Revised Quantum Resistant Public Key Encryption Scheme RLCE and IND-CCA2 Security for McEliece Schemes

Yongge Wang
Department of SIS, UNC Charlotte, USA.
yongge.wang@uncc.edu

February 28, 2017

Abstract

Recently, Wang (2016) introduced a random linear code based quantum resistant public encryption scheme RLCE which is a variant of McEliece encryption scheme. In this paper, we introduce a revised version of the RLCE encryption scheme. The revised RLCE schemes are more efficient than the original RLCE scheme. Specifically, it is shown that RLCE schemes have smaller public key sizes compared to binary Goppa code based McEliece encryption schemes for corresponding security levels. The paper further investigates message padding schemes for RLCE to achieve IND-CCA2 security. Practical RLCE parameters for the security levels of 128, 192, and 256 are recommended. Furthermore, we point out that the algorithm proposed by Sendrier (ISIT 2005) for encoding extra information symbols within error locations of McEliece encryption scheme is incorrect.

Software packages available at http://quantumca.org/

Key words: Random linear codes; McEliece encryption scheme; linear code based encryption scheme; message padding schemes; adaptive chosen ciphertext security.

MSC 2010 Codes: 94B05; 94A60; 11T71; 68P25

1 Introduction

With rapid development for quantum computing techniques, our society is concerned with the security of current Public Key Infrastructures (PKI) which are fundamental for Internet services. The core components for current PKI infrastructures are based on public cryptographic techniques such as RSA and DSA. However, it has been shown that these public key cryptographic techniques could be broken by quantum computers. Thus it is urgent to develop public key cryptographic systems that are secure against quantum computing.

Since McEliece encryption scheme [20] was introduced more than thirty years ago, it has withstood many attacks and still remains unbroken for general cases. It has been considered as one of the candidates for post-quantum cryptography since it is immune to existing quantum computer algorithm attacks. The original McEliece cryptographic system is based on binary Goppa codes. Several variants have been introduced to replace Goppa codes in the McEliece encryption scheme though most of them have been broken. Up to the writing of this paper, secure McEliece encryption schemes include MDPC/LDPC code based McEliece encryption schemes [1, 21], Wang's RLCE [30], and the original binary Goppa code based McEliece encryption scheme.

Recently, Wang's RLCE [30] presents a systematic approach of designing public key encryption schemes using any linear code. For example, one can use (generalized) Reed-Solomon codes to design McEliece based RLCE encryption scheme. Wang [30] used heuristics to show that the RLCE scheme is as secure as decoding random linear codes. The most powerful message recovery attacks (not key recovery attacks) on McEliece cryptosystem is the information-set decoding attack which was introduced by Prange [25]. Bernstein, Lange, and Peters [4] presented an exact complexity analysis on information-set decoding attacks against McEliece cryptosystem over binary linear codes. Peters [23] presented an exact complexity analysis on information-set decoding attacks against McEliece cryptosystem over $GF(p^m)$. Based on the exact complexity analysis of information-set decoding attacks, Wang [30] recommended example parameters for RLCE scheme.

In this paper, we propose a few variants of the RLCE scheme which will increase the message communication bandwidth, reduce the public key size, and improve the encryption and decryption performance. Experimental results are reported for different RLCE scheme parameter sizes. The paper will also analyze the security of RLCE scheme by investigating attacks on dual codes of RLCE public keys. We further investigate message padding schemes for RLCE to be secure against adaptive chosen ciphertext attacks (IND-CCA2).

Unless specified otherwise, we will use $q = p^m$ where $p = 2$ or p is a prime. Our discussion will be based on the field $GF(q)$ through out this paper. Bold face letters such as $\mathbf{a}, \mathbf{b}, \mathbf{e}, \mathbf{f}, \mathbf{g}$ are used to denote row or column vectors over $GF(q)$. It should be clear from the context whether a specific bold face letter represents a row vector or a column vector.

2 McEliece, Niederreiter, and RLCE Encryption schemes

For given parameters n, k and t, the McEliece scheme [20] chooses an $(n, k, 2t + 1)$ linear Goppa code C. Let G_s be the $k \times n$ generator matrix for the code C. Select a random dense $k \times k$ nonsingular matrix S and a random $n \times n$ permutation matrix P. Then the public key is $G = SG_sP$ and the private key is G_s. The following is a description of encryption and decryption processes.

`Mc.Enc`$(G, \mathbf{m}, \mathbf{e})$. For a message $\mathbf{m} \in \{0, 1\}^k$, choose a random vector $\mathbf{e} \in \{0, 1\}^n$ of weight t and compute the cipher text $\mathbf{c} = \mathbf{m}G + \mathbf{e}$

`Mc.Dec`$(S, G_s, P, \mathbf{c})$. For a received ciphertext $\mathbf{c}$, first compute $\mathbf{c}' = \mathbf{c}P^{-1} = \mathbf{m}SG$. Next use an error-correction algorithm to recover $\mathbf{m}' = \mathbf{m}S$ and compute the message $\mathbf{m}$ as $\mathbf{m} = \mathbf{m}'S^{-1}$.

For given parameters n, k, and t, the Niederreiter's scheme [22] chooses an $(n, k, 2t + 1)$ linear code C. Let H_s be an $(n - k) \times n$ parity check matrix of C. Select a random $(n - k) \times (n - k)$ nonsingular matrix S and a random $n \times n$ permutation matrix P. Then the public key is $H = SH_sP$ and the private key is S, H_s, P. The encryption and decryption processes are as follows.

`Nied.Enc`$(H, \mathbf{m})$. For a message $\mathbf{m} \in GF(q)^n$ of weight t, compute the cipher text $\mathbf{c} = \mathbf{m}H^T$ of length $n - k$.

`Nied.Dec`$(S, H_s, P, \mathbf{c})$. For a received ciphertext $\mathbf{c} = \mathbf{m}P^T H_s^T S^T$, compute $\mathbf{c}(S^T)^{-1} = \mathbf{m}P^T H_s^T$. Use an error-correction algorithm to recover $\mathbf{m}' = \mathbf{m}P^T$ and compute the message $\mathbf{m} = \mathbf{m}'(P^T)^{-1}$.

The protocol for the RLCE Encryption scheme by Wang [30] consists of the following three processes: `RLCE.KeySetup`, `RLCE.Enc`, and `RLCE.Dec`.

`RLCE.KeySetup`(n, k, d, t, r). Let $n, k, d, t > 0$, and $r \geq 1$ be given parameters such that $n-k+1 \geq d \geq 2t+1$. Let $G_s = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$ be a $k \times n$ generator matrix for an $[n, k, d]$ linear code such that there is an efficient decoding algorithm to correct at least t errors for this linear code given by G_s.

1. Let $C_0, C_1, \cdots, C_{n-1} \in GF(q)^{k \times r}$ be $k \times r$ matrices drawn uniformly at random and let

$$G_1 = [\mathbf{g}_0, C_0, \mathbf{g}_1, C_1 \cdots, \mathbf{g}_{n-1}, C_{n-1}] \tag{1}$$

be the $k \times n(r+1)$ matrix obtained by inserting the random matrices C_i into G_s.

2. Let $A_0, \cdots, A_{n-1} \in GF(q)^{(r+1)\times(r+1)}$ be dense nonsingular $(r+1) \times (r+1)$ matrices chosen uniformly at random and let $A = \text{diag}[A_0, \cdots, A_{n-1}]$ be an $n(r+1) \times n(r+1)$ nonsingular matrix.

3. Let S be a random dense $k \times k$ nonsingular matrix and P be an $n(r+1) \times n(r+1)$ permutation matrix.

4. The public key is the $k \times n(r+1)$ matrix $G = SG_1AP$ and the private key is (S, G_s, P, A).

`RLCE.Enc`$(G, \mathbf{m}, \mathbf{e})$. For a row vector message $\mathbf{m} \in GF(q)^k$, choose a random row vector $\mathbf{e} = [e_0, \ldots, e_{n(r+1)-1}] \in GF(q)^{n(r+1)}$ such that the Hamming weight of $\mathbf{e}$ is at most t. The cipher text is $\mathbf{c} = \mathbf{m}G + \mathbf{e}$.

`RLCE.Dec`$(S, G_s, P, A, \mathbf{c})$. For a received cipher text $\mathbf{c} = [c_0, \ldots, c_{n(r+1)-1}]$, compute

$$\mathbf{c}P^{-1}A^{-1} = \mathbf{m}SG_1 + \mathbf{e}P^{-1}A^{-1} = [c'_0, \ldots, c'_{n(r+1)-1}]$$

where $A^{-1} = \text{diag}[A^{-1}, \cdots, A^{-1}_{n-1}]$. Let $\mathbf{c}' = [c'_0, c'_{r+1}, \cdots, c'_{(n-1)(r+1)}]$ be the row vector of length n selected from the length $n(r+1)$ row vector $\mathbf{c}P^{-1}A^{-1}$. Then $\mathbf{c}' = \mathbf{m}SG_s + \mathbf{e}'$ for some error vector $\mathbf{e}' \in GF(q)^n$. Let $\mathbf{e}'' = \mathbf{e}P^{-1} = [e''_0, \cdots, e''_{n(r+1)-1}]$ and $\mathbf{e}''_i = [e''_{i(r+1)}, \ldots, e''_{i(r+1)+r}]$ be a sub-vector of $\mathbf{e}''$ for $i \le n-1$. Then $\mathbf{e}'[i]$ is the first element of $\mathbf{e}''_i A_i^{-1}$. Thus $\mathbf{e}'[i] \neq 0$ only if $\mathbf{e}''_i$ is non-zero. Since there are at most t non-zero sub-vectors $\mathbf{e}''_i$, the Hamming weight of $\mathbf{e}' \in GF(q)^n$ is at most t. Using the efficient decoding algorithm, one can compute $\mathbf{m}' = \mathbf{m}S$ and $\mathbf{m} = \mathbf{m}'S^{-1}$. Finally, calculate the Hamming weight $w = \texttt{weight}(\mathbf{c} - \mathbf{m}G)$. If $w \le t$ then output $\mathbf{m}$ as the decrypted plaintext. Otherwise, output error.

3 The dual RLCE scheme

It is straightforward to show that McEliece encryption scheme is equivalent to Niederreiter encryption scheme. That is, for each McEliece encryption scheme public key, one can derive a Niederreiter encryption scheme public key and, for each Niederreiter encryption scheme public key, one can derive a McEliece encryption scheme public key. One can break the Mcelience encryption scheme (respectively the Niederreiter encryption scheme) if and only if one can break the corresponding Niederreiter encryption scheme (respectively, the McEliece encryption scheme). In this section, we show that a similar equivalent result may not hold for RLCE schemes. We first try to give a natural candidate construction of Niederreiter RLCE scheme and show it is challenging (or infeasible) to design an efficient decryption algorithm. Thus it is not clear whether there exists an efficient equivalent Niederreiter RLCE encryption scheme corresponding to the McEliece RLCE encryption scheme.

`RLCEdual.KeySetup`(n, k, d, t, r). For an $(n, k, 2t+1)$ linear code C, let $H_s = [\mathbf{h}_0, \cdots, \mathbf{h}_{n-1}]$ be an $(n-k) \times n$ parity check matrix of C. The keys are generated using the following steps.

1. Let $C_0, C_1, \cdots, C_{n-1} \in GF(q)^{(n-k)\times r}$ be $(n-k) \times r$ matrices drawn uniformly at random and let

$$H_1 = [\mathbf{h}_0, C_0, \mathbf{g}_1, C_1 \cdots, \mathbf{h}_{n-1}, C_{n-1}] \tag{2}$$

be the $(n-k) \times n(r+1)$ matrix obtained by inserting the random matrices C_i into H_s.

2. Let $A_0, \cdots, A_{n-1} \in GF(q)^{(r+1)\times(r+1)}$ be dense nonsingular $(r+1) \times (r+1)$ matrices chosen uniformly at random and let $A = \text{diag}[A_0, \cdots, A_{n-1}]$ be an $n(r+1) \times n(r+1)$ nonsingular matrix.

3. Let S be a random dense $k \times k$ nonsingular matrix and P be an $n(r+1) \times n(r+1)$ permutation matrix.

4. The public key is the $(n-k)\times n(r+1)$ matrix $H = SH_1AP$ and the private key is (S, H_s, P, A).

$\texttt{RLCEdual.Enc}(H, \mathbf{m})$. For a row message $\mathbf{m} \in GF(q)^{n(r+1)}$ of weight t, compute the ciphertext $\mathbf{c} = \mathbf{m}H^T$.

Candidate decryption algorithms? For a received ciphertext $\mathbf{c} = \mathbf{m}H^T$, we have $\mathbf{c}(S^T)^{-1} = \mathbf{m}P^TA^TH_1^T$. Since the weight of $\mathbf{m}P^TA^T$ is at most $2t$, we can decrypt the ciphertext $\mathbf{c}$ only if we had an efficient $2t$-error-correcting algorithm for the code defined by the parity check matrix H_1. Since the matrices $C_0, C_1, \cdots, C_{n-1}$ are selected at random, it is unknown whether there is an efficient error correcting algorithm for the code defined by the parity check matrix H_1. In the following, we describe the natural candidate algorithm for decrypting the ciphertext and show that this algorithm will not work. Let $G_s = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$ be the $k\times n$ generator matrix for the linear code C such that $G_sH_s^T = 0$. Furthermore, let $D_0, D_1, \cdots, D_{n-1}$ be $k\times r$ matrices, such that $D_0C_0^T + D_1C_1^T + \cdots + D_{n-1}C_{n-1}^T = 0$ (for example, one may take $D_0 = D_1 = \cdots = D_{n-1} = 0$). Let $G_1 = [\mathbf{g}_0, D_0, \cdots, \mathbf{g}_{n-1}, D_{n-1}]$, and $G = G_1(A^T)^{-1}(P^T)^{-1}$. Then

$$GH^T = G_1(A^T)^{-1}(P^T)^{-1}P^TA^TH_1^TS^T = G_1H_1^T = 0.$$

For a received ciphertext $\mathbf{c}$ with $\mathbf{c}(S^T)^{-1} = \mathbf{m}P^TA^TH_1^T$, one can find a vector $\mathbf{a} \in GF(q)^{n(r+1)}$ such that $\mathbf{c}(S^T)^{-1} = \mathbf{a}H^T$. Then we have $(\mathbf{a} - \mathbf{m}P^TA^T)H^T = 0$. Since the space spanned by the rows of H is of dimension $n-k$, the orthogonal space to the space spanned by the rows of H is of dimension $nt+k$. However, the space spanned by the rows of G only has dimension k. Thus only with a negligible probability, the vector $\mathbf{a} - \mathbf{m}P^TA^T$ is in the code space generated by the rows of G. In other words, the above candidate decryption algorithm will succeed only with a negligible probability.

The arguments in the preceding paragraph shows that it is hard to design an equivalent Niederreiter encryption scheme for RLCE scheme. This provides certain evidence for the robustness of RLCE scheme.

4 Revised encryption scheme RLCE

In this section, we introduce a revised RLCE scheme to improve the message bandwidth and to reduce the public key size. The main difference between the revised scheme and the original scheme in [30] is that the revised scheme inserts random columns after randomly selected columns in the generator matrix. Specifically the revised RLCE scheme proceeds as follows.

$\texttt{RLCE.KeySetup}(n, k, d, t, w)$. Let $n, k, d, t > 0$, and $w \in \{1, \cdots, n\}$ be given parameters such that $n-k+1 \geq d \geq 2t+1$. Let G_s be a $k\times n$ generator matrix for an $[n, k, d]$ linear code C such that there is an efficient decoding algorithm to correct at least t errors for this linear code given by G_s. Let P_1 be a randomly chosen $n\times n$ permutation matrix and $G_sP_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$.

1. Let $\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{w-1} \in GF(q)^k$ be column vectors drawn uniformly at random and let

$$G_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-w}, \mathbf{r}_0, \cdots, \mathbf{g}_{n-1}, \mathbf{r}_{w-1}] \tag{3}$$

be the $k\times(n+w)$ matrix obtained by inserting column vectors $\mathbf{r}_i$ into G_s.

2. Let $A_0, \cdots, A_{w-1} \in GF(q)^{2\times 2}$ be dense nonsingular 2×2 matrices chosen uniformly at random and let $A = \text{diag}[1, \cdots, 1, A_0, \cdots, A_{w-1}]$ be an $(n+w)\times(n+w)$ nonsingular matrix.

3. Let S be a random dense $k\times k$ nonsingular matrix and P_2 be an $(n+w)\times(n+w)$ permutation matrix.

4. The public key is the $k\times(n+w)$ matrix $G = SG_1AP_2$ and the private key is (S, G_s, P_1, P_2, A).

`RLCE.Enc`$(G, \mathbf{m}, \mathbf{e})$. For a row vector message $\mathbf{m} \in GF(q)^k$, choose a random row vector $\mathbf{e} = [e_0, \ldots, e_{n+w-1}] \in GF(q)^{n+w}$ such that the Hamming weight of $\mathbf{e}$ is at most t. The cipher text is $\mathbf{c} = \mathbf{m}G + \mathbf{e}$.

`RLCE.Dec`$(S, G_s, P_1, P_2, A, \mathbf{c})$. For a received cipher text $\mathbf{c} = [c_0, \ldots, c_{n+w-1}]$, compute

$$\mathbf{c}P_2^{-1}A^{-1} = \mathbf{m}SG_1 + \mathbf{e}P_2^{-1}A^{-1} = [c'_0, \ldots, c'_{n+w-1}].$$

Let $\mathbf{c}' = [c'_0, c'_1, \cdots, c'_{n-w}, c'_{n-w+2}, \cdots, c'_{n+w-2}]$ be the row vector of length n selected from the length $n + w$ row vector $\mathbf{c}P_2^{-1}A^{-1}$. Then $\mathbf{c}'P_1^{-1} = \mathbf{m}SG_s + \mathbf{e}'$ for some error vector $\mathbf{e}' \in GF(q)^n$ where the Hamming weight of $\mathbf{e}' \in GF(q)^n$ is at most t. Using the efficient decoding algorithm, one can compute $\mathbf{m}' = \mathbf{m}S$ and $\mathbf{m} = \mathbf{m}'S^{-1}$. Finally, calculate the Hamming weight $w = \texttt{weight}(\mathbf{c} - \mathbf{m}G)$. If $w \leq t$ then output $\mathbf{m}$ as the decrypted plaintext. Otherwise, output error.

Remark 1: From the construction of RLCE scheme, it is clear that if we set $w = n$, then the revised RLCE scheme is the same as the original RLCE scheme with $r = 1$. It is recommended to use $w \geq (n-k)/2$ though a smaller w is also acceptable. The details for selecting the value of w will be presented in the next section.
Remark 2. It should be noted that the private key does not need to hold the entire matrix A^{-1}. It is sufficient to hold the first column of A_i^{-1} for each $i = 0, \cdots, w$.

5 Systematic Decoding RLCE schemes

In the revised RLCE encryption scheme discussed in Section 4, one recovers the message by recovering $\mathbf{m}' = \mathbf{m}S$ first and then calculates the message $\mathbf{m} = \mathbf{m}'S^{-1}$. To avoid these expensive operations, we can require that the public key is systematic and restrict the permutation P_2 in such a way that it only permutes the last $n + w - k$ columns of SG_1A and keep the order of the first k columns of SG_1A unchanged. Note that with the above restriction, if the public key is in echelon format, then the matrix SG_sP_1 is in echelon format also. In other words, S is the matrix that makes SG_sP_1 in echelon format.

With the revision in the preceding paragraph, one only needs to recover the codeword $\mathbf{m}SG_s$ (equivalently, the error values $\mathbf{e}'$) instead of the values $\mathbf{m}S$. Furthermore, the matrix S^{-1} is no longer used in the decryption process. Thus one does not need to include S^{-1} within the private key and reduce the private key size significantly. By requiring $w \leq n - k$ (which is recommended), the steps of the RLCE scheme remain unchanged except the following revised decryption process.

`Systematic-decoding-RLCE.Dec`$(G_s, P_1, P_2, A, \mathbf{c})$. For a received cipher text $\mathbf{c} = [c_0, \ldots, c_{n+w-1}]$, compute

$$\mathbf{c}P_2^{-1}A^{-1} = \mathbf{m}SG_1 + \mathbf{e}P_2^{-1}A^{-1} = [c'_0, \ldots, c'_{n+w-1}] = [c_0, \ldots, c_{k-1}, c'_k, \ldots, c'_{n+w-1}].$$

Let $\mathbf{c}' = [c_0, c_1, \cdots, c_{k-1}, \cdots, c'_{n-w}, c'_{n-w+2}, \cdots, c'_{n+w-2}]$ be the row vector of length n selected from the length $n + w$ row vector $\mathbf{c}P_2^{-1}A^{-1}$. Then $\mathbf{c}'P_1^{-1} = \mathbf{m}SG_s + \mathbf{e}'$ for some error vector $\mathbf{e}' \in GF(q)^n$ where the Hamming weight of $\mathbf{e}' \in GF(q)^n$ is at most t. Using the efficient decoding algorithm, one recovers $\mathbf{e}'$ from $\mathbf{c}'P_1^{-1}$. Since $\mathbf{c}' = \mathbf{m}SG_sP_1 + \mathbf{e}'P_1$ and SG_sP_1 is in echelon format, the message $\mathbf{m}$ equals to the first k elements in the vector $\mathbf{c}' - \mathbf{e}'P_1$. Finally, calculate the Hamming weight $w = \texttt{weight}(\mathbf{c} - \mathbf{m}G)$. If $w \leq t$ then output $\mathbf{m}$ as the decrypted plaintext. Otherwise, output error.

6 Security analysis

Similar to most cryptographic systems, each type of McElience schemes may contain some weak keys and one should avoid using these weak keys when setting up the scheme. For example, [19] pointed out some weak keys for binary Goppa code based McElience schemes. The second straightforward observation is that one can modify an McElience encryption scheme ciphertext $\mathbf{c} = \mathbf{m}G + \mathbf{e}$ without knowing the message m. For example, one can obtain a valid ciphertext for a message $m + m'$ by setting $\mathbf{c}' = \mathbf{c} + m'G$. This kind of attacks could be defeated by using IND-CCA2-secure message padding schemes which will be discussed in the next Section.

In the following sections, we carry out some heuristic analysis of the encryption scheme RLCE. We first show that if w is too small then, for certain codes C such as the generalized Reed-Solomon code, the filtration attacks may be mounted to identify a small portion of the sub-matrix $S[\mathbf{g}_w, \cdots, \mathbf{g}_{n-1}]$ from the public key G where $G_s = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]P_1^{-1}$ is the private generator matrix for C. Thus the parameter w should be large enough so that one may not be able to recover any columns of $S[\mathbf{g}_w, \cdots, \mathbf{g}_{n-1}]$ or the recovered columns of $S[\mathbf{g}_w, \cdots, \mathbf{g}_{n-1}]$ will not help for breaking the scheme.

6.1 Filtration attacks

Using distinguisher techniques [12], Couvreur et al. [9] designed a filtration technique to attack GRS code based McEliece scheme. The filtration technique was further developed by Couvreur et al [10] to attack wild Goppa code based McEliece scheme. In the following, we briefly review the filtration attack in [10]. For two codes C_1 and C_2 of length n, the star product code $C_1 * C_2$ is the vector space spanned by $\mathbf{a} * \mathbf{b}$ for all pairs $(\mathbf{a}, \mathbf{b}) \in C_1 \times C_2$ where $\mathbf{a} * \mathbf{b} = [a_0b_0, a_1b_1, \cdots, a_{n-1}b_{n-1}]$. For $C_1 = C_2$, $C_1 * C_1$ is called the square code of C_1. It is showed in [10] that

$$\dim C_1 \times C_2 \le \left\{ n, \dim C_1 \dim C_2 - \binom{\dim(C_1 \cap C_1)}{2} \right\}. \tag{4}$$

Furthermore, the equality in (4) is attained for most randomly selected codes C_1 and C_2 of a given length and dimension. Note that for $C = C_1 = C_2$ and $\dim C = k$, the equation (4) becomes $\dim C^{*2} \le \min \left\{ n, \binom{k+1}{2} \right\}$.

Couvreur et al [10] showed that the square code of an alternant code of extension degree 2 may have an unusually low dimension when its actual rate is larger than its designed rate. Specifically, Couvreur et al created a family of nested codes (called a filtration) defined as follows:

$$C^a(0) \supseteq C^a(1) \supseteq \cdots \supseteq C^a(q+1). \tag{5}$$

where $a \in \{0, \cdots, n-1\}$. Roughly speaking, $C^a(j)$ consists of codewords of C corresponding to polynomials which have a zero of order j at position a. The first two elements of this filtration are just punctured and shortened versions of C and the rest of them can be computed from C by computing star products and solving linear systems. The support values $\alpha_0, \cdots, \alpha_{n-1}$ (the private key) for the Goppa code could be recovered using this nested family of codes efficiently.

The crucial part of the filtration technique is the efficient algorithm to compute the nested family of codes in (5). If the underlying linear code is a generalized Reed-Solomon code, the first step in Couvreur et al. [9] and Couvreur et al [10] is to identify the column positions that do not contain mixed randomness. If we set $w = n - k$, then this first step will not work since the matrix $[\mathbf{g}_w, \cdots, \mathbf{g}_{n-1}]$ is a full-rank square matrix which is equivalent to any full-rank random square matrix. On the other hand, if $w < n - k$, then the filtration attacks may be used to recover some portion of the sub-matrix $S[\mathbf{g}_w, \cdots, \mathbf{g}_{n-1}]$ from the public key. Once these column positions are recovered, Couvreur et al. [9] proposed to use Sidelnikov and Shestakov attack to break the scheme. However, this attack cannot continue since one does not have enough non-random columns to carry out Sidelnikov and Shestakov attack. The details will be presented in the next section. For the columns with mixed randomness, the linear equations constructed in Couvreur et al [10] could not be solved and the nested family (5) could not be computed correctly. After the non-random column positions are identified, one may also mount attacks against the columns with randomness using the attacks from [9].

That is, one mount the attack against the shortened $[w, k]$ linear code C' which is obtained by excluding identified non-random columns. This attack will be successful only if one can find a related code C'_1 for C' of dimension k such that the dimension of the square code of C'_1 has a dimension significantly less than $\min\left\{w, \binom{k+1}{2}\right\}$.

The analysis in the preceding paragraph shows that the filtration attacks could be used to recover at most $n - w - k$ to $n - w$ non-random columns. One may use these non-random columns to obtain a length $n - w$ and dimension k shortened code. We distinguish the following two cases:

- $w > n - k$. In this case, the obtained shortened code has length $n - w < k$. Thus one cannot decode the shortened code for any given ciphertext.
- $w \le n-k$. In this case, we have $n-w \ge k$. Thus the obtained shortened code is an $[n-w, k]$ linear code. For a ciphertext $\mathbf{c}$, let $\mathbf{c}'$ be a shortened ciphertext of length $n - w$ by restricting $\mathbf{c}$ to these non-random columns. In case that there are at most $\frac{n-w-k}{2}$ errors within $\mathbf{c}'$, then one can decode $\mathbf{c}'$ efficiently using the shortened $[n - w, k]$ linear code. Note that the probability for $\mathbf{c}'$ to contain at most $\frac{n-w-k}{2}$ errors is bounded by the following value:
$$P_{n,w,t} = \frac{\sum_{i=0}^{\frac{n-w-k}{2}} \binom{n-w}{i}\binom{w}{t-i}}{\binom{n}{t}} \tag{6}$$
Thus the value of w should be chosen in such a way that for the given security parameter κ, we should have $P_{n,w,t} \le 2^{-\kappa}$.

6.2 Sidelnikov-Shestakov's attack

Let $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$ be n distinct elements of $GF(q)$ and let $v = (v_0, \ldots, v_{n-1})$ be nonzero (not necessarily distinct) elements of $GF(q)$. The generalized Reed-Solomon (GRS) code of dimension k, denoted by $GRS_k(\alpha, v)$, is defined by the following subspace.

$$GRS_k(\alpha, v) = \{(v_0 f(\alpha_0), \ldots, v_{n-1} f(\alpha_{n-1})) : f(x) \in GF(q)[x]_k)\}$$

where $GF(q)[x]_k$ is the set of polynomials in $GF(q)[x]$ of degree less than k. $GF(q)[x]_k$ is a vector space of dimension k over $GF(q)$. For each code word $c = (v_0 f(\alpha_0), \ldots, v_{n-1} f(\alpha_{n-1}))$, $f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}$ is called the associate polynomial of the code word c that encodes the message $(a_0, \ldots, a_{k-1}) \in GF(q)^k$. $GRS_k(\alpha, v)$ is an $[n, k, d]$ MDS code where $d = n - k + 1$.

Niederreiter's scheme [22] replaces the binary Goppa codes in McEliece scheme using GRS codes. The first attack on Niederreiter scheme is presented by Sidelnikov and Shestakov [28]. In Sidelnikov-Shestakov attack, one recovers an equivalent private key (α', v') from a public key G for the code $GRS_k(\alpha, v)$ as follows. For the given public key G, one first computes the systematic form $E(G) = [I|G']$ (also called echelon form) using Gaussian elimination. An equation system is then constructed from $E(G)$ to recover a decryption key.

$$E(G) = \begin{bmatrix} 1 & 0 & \cdots & 0 & b_{0,k} & \cdots & b_{0,n-1} \\ 0 & 1 & \cdots & 0 & b_{1,k} & \cdots & b_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{k-1,k} & \cdots & b_{k-1,n-1} \end{bmatrix} \tag{7}$$

For the ith row $\mathbf{b}_i$ of $E(G)$, assume the associated polynomial is $f_i(x)$. Since the only non-zero elements are

$b_{i,i}, b_{i,k+1}, \cdots, b_{i,n-1}$, we have

$$\begin{array}{l} v_0 f_i(\alpha_0) = 0 \\ \cdots \\ v_i f_i(\alpha_i) = 1 \\ \cdots \\ v_{n-1} f_i(\alpha_{n-1}) = b_{i,n-1} \end{array} \tag{8}$$

Thus f_i can be written as

$$f_i(x) = c_i \cdot \prod_{j=1, j\neq i}^{k} (x - \alpha_j) \tag{9}$$

for some $c_i \neq 0$. By the fact that

$$GRS_k(\alpha, v) = GRS_k(a\alpha + b, cv) \tag{10}$$

for all $a, b, c \in GF(q)$ with $ab \neq 0$, we may assume that $\alpha_0 = 0$ and $\alpha_1 = 1$. In the following, we try to recover $\alpha_2, \cdots, \alpha_{n-1}$. Using equation (9), one can divide the row entries in (7) by the corresponding nonzero entries in another row to get several equations. For example, if we divide entries in row i_0 by corresponding nonzero entries in row i_1, we get

$$\frac{b_{i_0,j}}{b_{i_1,j}} = \frac{v_j f_{i_0}(\alpha_j)}{v_j f_{i_1}(\alpha_j)} = \frac{c_{i_0}(\alpha_j - \alpha_{i_1})}{c_{i_1}(\alpha_j - \alpha_{i_0})} \tag{11}$$

for $j = k, \cdots, n-1$. First, by taking $i_0 = 0$ and $i_1 = 1$, equation (11) could be used to recover $\alpha_k, \cdots, \alpha_{n-1}$ by guessing the value of $\frac{c_0}{c_1}$ which is possible when q is small. By letting $i_0 = 0$ and $i_1 = 2, \cdots, k-1$ respectively, equation (11) could be used to recover α_{i_1}. Sidelnikov and Shestakov [28] also showed that the values of v can then be recovered by solving some linear equation systems based on $\alpha_0, \cdots, \alpha_{n-1}$.

The crucial step in Sidelnikov and Shestakov attack is to use the echelon form $E(G) = [I|G']$ of the public key to get minimum weight codewords that are co-related to each other supports. In the encryption scheme RLCE, w columns of the public key matrix G contain mixed randomness. Using the filtration attacks, one may identify $n - w - k$ to $n - w$ non-random columns and use the recovered non-random columns to form a $k \times (n - w - i_0)$ matrix G_N for some $i_0 \leq k$. Then one can compute an echelon form $E(G_N)$ for G_N that contains $n - w - i_0$ columns. From the echelon form, one can establish equations (8) and (9). However, for appropriately chosen w, there are not enough columns from $E(G_N)$ for one to build enough equations (11) to recover $\alpha_0, \cdots, \alpha_{n-1}$. In particular, if $w \geq n - k - i_0$, no equations in (11) could be established. Thus Sidelnikov and Shestakov attack could not be mounted on the RLCE scheme for appropriately chosen w.

6.3 Algebraic attacks

Faugere, Otmani, Perret, and Tillich [13] developed an algebraic attack against quasi-cyclic and dyadic structure based compact variants of McEliece encryption scheme. In a high level, the algebraic attack from [13] tries to find $\mathbf{x}^* = [\alpha_0, \cdots, \alpha_{n-1}], \mathbf{y}^* = \left[\frac{1}{g(\alpha_0)}, \cdots, \frac{1}{g(\alpha_{n-1})}\right] \in GF(q)^n$ such that

$$V_t(\mathbf{x}^*, \mathbf{y}^*) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0^1 & \alpha_1^1 & \cdots & \alpha_{n-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^t & \alpha_1^t & \cdots & \alpha_{n-1}^t \end{bmatrix} \begin{bmatrix} \frac{1}{g(\alpha_0)} & & \\ & \ddots & \\ & & \frac{1}{g(\alpha_{n-1})} \end{bmatrix} \tag{12}$$

is the parity check matrix for the underlying alternant codes of the compact variants of McEliece encryption scheme. $V_t(\mathbf{x}^*, \mathbf{y}^*)$ can then be used to break the McEliece scheme. Note that this $V_t(\mathbf{x}^*, \mathbf{y}^*)$ is generally different from the private parity check matrix $V_t(\mathbf{x}, \mathbf{y})$. The parity check matrix $V_t(\mathbf{x}^*, \mathbf{y}^*)$ was obtained by solving an equation system constructed from

$$V_t(\mathbf{x}^*, \mathbf{y}^*)G^T = 0, \tag{13}$$

where G is the public key. The authors of [13] employed the special properties of quasi-cyclic and dyadic structures (which provide additional linear equations) to rewrite the equation system obtained from (13) and then calculate $V_t(\mathbf{x}^*, \mathbf{y}^*)$ efficiently.

It is challenging to mount the above mentioned algebraic attacks on the RLCE encryption scheme. Assume that the RLCE scheme is based on a Reed-Solomon code. Let G be the public key and (S, G_s, A, P) be the private key. The parity check matrix for a Reed-Solomon code is in the format of

$$V_t(\alpha) = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{t+1} & \alpha^{2(t+1)} & \cdots & \alpha^{(t+1)(n-1)} \end{bmatrix}. \tag{14}$$

The algebraic attack in [12, 13] requires one to obtain a parity check matrix $V_t(\alpha^*)$ for the underlying Reed-Solomon code from the public key G, where α^* may be different from α. Assume that $V_t(\alpha^*) = [\mathbf{v}_0, \cdots, \mathbf{v_{n-1}}] \in GF(q)^{(t+1)\times n}$ is a parity check matrix for the underlying Reed-Solomon code. Let $V_t'(\alpha^*) \in GF(q)^{(t+1)\times(n+w)}$ be a $(t+1)\times(n+w)$ matrix obtained from $V_t(\alpha^*)$ by inserting w column vectors $\mathbf{0}$ after each of the first w column of $V_t(\alpha^*)$. That is,

$$V_t'(\alpha^*) = [\mathbf{v}_0, \mathbf{0}, \mathbf{v}_1, \mathbf{0}, \cdots, \mathbf{v_{n-1}}]. \tag{15}$$

Then we have

$$\begin{array}{rcl} V_t'(\alpha^*)G_1^T & = & V_t'(\alpha^*)[\mathbf{g}_0, \mathbf{r}_0, \cdots, \mathbf{g}_{n-1}]^T \\ & = & V_t(\alpha^*)[\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]^T \\ & = & V_t(\alpha^*)G_s^T \\ & = & \mathbf{0}. \end{array} \tag{16}$$

We cannot build an equation system for the unknown $V_t'(\alpha^*)$ from the public key $G = SG_1AP$ directly since the identity (16) only shows the relationship between $V_t'(\alpha^*)$ and G_1. In other words, in order to build an equation system for $V_t'(\alpha^*)$, one also needs to use unknown variables for the non-singular matrix A and the permutation matrix P. That is, we have

$$V_t'(\alpha^*)(A^{-1})^T(P^{-1})^TG^T = V_t'(\alpha^*)(GP^{-1}A^{-1})^T = V_t'(\alpha^*)G_1^TS^T = \mathbf{0}. \tag{17}$$

with an unknown α^*, an unknown matrix $A = \text{diag}[A_0, \cdots, A_{w-1}, 1, \cdots, 1]$ which consists of w dense non-singular 2×2 matrices $A_i \in GF(q)^{2\times 2}$, and an unknown permutation matrix P. In order to find a solution α^*, one first needs to take a potential permutation matrix P^{-1} to reorganize columns of the public key G. Then, using the identity $V_t'(\alpha^*)(A^{-1})^T(P^{-1})^TG^T = \mathbf{0}$, one can build a degree $(t+1)(n-1)+1$ equation system of $k(t+1)$ equations in $4w+1$ unknowns. In case that $k(t+1) \geq 4w+1$, one may use Buchberger's Gröbner basis algorithms as in [13] to find a solution α^*. However, this kind of algebraic attacks are infeasible due to the following two challenges. First the number of permutation matrices P is too large to be handled practically. Secondly, even if one can manage to handle the large number of permutation matrices P, the Gröbner basis are impractical for such kind of equation systems since the Buchberger's algorithm cannot solve nonlinear multicariate equation systems with more than 20 variables in practice (see, e.g., Courtois et al [8]).

6.4 Information-Set Decoding

As mentioned in the introduction section, the most powerful message recovery attack (not private key recovery attack) on McEliece encryption schemes is the information-set decoding attack. The state-of-the-art information-set decoding attack for non-binary McEliece scheme is the one presented in Peters [23], which integrated optimized Lee-Brickell's algorithm [17], Stern's algorithm [29], and Leon's minimum-weight-word-finding algorithm [18]. Peters's attack [23] also integrated analysis techniques for information-set decoding attacks on binary McEliece scheme discussed in [4]. For the RLCE encryption scheme, the information-set decoding attack is based on the number of columns in the public key G instead of the number of columns in the private key G_s. For the same error weight t, the probability to find error-free coordinates in $n + w$ coordinates is different from the probability to find error-free coordinates in n coordinates. Specifically, the cost of information-set decoding attacks on an $[n, k, t; w]$-RLCE scheme is equivalent to the cost of information-set decoding attacks on a standard $[n + w, k; t]$-McEliece scheme. It should be pointed out that the information set decoding attack is closely related to the finding low-weight codeword attacks.

6.5 Known partial plaintext [7]

For McEliece Encryption scheme, we have $\mathbf{c} = \mathbf{m}G + \mathbf{e}$. Let l, r be two positive integers such that $k = l + r$. Assume that $\mathbf{m} = [\mathbf{m}_l, \mathbf{m}_r]$ and $G = \begin{bmatrix} G_l \\ G_r \end{bmatrix}$. Then we have

$$\mathbf{c} = \mathbf{m}G + \mathbf{e} = [\mathbf{m}_l, \mathbf{m}_r]\begin{bmatrix} G_l \\ G_r \end{bmatrix} + \mathbf{e} = \mathbf{m}_l G_l + \mathbf{m}_r G_r + \mathbf{e}. \tag{18}$$

Thus if one knows the value of $\mathbf{m}_l$, the identity (18) becomes $\mathbf{c} - \mathbf{m}_l G_l = \mathbf{m}_r G_r + \mathbf{e}$ which could be much easy to decode than the original codeword $\mathbf{c}$ since $r < k$. The known-partial-plaintext-attack could be defeated using appropraite message padding for IND-CCA2-security that will be discussed in Section 7.

6.6 Related message attack [5]

Assume that $\mathbf{c}_1 = \mathbf{m}_1 G + \mathbf{e}_1$ and $\mathbf{c}_2 = \mathbf{m}_2 G + \mathbf{e}_2$. Furthermore, assume that the adversary knows the relation between $\mathbf{m}_1$ and $\mathbf{m}_2$. For example, assume that $\mathbf{m} = \mathbf{m}_1 + \mathbf{m}_2$ and that the adversary knows the value of $\mathbf{m}$. Then we have $\mathbf{c}_1 + \mathbf{c}_2 - \mathbf{m}G = \mathbf{e}_1 + \mathbf{e}_2$. Since $\mathbf{e}_1$ and $\mathbf{e}_1$ are different and both of them have low weight t, it could be easy for the adversary to recover both $\mathbf{e}_1$ and $\mathbf{e}_1$ by trying all combinations. Even if one cannot enumerate all combinations to recover either $\mathbf{e}_1$ or $\mathbf{e}_1$, one can use the 0 entries within $\mathbf{e}_1 + \mathbf{e}_2$ as a hint to speed up the information set decoding algorithm for recovering $\mathbf{m}_1$ from $\mathbf{c}_1 = \mathbf{m}_1 G + \mathbf{e}_1$. A special case of this attack is the attack on two ciphertexts of the identical message encrypted using different error vectors. The related-message-attack could be defeated using appropraite message padding for IND-CCA2 security that will be discussed in Section 7.

6.7 Reaction attack [15]

In this attack, one assumes that an McEliece decryption oracle outputs an error message each time when the given ciphertext contains too many errors to decrypt. For a given ciphertext $\mathbf{c}$, the adversary first randomly selects positions to add errors until the decryption oracle complains. That is, the adversary first obtains a ciphertext $\mathbf{c}'$ that contains maximum errors that the decryption oracle could handle. Then the adversary selects a random position i and add errors to this position. If the decryption oracle could decrypt the resulting ciphertext, it means that $\mathbf{c}'$ contains error at this position. Otherwise, this position is error-free. The adversary continues this process until she obtains k error-free positions for the ciphertext $\mathbf{c}$. These error-free positions could be used to recover the plaintext message for the ciphertext $\mathbf{c}$. The reaction-attack could be defeated using appropraite message padding for IND-CCA2 security that will be discussed in Section 7.

6.8 Reaction-attack based side channel attacks

Message padding schemes for IND-CCA2 security in Section 7 could be used to defeat the reaction attack. However, for a ciphertext that contains too many errors to decrypt and for a ciphertext with padding errors that decrypts successfully, the decryption oracle normally uses different amount of times. Thus an adversary may introduce errors in some positions of the ciphertext and observe the amount of time used for the decryption oracle to report errors. This will allow the adversary to distinguish whether the original ciphertext contains errors in these positions or not. The observed results could be used as in the reaction attack to recover the plaintext. In order to defeat such kind of reaction-attack based side-channel attacks, appropriate delays should be introduced in a decryption process of padded RLCE schemes so that the decryption process takes the same amount of times to report errors for padding errors and for decoding errors.

7 Message encoding and IND-CCA2 security

We mentioned several attacks on RLCE schemes in the preceding section. To avoid these attacks, it is necessary to use message padding schemes so that the encryption scheme is secure against adaptive chosen ciphertext attacks (IND-CCA2). In the following subsections, we present message padding schemes to make McEliece encryption scheme secure against adaptive chosen ciphertext attacks.

7.1 Message bandwidth

We first analyze the amount of information that could be encoded within each ciphertext. Let (n, k, t, w) be the parameters where the public key is of dimension $k \times (n + w)$ and $GF(2^m)$ is the underlying finite field. There are three approaches to encode messages within the ciphertext.

1. **basicEncoding**: Encode informaiton within the vector $\mathbf{m} \in GF(q)^k$ and the ciphertext is $\mathbf{c} = \mathbf{m}G + \mathbf{e}$. In this case, we can encode `mLen` $= mk$ bits information within each ciphertext.

2. **mediumEncoding**: In addition to **basicEncoding**, further information is encoded in the non-zero entries of $\mathbf{e}$. That is, let $e_{i_1}, \cdots, e_{i_t} \in GF(q) \setminus \{0\}$ be the non-zero elements within $\mathbf{e}$ and encode further information within $e_{i_1}, \cdots, e_{i_t}$. In this case, we can encode `mLen` $= m(k + t)$ bits information within each ciphertext. Strictly speaking, the encoded information is less than $m(k + t)$ bits since e_{i_j} cannot be zeros.

3. **advancedEncoding**: In addition to **mediumEncoding**, further information are encoded within within the choice of non-zero entries within $\mathbf{e}$. Since there are $\binom{n+w}{t}$ candidates for the choice of non-zero entries within $\mathbf{e}$, we can encode `mLen` $= m(k+t)+\left\lfloor \log_2 \binom{n+w}{t} \right\rfloor$ bits information within each ciphertext.

The basicEncoding approach is straightforward. For the mediumEncoding, after one recovers the vector $\mathbf{m}$, one needs to compute $\mathbf{m}G - \mathbf{c}$ to obtain the values of $e_{i_1}, \cdots, e_{i_t}$. For the advancedEncoding approach, we need to compute an invertible function

$$\varphi : W_{n+w,t} \leftrightarrow \left\{ i : 1 \le i \le \binom{n+w}{t} \right\} \tag{19}$$

where $W_{n+w,t} \subsetneq GF(2)^{n+w}$ is the set of all $(n+w)$-bit binary string of weight t. For the invertible function φ in (19), one may use the enumerative source encoding construction in Cover [11]:

$$\varphi : W_{n+w,t} \longleftrightarrow \left[0, \binom{n+w}{t}\right]$$

where $\varphi(i_1, \cdots, i_t) = \binom{i_t - 1}{t} + \cdots + \binom{i_1 - 1}{1}$ and $0 \le i_1 < i_2 < \cdots < i_t < n+w$ are the positions of ones. The function φ could be evaluated with the cost of $O\left(\left(\log_2 \left\lceil \binom{n+w}{t} \right\rceil\right)^2\right)$ operations (see, e.g., Sendrier [26]).

It should be noted that Sendrier [26] proposed a more efficient Golomb's run-length encoding construction of φ. Sendrier [26] claimed that their construction satisfies the condition in (19). However it can be shown that the construction in [26] is not a map. That is, for some elements $x \in \left\{i : 1 \le i \le \binom{n+w}{t}\right\}$, there does not exist $y \in W_{n+w,t}$ such that $x = \varphi(y)$ (see Section 8 for details).

7.2 Existing message encoding approaches

Several authors proposed to use message encoding (padding) approach to achieve IND-CCA2 security for McEliece encryption schemes. For example, Kobara and Imai [16] recommended the use of Pointcheval's generic conversion [24] or Fujisak-Okamato's generic conversion [14] to achieve adaptive chosen ciphertext security (IND-CCA2) for McEliece encryption scheme. Furthermore, they also proposed three new message encoding approaches to achieve adaptive chosen ciphertext security (IND-CCA2) for McEliece encryption scheme. Let H_1, H_2 be random oracles (e.g., they could be pseudo-random-bits generators or hash functions) that output random strings of appropriate lengths and let r_1, r_2 be randomly selected strings with appropriate length. Then the encryption processes with message padding schemes could be informally described as follows.

- Pointcheval padding: $\mathbf{c} = \texttt{Mc.Enc}(G, r_1, H_1(\mathbf{m}\|r_2))\|(H_2(r_1) \oplus (\mathbf{m}\|r_2))$.
- Fujisak-Okamato padding: $\mathbf{c} = \texttt{Mc.Enc}(G, r_1, H_1(\mathbf{m}\|r_1))\|(H_2(r_1) \oplus \mathbf{m})$.
- Kobara-Imai's α-padding: $\mathbf{c} = \texttt{Mc.Enc}(G, y_1, H_1(\mathbf{m}\|r_1))\|y_2$ where $y_1\|y_2 = H_2(H_1(\mathbf{m}\|r_1)) \oplus (\mathbf{m}\|r_1)$.
- Kobara-Imai's β-padding: $\mathbf{c} = y_1\|\texttt{Mc.Enc}(G, y_2, H_1(r_1))$ where $y_1\|y_2 = (r \oplus H_1(H_2(r) \oplus \mathbf{m}))\|(H_2(r) \oplus \mathbf{m})$.
- Kobara-Imai's γ-padding: $\mathbf{c} = y_3\|\texttt{Mc.Enc}(G, y_1, y_2)$ where $y_3\|y_2\|y_1 = (r \oplus H_1(H_2(r) \oplus (m\|\text{const})))\|(H_2(r) \oplus (m\|\text{const}))$.

Among these padding schemes, Pointcheval padding and Fujisak-Okamato padding require extra strings added after the McEliece ciphertext. This increases the ciphertext length and it is not a preferred choice for bandwidth efficiency. Though Kobara and Imai provided proof of security for their three padding schemes, it is not clear how to select the message and random bit lengths for a specific security strength. In particular, further analysis may be required to analyze the exact security corresponding to various parameter selections for Kobara-Imai padding schemes.

7.3 RLCE message padding schemes RLCEspad and RLCEpad

In this section, we assume that the message bandwidth is `mLen`-bits for each ciphertext. We present two efficient padding schemes for the RLCE encryption scheme. Our padding schemes are adapted from the well analyzed Optimal Asymmetric Encryption Padding (OAEP) for RSA/Rabin encryption schemes and its variants OAEP+ [27] and SAEP+ [6]. The first simple padding scheme RLCEspad is a one-round of a Feistel network that is similar to SAEP+. RLCEspad could be used to encrypt short messages (e.g., `mLen`/4-bits) and is sufficient for applications such as symmetric key transportation using the RLCE public key encryption scheme. The second padding scheme RLCEpad is a two-round Feistel network that is similar to OAEP+. RLCEpad could be used to encrypt messages that are almost as long as `mLen`-bits.

We assume that messages are binary strings. After padding, they will be converted to field elements and/or other information in the RLCE scheme (e.g., the information contained in the error vector $\mathbf{e}$ if mediumEncoding or advancedEncoding is used). For a RLCE setup process $\texttt{RLCE.KeySetup}(n, k, d, t, w)$, let the $k \times (n + w)$ matrix G be a public key and (S, G_s, P_1, P_2, A) be a corresponding private key. Assume that scheme is over a finite field $GF(2^m)$. The RLCEspad proceeds as follows.

$\texttt{RLCEspad}(\texttt{mLen}, k_1, k_2, k_3)$: Let k_1, k_2, k_3 be parameters such that $k_1 + k_2 + k_3 = \left\lceil \frac{\texttt{mLen}}{8} \right\rceil$, $k_1 + k_2 < k_3$, and $8k_1 \leq \texttt{mLen}/4$. Let $v = 8(k_1 + k_2 + k_3) - \texttt{mLen}$. Let H_1 be a random oracle that takes any-length inputs and outputs k_2-bytes and let H_2 be a random oracle that takes any-length inputs and outputs $(k_1 + k_2)$-bytes. Let $\mathbf{m} \in \{0, 1\}^{8k_1}$ be a message to be encrypted, $\mathbf{r}_0 \in \{0, 1\}^{8k_3 - v}$ be a randomly selected sequence, and $\mathbf{r} = \mathbf{r}_0 \| 0^v$. We distinguish the following three cases:

- basicEncoding: Select a random $\mathbf{e} \in GF(q)^{n+w}$ of weight t and set

$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e})) \oplus H_2(\mathbf{r}, \mathbf{e})) \| \mathbf{r}. \tag{20}$$

  Convert $\mathbf{y}$ to an element $\mathbf{y}_1 \in GF(q)^k$. Let the ciphertext be $\mathbf{c} = \mathbf{y}_1 G + \mathbf{e}$.

- mediumEncoding: Select random $0 \leq l_0 < l_1 < \cdots < l_{t-1} < n + w - 1$ and let $\mathbf{e}_0 = l_0 \| l_1 \cdots \| l_{t-1} \in \{0, 1\}^{16t}$. Set

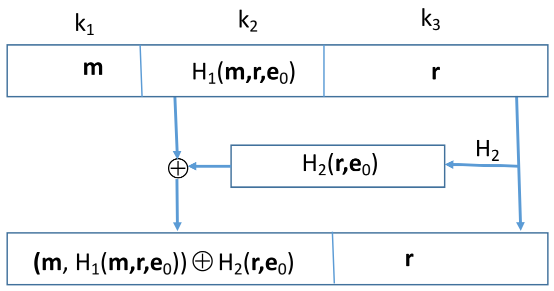
$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e}_0)) \oplus H_2(\mathbf{r}, \mathbf{e}_0)) \| \mathbf{r}. \tag{21}$$

  Convert $\mathbf{y}$ to an element $(\mathbf{y}_1, \mathbf{e}_1) \in GF(q)^{k+t}$ where $\mathbf{y}_1 \in GF(q)^k$ and $\mathbf{e}_1 \in GF(q)^t$. Let $\mathbf{e} \in GF(q)^{n+w}$ such that $\mathbf{e}[l_i] = \mathbf{e}_1[i]$ for $0 \leq i < t$ and $\mathbf{e}[j] = 0$ for $j \neq l_i$. Let the ciphertext be $\mathbf{c} = \mathbf{y}_1 G + \mathbf{e}$.

- advancedEncoding: Set $\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r})) \oplus H_2(\mathbf{r})) \| \mathbf{r}$. Convert $\mathbf{y}$ to an element $\mathbf{y}_1 \in GF(q)^k$ and a vector $\mathbf{e} \in GF(q)^{n+w}$ of weight t. Let the ciphertext be $\mathbf{c} = \mathbf{y}_1 G + \mathbf{e}$.

The mediumEncoding based RLCEspad is shown graphically in Figure 1.

Figure 1: mediumEncoding based RLCEspad



Assuming the hardness of decoding RLCE ciphertexts, a similar proof as in [6] could be used to show that RLCE-RLCEspad scheme is secure against IND-CCA2 attacks. As an example with $\kappa_c = 128$ bits security RLCE scheme $(600, 464, 68)$ over $GF(2^{10})$ in Table 2, we use $k_1 = k_2 = 160$-bytes for mediumEncoding and $k_1 = k_2 = 170$-bytes for advancedEncoding. Thus, we can encrypt $k_1 = 160$-bytes of information for mediumEncoding and $k_1 = 170$-bytes of information for advancedEncoding per RLCE-RLCEspad ciphertext.

Our next padding scheme RLCEpad is based on OAEP+ and proceeds as follows.

`RLCEpad(mLen`, k_1, k_2, k_3, t): Let k_1, k_2, k_3 be parameters such that $k_1 + k_2 + k_3 = \left\lceil \frac{\texttt{mLen}}{8} \right\rceil$, $\min\{k_2, k_3\} \geq \kappa_c$ where κ_c is the security parameter. Let H_1 be a random oracle that takes any-length inputs and outputs k_2 bytes, H_2 be a random oracle that takes any-length inputs and outputs $k_1 + k_2$ bytes, and H_3 be a random oracle that takes any-length inputs and outputs k_3 bytes. Let $\mathbf{m} \in \{0,1\}^{8k_1}$ be a message to be encrypted, $\mathbf{r}_0 \in \{0,1\}^{8k_3-v}$ be a randomly selected sequence, and $\mathbf{r} = \mathbf{r}_0 \| 0^v$. We distinguish the following three cases:

- basicEncoding: Select a random $\mathbf{e} \in GF(q)^{n+w,t}$ of weight t and set

$$\mathbf{y} = ((\mathbf{m}\|H_1(\mathbf{m},\mathbf{r},\mathbf{e})) \oplus H_2(\mathbf{r},\mathbf{e}))\,\|\mathbf{r} \oplus H_3(((\mathbf{m}\|H_1(\mathbf{m},\mathbf{r},\mathbf{e})) \oplus H_2(\mathbf{r},\mathbf{e}))) \tag{22}$$

  Convert $\mathbf{y}$ to an element $\mathbf{y}_1 \in GF(q)^k$. Let the ciphertext be $\mathbf{c} = \mathbf{y}_1 G + \mathbf{e}$.

- mediumEncoding: Select random $0 \leq l_0 < l_1 < \cdots < l_{t-1} < n + w - 1$ and let $\mathbf{e}_0 = l_0\|l_1 \cdots \|l_{t-1} \in \{0,1\}^{16t}$. Set

$$\mathbf{y} = ((\mathbf{m}\|H_1(\mathbf{m},\mathbf{r},\mathbf{e}_0)) \oplus H_2(\mathbf{r},\mathbf{e}_0))\,\|\mathbf{r} \oplus H_3(((\mathbf{m}\|H_1(\mathbf{m},\mathbf{r},\mathbf{e}_0)) \oplus H_2(\mathbf{r},\mathbf{e}_0))) \tag{23}$$

  Convert $\mathbf{y}$ to an element $(\mathbf{y}_1, \mathbf{e}_1) \in GF(q)^{k+t}$ where $\mathbf{y}_1 \in GF(q)^k$ and $\mathbf{e}_1 \in GF(q)^t$. Let $\mathbf{e} \in GF(q)^{n+w}$ such that $\mathbf{e}[l_i] = \mathbf{e}_1[i]$ for $0 \leq i < t$ and $\mathbf{e}[j] = 0$ for $j \neq l_i$. Let the ciphertext be $\mathbf{c} = \mathbf{y}_1 G + \mathbf{e}$.

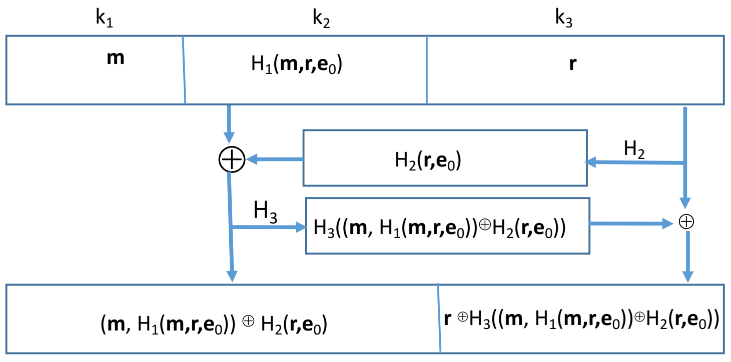
- advancedEncoding: Set

$$\mathbf{y} = ((\mathbf{m}\|H_1(\mathbf{m},\mathbf{r})) \oplus H_2(\mathbf{r}))\,\|\mathbf{r} \oplus H_3(((\mathbf{m}\|H_1(\mathbf{m},\mathbf{r})) \oplus H_2(\mathbf{r}))) \tag{24}$$

  Convert $\mathbf{y}$ to an element $\mathbf{y}_1 \in GF(q)^k$ and a vector $\mathbf{e} \in GF(q)^{n+w}$ of weight t. Let the ciphertext be $\mathbf{c} = \mathbf{y}_1 G + \mathbf{e}$.

The mediumEncoding based RLCEspad is shown graphically in Figure 2.

Figure 2: mediumEncoding based RLCEpad


k1
k2
k3
m
H1(m,r,e0)
r
H2(r,e0)
H2
H3
H3((m, H1(m,r,e0))⊕H2(r,e0))
(m, H1(m,r,e0)) ⊕ H2(r,e0)
r ⊕H3((m, H1(m,r,e0))⊕H2(r,e0))

Assuming the hardness of decoding RLCE ciphertexts, a similar proof as in [27] could be used to show that RLCE-RLCEpad scheme is secure against IND-CCA2 attacks. The proof in [27] shows that, for a given security parameter κ_c, it is sufficient to choose k_2, k_3 with

$$\max\left\{\frac{1}{q^{k_2}}, \frac{1}{q^{k_3}}\right\} \le \frac{1}{2^{\kappa_c}}. \tag{25}$$

As an example with $\kappa_c = 128$ bits security RLCE scheme $(600, 464, 68)$ over $GF(2^{10})$ in Table 2, we use $k_2 = k_3 = 32$-bytes for both mediumEncoding and advancedEncoding. Thus, we can encrypt $k_1 = 601$-bytes of information for mediumEncoding and $k_1 = 641$-bytes of information for advancedEncoding per RLCE-RLCEpad ciphertext.

Remark 1: In RLCE encryption scheme, either error positions $\mathbf{e}_0$ or error vector $\mathbf{e}$ is used in the RLCEspad/RLCEpad process and the message recipient needs to have the exact $\mathbf{e}_0$ or $\mathbf{e}$ for message decoding. In case that the randomly generated error values contain zero field elements, the corresponding error positions will be unavailable for the recipient. To avoid this potential issue, the message encryption process needs to guarantee that error values should never be zero. A simple approach to address this challenge is that, when calculated error values (using the given random value $\mathbf{r}$) contain zero field elements, one revises the random value $\mathbf{r}$ to a new value and tries the padding approach again. This process continues until all error values are non-zero.

Remark 2: In our scheme, we use $k_1 + k_2 + k_3 = \left\lceil \frac{\mathtt{mLen}}{8} \right\rceil$. Alternatively, one may use $k_1 + k_2 + k_3 = \left\lfloor \frac{\mathtt{mLen}}{8} \right\rfloor$ and adjust the schemes correspondingly.

8 Run length encoding (RLE)

For RLCE advancedEncoding scheme, one encodes information within the error vectors. Sendrier [26] uses Golomb's run length encoding to construct a linear algorithm for the map between $\left[0, \binom{n}{t}\right]$ and $W_{n,t}$. However, this map is not a bijection. Thus it cannot be used to map numbers $\left[0, \binom{n}{t}\right]$ to constant weight words. First, we briefly discuss run length encoding. We start with a simple example of encoding a run of zeros using four-bit binary sequences. A sequence: $0^{14}10^{9}110^{20}10^{30}1$ could be encoded as

$$(1110)(1001)(0000)(1111\ 0101)(1111\ 1111\ 0000)$$

where we used 32-bits to encode a sequnce of 78-bits. The above scheme could be improved by using non-fixed length encoding for the length of a run. In Golomb's RLE, let d be an appropriately chosen non-negative integer and $f_d : \{0, 1, \cdots, d\} \to \{0, 1\}^*$ be a prefix-free code with $f_d(d) = 1$. Then a zero run of length $n = qd + r$ can be encoded as $1^q f_d(r)$. As an example, let $d = 5$ and f_d be defined as:

$$f_5(\emptyset) = 0, f_5(0) = 001, f_5(00) = 010, f_5(000) = 0110, f_5(0000) = 0111, f_5(00000) = 1.$$

With this RLE, the sequence

$$00000\ 0001\ 00000\ 00001\ 1\ 0001\ 01$$

can be encoded as 101101011100110001. In other words, a 26-bit sequence is encoded to a 18-bit sequence. Assume that 0 has a probability $p \ge \frac{1}{2}$. Then Golomb RLE is optimal when $d = \left\lceil \frac{-1}{\log_2 p} \right\rceil$. Note that $(1-p)^d \sim \frac{1}{2}$ is the probability for any string of zeroes to have length $\ge d$.

For $t << n$, the set $W_{n,t}$ contains binary sequences with majority 0. Thus Golomb RLE could be used to encde elements of $W_{n,t}$. Specifically, each sequence $0^{\delta_1}10^{\delta_2}\cdots 10^{\delta_{t+1}} \in W_{n,t}$ could be encoded as

$f(\delta_1)\cdots f(\delta_t)$. However the encoded words cannot be one-to-one mapped to numbers in the range $\left[0, \binom{n}{t}\right]$. Sendrier [26] showed that if one considers the probability

$$P_1(s) = Prob(\delta_1 = s) = P_{n,t}(s) = \frac{\binom{n-s-1}{t-1}}{\binom{n}{t}}$$

and the conditional probability

$$P_{j+1}(s) = Prob(\delta_{j+1} = s|\delta_1, \cdots, \delta_j) = P_{n-\delta_1-\cdots-\delta_j-j,t-j}(s),$$

one may dynamically construct an efficient fully decodable encoding φ in (19). Unfortunately, this claim is not true. In the following, we point out the issues within the map by [26]. The construction in [26] proceeds as follows.

1. At step 1, let d_1 be the smallest integer such that $\sum_{s\geq d_1} P_1(s) < 1/2$ and use f_{d_1} to encode δ_1.
2. At step $j+1$, d_{j+1} is calculated using the conditional probability $P_{j+1}(s)$. Use $f_{d_{j+1}}$ to encode δ_{j+1}.

The above encoding process gives an algorithm to convert each constant weight string in $W_{n,t}$ to a binary string. Sendrier [26] claimed without a proof that given sufficient input bits, the inverse of the above algorithm can convert any binary string to constant weight strings in $W_{n,t}$ using the prefix-free coding

$$f_d(i) = \begin{cases} \text{base2}(i, u-1) & \text{if } 0 \leq i < 2^u - d \\ \text{base2}(i + 2^u - d, u-1) & \text{if } 2^u - d \leq d \end{cases}$$

where $2^{u-1} < d \leq 2^u$ and base2(x, l) denotes the l least significant bits of x. However, this claim is not true. For given parameters n and t, let x be a binary string such that x contain no prefix from the set $\{f_{d_1}(y) : |y| \leq d_1\}$. Then it is straightforward to show that no element in $W_{n,t}$ is mapped to x. In other words, for the information symbol x, there is no way to encode it as an error vector in $W_{n,t}$. Since the above construction map is one-to-one and $|W_{n,t}| = \binom{n}{t}$, this implies that for many elements $x \in W_{n,t}$, we have $\varphi(x) \notin \left[0, \binom{n}{t}\right]$.

9 Recommended parameters and performance evaluation

Taking into account of the cost of Sidelnikov-Shestakov attacks, the cost of recovering McEliece encryption scheme secret keys from the public keys, and the cost of recovering plaintext messages from ciphertexts using the information-set decoding methods, we generated a recommended list of parameters for RLCE scheme in Table 1. In Table 1, κ_c denotes the conventional security strength. For example, $\kappa_c = 128$ means an equivalent security of AES-128. For the naive information set decoding, one first uniformly selects k columns from the public key and checks whether it could be inversed. If it could be inversed, one multiplies the inverse with the ciphertext. If these coordinates contain no errors in the ciphertext, one recovers the plain text. To be conservative, we may assume that randomly selected k columns from the public key is invertible. For each $k \times k$ matrix inversion, Strassen algorithm takes $O(n^{2.807})$ field operations (though Coppersmith-Winograd algorithm takes $O(n^{2.376})$ field operations in theory, it may not be practical for the matrices involved in RLCE encryption schemes). Thus the naive information-set decoding algorithm takes more than $2^{\kappa_c'}$ steps to find k-error free coordinates where, by Sterling's approximation,

$$\kappa_c' = \log_2\left(\frac{\binom{n+w}{k}k^3}{\binom{n+w-t}{k}}\right) + O(1) \simeq (n+w)I\left(\frac{k}{n+w}\right) - (n+w-t)I\left(\frac{k}{n+w-t}\right) + \log_2(k^3) + O(1) \quad (26)$$

and $I(x) = -x \log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy of x. There are several improved information-set decoding algorithms in the literature such as the one by Peters [23]. The values of κ_c in Table 1 are mainly calculated using the PARI/GP script from Peters [23]. Normally, we have $\kappa_c = \kappa_c' - 6 + o(1)$. For the recommended parameters, the default underlying linear code is assumed to be any MDS code (e.g., generalized Reed-Solomon code) over $GF(q)$ where $q = 2^{\lceil \log_2 n \rceil}$ or $q = 2^{12}$ (for convenient data conversion over 32 or 64 bit computers). For generalized Reed-Solomon code, the natural construction requires $n = q-1$. However, generalized Reed-Solomon code could be shortened to length $n < q-1$ codes by interpreting the unused $q-1-n$ information symbols as zeros. For the value of w, we consider the following two cases: $w = n-k$ and $w = \frac{n-k}{2}$. For the purpose of comparison, we also list the recommended parameters from [4] for the binary Goppa code based McEliece encryption scheme.

To reduce the public key sizes, the authors in [4, 23] proposed the use of semantic secure message coding approach so that one can store the public key as a systematic generator matrix. For a McEliece encryption scheme over $GF(q)$, one needs to store $k(n-k)$ elements from $GF(q)$ for a systematic generator matrix public key instead of nk elements from $GF(q)$ for a non-systematic generator matrix public key. For RLCE encryption scheme over $GF(q)$, the systematic generator matrix public key is $k(n+w-k)\log q$ bits. It is observed that RLCE schemes with all parameters have smaller public key sizes than binary Goppa code based McEliece scheme. Specifically, for a security level of 128 bits, the public key for the RLCE scheme with $w = n-k$ is 154KB, the public key for the RLCE scheme with $w = \frac{n-k}{2}$ is 62KB while the binary Goppa code based McEliece encryption scheme has a public key size of 187.7KB.

The value κ_q in Table 1 denotes the quantum security strength under quantum information-set decoding using Grover's algorithm (e.g., Bernstein [3]). For RLCE scheme, quantum information-set decoding could be carried out similarly as in Bernstein's [3]. One first uniformly selects k columns from the public key and checks whether it could be inversed. If it could be inversed, one multiplies the inverse with the ciphertext. If these coordinates contain no errors in the ciphertext, one recovers the plain text. In order for one to obtain k-error free coordinates, Grover's algorithm could be applied. To be conservative, we may assume that a randomly selected k columns from the public key is invertible. Thus Grover's algorithm take more than $2^{\kappa_c'/2}$ steps to find k-error free coordinates where κ_c' is given by the formula in (26). For each chosen k columns from the public key, similar analysis as in [3] could be used to show that the quantum computer needs $O(n^3 q^{1.585})$ bit operations to test whether the selected columns are error-free, where we assume that field multiplications are done using Karatsuba algorithm. In a summary, Grove's algorithm requires approximately $O(n^3 q^{1.585} 2^{\kappa_1})$ steps for the information-set decoding. The quantum security parameter κ_q is derived from these attacks. Advanced quantum information-set decoding techniques may be developed based on the algorithms information-set decoding algorithm. However our analysis shows that it will not reduce the quantum security strength for the schemes proposed in Table 1.

Table 1: RLCE parameters: "600, 464, 68, **10**, 154KB"' represents $n = 600, k = 464, t = 68, q = 2^{10}$. The bold face security parameters correspond to NIST post call for proposal security parameters

κ_c	κ_q	RLCE ($w = n-k$)	RLCE $\left(w = \frac{n-k}{2}\right)$	binary Goppa code [4]
60	59	255,155,50,**8**,30KB	200,120,40,**8**,12.89KB	1024, 524, 50, 19.8KB
80	70	360,200, 80, **9**, 101KB	300,140,80,**9**, 36.91KB	1632, 1269, 34, 56.2KB
128	**96**	600,464,68,**10**,154KB	511,381,65,**9**,82KB	2960, 2288, 57, 188KB
128	100	600,440,80,**12**,206KB	502,378,62, **12**,103KB	
160	115	780,580,100,**10**,212KB	620,440,90,**10**,177KB	3100,2300,80,302KB
160	117	760,540,110,**12**, 348KB	620,440,90,**12**,174KB	
192	**132**	1000,790,105,**10**,405KB	800,600,100,**10**,220KB	4624, 3468, 97, 490KB
192	135	990,780,105, **12**, 480KB	790,590,100,**12**,259KB	
256	**165**	1300,800,250,**11**, 1.05MB	1023,663,180,**10**,437KB	6624, 5129, 117, 900KB
256	167	1300,800,250,**12**, 1.14MB	1023,663,180,**12**,524KB	

Parameters in Table 1 could be used for any MDS code based RLCE scheme. In practice, one may also use non-MDS codes such as LDPC codes, Polar codes, and other to construct RLCE schemes. In addition to QC-LDPC codes [1], other LDPC codes could be used to design RLCE scheme also. Polar code based McEliece encryption scheme has been broken in [2] using the fact that, for given parameters n, k, there is only one (n, k) polar code. However, secure polar code based McEliece encryption schemes could be designed using RLCE scheme. Since decoding algorithm for polar codes can produce as close as possible codeword from any given binary string, it may be possible to design efficient digital signature schemes using polar code based RLCE scheme.

Table 2 lists the message bandwidth and message padding scheme parameters for the recommended schemes. For each security strength (κ_c, κ_q), the odd-ID is for RLCE $(w = n - k)$ and the even-ID is for RLCE $\left(w = \frac{n-k}{2}\right)$. In case that $v = 8(k_1 + k_2 + k_3) - \texttt{mLen}_i > 0$, the last v-bits of the k_3-bytes random seed $\mathbf{r}$ should be set to zero and the last v-bit of the encoded string $\mathbf{y}$ is discarded. For RLCEspad with $v > 0$, the encoding and decoding process are straightforward. For RLCEpad with $v > 0$, the decoding process produces an encoded string $\mathbf{y}$ with last v-bits missing. After using H_3 to hash the first part of $\mathbf{y}$ resulting in k_3-bytes hash output, one discards the last v-bits from the hash output and $\oplus$ the remaining $(8k_3 - v)$-bits with the second half of $\mathbf{y}$ to obtain the $(8k_3 - v)$-bits of $\mathbf{r}$ without the v-bits zero trailer.

Table 3 lists the performance results for RLCE encryption scheme that was tested with MacOS Sierra on a MacBook Pro (Retina 2013 model) with 2.4 GHz Intel Core i5. The first column contains the encryption scheme ID from Table 2. The second column contains the time needed for a public/private key pair generation. The third two-column group contains the time needed for one ciphertext encryption. The fourth two-column group contains kilo-bytes of plaintext message that could be encrypted within one second. The fifth two-column group contains the time needed for one ciphertext decryption and the last two-column group contains kilo-bytes of plaintext message that could be decrypted within one second. The message size refers to pre-padded message size.

Table 4 lists the performance results for RLCE encryption scheme that was tested with Dell Optiplex 9010 Desktop Computer with Intel(R) Core(TM) i7-3770 CPU @3.40GHz and 16GB RAM. It runs Cygwin within Windows 10.

10 Conclusions

In this paper, we presented techniques for designing general random linear code based public encryption schemes using any linear code. The proposed scheme generally has smaller public key sizes compared to binary Goppa code based McEliece encryption schemes. Furthermore, the proposed schemes could use any linear codes such as (generalized) Reed-Solomon code, LDPC code, Turbo code, or Polar code. Heuristics and experiments encourages us to think that the proposed schemes are immune against existing attacks on linear code based encryption schemes such as Sidelnikov-Shestakov attack, filtration attacks, and algebraic attacks. For related documents, see Wang [31, 32].

References

[1] M. Baldi, M. Bodrato, and F. Chiaraluce. A new analysis of the mceliece cryptosystem based on qc-ldpc codes. In *Security and Cryptography for Networks*, pages 246–262. Springer, 2008.

[2] M. Bardet, J. Chaulet, V. Dragoi, A. Otmani, and J.-P. Tillich. Cryptanalysis of the McEliece public key cryptosystem based on polar codes. In *International Workshop on Post-Quantum Cryptography*, pages 118–143. Springer, 2016.

Table 2: Padding parameters (odd-ID schemes use $w = n - k$ and even-ID schemes use $w = \frac{n-k}{2}$): bE for basicEncoding, mE for mediumEncoding and aE for advancedEncoding

ID	κ_c	κ_q	n	k	t	m	sys `sk`	`sk`	`pk`		`mLen`	RLCEspad		RLCEpad	
												$k_1(k_2)$	k_3	k_1	$k_2(k_3)$
1	60	59	255	155	50	**8**	32467	56802	31001	bE	1240	38	79	135	10
										mE	1640	50	105	185	10
										aE	1848	55	122	211	10
2	60	59	200	120	40	**8**	15402	30042	14401	bE	960	30	60	100	10
										mE	1280	38	84	140	10
										aE	1436	44	92	160	10
3	80	70	360	200	80	**9**	74308	119708	72001	bE	1800	56	113	195	15
										mE	2520	75	165	285	15
										aE	2842	85	185	326	15
4	80	70	300	140	80	**9**	39580	61910	37801	bE	1260	39	80	128	15
										mE	1980	60	128	218	15
										aE	2262	70	143	243	15
5	**128**	**96**	600	464	68	**10**	160767	430815	157761	bE	4640	145	290	516	32
										mE	5320	160	345	601	32
										aE	5647	170	365	641	32
6	**128**	**96**	511	381	65	**9**	85864	249932	83583	bE	3429	107	215	365	32
										mE	4014	125	252	438	32
										aE	4306	134	270	475	32
7	128	100	600	440	80	**12**	214663	505943	211201	bE	5280	165	330	596	32
										mE	6240	190	400	716	32
										aE	6608	200	427	763	32
8	128	100	502	378	62	**12**	107966	323048	105463	bE	4536	141	285	503	32
										mE	5280	160	340	596	32
										aE	5561	170	356	632	32
9	160	115	780	580	100	**10**	294088	715748	290001	bE	5800	181	363	645	40
										mE	6800	210	430	770	40
										aE	7265	220	469	829	40
10	160	115	620	440	90	**10**	151508	394388	148501	bE	4400	137	276	470	40
										mE	5300	160	343	583	40
										aE	5689	170	372	632	40
11	160	117	760	540	110	**12**	360933	799413	356401	bE	6480	202	406	730	40
										mE	7800	240	495	895	40
										aE	8296	250	538	958	40
12	160	117	620	440	90	**12**	181453	472733	178201	bE	5280	165	330	580	40
										mE	6360	190	415	715	40
										aE	6749	200	444	764	40
13	**192**	**132**	1000	790	105	**10**	419630	1201335	414751	bE	7900	246	496	892	48
										mE	8950	270	579	1023	48
										aE	9464	290	604	1088	48
14	**192**	**132**	800	600	100	**10**	228703	679903	225001	bE	6000	187	376	654	48
										mE	7000	215	445	779	48
										aE	7452	230	472	836	48
15	192	135	990	780	105	**12**	496653	1410813	491401	bE	9360	292	586	1074	48
										mE	10620	330	668	1232	48
										aE	11133	345	702	1296	48
16	192	135	790	590	100	**12**	269468	792798	265501	bE	7080	221	443	789	48
										mE	8280	255	525	939	48
										aE	8731	270	552	996	48
17	**256**	**165**	1300	800	250	**11**	1108453	1990053	1100001	bE	8800	275	550	980	60
										mE	11550	360	724	1324	60
										aE	12596	390	795	1455	60
18	**256**	**165**	1023	663	180	**10**	452832	1003620	447526	bE	6630	207	415	709	60
										mE	8430	260	534	934	60
										aE	9162	285	576	1026	60
19	256	167	1300	800	250	**12**	1208803	2170403	1200001	bE	9600	300	600	1080	60
										mE	12600	390	795	1455	60
										aE	13646	425	856	1586	60
20	256	167	1023	663	180	**12**	542773	1203453	537031	bE	7956	248	499	875	60
										mE	10116	260	745	1145	60
										aE	10848	285	787	1237	60

Table 3: RLCE performance on MacOS 2.4GHz Intel Core i5

ID	sec/key	seconds/encryption		KB per/sec		seconds/decryption		KB/sec	
		RLCEspad	RLCEpad	RLCEspad	RLCEpad	RLCEspad	RLCEpad	RLCEspad	RLCEpad
1	0.059447	0.004360	0.004599	79.426	278.611	0.009634	0.00967	35.944	132.454
2	0.027372	0.002630	0.003003	100.060	322.892	0.006711	0.006857	39.213	141.400
3	0.158536	0.007747	0.007864	67.048	250.983	0.024781	0.025218	20.961	78.269
4	0.073822	0.004287	0.004810	96.936	313.905	0.028472	0.029469	14.595	51.233
5	1.038965	0.018558	0.018448	59.709	225.630	0.037294	0.038537	29.713	108.009
6	0.596978	0.011261	0.011613	76.876	261.217	0.022252	0.022106	38.904	137.222
7	1.025386	0.019920	0.019850	66.059	249.816	0.142494	0.152560	9.235	32.504
8	0.645071	0.013388	0.012052	82.771	342.491	0.112433	0.115949	9.856	35.599
9	2.369620	0.031716	0.031144	45.856	171.230	0.070691	0.070083	20.574	76.092
10	1.119982	0.016762	0.016368	66.109	246.678	0.053734	0.050209	20.622	80.417
11	2.141296	0.029142	0.028558	57.037	217.050	0.204380	0.204203	8.132	30.354
12	1.140381	0.017647	0.017313	74.567	286.019	0.164515	0.161873	7.998	30.591
13	5.370898	0.046863	0.047414	39.902	149.428	0.083115	0.078595	22.498	90.145
14	2.451222	0.027551	0.028979	54.045	186.175	0.059952	0.061553	24.837	87.650
15	4.904205	0.044599	0.048186	51.245	177.071	0.193810	0.199046	11.792	42.866
16	2.436107	0.025108	0.026074	70.339	249.415	0.178947	0.182934	9.869	35.549
17	9.316449	0.087165	0.086902	28.604	105.516	0.312599	0.309396	7.976	29.637
18	4.677360	0.042546	0.043558	42.323	148.506	0.130921	0.129737	13.754	49.859
19	9.410073	0.086897	0.086147	31.083	116.972	0.532154	0.516498	5.076	19.510
20	4.794430	0.042819	0.044252	42.053	179.198	0.344813	0.338056	5.222	23.457

[3] D.J. Bernstein. Grover vs. McEliece. In *International Workshop on Post-Quantum Cryptography*, pages 73–80. Springer, 2010.

[4] D.J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *Post-Quantum Cryptography*, pages 31–46. Springer, 2008.

[5] T.A Berson. Failure of the mceliece public-key cryptosystem under message-resend and related-message attack. In *Proc Crypto*, pages 213–220. Springer, 1997.

[6] D. Boneh. Simplified OAEP for the RSA and rabin functions. In *Advances in Cryptology–CRYPTO 2001*, pages 275–291. Springer, 2001.

[7] A. Canteaut and N. Sendrier. Cryptanalysis of the original mceliece cryptosystem. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 187–199. Springer, 1998.

[8] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT 2000*, pages 392–407. Springer, 2000.

[9] A. Couvreur, P. Gaborit, V. Gauthier-Umaña, A. Otmani, and J.-P. Tillich. Distinguisher-based attacks on public-key cryptosystems using Reed–Solomon codes. *Designs, Codes and Cryptography*, pages 1–26, 2013.

[10] A. Couvreur, A. Otmañi, and J.-P. Tillich. Polynomial time attack on wild McEliece over quadratic extensions. In *Advances in Cryptology–EUROCRYPT 2014*, pages 17–39. Springer, 2014.

[11] T. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19(1):73–77, 1973.

Table 4: RLCE performance on DELL Optiplex 9010 Intel(R) Core(TM) i7-3770 CPU@3.40GHz 16GB RAM

ID	sec/key	seconds/encryption		KB per/sec		seconds/decryption		KB/sec	
		RLCEspad	RLCEpad	RLCEspad	RLCEpad	RLCEspad	RLCEpad	RLCEspad	RLCEpad
1	0.046667	0.000488	0.000522	100.140	346.100	0.000956	0.000972	51.065	185.907
2	0.020667	0.000284	0.000322	130.483	424.856	0.000669	0.000684	55.487	199.764
3	0.125000	0.000881	0.000831	88.116	315.771	0.002397	0.002450	30.558	113.600
4	0.057333	0.000481	0.000522	121.766	407.836	0.002278	0.002300	25.722	92.561
5	0.812333	0.001981	0.002034	78.858	288.523	0.003506	0.003538	44.564	165.907
6	0.479000	0.001294	0.001337	94.350	319.825	0.002103	0.002134	58.040	200.400
7	0.791667	0.002084	0.002156	89.017	324.253	0.013056	0.013047	14.211	53.593
8	0.484667	0.001325	0.001334	117.925	436.240	0.010066	0.009969	15.523	58.385
9	1.828000	0.003275	0.003337	62.619	225.311	0.005738	0.005719	35.743	131.488
10	0.875000	0.001837	0.001894	85.039	300.632	0.004850	0.004850	32.216	117.389
11	1.698000	0.003244	0.003272	72.258	267.138	0.018522	0.018541	12.654	47.141
12	0.885333	0.001856	0.001900	99.950	367.496	0.014722	0.014781	12.604	47.239
13	3.968667	0.005044	0.005125	52.276	194.924	0.006378	0.006450	41.340	154.887
14	1.973667	0.002900	0.002972	72.400	255.987	0.005588	0.005591	37.576	136.075
15	3.864667	0.005063	0.005072	63.656	237.209	0.017959	0.017997	17.944	66.852
16	1.922000	0.002894	0.002916	86.060	314.512	0.016491	0.016588	15.100	55.282
17	7.229333	0.009769	0.009775	35.989	132.273	0.028684	0.028566	12.256	45.263
18	3.703333	0.004672	0.004800	54.349	190.023	0.011881	0.011950	21.370	76.327
19	7.349000	0.009928	0.009931	38.362	143.074	0.047234	0.047344	8.063	30.012
20	3.755000	0.004778	0.004822	53.141	231.898	0.031506	0.031650	8.059	35.329

[12] J.-C. Faugere, V. Gauthier-Umaña, A. Otmani, L. Perret, and J.-P. Tillich. A distinguisher for high-rate mceliece cryptosystems. *IEEE Tran. Information Theory*, 59(10):6830–6844, 2013.

[13] J.-C. Faugere, A. Otmani, L. Perret, and J.-P. Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Eurocrypt 2010*, pages 279–298. Springer, 2010.

[14] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Annual International Cryptology Conference*, pages 537–554. Springer, 1999.

[15] C. Hall, I. Goldberg, and B. Schneier. Reaction attacks against several public-key cryptosystem. In *International Conference on Information and Communications Security*, pages 2–12. Springer, 1999.

[16] K. Kobara and H. Imai. Semantically secure mceliece public-key cryptosystems-conversions for mceliece pkc. In *International Workshop on Public Key Cryptography*, pages 19–35. Springer, 2001.

[17] P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *EUROCRYPT'88*, pages 275–280. Springer, 1988.

[18] J. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Information Theory*, 34(5):1354–1359, 1988.

[19] P. Loidreau and N. Sendrier. Some weak keys in mceliece public-key cryptosystem. In *IEEE International symposium on Information Theory*, pages 382–382. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1998.

[20] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44):114–116, 1978.

[21] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE ISIT 2013*, pages 2069–2073, 2013.

[22] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Control and Information Theory*, 15(2):159–166, 1986.

[23] C. Peters. Information-set decoding for linear codes over F_q. In *Post-Quantum Cryptography*, pages 81–94. Springer, 2010.

[24] D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *International Workshop on Public Key Cryptography*, pages 129–146. Springer, 2000.

[25] E. Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Information Theory*, 8(5):5–9, 1962.

[26] N. Sendrier. Encoding information into constant weight words. In *Proc. ISIT 2005*, pages 435–438. IEEE, 2005.

[27] V. Shoup. OAEP reconsidered. In *CRYPTO 2001*, pages 239–259. Springer, 2001.

[28] V. M Sidelnikov and S. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics and Applications*, 2(4):439–444, 1992.

[29] J. Stern. A method for finding codewords of small weight. In *Coding theory and applications*, pages 106–113. Springer, 1989.

[30] Y. Wang. Quantum resistant random linear code based public key encryption scheme RLCE. In *Proc. IEEE ISIT*, pages 2519–2523, July 2016.

[31] Y. Wang. Quantum resistant encryption scheme RLCE and IND-CCA2 security for McEliece schemes. In *Sumbitted*, pages ??–??, July 2017.

[32] Yongge Wang. Decoding generalized reed-solomon codes and its application to RLCE encryption schemes. *arXiv preprint: https://arxiv.org/abs/1702.07737*, 2017.