A Study on Bernstein-Lange-Peters' Attack on McEliece Cryptosystem

Kittiphon Phalakarn

Abstract. This work studies the attack on McEliece public-key cryptosystem presented by Bernstein, Lange, and Peters in 2008. The attack can be considered as an optimized version of Stern's attack that finds codewords of low weight in a binary linear code. Their implementation breaks the original parameters proposed by McEliece using $2^{60.55}$ bit operations, comparing to $2^{64.1}$ bit operations of the fastest attack at that time, and new parameters were proposed in their paper to strengthen the cryptosystem. This work also gives a numerical example of applying the attack, and discusses state-of-the-art attacks and implementations related to McEliece cryptosystem.

1 Introduction

As Shor's algorithm, a polynomial-time quantum algorithm for factoring integer and solving the discrete logarithm problem, was discovered in 1994, the public-key cryptosystems used in practice, e.g. RSA cryptosystem, Diffie-Hellman key exchange, and elliptic curve cryptosystem, can be considered insecure against quantum computers. Even though large-scale quantum computers cannot be built now, it is probable that they can be built in the future. Thus, quantum-resistant cryptosystems are required to be studied in order to secure the communications at the time when quantum computers are ready. NIST has already started a process of soliciting, evaluating, and standardizing quantum-resistant public-key cryptosystems in 2017.

Among the quantum-resistant candidates is Classic McEliece cryptosystem [5] used for public-key encryption and key encapsulation mechanism. Classic McEliece is based on McEliece cryptosystem [15], the first code-based public-key cryptosystem introduced by McEliece in 1978. The security of McEliece has long been studied, and its parameters have been improved to prevent discovered attacks. One attack on McEliece cryptosystem using original parameters was presented by Bernstein, Lange, and Peters in 2008 [7]. It is based on Stern's attack [20] which finds codewords of low weight in a binary linear code.

The attack finds the corresponding plaintext from the given ciphertext using a reasonable amount of computation time: $2^{60.55}$ bit operations on average, which is 2^{58} CPU cycles or 1400 days on a single computer. Performing the attack on 200 such computers reduces the time to just one week. This result is remarkable as the fastest attack at that time, published by Canteaut, Chabaud, and Sendrier [9, 10], spends $2^{64.1}$ bit operations, which is 2^{68} CPU cycles or 220000 days on a single computer. New parameters were also proposed as a countermeasure for preventing this attack.

In this work, we study Bernstein-Lange-Peters' attack on McEliece cryptosystem as presented in [7] and discuss some recent works on McEliece cryptosystem. We first describe McEliece cryptosystem with some backgrounds on error-correcting codes in Section 2. Then, in Section 3, the attack is described with a numerical example. We discuss state-of-the-art research related to McEliece cryptosystem, including Classic McEliece, in Section 4, and we conclude our work in Section 5.

2 McEliece Cryptosystem

McEliece cryptosystem is a code-based public-key cryptosystem, which uses error-correcting codes. In this section, we describe linear codes, Goppa codes, and the key generation-encryption-decryption processes of McEliece cryptosystem.

2.1 Error-Correcting Codes and Linear Codes

When message is transmitted over channels that are subject to noise, there is a chance that parts or all of the message is modified, e.g. some bits are flipped. The goal of using error-correcting codes is to provide an ability to correct these errors. The process of applying error-correcting codes is depicted as in Fig. 1.

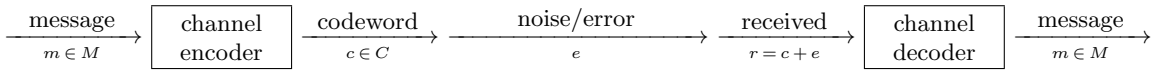


Fig. 1. The process of applying error-correcting codes

In this work, we consider m, c, e, r as vectors over the finite field GF(2). We define the *Hamming distance* $d(x, y)$ of $x, y \in \{0,1\}^n$ as the number of coordinate positions in which x and y differ. To transmit a message m over a noisy channel, we first encode m to a codeword $c \in C$ and transmit c over the channel. Suppose r is received. We then "correct the error" by finding $c \in C$ with smallest $d(r, c)$ and decode c to obtain m.

For $x \in \{0,1\}^n$, let the *Hamming weight* $w(x)$ of x denote the number of nonzero coordinates in x, the *weight* of C is $w(C) = \min\{w(c) : c \in C, c \neq 0\}$, and $t = \lfloor \frac{w(C)-1}{2} \rfloor$. It can be proved that if $w(e) \leq t$, then for $r = c + e$, there is a unique $c \in C$ with smallest $d(r, c)$. We call t as the *error-correcting capability* of C.

When $M = \{0,1\}^k$ and C is a k-dimensional subspace of $\{0,1\}^n$, we call C as a *binary linear* (n, k)*-code*. Since $\dim(C) = k$, there exists a *generator matrix* $G_{k\times n}$ for C such that $C = \{xG : x \in \{0,1\}^k\}$. Thus, the channel encoder can encode m to $c = mG$. Also, there exists a *parity-check matrix* $H_{(n-k)\times n}$ for C such that $c \in C$ if and only if $Hc^\top = 0$. Note that $\dim(G) = k$, $\dim(H) = n - k$, and $GH^\top = 0$.

If G has a form $[\, I_k \,|\, A_{k\times(n-k)} \,]_{k\times n}$, then G is in *standard form* and $H = [\, A^\top_{(n-k)\times k} \,|\, I_{n-k} \,]_{(n-k)\times n}$ is a parity-check matrix for C. We can see that a result of performing elementary row operations on G is still a generator matrix for the same code C (with different encoding rule). Thus, if the reduced row echelon form of G is not in standard form, it is not possible to find a standard form G for C. However, we can find a standard form G' for an equivalent code C' obtained by permuting columns of C.

2.2 Classical Goppa Codes

To construct a Goppa code Γ of length n and error-correcting capability t, first choose a finite field $\mathrm{GF}(2^d)$, its basis $\langle \beta_1, \ldots, \beta_d \rangle$, its n distinct elements $\{\alpha_1, \ldots, \alpha_n\}$, and an irreducible polynomial $g \in \mathrm{GF}(2^d)[x]$ of degree t, where $2 \leq t \leq (n-1)/d$. The Goppa code Γ consists of codewords $c = [c_1, \ldots, c_n] \in \{0,1\}^n$ where

$$\sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \quad \text{in} \quad \mathrm{GF}(2^d)[x]/g. \tag{1}$$

This gives $w(\Gamma) \geq 2t+1$, thus the error-correcting capability of Γ is at least t. When $w(e) \leq t$, the error can be corrected using Patterson's algorithm [19]. We also have $\dim(\Gamma) \geq n - td$. For constructing cryptosystems, we assume that $\dim(\Gamma) = n - td$. Hence, Γ is $(n, n - td)$-code. A parity-check matrix H for Γ is

$$H = \begin{bmatrix} 1/g(\alpha_1) & \cdots & 1/g(\alpha_n) \\ \alpha_1/g(\alpha_1) & \cdots & \alpha_n/g(\alpha_n) \\ \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/g(\alpha_1) & \cdots & \alpha_n^{t-1}/g(\alpha_n) \end{bmatrix}_{td\times n} \text{where} \quad \alpha_j^i/g(\alpha_j) = \sum_{k=1}^{d} a_k \beta_k \quad \text{is represented as} \quad \begin{bmatrix} a_1 \\ \vdots \\ a_d \end{bmatrix}.$$

Example 1. We construct a $(15, 3)$-Goppa code with length $n = 15$ and error-correcting capability $t = 3$ by choosing $d = 4$, $\mathrm{GF}(2^4) = \mathrm{GF}(2)[y]/(y^4 + y + 1)$ with a basis $\langle 1, y, y^2, y^3 \rangle$, and $g = x^3 + x + 1 \in \mathrm{GF}(2^4)[x]$. Since y is a generator of $\mathrm{GF}(2^4)$, we choose its 15 elements $\{y^0, \ldots, y^{14}\}$. A parity-check matrix H for Γ is

$$H = \begin{bmatrix} \frac{1}{g(y^0)} & \frac{1}{g(y^1)} & \cdots & \frac{1}{g(y^{14})} \\ \frac{y^0}{g(y^0)} & \frac{y^1}{g(y^1)} & \cdots & \frac{y^{14}}{g(y^{14})} \\ \frac{y^{0\cdot 2}}{g(y^0)} & \frac{y^{1\cdot 2}}{g(y^1)} & \cdots & \frac{y^{14\cdot 2}}{g(y^{14})} \end{bmatrix} = \begin{bmatrix} y^0 & y^8 & y^1 & y^{11} & y^2 & y^{10} & y^7 & y^{10} & y^4 & y^{13} & y^5 & y^5 & y^{14} & y^{10} & y^5 \\ y^0 & y^9 & y^3 & y^{14} & y^6 & y^0 & y^{13} & y^2 & y^{12} & y^7 & y^0 & y^1 & y^{11} & y^8 & y^4 \\ y^0 & y^{10} & y^5 & y^2 & y^{10} & y^5 & y^4 & y^9 & y^5 & y^1 & y^{10} & y^{12} & y^8 & y^6 & y^3 \end{bmatrix}.$$

Each element is then converted to a column of four elements corresponding to its representation with respect to the basis $\langle 1, y, y^2, y^3 \rangle$. For instance, $y^7 = 1 + y + y^3$ is converted to

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Thus, a parity-check matrix H and a generator matrix G (computed from H) for Γ are

$$H = \begin{bmatrix} 1&1&0&0&0&1&1&1&1&1&0&0&1&1&0 \\ 0&0&1&1&0&1&1&1&1&0&1&1&0&1&1 \\ 0&1&0&1&1&1&0&1&0&1&1&1&0&1&1 \\ 0&0&0&1&0&0&1&0&0&1&0&0&1&0&0 \\ 1&0&0&1&0&1&1&0&1&1&1&0&0&1&1 \\ 0&1&0&0&0&0&0&0&1&1&0&1&1&0&1 \\ 0&0&0&0&1&0&1&1&1&0&0&0&1&1&0 \\ 0&1&1&1&1&0&1&0&1&1&0&0&1&0&0 \\ 1&1&0&0&1&0&1&0&0&0&1&1&1&0&0 \\ 0&1&1&0&1&1&1&1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0&1&0&1&1&1&1&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1 \end{bmatrix}_{12\times 15} \qquad G = \begin{bmatrix} 1&0&0&1&0&0&1&1&0&1&0&1&1&1&1 \\ 0&1&0&0&1&1&1&0&0&1&0&1&0&0&1 \\ 0&0&1&1&0&0&0&1&1&0&1&0&1&1&0 \end{bmatrix}_{3\times 15}.$$

This gives a $(15, 3)$-Goppa code

$$\Gamma = \{000000000000000, 001100011010110, 010011100101001, 011111111111111, \\ 100100110101111, 101000101111001, 110111010000110, 111011001010000\}.$$

One can verify that all codewords in Γ satisfy Equation (1). We have $w(\Gamma) = w(001100011010110) = 7$, hence its error-correcting capability is $t = \lfloor \frac{w(\Gamma)-1}{2} \rfloor = 3$. Note that the above G is in standard form.

To send a message $m = 110$, we encode m to $c = mG = 110111010000110$ and transmit c over the channel. Suppose $r = 100110010010110$ is received. We see that $c = 110111010000110$ has smallest $d(r, c)$ among all codewords in Γ. So, we correct r to c and decode c to $m = 110$. When $|\Gamma|$ is large, comparing $d(r, c)$ among all $c \in \Gamma$ is not efficient. Patterson's algorithm can be used to efficiently correct the errors. □

2.3 McEliece Cryptosystem

McEliece proposed a code-based public-key cryptosystem using the classical Goppa codes as follows:

- Domain parameters: n, k, t
- Key generation: Alice does
  1. Construct an (n, k)-Goppa code Γ with error-correcting capability t where α_1, ..., α_n, and g are randomly chosen. Let $G_{k\times n}$ be a generator matrix of Γ.
  2. Select a random invertible matrix $S_{k\times k}$ and a random permutation matrix $P_{n\times n}$.
  3. Alice's public key is $G'_{k\times n} = SGP$. Her private key is (S, G, P).
- Encryption: To encrypt a message $m \in \{0, 1\}^k$ to Alice, Bob does
  1. Obtain an authentic copy of Alice's public key G'.
  2. Select a random $e \in \{0, 1\}^n$ with $w(e) = t$.
  3. A ciphertext of m is $r = mG' + e$.
- Decryption: To decrypt a ciphertext $r \in \{0, 1\}^n$ from Bob, Alice does
  1. Compute $r' = rP^{-1}$.
  2. Use a decoding algorithm, e.g. Patterson's, to correct r' to $c' \in \Gamma$ and decode c' to $m' \in \{0, 1\}^k$.
  3. The plaintext of r is $m = m'S^{-1}$.

In the McEliece original paper, the domain parameters were proposed as $(n, k, t) = (1024, 524, 50)$.

Correctness: We have $r' = rP^{-1} = (mSGP + e)P^{-1} = mSG + eP^{-1}$. As $mSG \in \Gamma$ and $w(eP^{-1}) = w(e) = t$, one can correct r' to $mSG = c'$ using a decoding algorithm. We can then obtain $m' = mS$ from $c' = (mS)G$. Therefore, in the last step, we obtain $m'S^{-1} = (mS)S^{-1} = m$, the plaintext.

Security: The security of McEliece cryptosystem is based on the problem of syndrome decoding. From $Hc^\top = 0$ and $r = c + e$, we have $Hr^\top = Hc^\top + He^\top = He^\top$. We call $s = Hr^\top = He^\top$ as the *syndrome* of r. The problem is stated as follows:

Computational Syndrome Decoding [18]: Given $H_{(n-k)\times n}$, $r \in \{0, 1\}^n$, and an integer t, find $e \in \{0, 1\}^n$ such that $Hr^\top = He^\top$ and $w(e) \leq t$.

This problem states a general approach for decoding an arbitrary linear code. In McEliece cryptosystem, we think of G' as a generator matrix of a random linear code, so the best attack is to solve the computational syndrome decoding problem. Note that a corresponding H' can be easily computed from G'. If the above problem can be solved efficiently, given H', r, and t, we can efficiently find e, obtain $mG' = r - e$, and recover the plaintext m. However, this problem is NP-hard. Specifically, the associated decision problem was proved to be NP-complete [3]. So, we believe that McEliece cryptosystem is secure.

Furthermore, McEliece cryptosystem is considered secure against quantum computers. It is stated in [18] that there is no connection between the decoding problem and Shor's algorithm, and Grover's algorithm cannot significantly speed-up the attack. These explain why code-based cryptosystems are quantum-resistant.

Example 2. Suppose Bob wants to send a message to Alice using McEliece cryptosystem. They first agree on $(n, k, t) = (15, 3, 3)$. Assume that Alice generates her keys by using $(15, 3)$-Goppa code Γ from Example 1, an invertible matrix S as shown below, and a permutation matrix P equivalent to one right cyclic shift. Thus, her public key G' is as shown below.

$$S = \begin{bmatrix} 1\,0\,1 \\ 0\,1\,1 \\ 1\,0\,0 \end{bmatrix} \qquad G' = SGP = \begin{bmatrix} 1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0 \\ 1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\ 1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1 \end{bmatrix}$$

To send $m = 101$ to Alice, Bob randomly selects $e = 000010010000010$ and encrypts m to $r = mG' + e = 000100011101001$. Upon receiving r, Alice computes $r' = rP^{-1} = 001000111010010$ and corrects r' to $c' = 001100011010110$ which corresponds to $m' = 001$. Finally, she obtains $m = m'S^{-1} = 101$.

For an eavesdropper Eve who knows G' and r, she cannot distinguish G' from a random matrix of rank k. Thus, she has to solve the computational syndrome decoding problem in order to obtain the plaintext m. □

We note that the described McEliece cryptosystem is not secure against chosen-ciphertext attacks. For example, if an adversary Eve modifies r in Example 2 by flipping last two bits, i.e. $\widetilde{r} = 000100011101010$, this $\widetilde{r}$ can be considered as a ciphertext of m with $e = 000010010000001$. When Eve asks Alice to decrypt $\widetilde{r}$, she will obtain m which breaks the cryptosystem. To prevent chosen-ciphertext attacks, many techniques were proposed. The main idea is to scramble m in order to destroy relations of two dependent messages. When these techniques are applied, one can use the public key G' in standard form $[\, I_k \mid A_{k\times(n-k)} \,]_{k\times n}$. Hence, we can use A as a public key, reducing the public-key size from kn to $k(n-k)$ bits.

2.4 Niederreiter Cryptosystem

Niederreiter cryptosystem is one variant of McEliece cryptosystem proposed in [17]. It is described as follows:

- Domain parameters: n, k, t
- Key generation: Alice does
  1. Construct an (n, k)-Goppa code Γ with error-correcting capability t where $\alpha_1, \ldots, \alpha_n$, and g are randomly chosen. Let $H_{(n-k)\times n}$ be a parity-check matrix of Γ.
  2. Select a random invertible matrix $S_{(n-k)\times(n-k)}$ and a random permutation matrix $P_{n\times n}$.
  3. Alice's public key is $H'_{(n-k)\times n} = SHP$. Her private key is (S, H, P).
- Encryption: To encrypt a message $m \in \{0,1\}^n$ with $w(m) = t$ to Alice, Bob does
  1. Obtain an authentic copy of Alice's public key H'.
  2. A ciphertext of m is $s = H'm^\top$.
- Decryption: To decrypt a ciphertext $s \in \{0,1\}^{(n-k)}$ from Bob, Alice does
  1. Use linear algebra to find $r' \in \{0,1\}^n$ such that $Hr'^\top = S^{-1}s$.
  2. Use a decoding algorithm, e.g. Patterson's, to correct r' to $c' \in \Gamma$. Let $e' = r' - c'$.
  3. The plaintext of s is $m = e'(P^\top)^{-1}$.

Correctness: We consider m as an error vector of weight t and s as its syndrome. We have $S^{-1}s = S^{-1}(SHPm^\top) = HPm^\top$. Thus, for $r' \in \{0,1\}^n$ that $Hr'^\top = S^{-1}s = HPm^\top$, we have $H(r'^\top - Pm^\top) = 0$ which implies $(r'^\top - Pm^\top)^\top = r' - mP^\top \in \Gamma$. Since $w(mP^\top) = t$, when correcting r' to the unique $c' = r' - mP^\top$, the error vector is $e' = r' - c' = mP^\top$. Therefore, in the last step, we obtain $e'(P^\top)^{-1} = (mP^\top)(P^\top)^{-1} = m$, the plaintext.

Security: It is shown in [12] that the security of Niederreiter cryptosystem is equivalent to that of McEliece cryptosystem. Thus, in this work, we will mainly focus on the attacks on McEliece cryptosystem.

3 Bernstein-Lange-Peters' Attack

Bernstein, Lange, and Peters presented an attack on McEliece cryptosystem which breaks the original proposed parameters in reasonable amount of time. This attack is based on Stern's attack for finding low-weight codewords in a linear code. We first consider how the low-weight codewords finding problem can be applied to decrypt a ciphertext, and then describe Stern's and Bernstein-Lange-Peters's attacks.

3.1 Decryption by Finding Low-Weight Codewords

Suppose we want to find m from the generator matrix G' and the ciphertext r. Let C denote a linear code generating by G', $\widehat{c} = mG'$, and $r = \widehat{c} + e$ (so $r \notin C$ which implies $r \neq 0$). We define $C_r = \{r + c : c \in C\}$. Note that $r + c = r - c$ for vectors over GF(2). We construct a new linear code $\mathcal{C} = C \cup C_r$, and consider the weight of each codeword in $\mathcal{C}$ as follows:

- $w(0) = 0$.
- For $c \in C$ and $c \neq 0$, we have $w(c) > t$ since $w(C) \geq 2t + 1$.
- $w(r + \widehat{c}) = w(e) = t$.
- For $c \in C$ and $c \neq \widehat{c}$, we have $d(r, c) > t$ and $w(r + c) = d(r, c) > t$.

We see that the only codeword in $\mathcal{C}$ that has weight t is e. Hence, the attack is as follows:

1. Construct $\mathcal{C}$. A generator matrix $\mathcal{G}$ for $\mathcal{C}$ can be created by appending r as a last new row of G'.
2. Find the unique codeword e in $\mathcal{C}$ with $w(e) = t$.
3. Correct r to $\widehat{c} = r - e \in C$, and decode $\widehat{c}$ to m.

Example 3. Suppose, in Example 2, Eve wants to find m from G' and r. She first constructs $\mathcal{G}$ as

$$\mathcal{G} = \begin{bmatrix} \\ G' \\ \\ -\,r\,- \end{bmatrix} = \begin{bmatrix} 1\,1\,0\,1\,0\,0\,0\,1\,0\,1\,1\,1\,1\,0\,0 \\ 1\,0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\ 1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1 \\ 0\,0\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1 \end{bmatrix}.$$

All codewords in $\mathcal{C}$ (computed from $\mathcal{G}$) are

$$\begin{aligned} \mathcal{C} = \{&000000000000000, 000010010000010, 000100011101001, 000110001101011, \\ &011001111000001, 011011101000011, 011101100101000, 011111110101010, \\ &101001110010100, 101011100010110, 101101101111101, 101111111111111, \\ &110000001010101, 110010011010111, 110100010111100, 110110000111110\}, \end{aligned}$$

and the unique codeword e with $w(e) = 3$ is $e = 000010010000010$. Thus, Eve can compute $\widehat{c} = r - e = 000110001101011 \in C$ which corresponds to $m = 101$. □

3.2 Stern's Attack

When $|\mathcal{C}|$ is large, enumerating all codewords in $\mathcal{C}$ is not an efficient approach. In 1988, Stern published a probabilistic algorithm which finds a codeword in $\mathcal{C}$ of weight t. This algorithm receives a parity-check matrix $\mathcal{H}$ of $\mathcal{C}$ as input. From $\mathcal{H}c^\top = 0$ for all $c \in \mathcal{C}$, we can write

$$\mathcal{H} = \begin{bmatrix} | & | & & | \\ h_1 & h_2 & \cdots & h_n \\ | & | & & | \end{bmatrix}_{(n-k)\times n} \qquad c = \begin{bmatrix} c_1\ c_2 \cdots c_n \end{bmatrix}_{1\times n} \qquad \mathcal{H}c^\top = \sum_{i=1}^{n} c_i h_i = 0.$$

Thus, finding a codeword in $\mathcal{C}$ of weight t is equivalent to finding t columns of $\mathcal{H}$ whose sum is zero. Stern's attack tries to search for these t columns of $\mathcal{H}$ by partitioning columns of $\mathcal{H}$ into four sets: X, Y, Z, W and guessing that p columns in X, p columns in Y, and $t - 2p$ columns in W form t columns in $\mathcal{H}$ whose sum is zero. Here, p is an algorithm parameter that can be adjusted. We assume that t is a small integer which is true for our settings, e.g. the original paper proposed $t = 50$.

Stern's attack can be described as follows:

1. Select $n-k$ columns of $\mathcal{H}$ at random. From these $n-k$ columns, select ℓ columns to be in Z, and the remaining $n-k-\ell$ columns to be in W. Here, ℓ is another algorithm parameter.
2. Partition the remaining k columns of $\mathcal{H}$ not in Z nor W into X and Y. Each column joins X and Y independently and with equal probability. Thus, it is possible that $|X| \neq |Y|$.
3. Convert $n-k$ columns in Z and W into the identity matrix, i.e. reduced row echelon form, using elementary row operations. If this fails, restart the algorithm from Step 1. Now, each column in Z and W will consist of only one 1. There will be ℓ rows corresponding to all 1s in Z.
4. For each subset $A \subseteq X$ of size p, compute the sum of all columns in A for ℓ rows from Step 3. The result is a column vector $\pi(A)$ of size ℓ. Store $[\pi(A), A]$ in a table sorted by the first component.
5. For each subset $B \subseteq Y$ of size p:
   5.1 Compute a column vector $\pi(B)$ of size ℓ by the same way.
   5.2 Search for $\pi(B)$ in the table. (We say that $\pi(B)$ matches $[\pi(A), A]$ if $\pi(A) = \pi(B)$.)
   5.3 For each match $[\pi(A), A]$ in the table,
      5.3.1 Compute the sum of $2p$ columns in A and B. The result is a column vector v of size $n-k$.
      5.3.2 If $w(v) = t-2p$, select the subset $W' \subseteq W$ of size $t-2p$ corresponding to each 1 in v. Output a codeword c of weight t corresponding to t columns in $A \cup B \cup W'$ and STOP.
6. If no solution is found, restart the algorithm from Step 1.

To avoid a restart in Step 3, one can perform row operations after selecting each column for Z and W in Step 1, and select the next column which ensures that the identity matrix can still be formed. This could bias the selection of (X, Y, Z, W) but it is mentioned that the bias does not noticably affect the performance.

Example 4. We apply Stern's attack to find a codeword in $\mathcal{C}$ of weight $t = 3$ from Example 3. Suppose we use $p = 1$ and $\ell = 5$. First, we find $\mathcal{H}$ for $\mathcal{C}$ from $\mathcal{G}$, and then randomly select X, Y, Z, W as follows:

$$\mathcal{H} = \begin{bmatrix} 1&1&0&1&1&0&0&1&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&1&0&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&1&0 \\ 1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&1&0&0&0&0&0 \\ 1&1&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&1&0&1&0&0&0&0&0&0&0&0&0&0&1 \\ 1&1&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&0&0&0 \end{bmatrix} = \begin{bmatrix} | & | & | & | & | & | & | & | & | & | & | & | & | & | \\ h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 & h_{10} & h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ | & | & | & | & | & | & | & | & | & | & | & | & | & | \end{bmatrix}$$

$$Z = \{h_1, h_4, h_9, h_{12}, h_{15}\}$$
$$W = \{h_3, h_6, h_7, h_{10}, h_{13}, h_{14}\}$$
$$X = \{h_5, h_{11}\}$$
$$Y = \{h_2, h_8\}$$

Next, we perform row operations on $\mathcal{H}$ so that columns in Z and W form I_{11}. All 1s in Z are underlined.

$$\mathcal{H}' = \begin{bmatrix} \underline{1}&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&1&1&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&1&0&\underline{1}&1&0&0&1&0&0&1&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&1&0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&1&\underline{1}&0&1&0&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&1&0&0&0&0 \\ 0&1&0&0&1&0&0&1&0&0&1&\underline{1}&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&1&0&0&1&0&0&0&\underline{1} \end{bmatrix}$$

We consider all subsets of X and Y of size $p = 1$: $A_1 = \{h_5\}$, $A_2 = \{h_{11}\}$, $B_1 = \{h_2\}$, $B_2 = \{h_8\}$, with

$$\pi(A_1) = \begin{bmatrix}0\\1\\1\\1\\1\end{bmatrix} \qquad \pi(A_2) = \begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} \qquad \pi(B_1) = \begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix} \qquad \pi(B_2) = \begin{bmatrix}0\\1\\1\\1\\1\end{bmatrix}.$$

We found a match $\pi(A_1) = \pi(B_2)$, so we compute $v = h_5 + h_8 = \begin{bmatrix}0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\end{bmatrix}^\top$. Since $w(v) = 1 = t - 2p$ and h_{14} has 1 in its tenth row, we select $W' = \{h_{14}\}$. Therefore, we have $A_1 \cup B_2 \cup W' = \{h_5, h_8, h_{14}\}$ and obtain $c = 000010010000010 \in \mathcal{C}$ with $w(c) = 3$, matches with the result in Example 3. □

Analysis: We consider the probability that a codeword is found in each round of the algorithm. Let

- e_1 denote the event that $X \cup Y$ (of size k) has exactly $2p$ columns corresponding to the error,
- e_2 denote the event that, given e_1, X has exactly p columns corresponding to the error, and
- e_3 denote the event that, given e_1, W has exactly $t - 2p$ columns corresponding to the error.

The probability of each event is

$$\Pr(e_1) = \frac{\binom{t}{2p}\binom{n-t}{k-2p}}{\binom{n}{k}} \qquad \Pr(e_2) = \frac{\binom{2p}{p}}{2^{2p}} \qquad \Pr(e_3) = \frac{\binom{n-k-(t-2p)}{\ell}}{\binom{n-k}{\ell}},$$

and the probability of the event $e = e_1 \cap e_2 \cap e_3$ is $\Pr(e) = \Pr(e_1) \cdot \Pr(e_2) \cdot \Pr(e_3)$. Thus, the expected number of rounds before the algorithm finds a codeword is $1/\Pr(e)$. Note that this is much better than the brute force algorithm which runs $\binom{n}{t}$ rounds on average.

Next, we consider the number of bit operations taken in each round of the algorithm.

- We assume that Step 1 and 2 take negligible time.
- The row operations in Step 3, i.e. Gauss-Jordan elimination, are performed for $n-k$ pivots. For the first pivot, $n-1$ entries of the first row are added to at most $n-k$ rows. For the second pivot, $n-2$ entries of the second row are added to at most $n-k$ rows, and so on. So, the number of bit operations is at most

$$\sum_{i=1}^{n-k}(n-i)(n-k) = (1/2)(n-k)^3 + (k-1/2)(n-k)^2.$$

- In Step 4, the number of subsets $A \subseteq X$ is approximately $\binom{k/2}{p}$ and adding ℓ rows of all columns in A takes roughly ℓp bit operations. The same is applied for subsets $B \subseteq Y$ in Step 5.1. Hence, computing $\pi(A)$ and $\pi(B)$ for all possible A and B takes $2\ell p\binom{k/2}{p}$ bit operations.
- Assume that π has a uniform distribution. For each B, the probability that $\pi(B)$ is in the table is $\binom{k/2}{p}/2^\ell$, and there are $\binom{k/2}{p}$ such B. Thus, the expected number of matches in Step 5.3 is $\binom{k/2}{p}^2/2^\ell$.
- For each match, all $n-k$ entries of $2p$ columns are added together. So, the number of bit operations in Step 5.3.1 is $2p(n-k)\binom{k/2}{p}^2/2^\ell$.

Therefore, the total number of bit operations taken in each round is approximately

$$[(1/2)(n-k)^3 + (k-1/2)(n-k)^2] + 2\ell p\binom{k/2}{p} + 2p(n-k)\binom{k/2}{p}^2/2^\ell.$$

Example 5. We apply the results of the analysis to the original proposed parameters $(n, k, t) = (1024, 524, 50)$ with $(p, \ell) = (2, 18)$. We have

- $\Pr(e) \approx 1.899 \cdot 10^{-12}$,
- the expected number of rounds is $5.265 \cdot 10^{11}$,
- the total number of bit operations taken in each round is $2.754 \cdot 10^6$,

and hence the total number of bit operations taken by the algorithm is approximately

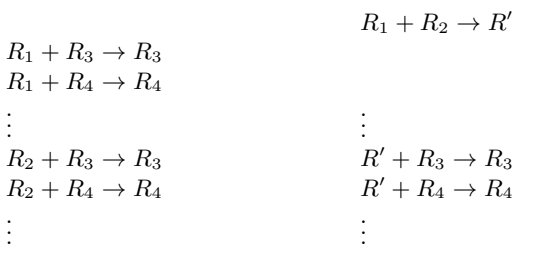
$$(5.265 \cdot 10^{11})(2.754 \cdot 10^6) \approx 1.450 \cdot 10^{18} \approx 2^{60.331}$$

which is considered feasible. Note that the brute force attack will take $\binom{1024}{50} \approx 2^{284}$ bit operations. □

3.3 Bernstein-Lange-Peters' Attack

The attack presented by Bernstein, Lange, and Peters is based on Stern's attack. It improved Stern's attack in various aspects as listed below.

1. Reusing existing pivots. When the algorithm restarts in Step 6, instead of performing row operations from the original values in $\mathcal{H}$, we can start row operations from the result of the previous round. When selecting $n-k$ columns in Step 1, the chance that each column is already reduced is $(n-k)/n$, so around $(n-k)^2/n$ columns are reduced on average. Thus, the number of columns needs to be pivoted is $(n-k)-(n-k)^2/n = (k/n)(n-k)$. Also, we can make an assumption that, on average, half of $n-k$ rows are modified in the row operations for each pivot. This leads to the improved number of bit operations as $k^2(n-k)(n-k-1)(3n-k)/4n^2$.
2. Forcing more existing pivots. When selecting $n-k$ columns in Step 1 after the restart, one can randomly choose only c new columns and reuse $n-k-c$ columns of the identity matrix from the previous round. Here, c is another algorithm parameter. When $c < (k/n)(n-k)$, the number of bit operations is improved. The authors mentioned that if c is too small, there will be a dependence between two rounds, which in turn increases the total number of rounds. The number of rounds is maximized when $c = 1$.
3. Faster pivoting. Consider an example of pivoting in Fig. 2. In Fig. 2(a), when we pivot the first column, we add Row 1 to other rows, including Row 3 and 4. When pivoting the second column, we add Row 2 to other rows, including Row 3 and 4. To reduce the number of additions in this case, we can first compute the summation of Row 1 and 2, called R', and then add R' to Row 3 and 4, which reduces the number of additions from 4 to 3.



(a) Normal pivoting (b) Faster pivoting

Fig. 2. An example of faster pivoting

This can be generalized as follows: when pivoting r columns and r rows are to be added to other rows, we compute all $2^r - 1$ sums of nonempty subsets of these r rows in advance. Here, r is another algorithm parameter. Then, we add the corresponding precomputed value to each row when pivoting. For each row, the expected number of times that it is modified is reduced from $r/2$ to $1 - 1/2^r$. The authors stated that the optimal value of r depends on the value of c and is around $\log(n-k) - \log\log(n-k)$.

4. Multiple choices of Z. The attack tries several Zs in each round. After $n-k$ columns are converted to the identity matrix, the algorithm tries m disjoint sets $Z_1, Z_2, \ldots, Z_m$ as Z in Stern's attack with the same X and Y. Here, m is another algorithm parameter. The benefits of this are that the Gauss-Jordan elimination is performed once but many Zs can be tried, and the probability that the algorithm finds a codeword in each round increases by a factor of nearly m.
5. Reusing additions of the ℓ-bit vectors. In Step 4 and 5.1, we add p vectors in A and B to obtain $\pi(A)$ and $\pi(B)$. Suppose $A_1 = \{h_1, h_2, h_3, h_4\}$ and $A_2 = \{h_1, h_2, h_3, h_5\}$. When computing $\pi(A_1)$, we compute $h_1 + h_2 + h_3$ which is also a part of $\pi(A_2)$. So, the number of bit operations for computing $\pi(A)$ for each A is reduced from ℓp to roughly ℓ. This caching technique becomes more important when p is large.
6. Faster additions after collisions. For each match $\pi(A) = \pi(B)$, we add all columns in A and B and check if the weight is $t - 2p$. It is possible that columns in $A \cup B$ overlap with previous calculations, and those can be used to reduce the number of bit operations. Another point is we can stop the addition when we find that the weight of the result is at least $t - 2p + 1$. This happens after adding $2(t - 2p + 1)$ rows on average. Hence, the expected number of bit operations for each match is smaller than $2p(n-k)$.

Analysis: First, we recall all parameters of the attack:

- p: the number of columns in X and Y corresponding to the error.
- m: the number of Zs tried in each round.
- ℓ: the size of each Z.
- c: the number of new columns to be pivoted.
- r: the number of rows for faster pivoting.

We use the same notation as in the analysis of Stern's attack. In Bernstein-Lange-Peters' attack, there are some modifications on the probabilities.

- $\Pr(e_1)$ is the same.
- This attack constructs X by choosing exactly $\lfloor k/2 \rfloor$ columns at random instead of having each column decides independently. This gives $\Pr(e_2) = \frac{\binom{\lfloor k/2 \rfloor}{p}\binom{\lceil k/2 \rceil}{p}}{\binom{k}{2p}}$.
- The event e_3 is equivalent to the event that Z has no columns corresponding to the error. When using m disjoint Zs, the probability that at least one of Zs has no columns corresponding to the error, from $t - 2p$ columns, is (by inclusion-exclusion principle)

$$\Pr(e_3) = \sum_{i=1}^{m} (-1)^{i-1} \binom{m}{i} \frac{\binom{n-k-(t-2p)}{i\ell}}{\binom{n-k}{i\ell}}.$$

The authors noted that $\Pr(e_1) \cdot \Pr(e_2) \cdot \Pr(e_3)$ is the probability of success only for the first round. This is because each round depends on the previous round by the parameter c. Here, the analysis can be done by constructing a Markov chain with t states: state i denotes the event that there are i errors in $X \cup Y$. From state i, there are transitions to states $i - c, \ldots, i + c$, as we introduce c new columns to be pivoted.

Experimental Results: When choosing parameters $p = 2$, $m = 2$, $\ell = 20$, $c = 7$, and $r = 7$, the attack breaks the original parameters $(n, k, t) = (1024, 524, 50)$ in $2^{60.55}$ bit operations: $4.21 \cdot 10^{11}$ rounds with $4 \cdot 10^6$ bit operations per round. The best attack at that time by Canteaut, Chabaud, and Sendrier takes $2^{64.1}$ bit operations: $9.85 \cdot 10^{11}$ rounds with $20 \cdot 10^6$ bit operations per round – almost $12\times$ improvement.

The software implementation spends nearly 2^{58} CPU cycles or 1400 days on a single computer. The attack can be parallelized on more machines with no communications between them. Performing the attack on 200 such computers reduces the time to just one week. The attack by Canteaut, Chabaud, and Sendrier was reported to spend 2^{68} CPU cycles or 220000 days (7400000 days at the time when their work was published in 1998). The $150\times$ speed up factor is a result of this attack with low-level CPU cycles optimization.

Countermeasure: Two countermeasures were presented to prevent the attack.

1. Increasing n: The attack takes more time to search for the codeword as n grows.
2. Increasing t: For a Goppa code with error-correcting capability t, we cannot use an error vector whose weight is more than t if we are decoding it in a traditional way. However, the list decoding, presented in [4], can decode even when the weight of an error vector is more than t. It is applicable for an error of weight roughly $n - \sqrt{n(n - 2t - 2)}$, which is at least $t + 1$. When the weight of an error increases, the attack takes more time on the search.

To strengthen McEliece cryptosystem, new parameters were also proposed. These parameters are for the variants that prevent chosen-ciphertext attacks, where the public-key size is $k(n - k)$ bits.

Table 1. Proposed parameters for strengthening McEliece cryptosystem

Security level	n	k	t	**Error-correcting capability by list decoding**	**Public-key size $k(n-k)$ bits**
80	1632	1269	33	34	460647
128	2960	2288	56	57	1537536
256	6624	5129	115	117	7667855

Another set of parameters were proposed for McEliece cryptosystem with restricted key size.

Table 2. Proposed parameters for strengthening McEliece cryptosystem with restricted key size

Key size (bytes)	n	t	Error-correcting capability by list decoding	Security level
2^{16}	1744	35	36	84.88
2^{17}	2480	45	46	107.41
2^{18}	3408	67	68	147.94
2^{19}	4624	95	97	191.18
2^{20}	6960	119	121	266.94

4 State-of-the-art Research

Many attacks and implementations of McEliece cryptosystem were published after the attack by Bernstein et al. In this section, we look into some recent works, including McBits and Classic McEliece.

4.1 Recent Attacks

In Stern's attack, the distribution of columns corresponding to the error in X, Y, Z, W is assumed to be $(p, p, 0, t-2p)$. This distribution can be adjusted to achieve a better result. In 2011, Bernstein, Lange, and Peters presented ball-collision decoding in [8] which speeds up Stern's attack. For instance, the ball-collision decoding breaks McEliece cryptosystem with parameters $(n, k, t) = (6624, 5129, 117)$ using $2^{254.15}$ bit operations while their attack in 2008 takes around 2^{256} bit operations as shown in Table 1. Here, the distribution is assumed to be $(p, p, q, q, t-2p-2q)$.

Another improvement of Stern's attack was presented in 2011 by Johansson and Löndahl [11]. Their work assumes the distribution as $(2p, 0, 0, t-2p)$, and takes only $2^{247.29}$ bit operations with parameters $(n, k, t) = (6624, 5129, 117)$. Fig. 3 from [11] compares the assumed distribution of errors for some attacks in literature. The figure uses w to denote the error-correcting capability t.

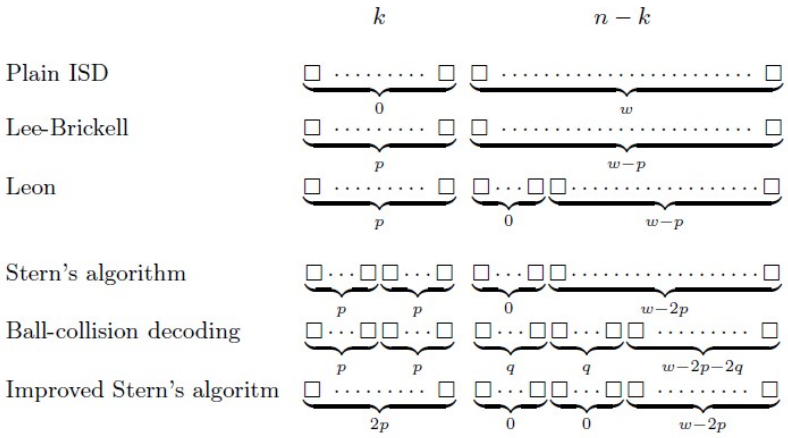


Fig. 3. The assumed distribution of errors for some attacks [11]

When considering the asymptotic number of bit operations taken by the attacks, Stern's attack takes $O(2^{0.0557n})$ bit operations and ball-collision decoding takes $O(2^{0.0556n})$ bit operations. There are many recent works trying to improve this number. In 2011, May, Meurer, and Thomae [13] presented an attack whose running-time is $O(2^{0.0536n})$ bit operations. One year later, another attack by Becker, Joux, May, and Meurer [2] was published, using $O(2^{0.0494n})$ bit operations. The most recent attack is the work of May and Ozerov [14] presented in 2015 which takes $O(2^{0.0473n})$ bit operations. Note that the reduced time complexity is obtained at the expense of an increased memory complexity.

4.2 Recent Implementations

McBits [6]: This implementation was proposed in 2013 by Bernstein, Chou, and Schwabe. It is based on Niederreiter cryptosystem. Here, they reduced the decryption time significantly comparing to previous works. They also claimed that their implementation is fully protected against timing attacks. Some components of their implementation are an additive FFT for fast root computation, a transposed additive FFT for fast syndrome computation, and a sorting network to avoid cache-timing attacks. As a result, they achieved decryption time of

- 26544 cycles for 87-bit security, while the previous work spends 288649 cycles,
- 60493 cycles for 128-bit security, while the previous work spends 540952 cycles.

Classic McEliece [5]: This cryptosystem is one candidate for NIST post-quantum cryptography standard submitted by Bernstein et al. The first submission was made in 2017, and the modified version was submitted in 2019. The authors stated that Classic McEliece is designed as a key encapsulation mechanism and claimed that it is secure against chosen-ciphertext attacks with very high security level and is quantum-resistant. Classic McEliece is based on Niederreiter cryptosystem. The cryptosystem is briefly described below, and its proposed parameters are listed with their security levels in Table 3.

- Let H denote a cryptographic hash function.
- Key generation: Alice does
  1. Select g, α_1, …, α_n at random.
  2. Construct a parity-check matrix $H_{(n-k)\times n}$ of a form $[\, I_{n-k} \,|\, T_{(n-k)\times k} \,]$ of a Goppa code.
  3. Select a random $s \in \{0,1\}^n$.
  4. Alice's public key is T. Her private key is $(s, g, \alpha_1, \ldots, \alpha_n)$.
- Encryption: To encrypt a message $m \in \{0,1\}^n$ with $w(m) = t$ to Alice, Bob does
  1. Obtain an authentic copy of Alice's public key T.
  2. Construct $H = [\, I_{n-k} \,|\, T \,]$
  3. A ciphertext of m is $c_0 = Hm^\top$.
- Decryption: To decrypt a ciphertext $c_0 \in \{0,1\}^{(n-k)}$ from Bob, Alice does
  1. Extend c_0 to $v = [c_0, 0, \ldots, 0] \in \{0,1\}^n$ by appending k zeros.
  2. Find the unique codeword c such that $d(c, v) \leq t$. If none exists, return $\perp$.
  3. Compute $m = v + c$.
  4. If $w(m) = t$ and $c_0 = Hm^\top$, return m. Otherwise, return $\perp$.
- Encapsulation: To establish a shared session key with Alice, Bob does
  1. Obtain an authentic copy of Alice's public key T.
  2. Select $e \in \{0,1\}^n$ with $w(e) = t$ at random.
  3. Encrypt e and obtain c_0.
  4. Compute $c_1 = \mathsf{H}(2, e)$ and construct $c = (c_0, c_1)$. Send c to Alice.
  5. Compute a session key $k = \mathsf{H}(1, e, c)$.
- Decapsulation: After receiving $c = (c_0, c_1)$ from Bob, to recover the session key k, Alice does
  1. Set $b \leftarrow 1$.
  2. Decrypt c_0 to obtain e. If the decryption returns $\perp$, set $e \leftarrow s$ and $b \leftarrow 0$.
  3. Compute $c_1' = \mathsf{H}(2, e)$. If $c_1' \neq c_1$, set $e \leftarrow s$ and $b \leftarrow 0$.
  4. Compute the session key $k = \mathsf{H}(b, e, c)$.

Table 3. Proposed parameters of Classic McEliece

Parameter set	n	k	t	Length of a hash digest	Security level
`mceliece348864`	3488	2720	64	256	≥ 128
`mceliece460896`	4608	3360	96	256	≥ 192
`mceliece6688128`	6688	5024	128	256	≥ 256
`mceliece6960119`	6960	5413	119	256	≥ 256
`mceliece8192128`	8192	6528	128	256	≥ 256

4.3 Comparing McEliece Cryptosystem with Other Quantum-Resistant Cryptosystems

NIST discussed some advantages and limitations of Classic McEliece in [1]. It is stated that the size of the ciphertext of Classic McEliece is very small (around 200 bytes) and the performance of encapsulation and decapsulation process is good. However, the main drawback is that the public-key size is very large. Below are figures from [16] that compares candidates of NIST post-quantum cryptography standardization process when considering security level of 128 bits.

One can see that, for the cryptosystems based on McEliece cryptosystem, the public-key sizes are at least 32× larger than that of other candidates but their ciphertexts are the smallest among them. When comparing the speed, Classic McEliece (`McEliece348864` and `McEliece348864f`) take longest time on key generation, but small time on encryption and decryption. It is mentioned in [5] that applications will benefit from Classic McEliece if key generation and distribution are not done so often and most of their operations are encryption, decryption, and sending ciphertexts through the network.

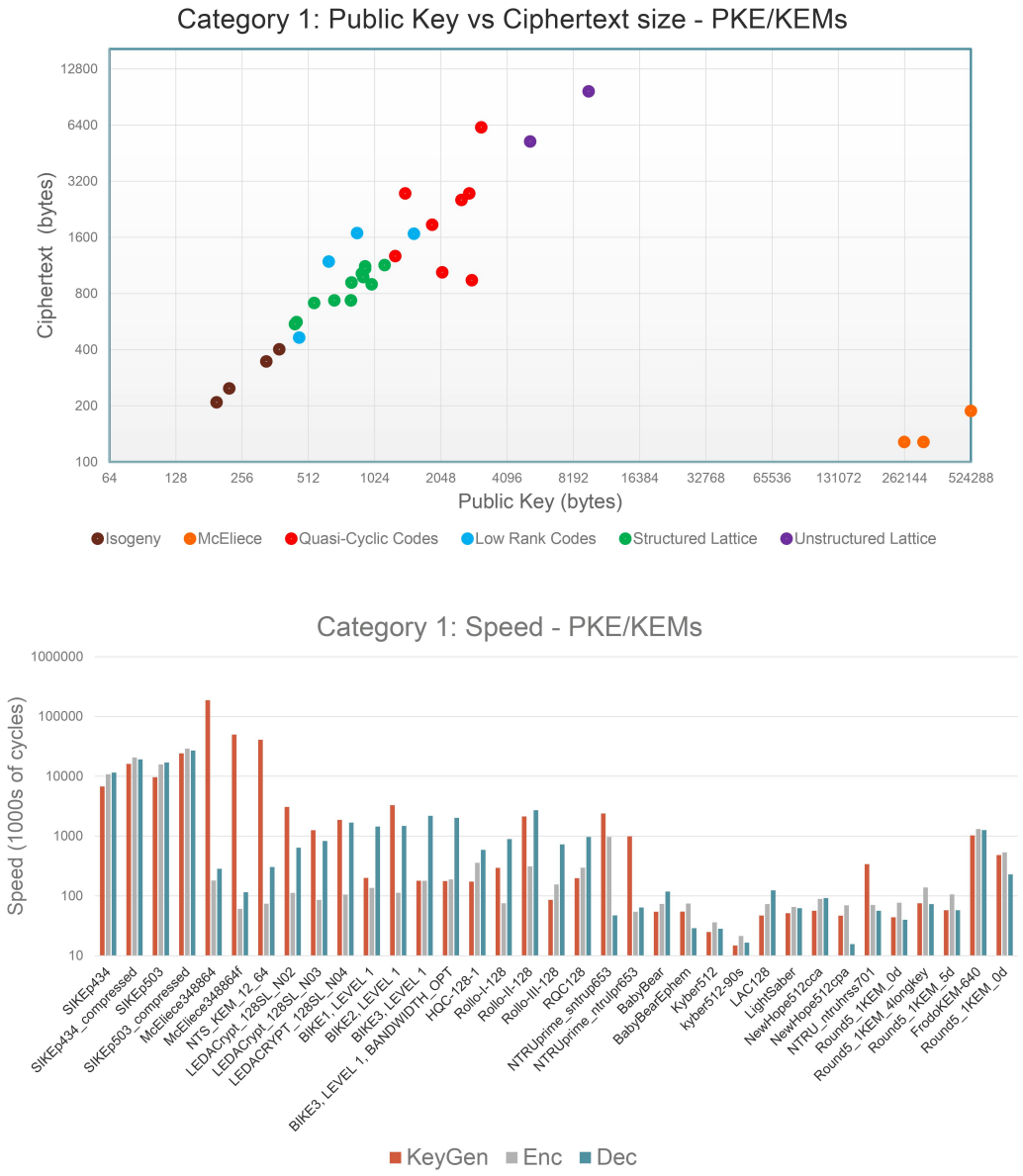


Fig. 4. A comparison between candidates of NIST PQC standardization process [16]

5 Conclusion

We study McEliece cryptosystem, the first code-based public-key cryptosystem, together with one of the attack by Bernstein, Lange, and Peters. Their attack is based on Stern's attack for finding low-weight codewords of a linear code. Their implementation breaks the original proposed parameters using reasonable amount of time, but increasing the sizes of parameters can prevent the attack. We also discuss some recent attacks and implementations based on McEliece cryptosystem. As the problem of decoding random linear codes is NP-hard, and Shor's nor Grover's quantum algorithm cannot significantly speed-up the attack, code-based cryptosystem can be considered as a good candidate for quantum-resistant cryptosystem.

The lesson learned from this study is that attacks only get better. McEliece cryptosystem was considered secure when it was proposed in 1978. However, as time passed, more efficient attacks were discovered and the original proposed parameters are not secure anymore. This is one reason that the thorough analysis is needed before cryptographic standards can be established.

References

1. Alagic, G., Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.K., Miller, C., Moody, D., Peralta, R., et al.: Status report on the first round of the NIST post-quantum cryptography standardization process. US Department of Commerce, National Institute of Standards and Technology (2019)
2. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How $1+1=0$ improves information set decoding. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 520–536. Springer (2012)
3. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory **24**(3), 384–386 (1978)
4. Bernstein, D.J.: List decoding for binary goppa codes. In: International Conference on Coding and Cryptology. pp. 62–80. Springer (2011)
5. Bernstein, D.J., Chou, T., Lange, T., Misoczki, R., Niederhagen, R., Persichetti, E., Schwabe, P., Szefer, J., Wang, W.: Classic mceliece: conservative code-based cryptography 30 march 2019 (2019)
6. Bernstein, D.J., Chou, T., Schwabe, P.: Mcbits: fast constant-time code-based cryptography. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 250–272. Springer (2013)
7. Bernstein, D.J., Lange, T., Peters, C.: Attacking and defending the mceliece cryptosystem. In: International Workshop on Post-Quantum Cryptography. pp. 31–46. Springer (2008)
8. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Annual Cryptology Conference. pp. 743–760. Springer (2011)
9. Canteaut, A., Chabaud, F.: 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)
10. Canteaut, A., Sendrier, N.: Cryptanalysis of the original mceliece cryptosystem. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 187–199. Springer (1998)
11. Johansson, T., Löndahl, C.: An improvement to stern's algorithm. Report, Lund University p. 115 (2011)
12. Li, Y.X., Deng, R.H., Wang, X.M.: On the equivalence of mceliece's and niederreiter's public-key cryptosystems. IEEE Transactions on Information Theory **40**(1), 271–273 (1994)
13. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $O(2^{0.054n})$. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 107–124. Springer (2011)
14. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 203–228. Springer (2015)
15. McEliece, R.J.: A public-key cryptosystem based on algebraic. Coding Thv **4244**, 114–116 (1978)
16. Moody, D.: The 2nd round of the nist pqc standardization process (2019), https://csrc.nist.gov/CSRC/media/Presentations/the-2nd-round-of-the-nist-pqc-standardization-proc/images-media/moody-opening-remarks.pdf, [Online; accessed 27-April-2020]
17. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Prob. Control and Inf. Theory **15**(2), 159–166 (1986)
18. Overbeck, R., Sendrier, N.: Code-based cryptography. In: Post-quantum cryptography, pp. 95–145. Springer (2009)
19. Patterson, N.: The algebraic decoding of goppa codes. IEEE Transactions on Information Theory **21**(2), 203–207 (1975)
20. Stern, J.: A method for finding codewords of small weight. In: International Colloquium on Coding Theory and Applications. pp. 106–113. Springer (1988)