear codes: representation, constructions, minimum distance computation and decoding*. PhD thesis, Universitat Autònoma de Barcelona, 2014.

[25] V. M. Sidelnikov and S. O. Shestakov, "On insecurity of cryptosystems based on generalized reed-solomon codes," *Discrete Mathematics and Applications*, vol. 2, no. 4, pp. 439–444, 1992.

[26] R. Nojima, H. Imai, K. Kobara, and K. Morozov, "Semantic security for the mceliece cryptosystem without random oracles," *Designs, Codes and Cryptography*, vol. 49, no. 1, pp. 289–305, 2008.

[27] N. Dottling, R. Dowsley, J. Muller-Quade, and A. C. Nascimento, "A cca2 secure variant of the mceliece cryptosystem," *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6672–6680, 2012.

[28] S. Au, C. Eubanks-Turner, and J. Everson, "The mceliece cryptosystem," *Unpublished manuscript*, vol. 5, 2003.

[29] R. Niebuhr, M. Meziani, S. Bulygin, and J. Buchmann, "Selecting parameters for secure mceliece-based cryptosystems," *International Journal of Information Security*, vol. 11, no. 3, pp. 137–147, 2012.

[30] M. Finiasz and N. Sendrier, “Security bounds for the design of code-based cryptosystems,” in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 88–105, Springer, 2009.

[31] D. J. Bernstein, T. Lange, and C. Peters, “Wild mceliece,” in *International Workshop on Selected Areas in Cryptography*, pp. 143–158, Springer, 2010.

[32] B.-Y. Yang, *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29-December 2, 2011, Proceedings*, vol. 7071. Springer, 2011.

[33] T. P. Berger and P. Loidreau, “How to mask the structure of codes for a cryptographic use,” *Designs, Codes and Cryptography*, vol. 35, no. 1, pp. 63–79, 2005.