McEliece Cryptosystem, Variants, Attacks

Alison Lin

May 11, 2011

1 Overview

The security of most popular public key cryptosystems such as RSA, ElGamal, ECC(elliptic curve cryptography), and Diffie-Hellman key exchange is based on the difficulties of discrete log problem or integer factorization. These problem can be efficiently solved by a quantum computer using Shor's algorithm. The quantum computer has not been built so far and therefore those are still popular cryptosystems today. On the other hand, there are also other public key cryptosystems such as McEliece and NTRU whose security does not depend on these problems. They are both candidates for "post-quantum cryptography", i.e., they are immune to the known quantum computer attacks.

McEliece is one of the oldest public key cryptosystems. Its main advantage is its high speed encryption and decryption functions compared to other public key cryptosystem. Compared to RSA, the encryption and decryption of McEliece are faster and the security grows much faster as the key size grows. The structure of the secret key is hidden by a matrix product. The structural attacks look for any structure that might remain visible after the matrix product but have not produced effective attacks so far. So most of the attacks that have been published so far are non-structural attacks. In a non-structural attack, an attacker's objective is to recover the message without using the key's structure. So it is usually said that the security of McEliece is based on the difficulty of decoding a randomly looking code without any obvious structure. An attacker is faced with the classical decoding problem: Find the closest codeword in a linear code of a given vector, assuming that there is a unique closest codeword. The classical decoding problem is proven to be very hard. The best method of this problem currently is information-set decoding algorithm. The algorithm was first proposed by Prange [2] and then a variety of variants have been developed over the last few decades. The best of them is considered to be Bernstein's algorithm [15]. A new parameter set for the McEliece cryptosystem was suggested in order to provide adequate security. We will introduce McEliece cryptosystem in Section 4 and some of the information-set decoding algorithms in Section 5.

On the other hand, McEliece cryptosystem is not currently used because of its relatively large public key size (as the following tables show) and low data rate (the length of a ciphertext is about twice of the length of its plaintext). There are some variants of the McEliece system that try to reduce the key size but all have been broken. In Section 6, one of the most recent variants proposed by Berger with smaller key size will be presented as well as the two structural attacks on them. In Section 7, we propose a possible improvement of one of the attacks against the variant.

Table: currently suggested key sizes for the various public key cryptosystems. Note that the security is a bits means the best attack algorithm at present takes 2^a bit operation to break the cryptosystem. So the number might become slower in the future if a better attack algorithm is obtained.

McEliece cryptosystem			
Security (bit)	60	80	128
Size (bit)	262000	520047	1537536
Encryption/Decryption speed	O(n^2)		

RSA cryptosystem				
Security (bit)	56	80	112	128
Size (bit)	512	1024	2048	3072
Encryption/Decryption speed	O(n^3)			

ECC cryptosystem				
Security (bit)	56	80	112	128
Size (bit)	112	161	224	256
Encryption/Decryption speed	$\leq$ O(n^3); faster than RSA because of its smaller key size			

2 Quantum Computer

The idea of quantum computing is identifying 0/1 bits and logic gates with particle behaviors. For example, represent 0/1 by the spin direction of an electron or the path of a photon. The computation result is obtained by measuring the result of related physics operations. This should be implemented by physics researches. For example, $\mathbb{F}_2$ addition is in fact equivalent to XOR gate. Currently, XOR gate can be implemented in several ways [14]. Comparing to the rapid development of quantum computing theories, the experiments and implements are still at baby steps. How to control those particles well is still a big problem.

Shor's algorithm is an algorithm running on a quantum computer that can find the factorization of an integer. The idea is to convert the problem into finding the period of an exponential function $f(x) = a^x \mod N$ for some integer a. ($a^{2t} = 1 \mod N$ implies N divides $a^t + 1$ or $a^t - 1$ and hence $\gcd(a^t \pm 1, N)$ is a possible factor of N). Then use quantum computing to find the period of $f(x)$. To do this, consider the quantum state $\sum_{x=0}^{Q} \frac{1}{\sqrt{Q}}|x\rangle$ for some integer $Q = 2^q$ large enough. By quantum Fourier transform, $\sum_{x=0}^{Q} \frac{1}{\sqrt{Q}}|x\rangle = \frac{1}{Q}\sum_{y=0}^{Q-1}\sum_{x=0}^{Q-1} w^{xy}|y\rangle|f(x)\rangle$. If $f(x)$ has period r, then it causes some repeated terms in the above state. Compute the probability of the appearance of those terms in terms of r. Then keep measuring the output of the quantum computing to get those probabilities and recover r.

Remark 2.1. Currently post-quantum cryptography is mostly focused on four different approaches:
Lattice-based cryptography such as NTRU and GGH
Multivariate cryptography such as Unbalanced Oil and Vinegar
Hash-based signatures such as Lamport signatures and Merkle signature scheme
Code-based cryptography that relies on error-correcting codes, such as McEliece encryption and Niederreiter signatures

3 Brief Review of Linear Codes

Since the McEliece cryptosystem is a code-based cryptosystem, we need to review some concepts of linear codes before introducing it in the next section.

Let $\mathbb{F}_q$ be a finite field with q elements. An $[n,k]$-linear code C is a linear subspace of $\mathbb{F}_q^n$ with dimension k. The elements in C are called codewords. Let G be a $k \times n$ matrix whose rows form a basis of C. Thus, for a message $m = (m_1, ..., m_k) \in \mathbb{F}_q^k$, $m \mapsto mG$ defines a map from the message space $\mathbb{F}_q^k$ to $C \subseteq \mathbb{F}_q^n$. We call this process encoding and call G the generating matrix of the code C. If $\mathbb{F}_q = \mathbb{F}_2$, C is called a binary code.

When A wants to convey a message m to B, A transmits its codeword mG(n bits) instead of m(k bits) to B. This is because messages are usually corrupted on the way to their destination. If A sends the message m to B and than B receives a corrupted m with some error bits (position of those errors are unknown), than B is not able to recover m. Instead, if A sends B its codeword mG which is longer than m, and mG is corrupted on its way, then B has chance to recover m since mG carries more information than m. k/n is called the transmission rate, i.e., each k-bits message is carried by an n-bits codeword. A good code has a high transmission rate and an efficient decoding algorithm to recover m from $mG+e$ (corrupted codeword where e represents the error). Such a code and its decoding algorithm should be generated together by using some nice math structures. However, if G is just a randomly chosen matrix, then the code generating by G has no well-organized structure and a corresponding efficient decoding algorithm. In such cases, it is very hard to recover the original message or correct the error bits. This problem has been proved to be NP-complete by Berlekamp et al. [1].

For $x \in C$, the Hamming weight of x, denoted by $\mathtt{wt}(x)$, is the number of nonzero components of x. For $x, y \in C$, the Hamming distance of x and y, denoted by $d(x,y)$, is the number of coordinates where x and y are differ. Let $\min C$ denotes the minimum distance of any two distinct vectors in C.

Let G be a generating matrix of an $[n,k]$-code C. A parity-check matrix H of G is an $(n-k) \times n$ matrix whose rows form a basis of $C^{\perp}$. So $GH^T = 0$. If G has the form $G = [I_k|P]$, then $H = [-P^T|I_{n-k}]$ is a parity-check matrix.

4 Introduction of McEliece Cryptosystem and Its Dual - Niederreiter Cryptosystem

The encryption of McEliece is like encoding a message by a random matrix without obvious structure. The ciphertext looks like a codeword of a code without obvious structure that is corrupted by an error vector. For a receiver with the secret key, the ciphertext can be converted into a codeword of a code with nice structure that is corrupted by an error vector and hence can be decoded to recover the message. So the security relies on the difficulty of decoding a general linear code without known structure. Berlekamp, McEliece, and Tilborg [1] have shown that this problem for linear binary codes is NP-complete. The best known way to decode a random-looking code is information-set decoding introduced by Prange [2]. Before introducing information-set decoding, let's look at McEliece cryptosystem first.

McEliece Cryptosystem The secret key of the cryptosystem is a generating matrix of a binary linear code and the public key is a scrambled version of this secret matrix.

Key generation and materials

C: an $[n,k]$-linear code C which can correct t errors with an efficient decoding algorithm.
G: $k \times n$ matrix which is a generating matrix of C.
S: invertible $k \times k$ matrix
P: $n \times n$ permutation matrix
Public key: the matrix SGP
Private key: (S,G,P) and an efficient decoding algorithm.

Encryption

message m of length $k \longrightarrow$ ciphertext $y = mSGP + e$ of length n,
where e is a randomly chosen error vector of weight at most t.

Decryption

$$y \longrightarrow yP^{-1} = mSG + eP^{-1} \text{ (a codeword of } C \text{ with an error } eP^{-1}) \overset{\text{decoding}}{\longrightarrow} mS \longrightarrow mSS^{-1} = m$$

The complexity of encryption is O(n^2); Key size is kn or $k(n-k)$; Transmission rate is k/n. The suggested parameters are $n = 1024, k = 524, t = 50$.

On the other hand, since the generating matrix G and its parity-check matrix H determine each other, McEliece cryptosystem can be translated into its "parity-check matrix version" called Niederreiter cryptosystem. The two algorithms are equivalent.

Niederreiter Cryptosystem

Key generation and materials

C: an $[n, k]$-linear code C which can correct t errors with an efficient decoding algorithm.
H: $(n-k) \times n$ parity-check matrix of a generating matrix G.
S: invertible $(n-k) \times (n-k)$ matrix
P: $n \times n$ permutation matrix
Public key: the matrix SHP
Private key: (S, H, P) and an efficient decoding algorithm.

Encryption

message m of length $n \longrightarrow$ ciphertext $y = (SHP)m^T$,
where e is a randomly chosen error vector of weight at most t.

Decryption

$$y \longrightarrow S^{-1}y = HPm^T \overset{\text{decoding}}{\longrightarrow} Pm^T \longrightarrow P^{-1}Pm^T = m^T$$

5 Information-set Decoding and Attacks

In an attacker's view, the ciphertext $m(SGP) + e$ looks like a codeword of a code that is corrupted by an addition of a low-weight error vector (e). So to recover m is equivalent to decode $m(SGP) + e$ with respect to the code generated by (SGP). However, the scrambled matrix SGP conceals the structure of G. So the problem of an attacker is how to decode a random-looking code. The best known algorithm is information-set decoding. So in fact, this kind of attack is a message attack instead of a key recovery attack. The objective is to recover the message by trying to decode a corrupted codeword. In this section, we will see the two primary information-set decoding algorithms - Brickell's algorithm [3] and Stern's algorithm [4]. In fact, they are both algorithms for finding low-weight codeword of a given code because the error vector e can be viewed as a low-weight codeword in a slightly larger code. The following paragraphs explain the details of this relation.

The relation between recovering e and finding low-weight codeword

First of all, now we have a corrupted codeword (ciphertext) $y \in \mathbb{F}_2^n$ and we know $y = c + e$ for some codeword c and an error e with $\mathtt{wt}(e) = w \le t$. So $e = y - c \in C + \{0, y\}$. Moreover, for any v with $\mathtt{wt}(v) \le t$, $t \notin C$ because C can correct t errors which means $\min C \ge 2t + 1$.

Moreover, e is the only weight-w codeword in $C + \{0, y\} \setminus C$. If $c_1 - y \ne c_2 - y \in C + \{0, y\} \setminus C$, where $\mathtt{wt}(c_1 - y) = \mathtt{wt}(c_2 - y) = w \le t$, then $\mathtt{wt}(c_1 - c_2) \le \mathtt{wt}(c_1 - y) + \mathtt{wt}(c_2 - y) \le t + t = 2t < 2t + 1 = \min C$ which is impossible because $c_1 - c_2 \in C$.

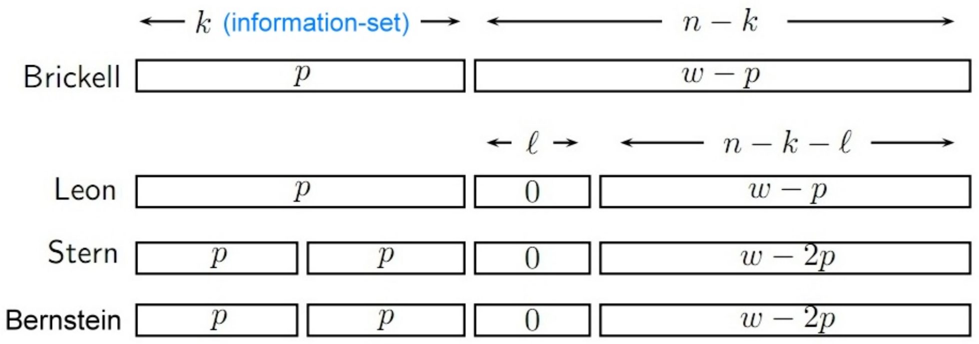
So the idea is to recover e by finding this small weight codeword in $C + \{0, y\} \setminus C$.

Information set

Given generating matrix G of an $[n,k]$-code. An information set is a size-k subset $I \subseteq \{1,2,...,n\}$ such that the I-indexed columns of G are invertible. Denote the matrix formed by the I-indexed columns of G by G_I . The I-indexed columns of $G_I^{-1}G$ form the identity matrix I_k. For any binary vector x denote by x_I the restriction of x to the coordinates indexed by I.

All of the following three algorithms are built upon a shared concept. That is, randomly select a fixed number of rows from $G_I^{-1}G$, thus the sum of these rows must automatically have the fixed weight among the coordinates indexed by I since the I-indexed columns of $G_I^{-1}G$ have the form I_k. Thus, just need to consider the weight at among the rest coordinates. Brickell's, Stern's, and Bernstein's algorithms are developed based on the same concept; however, each algorithm is increasingly elaborated and includes additional parameters (Brickell's < Stern's < Bernstein's). The following graph shows how these algorithms arrange the weight distribution of a target vector that it looks for. The performance of these algorithms is printed at the end of this section.

Figure: weight distribution of different information-set decoding algorithms



5.1 Brickell's algorithm

Input: code C, weight w, integer parameter $0 \leq \mathbf{p} \leq w$.
Output: $x \in C$ with $\texttt{wt}(x) = w$ where $\texttt{wt}(x_I) = p$ and $\texttt{wt}(x_{[n]\setminus I}) = w - p$.

Step 1 Choose an information set I from $\{1,...,n\}$.

Step 2 Replace G by $G_I^{-1}G$.

Step 3 For each size-p subset $A \subseteq I$ and compute $x := \sum_{i\in A} G_i$, where G_i is the ith row of G.
If $\texttt{wt}(x) = w$, output x. Else go to Step 1.

Step 2 makes the I indexed columns in G form the $k \times k$ identity matrix. Since $|A| = p$, $x := \sum_{i\in A} G_i$ is a sum of p rows and hence x_I automatically has weight p. So Step 3 is actually looking for a vector x having weight $w-p$ at the other coordinates. Moreover, the parameter p is optimal when $p = 2$. (leads to the fewest iterations) [16].

5.2 Stern's algorithm

Stern's algorithm divides the information set I into 2 subsets X and Y. Then it looks for the codewords having weight p among the columns indexed by X, weight p among the columns indexed by Y , and weight 0 on a random set of l positions outside the I-indexed columns.

Input: code $\mathbf{C}$, weight $\mathbf{w}$, $(n-k)\times n$ parity check matrix $\mathbf{H}$ for C, parameter $\mathbf{l}$, parameter $\mathbf{p}$.
Output: $x \in C$ with $\mathtt{wt}(x) = w$ where
$\mathtt{wt}(x_I) = 2p$ and $\mathtt{wt}(x_{[n]\setminus I}) = w - 2p$ and $\mathtt{wt}(x_Z) = 0$ for some l-size subset $Z \subseteq [n] \setminus I$.

Step 1 Randomly select linearly independent $n-k$ columns from the n columns of H.

Step 2 Randomly select a size-l subset Z from the above $n-k$ columns.

Step 3 Divide remaining k columns into two disjoint subsets X and Y.

Now, the following steps start to find those codewords with exactly p nonzero bits in X, exactly p nonzero bits in Y, all zeros in Z, exactly $w-2p$ nonzero bits in the remaining columns. Obviously, such codewords have weight w.

Step 4 Do row operations on H such that the $n-k$ columns chosen in Step 1 become the identity matrix. From the above identity matrix, the set Z with l columns correspond to l rows of H - the rows where the columns have "one". Let $\{r_1, ..., r_l\}$ denote these l rows.

Step 5 For every size-p subset A of X, compute

$$\pi(A) := \begin{pmatrix} [\pi(A)]_1 \\ [\pi(A)]_2 \\ \vdots \\ [\pi(A)]_l \end{pmatrix}_{l\times 1} = \begin{pmatrix} \sum_{j\in A} H_{r_1,j} \\ \sum_{j\in A} H_{r_2,j} \\ \vdots \\ \sum_{j\in A} H_{r_l,j} \end{pmatrix} \pmod 2.$$

Similarly, compute $\pi(B)$ for every size-p subset B of Y.

Step 6 Whenever $\pi(A) = \pi(B)$, compute the sum of the $2p$ columns in $A \cup B$. This sum is an $(n-k)$-bit column vector.

Step 7

If the sum has weight $w - 2p$

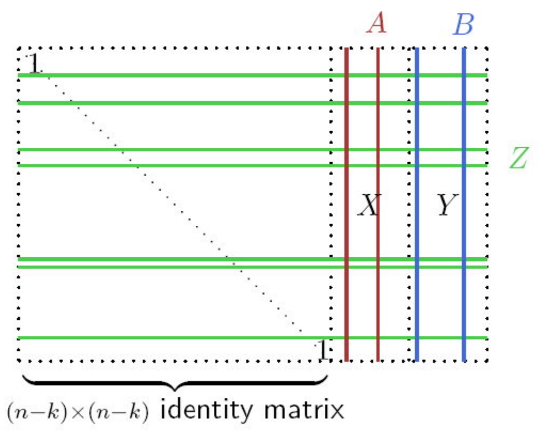
then return $x = (x_1, ..., x_n)$ where

for $i \le n-k$, $x_i = 1$ iff i is in one of the $w-2p$ set;

for $i > n-k$, $x_i = 1$ iff $i \in A \cup B$.

Else go the Step 1 to select $n-k$ new columns again.

Figure: weight distribution of Stern's algorithm



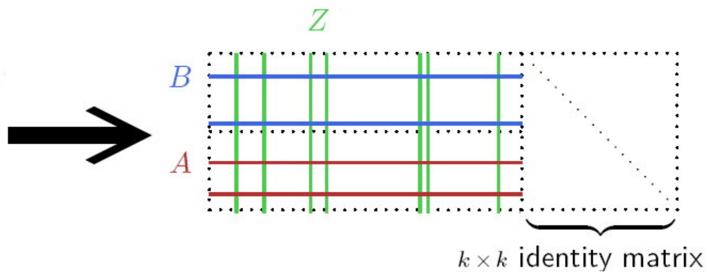


Figure1: parity check matrix H=(I|P)

Figure2: generating matrix G=(P^T|I)

Explanation of the above algorithm

In this part, we explain why the output vector satisfies all the demands of the algorithm. First, we know if $H = (I|P)$, $G = (-P^T|I) = (P^T|I)$ over $\mathbb{F}_2$. Let's clarify the algorithm step by step. Without loss of generality, we can assume H and G have the form in Figure 2.

1. **x is a codeword of C.**

   In fact, the definition of x means that x is the sum of rows of A and B in Figure 2. So x is composed by some rows of the generating matrix G and hence x is a codeword.

2. **x has $2p$ ones in the last k components. p of them are in X part and the other p of them are in Y part.**
   $\texttt{wt}(x_{n-k+1}, ..., x_n) = 2p$ because x is the sum of $2p$ rows where p rows are from A and B respectively. Moreover, since the right part of G is an identity matrix, the rows of B correspond to X part and the rows of A correspond to Y part in G.

3. **The l components of x at Z part are all zeros.**

   This is because $\pi(A) = \pi(B)$ and by definition the l components of x is $\pi(A) + \pi(B) = 0 \pmod 2$.

4. **The rest $n-k-l$ components has $w-2p$ ones.** By the condition of Step 6, we know the first k elements of x has weight $w-2p$. And by the Remark 3 above, we know those ones can not be in part Z. So those $w-2p$ ones are distributed in the rest $n-k-l$ components.

5. **wt(x)=w-2p+2p=w**

   This is an obvious result of the previous remarks.

5.3 Bernstein's improvement to Stern's algorithm

Bernstein's algorithm is an improvement of Stern's algorithm. The weight distribution of the target vector is the same as in Stern's setting. The difference is its more elaborate way to implement each step. Stern's algorithm contains 2 parameters **l** and **p**. Bernstein adds 3 new parameters **m**, **c**, **r** which provides more chances for a better optimization.

- the new parameter c

  To avoid the heavy computation of Gaussian elimination in iterations, start from the parity check matrix produced in the previous iteration. This means in each new iteration, there are already a an $(n-k)\times(n-k)$ identify matrix in the new H. When randomly choose $(n-k)$ columns from it, we expect $(n-k)\cdot\frac{(n-k)}{n}$ of the newly selected columns were just selected in the last iteration, i.e., those columns consisting one and zeros. Having these precalculated columns, when doing Gaussian elimination, one can simply do row operations of them to place those ones in the correct position without those heavy row-by-row elimination.

  Moreover, Bernstein formalizes this process. He reuses exactly $n-k-c$ column selections, and select the remaining c new columns from the $\underline{k \text{ deselected columns}}$. In fact, this is an extension of Canteaut's paper [17]. By setting c as a parameter, Stern's and Canteaut's approaches become special cases of Bernstein's approach. That is, $c = n-k$ in Stern's approach; $c = 1$ in Canteaut's approach. In fact, the smaller of c, the more iterations to do, but the less work on Gaussian elimination. So Bernstein takes this as a parameter in his improvement.

- the new parameter r

  For example, sometimes we need to add the 1st row to 5th row to cancel one at the first place of the 5th row, and also need to add the 2nd row to 5th row to cancel one at the second place of the 5th row. The idea is to precompute the sum of the 1st and 2nd row instead of doing the addition to the 5th row twice. More generally, choose r rows before doing Gaussian elimination. Precompute all 2^r-1 sums of any nonempty subset of the r rows. This can reduce the number of additions.

- the new parameter m

  Consider the l-set Z of Stern's attack. Bernstein chooses m disjoint l-set $Z_1, ..., Z_m$. If $Z = Z_i$ is failed, then let $Z = Z_{i+1}$ and do Step 5,6,7 again. This provides m new chances of success in each iteration. In other words, we don't need to touch H or do Gaussian elimination to start a new test. Instead, changing choice of Z can also be a new chance. Thus, Stern's attack can be considered as choosing $m = 1$ because he only prepares one Z for each iteration.

- When computing $\pi(A), \pi(B)$ for all p-subset of X, Y at Step 5, it is not necessary to compute those $p-1$ additions every time for those overlapping p-subsets. For example, one can memorize the sum of $p-1$ vectors and add the other one. If this fails ($\pi(A) \neq \pi(B)$) then substitute that vector by another one and add it to the precomputed sum of $p-1$ vectors to get new $\pi(A)$.

- This is just a little trick. At Step 6, when $\pi(A) = \pi(B)$, it needs to add up all the columns in $A \cup B$ but once the weight exceeds $w-2p$, one can give up this (A, B) pair because in that case the sum of columns in $A \cup B$ can never be $w-2p$.

Table: Complexity of information-set decoding algorithms against [1024, 524, 50] McEliece cryptosystem

Year	Algorithm	Log of bit operations	Optimal parameters choice
1986	Adams	80.7	
1988	Brickell	70.89	$p=2$
1989	Stern	66.21	$(p,l)=(2,18)$
1998	Canteaut	64.1	
2008	Bernstein	60.4	$(p,m,l,c,r)=(2,2,20,7,7)$
2009	Finiasz	59.9	

6 Variants of McEliece Cryptosystem and Attacks

6.1 Variants

The original McEliece cryptosystem is still unbroken up to now, but has two major drawbacks: large key size and low transmission rate. So some studies tried to use GRS or quasi-cyclic codes instead to reduce key length. But they have all been proved insecure. Two of the newest two variants are introduced by Berger in 2009 [10]. They are quasi-cyclic variant and dyadic variant. Since their structures are similar, we only discuss about the former one here. To reduce the key size, the variants use less redundancy and more structures to construct the generating matrix of the code. Expectedly, this reduces the number of unknown variables to solve for an attacker. It has been proven that even after the pubic matrix has being scrambled, it remains to be too "organized" (i.e., predictable). We will see the two attacks against the variants which are both key recovery attacks, i.e., the attacks that try to recover the secret key from the public key.

Key generation and materials The variants define the keys by defining the parity-check matrix (Niederreiter cryptosystem).

key generation: (parity-check matrix generation): Let α be a primitive element of $\mathbb{F}_{q^2}$ and β be an element of order l. Let $s \in \{1, ..., l-1\}$. Let $a_1, ..., a_b \in \mathbb{F}_{q^2}$. Let $\widetilde{H} = (B_1|\cdots|B_b)$ where

$$B_i = \begin{pmatrix} a_i & a_i\beta^s & \cdots & a_i(\beta^s)^{l-1}\alpha^{j_i} \\ a_i\alpha^{j_i} & a_i\beta^s\alpha^{j_i}\beta & \cdots & a_i(\beta^s)^{l-1}\alpha^{j_i}\beta^{l-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_i(\alpha^{j_i})^{t-1} & a_i\beta^s(\alpha^{j_i}\beta)^{t-1} & \cdots & a_i(\beta^s)^{l-1}(\alpha^{j_i}\beta^{l-1})^{t-1} \end{pmatrix}.$$

Now $\widetilde{H} \in \mathbb{F}_{q^2}^{t\times n}$. Consider each entry as two element of $\mathbb{F}_q$ by fixing a basis of $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$. Thus, $\widetilde{H}$ becomes a matrix in $\mathbb{F}_q^{2t\times n}$, called H'.

Let $S \in \mathbb{F}_q^{2t\times 2t}$ be an invertible matrix and let $M = \begin{pmatrix} \underbrace{P_{\sigma_1}}_{l\times l} & & \\ & \ddots & \\ & & \underbrace{P_{\sigma_b}}_{l\times l} \end{pmatrix}_{n\times n}$, where each P_{σ_i} denotes a cyclic shift on a l-tuple.

Public key: $(SH'M)$
Private key: $S, H', M, s, a_1, ..., a_b$.

Suggested parameters

q	q^m	t	l	b	n	assumed security	public key size(bits)
2^8	2^{16}	100	51	9	459	2^{80}	8,160
		100	51	10	510	2^{90}	9,792
		100	51	12	612	2^{100}	13,056
		100	51	15	765	2^{120}	20,400
2^{10}	2^{20}	112	75	6	450	2^{80}	6,750
		126	93	6	558	2^{90}	8,370
		108	93	8	744	2^{110}	14,880

Remark: The code has quasi-cyclic structure, i.e., the matrices are composed by circulant blocks with size $l \times l$. So the key size is $8 \cdot l \cdot$(the number of $l \times l$ blocks) bits where 8 comes from the base field $\mathbb{F}_q$. Indeed, each element of $\mathbb{F}_q$ has 8 bits. Let δ be the closest integer such that $2t \approx n - \delta l$. The number of rows of H' is

$2t \approx n - \delta l = bl - \delta l = (b-\delta)l$, and $k = \delta l$. The key size is $8 \cdot l \cdot$(the number of $l \times l$ blocks) $= 8 \cdot l \cdot \delta(b-\delta)$. Take the first case of the above table as an example, H' has size 200×459 which is $(n-k) \times n$. Since $200 \approx 4 \cdot 51$, $k = 459 - 4 \cdot 51 = 5 \cdot 51$. Thus, the systematic form of H' are composed by around $4 \cdot 5$ $l \times l$ circulant blocks. Hence the key size is $8 \cdot 51 \cdot 20 = 8160$ bits.

Remark: McEliece originally suggested security parameters (n = 1024,k = 524), resulting in a public key size of 524(1024-524) = 262,000 bits. Recent analysis suggests parameters (n = 2048,k = 1751) with key size 1751(2048 − 1751)=520,047 for 2^{80} of security, or (n = 1632,k = 1269) resulting in 1269(1632 − 1269)=460,647 bits key size.

6.2 Umaña and Leander's Attack

Recall that the public matrix is $P = (SH'M)$ where $H'M$ has a well-organized structure. In fact, the block diagonal matrix M almost doesn't influence the form of H'. The unknown parameters of H' have already taken account of the change by M well. Hence M can be considered as a part of the secret matrix H'. So one can assume $P = SH$ where P is the public matrix and S, H are secret matrices. Umaña's attack is to recover H from P. By classifying all γ $\mathbb{F}_q^{1\times 2t}$ based on the form of γP, the attack obtains l subspaces(Γ_d's) of $\mathbb{F}_q^{1\times 2t}$ with small dimensions and constructs equations from each subspace.

Definition 6.1. Let $\phi : \mathbb{F}_{q^2} \to \mathbb{F}_q$ defined by $\phi : x \mapsto x + \overline{x} = x + x^q$ which is in fact the trace and hence ϕ is $\mathbb{F}_q$-linear. Moreover, ϕ is a q-to-1 function mapping the whole $\mathbb{F}_q$ to 0.

Let b, l, n be integers such that $n = bl$. Let $\beta \in \mathbb{F}_q$ with order $l = 51$. Let $a_j, y_j \in \mathbb{F}_{q^2}$ for $j = 0, ..., b-1$. Define $c_0, ..., c_{n-1}$ and $x_0, ..., x_{n-1}$ by

$$c_{lj+i} := \beta^{is} a_j \text{ and } x_{lj+i} := \beta^i y_j$$

where $j = 0, ..., b-1$ and $i = 0, ...l-1$. The definition can be represented by the following tables.

The secret key H can be written in the form

$$H = \begin{pmatrix} \phi(c_0) & \phi(c_1) & \cdots & \phi(c_{n-1}) \\ \phi(\theta c_0) & \phi(\theta c_1) & \cdots & \phi(\theta c_{n-1}) \\ \phi(c_0 x_0) & \phi(c_1 x_1) & \cdots & \phi(c_{n-1} x_{n-1}) \\ \phi(\theta c_0 x_0) & \phi(\theta c_1 x_1) & \cdots & \phi(\theta c_{n-1} x_{n-1}) \\ \vdots & \vdots & & \vdots \\ \phi(c_0 x_0^{t-1}) & \phi(c_1 x_1^{t-1}) & \cdots & \phi(c_{n-1} x_{n-1}^{t-1}) \\ \phi(\theta c_0 x_0^{t-1}) & \phi(\theta c_1 x_1^{t-1}) & \cdots & \phi(\theta c_{n-1} x_{n-1}^{t-1}) \end{pmatrix} = \begin{pmatrix} sk_0 \\ \vdots \\ sk_{2t-1} \end{pmatrix}$$

where $sk_0, ..., sk_{2t-1}$ are the row vectors of H, and $\theta \in \mathbb{F}_{q^2} \setminus \mathbb{F}_q$.

Definition 6.2. Let $S \in \mathbb{F}_q^{2t\times 2t}$ be a secret invertible matrix . Let $P := SH \in \mathbb{F}_q^{2t\times n}$. Note that P has the form $P = SH = \begin{pmatrix} \phi(c_0 g_0(x_0)) & \phi(c_1 g_0(x_1)) & \cdots & \phi(c_{n-1} g_0(x_{n-1})) \\ \phi(c_0 g_1(x_0)) & \phi(c_1 g_1(x_1)) & \cdots & \phi(c_{n-1} g_1(x_{n-1})) \\ \vdots & \vdots & & \vdots \\ \phi(c_0 g_{2t-1}(x_0)) & \phi(c_1 g_{2t-1}(x_1)) & \cdots & \phi(c_{n-1} g_{2t-1}(x_{n-1})) \end{pmatrix}$ where $g_i \in \mathbb{F}_{q^2}[x]$ of degree at most $t-1$.

Proposition 6.3. Thus for any $\gamma \in \mathbb{F}_q^{2t}$, γP can be written as $\gamma P = (\phi(c_0 g_\gamma(x_0)), ..., \phi(c_{n-1} g_\gamma(x_{n-1})))$, where $g_\gamma(x) = \sum_{i=0}^{2t-1} \gamma_i g_i(x)$.

Remark 6.4. In an attacker's view, P is a public matrix and H is a secret key. The goal of this attack is to recover H from the given P.

Definition 6.5. More generally, for all $d = 0, ..., l-1$, define

$$\Gamma_d := \{\gamma \in \mathbb{F}_q^{2t} \mid \gamma P = (A_1\overrightarrow{\beta_d}, ..., A_b\overrightarrow{\beta_d}), \text{ for some } A_i \in \mathbb{F}_q\}$$

where $\overrightarrow{\beta_d} = (1, \beta^{s+d}, \beta^{2(s+d)}, ..., \beta^{(l-1)(s+d)}) \in \mathbb{F}_q^{1\times l}$.

Lemma 6.6. $\dim_{\mathbb{F}_q} \Gamma_d = 4$ for $d < t-l$. $\dim_{\mathbb{F}_q} \Gamma_d = 2$ for $t-l \le d < l$.
Moreover,

$$\Gamma_d = \{\gamma \in \mathbb{F}_q^{1\times 2t} \mid g_\gamma = \alpha_1 x^d + \alpha_2 x^{d+l}, \alpha_1, \alpha_2 \in \mathbb{F}_{q^2}\}.$$

Lemma 6.7. For $d < t-l$, suppose $\{\gamma_{(d)1}, \gamma_{(d)2}, \gamma_{(d)3}, \gamma_{(d)4}\}$ is a $\mathbb{F}_q$−basis of Γ_d. Then there exists an invertible matrix $M_d \in \mathbb{F}_q^{4\times 4}$ such that

$$\begin{pmatrix} \gamma_{(d)1} \\ \gamma_{(d)2} \\ \gamma_{(d)3} \\ \gamma_{(d)4} \end{pmatrix} P = M_d \begin{pmatrix} sk_{2d} \\ sk_{2d+1} \\ sk_{2(l+d)} \\ sk_{2(l+d)+1} \end{pmatrix}.$$

Similarly, for $t-l \le d < l$, suppose $\{\gamma_{(d)1}, \gamma_{(d)2}\}$ is a $\mathbb{F}_q$−basis of Γ_d. Then there exists an invertible matrix $M \in \mathbb{F}_q^{2\times 2}$ such that

$$\begin{pmatrix} \gamma_{(d)1} \\ \gamma_{(d)2} \end{pmatrix} P = M \begin{pmatrix} sk_{2d} \\ sk_{2d+1} \end{pmatrix}.$$

For example, when $d = 0$,

$$M^{-1}\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix} P = \begin{pmatrix} sk_0 \\ sk_1 \\ sk_{102} \\ sk_{103} \end{pmatrix} = \begin{pmatrix} \phi[c_0] & \phi[c_1] & \cdots & \phi[c_{n-1}] \\ \phi[\theta c_0] & \phi[\theta c_1] & \cdots & \phi[\theta c_{n-1}] \\ \phi[c_0 x_0^{51}] & \phi[c_1 x_0^{51}] & \cdots & \phi[c_{n-1} x_0^{51}] \\ \phi[\theta c_0 x_0^{51}] & \phi[\theta c_1 x_0^{51}] & \cdots & \phi[\theta c_{n-1} x_0^{51}] \end{pmatrix}$$

Denote the entries of M^{-1} by (β_{ij}). The k^{th} component of the first 2 rows of the above equation can be written as

$$\begin{aligned} \beta_{00}(\gamma_1 P)^{(k)} + \beta_{01}(\gamma_2 P)^{(k)} + \beta_{02}(\gamma_3 P)^{(k)} + \beta_{03}(\gamma_4 P)^{(k)} &= sk_0^{(k)} = \phi[c_k] = c_k + \overline{c_k} \\ \beta_{10}(\gamma_1 P)^{(k)} + \beta_{11}(\gamma_2 P)^{(k)} + \beta_{12}(\gamma_3 P)^{(k)} + \beta_{13}(\gamma_4 P)^{(k)} &= sk_1^{(k)} = \phi[\theta c_k] = \theta c_k + \overline{\theta c_k} \end{aligned}$$

Dividing the second equation by $\overline{\theta}$ and adding the two implies

$$\delta_0(\gamma_1 P)^{(k)} + \delta_1(\gamma_2 P)^{(k)} + \delta_2(\gamma_3 P)^{(k)} + \delta_3(\gamma_4 P)^{(k)} = \left(\frac{\theta}{\overline{\theta}} + 1\right) c_k,$$

where

$$\delta_0 = \left(\beta_{00} + \frac{\beta_{10}}{\overline{\theta}}\right), \delta_1 = \left(\beta_{01} + \frac{\beta_{11}}{\overline{\theta}}\right), \delta_2 = \left(\beta_{02} + \frac{\beta_{12}}{\overline{\theta}}\right), \delta_3 = \left(\beta_{03} + \frac{\beta_{13}}{\overline{\theta}}\right) \in \mathbb{F}_{q^2}.$$

The structure of this attack is basically collect this kind of equations to build a linear system and solve it. The unknown variables in the above equation are $\delta_0, \delta_1, \delta_2, \delta_3, c_k$.

Attack performance This shows that the attack is not practical.

q	q^m	t	l	b	n	assumed security	attack complexity
2^8	2^{16}	100	51	9	459	2^{80}	$2^{74.9}$
		100	51	10	510	2^{90}	$2^{75.1}$
		100	51	12	612	2^{100}	$2^{75.3}$
		100	51	15	765	2^{120}	$2^{75.6}$
2^{10}	2^{20}	112	75	6	450	2^{80}	-
		126	93	6	558	2^{90}	$2^{87.3}$
		108	93	8	744	2^{110}	$2^{86.0}$

6.3 The Attack of Faugère et al.

The above attack looks for linear equations but Faugère's attack focuses on nonlinear equations that can be easily obtained. This attack starts with the same problem pattern - to recover H from a given $P = SH$. But they consider a slightly different form of H. They consider H as a matrix in $\mathbb{F}_{q^2}^{t\times n}$ instead of the row-spanning form in $\mathbb{F}_q^{2t\times n}$. Compared to Umaña's attack, they don't use the trace map ϕ and this makes the unknown variables fewer. This attack makes the full use of the insufficiency of unknown variables. The idea of this attack is much simpler but they faces the difficulty of solving high degree nonlinear equations. Faugère et al. developed an algorithm for efficiently computing Gröbner basis [13] and then applied it to the attack to solve the variables of the nonlinear equations given by the matrix multiplication.

We use the same notation as much as possible here. Consider the secret parity-check matrix in the form

$$H = \begin{pmatrix} Y_0 & Y_1 & \cdots & Y_{n-1} \\ Y_0X_0 & Y_1X_1 & \cdots & Y_{n-1}X_{n-1} \\ Y_0X_0^2 & Y_1X_1^2 & \cdots & Y_{n-1}X_{n-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ Y_0X_0^{t-1} & Y_1X_1^{t-1} & \cdots & Y_{n-1}X_{n-1}^{t-1} \end{pmatrix}_{t\times n}$$

where $X_i, Y_j \in \mathbb{F}_{q^2}$. $P = SH$ is given. Since $PG^T = 0$, $HG^T = 0$, where G is the public generating matrix. This gives $t \times k$ nonlinear equations. Let $\mathcal{I}$ be the ideal generated by these nonlinear equations and $\mathcal{V}$ be the corresponding variety. Then compute the Gröbner basis of $\mathcal{I} \cap \mathbb{F}_{q^2}[Y]$ by F_4 algorithm [13]. Make use of $X_{lj+i} = \beta^i y_j$ and $Y_{lj+i} = \beta^{is} a_j$ to reduce the number of variables. Then the variables are $a_0, ..., a_{b-1}$ and $y_0, ..., y_{b-1}$.

Attack performance This shows that the attack is practical.

q	q^m	t	l	b	n	assumed security	attack complexity
2^8	2^{16}	100	51	9	459	2^{80}	$2^{18.9}$
		100	51	10	510	2^{90}	$2^{17.1}$
		100	51	12	612	2^{100}	$2^{16.2}$
		100	51	15	765	2^{120}	$2^{14.7}$
2^{10}	2^{20}	112	75	6	450	2^{80}	$2^{15.8}$
		126	93	6	558	2^{90}	$2^{17.1}$
		108	93	8	744	2^{110}	$2^{14.5}$

7 A Possible Improvement of Umaña's Attack

A possible improvement of Umaña's attack is to obtain more structures and then more equations and also consider the secret matrix S. We choose some $\gamma \in \mathbb{F}_q^{1\times 2t}$ that may reveal the value of some items. Then represent those items in another form and then add up some of them to cancel most of terms by using $|\beta| = l$. Thus, there comes out a number of nonlinear equations. Then for each Γ_d, and $\gamma \in \Gamma_d$, by comparing the coefficients of g_γ, we can get more linear equations. The observation shows that each column of the secret matrix S can be expressed by a linear combination of only 2 or 4 entries in the column. For convenience of expression, assume $s = 1$, $l = 51$, $t = 100$, $b = 9$. Note that $s \in \{1, ..., l-1\}$ is a secret number but only has $l = 51$ choices.

Let $\gamma = (10\cdots 0) \in \mathbb{F}_q^{200}$. Then $g_\gamma = \sum_{0}^{199} \gamma_k g_k = g_0$, and hence

$$\begin{aligned} \text{1st row of } P &= \gamma P \\ &= (\boldsymbol{\beta^0}\phi[a_0 g_0(\beta^0 y_0)],\ \ \boldsymbol{\beta^1}\phi[a_0 g_0(\beta^1 y_0)],\ \ \boldsymbol{\beta^2}\phi[a_0 g_0(\beta^2 y_0)],\ \ ......\ ) \end{aligned}$$

The left hand side is known, so for all $i = 0, ..., 50$, $\phi[a_0 g_0(\beta^i y_0)] = known$ are all known numbers.

Note that we use $x = known$ to represent that the value of x is known or can be figured out.

Write $g_0 = \sum_{d=0}^{t-1} C_{0d} x^d = C_{00} + C_{01}x + \cdots + C_{0,99}x^{99}$, where $C_{0d} \in \mathbb{F}_{q^2}$. In fact, $C_{0d} = S_{0,2d} + \theta S_{0,2d+1}$. (The rows of S actually represents the coefficients of $g_0, ..., g_{2t-1}$.)Then

$$\begin{array}{rcll}
\phi[a_0 g_0(y_0)] & = & \phi[\mathbf{a_0 C_{00}} + a_0 C_{01} y_0 + \cdots + \mathbf{a_0 C_{0,51} y_0^{51}} + a_0 C_{0,52} y_0^{52} + \cdots + a_0 C_{0,99} y_0^{99}] & (0) \\
\phi[a_0 g_0(\beta y_0)] & = & \phi[\mathbf{a_0 C_{00}} + a_0 C_{01} \beta y_0 + \cdots + \mathbf{a_0 C_{0,51} y_0^{51}} + a_0 C_{0,52} \beta y_0^{52} + \cdots + a_0 C_{0,99} \beta^{48} y_0^{99}] & (1) \\
\phi[a_0 g_0(\beta^2 y_0)] & = & \phi[\mathbf{a_0 C_{00}} + a_0 C_{01} \beta^2 y_0 + \cdots + \mathbf{a_0 C_{0,51} y_0^{51}} + a_0 C_{0,52} \beta^{2\cdot 1} y_0^{52} + \cdots + a_0 C_{0,99} \beta^{2\cdot 48} y_0^{99}] & (2) \\
\vdots & & \vdots & \vdots \\
\phi[a_0 g_0(\beta^{50} y_0)] & = & \phi[\mathbf{a_0 C_{00}} + a_0 C_{01} \beta^{50} y_0 + \cdots + \mathbf{a_0 C_{0,51} y_0^{51}} + a_0 C_{0,52} \beta^{50\cdot 1} y_0^{52} + \cdots + a_0 C_{0,99} \beta^{50\cdot 48} y_0^{99}] & (50)
\end{array}$$

The items on the left hand side have all become known. Add up the above 51 equations. Then we see

$$known = \phi[a_0 C_{00} + a_0 C_{0,51} y_0^{51}] \quad (*)$$

Similarly, for $j = 1, ..., 8$, we also have ($known = a_j C_{00} + a_j C_{0,51} y_j^{51}$). Similarly, for $g_1, ..., g_{199}$, we also have

$$known = \phi[a_j C_{k0} + a_j C_{k,51} y_j^{51}], \quad \forall k = 1, ..., 199$$

To summarize, for $j = 0, ..., 8$, $k = 0, ..., 199$, we have

$$\phi[a_j C_{k0} + a_j C_{k,51} y_j^{51}] = known$$

which means we have obtained $9 \cdot 200 = 1800$ nonlinear equations with $18 + 2 \cdot 200 = 418$ variables. For convenience, rename $C_{0,k}$ by C_k and rename $C_{51,k}$ by C'_k, we have for all $j \in \{0, ..., 8\}$, $k \in \{0, ...99\}$,

$$\phi[a_j C_k + a_j C'_k y_j^{51}] = known$$

Or say

$$\begin{array}{llll}
\phi[a_0 C_0 + a_0 C'_0 y_0^{51}] = known, & \phi[a_0 C_1 + a_0 C'_1 y_0^{51}] = known, & ......, & \phi[a_0 C_{99} + a_0 C'_{99} y_0^{51}] = known \\
\phi[a_1 C_0 + a_1 C'_0 y_1^{51}] = known, & \phi[a_1 C_1 + a_1 C'_1 y_1^{51}] = known, & ......, & \phi[a_1 C_{99} + a_1 C'_{99} y_1^{51}] = known \\
\vdots & & & \vdots \\
\phi[a_8 C_0 + a_8 C'_0 y_8^{51}] = known, & \phi[a_8 C_1 + a_8 C'_1 y_8^{51}] = known, & ......, & \phi[a_8 C_{99} + a_8 C'_{99} y_8^{51}] = known
\end{array}$$

Similarly, $(0) + \beta^{-1}(1) + \beta^{-2}(2) + \cdots + \beta^{-50}(50)$ gives

$$known = \phi[a_0 C_{k1} y_0 + a_0 C_{k,52} y_0^{52}] \ , \quad \forall j \in \{0, ..., 8\}, \ \forall k \in \{0, ..., 199\}$$

and thus we got another 1800 nonlinear equations with 418 variables. To summarize, for $\xi \in \{0, ..., 48\}$, $\sum_{i=0}^{50} \beta^{-\xi i}(i)$ totally give us $49 \cdot 1800$ nonlinear equations with $100 \cdot 200 = 20000$ variables. For $\xi = 49$, we get $known = \phi[a_j C_{k,49} y_j^{49}]$. For $\xi = 50$, we get $known = \phi[a_j C_{k,50} y_j^{50}]$. So for $\xi \in \{49, 50\}$ we get $9 \cdot 200 \cdot 2$ nonlinear equations with $18 + 2 \cdot 200 = 418$ variables.

Similarly, $(0) + (3) + (6) + (9) + \cdots + (48)$ gives

$$known = \phi[a_0 C_{00} + a_0 C_{0,17} + a_0 C_{0,34} + a_0 C_{0,51} + a_0 C_{0,68} + a_0 C_{0,85}] \quad (**)$$

Combine with (*), get

$$known = \phi[a_0C_{0,17} + a_0C_{0,34} + a_0C_{0,68} + a_0C_{0,85}].$$

So $\forall j \in \{0, ...8\}, k \in \{0, ...199\}, known = \phi[a_jC_{k,17} + a_jC_{k,34} + a_jC_{k,68} + a_jC_{k,85}]$. So we got $9 \cdot 200 = 1800$ equations and $18 + 4 \cdot 200 = 818$ variables.

On the other hand, for $\gamma \in \Gamma_0$, g_γ has the form $g_\gamma = C_{\gamma,0} + C_{\gamma,51}x^{51} = (\sum_{k=0}^{199} \gamma_k C_{k0}) + (\sum_{k=0}^{199} \gamma_k C_{k,51})x^{51}$ and hence for

$d \notin \{0, 51\}$, $\sum_{k=0}^{199} \gamma_k C_{kd} = 0$. Since Γ_0 is a known space with dimension 4 over $\mathbb{F}_q$, we got 4 linear equations with 200 variables for each d. So consider all $d \notin \{0, 51\}$, there are totally $4 \cdot 98 = 392$ linear equations with $98 \cdot 200 = 19600$ variables.

Recall that $\Gamma_d := \{\gamma \in \mathbb{F}_q^{200} \mid \gamma P \in U_d\}$ where Γ_0 is just a special case at $d = 0$. For $0 \leq d < t - l = 49$, $\dim_{\mathbb{F}_q} \Gamma_d = 4$ and g_γ has the form $g_\gamma = C_{\gamma,d}x^d + C_{\gamma,d+l}x^{d+l}$. Thus, for $\Gamma_0, ..., \Gamma_{48}$, we can apply the above argument and since those Γ_d spaces are mutually disjoint, we got $4 \cdot 98 \cdot 49 = 19208$ linear equations with coefficients in $\mathbb{F}_q$ and $100 \cdot 200 = 20000$ variables in $\mathbb{F}_{q^2}$. A special case is at $d \in \{49, 50\}$, $\dim_{\mathbb{F}_q} \Gamma_d = 2$ and for $\gamma \in \Gamma_d$, g_γ has the form $g_\gamma = C_{\gamma,d}x^d$. From this we got $2 \cdot 2 \cdot 99 = 396$ linear equations with coefficients in $\mathbb{F}_q$ and $99 \cdot 200 = 19800$ variables in $\mathbb{F}_{q^2}$. Combine the case for all $d = 0, ..., 50$. We get $19208 + 396 = 19604$ linear equations with coefficients in $\mathbb{F}_q$ and $100 \cdot 200 = 20000$ variables in $\mathbb{F}_{q^2}$.

Since $C_{kd} = S_{k,2d} + \theta S_{k,2d+1}$ for all $k = 0, ..., 199$, $d = 0, ..., 99$. We can view the above linear equations given by Γ_d's as $2 \cdot 19604 = 39208$ linear equations with coefficients in $\mathbb{F}_q$ and $2 \cdot 20000 = 40000$ variables in $\mathbb{F}_q$.

For example, we can start from $d = 50$. The cases $d = 49$ and $d = 50$ are slightly easier since both Γ_{49} and Γ_{50} have dimension only 2. The structure is weaker at $d = 49, 50$. First, $\dim_{\mathbb{F}_q} \Gamma_{49} = \dim_{\mathbb{F}_q} \Gamma_{50} = 2$ and we also know the value of $\phi[a_jC_{k,49}y_j^{49}]$, $\phi[a_jC_{k,50}y_j^{50}]$ for all $j = 0, ..., 8$, $k = 0, ..., 199$, where $C_{kd} \in \mathbb{F}_{q^2}$ denotes the coefficient of the x^d term of g_k and in fact $C_{kd} = S_{k,2d} + \theta S_{k,2d+1}$. So we start the attack from $d = 50$.

An observation is that $\Gamma_1, ..., \Gamma_{50}$ are mutually disjoint (this observation actually proves the experimental observations in Umaña's paper) and $\dim_{\mathbb{F}_q} \Gamma_1 \oplus \cdots \oplus \Gamma_{49} = 198$. It can be proved that for $d = 0, ..., 48$, if $\gamma \in \Gamma_d$ then

$$C_{\gamma,d}x^d + C_{\gamma,d+51}x^{d+51} = g_\gamma = \sum_{k=0}^{199} \gamma_k g_k = \sum_{k=0}^{199} \gamma_k (\sum_{d=0}^{99} C_{kd}x^d) = \sum_{d=0}^{99} (\sum_{k=0}^{199} \gamma_k C_{kd})x^d.$$

Similarly for $d = 49$, $\gamma \in \Gamma_{49}$ implies

$$C_{\gamma,d}x^{49} = g_\gamma = \sum_{k=0}^{199} \gamma_k g_k = \sum_{k=0}^{199} \gamma_k (\sum_{d=0}^{99} C_{kd}x^d) = \sum_{d=0}^{99} (\sum_{k=0}^{199} \gamma_k C_{kd})x^d.$$

The x^{50} terms of the right hand side of the above equations give 198 linear equations for $C_{0,50}, ..., C_{199,50} \in \mathbb{F}_{q^2}$. In other words, for all $\gamma \in \Gamma_1 \oplus \cdots \oplus \Gamma_{49} \subseteq \mathbb{F}_q^{200}$, we have $\sum_{k=0}^{199} \gamma_k C_{k,50} = 0$. Since $C_{kd} = S_{k,2d} + \theta S_{k,2d+1}$, we can view these as 396 linear equations for $S_{0,100}, ..., S_{199,100}, S_{0,101}, ..., S_{199,101} \in \mathbb{F}_q$ by splitting every equation over $\mathbb{F}_{q^2}$ into two equations over $\mathbb{F}_q$, i.e.,

$$0 = \sum_{k=0}^{199} \gamma_k C_{k,50} = \sum_{k=0}^{199} \gamma_k (S_{k,100} + \theta S_{k,101}) \Longrightarrow 0 = \sum_{k=0}^{199} \gamma_k S_{k,100} \text{ and } 0 = \sum_{k=0}^{199} \gamma_k S_{k,101}.$$

Thus $S_{0,100}$ and $S_{1,100}$ determine $S_{2,100}, ..., S_{199,100}$; $S_{0,101}$ and $S_{1,101}$ determine $S_{2,101}, ..., S_{199,101}$. So write

$$\begin{aligned}
S_{2,100} &= A_2 S_{0,100} + B_2 S_{1,100} \\
S_{2,101} &= A'_2 S_{0,101} + B'_2 S_{1,101} \\
S_{3,100} &= A_3 S_{0,100} + B_3 S_{1,100} \\
S_{3,101} &= A'_3 S_{0,101} + B'_3 S_{1,101}
\end{aligned}$$

where $A_2, B_2, A_3, B_3 \in \mathbb{F}_q$ are known numbers. Now make use of

$$\begin{aligned}
known &= \phi[a_j C_{0,50} y_j^{50}] = \phi[S_{0,100} a_j y_j^{50}] + \phi[S_{0,101} \theta a_j y_j^{50}] \\
known &= \phi[a_j C_{1,50} y_j^{50}] = \phi[S_{1,100} a_j y_j^{50}] + \phi[S_{1,101} \theta a_j y_j^{50}] \\
known &= \phi[a_j C_{2,50} y_j^{50}] = A_2 \phi[S_{0,100} a_j y_j^{50}] + B_2 \phi[S_{1,100} a_j y_j^{50}] + A'_2 \phi[S_{0,101} \theta a_j y_j^{50}] + B'_2 \phi[S_{0,101} \theta a_j y_j^{50}] \\
known &= \phi[a_j C_{3,50} y_j^{50}] = A_3 \phi[S_{0,100} a_j y_j^{50}] + B_3 \phi[S_{1,100} a_j y_j^{50}] + A'_3 \phi[S_{0,101} \theta a_j y_j^{50}] + B'_3 \phi[S_{0,101} \theta a_j y_j^{50}]
\end{aligned}$$

to solve $\phi[S_{0,100} a_j y_j^{50}], \phi[S_{0,101} a_j y_j^{50}], \phi[S_{1,100} a_j y_j^{50}], \phi[S_{1,100} a_j y_j^{50}]$. Thus, for all $j = 0, ..., 8$,

$$S_{0,100} \phi[a_j y_j^{50}] = known, \; S_{1,100} \phi[a_j y_j^{50}] = known, \; S_{0,101} \phi[\theta a_j y_j^{50}] = known, \; S_{1,100} \phi[\theta a_j y_j^{50}] = known.$$

We can also make use of the following fact obtained by Umaña's paper. By the paper, there exists invertible 2×2 matrix $M = \begin{pmatrix} * & * \\ * & * \end{pmatrix} \in M_2(\mathbb{F}_q)$ such that

$$\begin{pmatrix} * & * \\ * & * \end{pmatrix} \underbrace{\begin{pmatrix} \gamma_{(50)1} \\ \gamma_{(50)2} \end{pmatrix} P}_{known} = \begin{pmatrix} sk_{100} \\ sk_{101} \end{pmatrix}$$

where $\{\gamma_{(50)1}, \gamma_{(50)2}\}$ denotes a basis of Γ_{50}.

If we solve $\phi[a_j y_j^{49}], \phi[a_j y_j^{50}], \phi[\theta a_j y_j^{49}], \phi[\theta a_j y_j^{50}]$, then we can solve $a_j y_j^{49}$ and $a_j y_j^{50}$ by

$$\begin{aligned}
known = \phi[a_j y_j^{49}] + \frac{1}{\bar{\theta}} \phi[\theta a_j y_j^{49}] &= (1 + \frac{\theta}{\bar{\theta}})(a_j y_j^{49}) \\
known = \phi[a_j y_j^{50}] + \frac{1}{\bar{\theta}} \phi[\theta a_j y_j^{50}] &= (1 + \frac{\theta}{\bar{\theta}})(a_j y_j^{50}).
\end{aligned}$$

References

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

[2] Eugene Prange. The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory, 8(5):5-9, September 1962

[3] Pil Joong Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In Christoph G. G*ü*nther, editor, Advances in cryptology—EUROCRYPT '88, volume 330 of Lecture Notes in Computer Science, pages 275-280. Springer, Berlin, 1988.

[4] Jacques Stern. A method for finding codewords of small weight. In G*é*rard D. Cohen and Jacques Wolfmann, editors, Coding theory and applications, volume 388 of Lecture Notes in Computer Science, pages 106-113. Springer, New York, 1989.

[5] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory, 1978. Jet Propulsion Laboratory DSN Progress Report 42-44. URL: http://ipnpr.jpl.nasa.gov/progress report2/ 42-44/44N.PDF.

[6] Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Contr. and Inform. Theory 15, 159-166 (1986).

[7] Li, Y.X., Deng, R., Wang, X.M.: On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Trans. Inform. Theory 40, 271-273 (1994).

[8] Baldi, M., Chiaraluce, F.: Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In: Proc. IEEE ISIT 2007, Nice, France, June 2007, pp. 2591-2595 (2007).

[9] Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. In: Proc. First International Conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China (April 2008).

[10] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit and Ayoub Otmani, Reducing Key Length of the McEliece Cryptosystem, 2009

[11] V. G. Umaña and G. Leander. Practical key recovery attacks on two McEliece variants. In International Conference on Symbolic Computation and Cryptography - SCC 2010, 2010. To appear. URL: http://eprint.iacr.org/2009/509.pdf.

[12] Faugère, J.-C. ; Otmani, A. ; Perret, L. ; Tillich, J.-P.: Algebraic Cryptanalysis of McEliece Variants with Compact Keys. To appear. URL: http://www-rocq.inria.fr/secret/Jean-Pierre.Tillich/publications/algebraic.pdf.

[13] Faugère, J. C. A new efficient algorithm for computing Grobner bases (F4). Journal of Pure and Applied Algebra 139, 1-3 (June 1999), 61-88.

[14] Monroe, C., Meekhof, D. M., King, B. E., Itano, W. M. & Wineland, D. J. 1995 Demonstration of a fundamental quantum logic gate. Phys. Rev. Lett. 75, 4714.

[15] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In Johannes Buchmann and Jintai Ding, editors, Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, volume 5299 of Lecture Notes in Computer Science, pages 31-46. Springer, Berlin, 2008.

[16] Daniel J. Bernstein, Tanja Lange, Christiane Peters, and Henk C. A. van Tilborg. Explicit bounds for generic decoding algorithms for code-based cryptography. Pre-proceedings of WCC 2009, pages 168-180, 2009.

[17] A. Canteaut and F. Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Transactions on Information Theory, 44(1):367-378, 1998.