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
April 11, 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 classical security levels of 128, 192, and 256 and for the quantum security levels of 85, 100, 120, and 150 are recommended. 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.

1 Introduction

Since McEliece encryption scheme [22] 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, 23], Wang’s RLCE [31], and the original binary Goppa code based McEliece encryption scheme.

Recently, Wang’s RLCE [31] presents a systematic approach of designing public key encryption schemes using any linear code. For example, one can use generalized Reed-Solomon (GRS) codes to design McEliece based RLCE encryption scheme. In this paper, we propose 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. Let \( k < n \leq q \). The generalized Reed-Solomon code \( \text{GRS}_k(x, y) \) of dimension \( k \) is defined as

\[
\text{GRS}_k(x, y) = \{(y_0p(x_0), \cdots, y_{n-1}p(x_{n-1})) : p(x) \in GF(q)[x], \deg(p) < k\}
\]

where \( x = (x_0, \cdots, x_{n-1}) \in GF(q)^n \) is an \( n \)-tuple of distinct elements and \( y = (y_0, \cdots, y_{n-1}) \in GF(q)^n \) is an \( n \)-tuple of nonzero (not necessarily distinct) elements.

## 2 McEliece, Niederreiter, and RLCE Encryption schemes

For given parameters \( n, k \) and \( t \), the McEliece scheme [22] 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(\( G, m, e \)).** For a message \( m \in \{0, 1\}^k \), choose a random vector \( e \in \{0, 1\}^n \) of weight \( t \) and compute the cipher text \( c = mG + e \).

**Mc.Dec(\( S, G_s, P, e \)).** For a received ciphertext \( c \), first compute \( c' = cP^{-1} = mSG \). Next use an error-correction algorithm to recover \( m' = cS \) and compute the message \( m = m'S^{-1} \).

For given parameters \( n, k \), and \( t \), the Niederreiter’s scheme [24] 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(\( H, m \)).** For a message \( m \in GF(q)^n \) of weight \( t \), compute the cipher text \( c = mH_T \) of length \( n - k \).

**Nied.Dec(\( S, H_s, P, e \)).** For a received ciphertext \( c = mH_T \) of weight \( t \), compute \( c(S_T)^{-1} = mP_T H_s^T \). Use an error-correction algorithm to recover \( m' = mP_T \) and compute the message \( m = m'(P_T)^{-1} \).

The protocol for the RLCE Encryption scheme by Wang [31] 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 \( G_s = [g_0, \cdots, g_{n-1}] \) 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

\[
G_1 = [g_0, C_0, g_1, C_1, \cdots, g_{n-1}, C_{n-1}]
\]  

(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(\( G, m, e \)).** For a row vector message \( m \in GF(q)^k \), choose a random row vector \( e = [e_0, \cdots, e_{n(r+1)-1}] \in GF(q)^{n(r+1)} \) such that the Hamming weight of \( e \) is at most \( t \). The cipher text is \( c = mG + e \).
RLCE.\texttt{Dec}(S, G_s, P, A, \mathbf{c}). For a received ciphertext \( \mathbf{c} = [c_0, \ldots, c_{n(r+1)−1}] \), compute

\[
\mathbf{c}P^{-1}A^{-1} = \mathbf{m}S\mathbf{G}_1 + \mathbf{e}P^{-1}A^{-1} = [c'_0, \ldots, c'_{n(r+1)−1}]
\]

where \( A^{-1} = \text{diag}[A_{−1}, \ldots, A_{n−1}^−1] \). Let \( \mathbf{c}' = [c'_0, c'_{r+1}, \ldots, c'_{(n−1)(r+1)}] \) be the row vector of length \( n \) selected from the length \( n(r+1) \) row vector \( \mathbf{c}P^{-1}A^{-1} \). Then \( \mathbf{c}' = \mathbf{m}S\mathbf{G}_1 + \mathbf{e}' \) for some error vector \( \mathbf{e}' \in GF(q)^n \). Let \( \mathbf{e}'' = \mathbf{e}P^{-1} = [e''_0, \ldots, e''_{n(r+1)−1}] \) and \( \mathbf{e}''' = [e''_{r+1}, \ldots, e''_{(n−1)(r+1)}] \) be a sub-vector of \( \mathbf{e}'' \) for \( i \leq n−1 \). Then \( \mathbf{e}'[i] \) is the first element of \( \mathbf{e}'''A_1^{-1} \). Thus \( \mathbf{e}'[i] \neq 0 \) only if \( \mathbf{e}'' \) is non-zero. Since there are at most \( t \) non-zero sub-vectors \( \mathbf{e}''' \), 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 \( w = \text{wt}(\mathbf{e} - \mathbf{m}G) \). If \( w \leq t \) 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 McEliece 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.

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

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

\[
H_1 = [\mathbf{h}_0, C_0, C_1, \ldots, \mathbf{h}_{n−1}, C_{n−1}]
\]

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

2. Let \( A_0, \ldots, A_{n−1} \in GF(q)^{(r+1)\times (r+1)} \) be dense nonsingular \((r+1) \times (r+1)\) matrices chosen uniformly at random and let \( A = \text{diag}[A_0, \ldots, A_{n−1}] \) 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 \( \mathbf{c}(S^T)^{-1} = \mathbf{m}P^T A^T H_1^T \). Since the weight of \( \mathbf{m}P^T A^T \) 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 \( C_0, C_1, \ldots, C_{n−1} \) 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 \( G_s = [\mathbf{g}_0, \ldots, \mathbf{g}_{n−1}] \) be the \( k \times n \) generator matrix for the linear code \( C \) such that \( G_s H_1^T = 0 \). Furthermore, let \( D_0, D_1, \ldots, D_{n−1} \) 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 $G_1 = [g_0, D_0, \cdots, g_{n-1}, D_{n-1}]$, and $G = G_1(A^T)^{-1}(P^T)^{-1}$. Then

$$GH^T = G_1(A^T)^{-1}(P^T)^{-1}P^T A^T H_1^T S^T = G_1 H_1^T = 0.$$  

For a received ciphertext $c$ with $c(S^T)^{-1} = mP^T A^T H_1^T$, one can find a vector $a \in GF(q)^w$ such that $c(S^T)^{-1} = ah^T$. Then we have $(a - mP^T A^T)H_1^T = 0$. 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 $a - mP^T A^T$ 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 [31] is that the revised scheme inserts random columns after randomly selected columns in the generator matrix. Specifically the revised RLCE scheme proceeds as follows.

**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 $G_sP_1 = [g_0, \cdots, g_{n-1}]$.

1. Let $r_0, r_1, \cdots, r_{w-1} \in GF(q)^k$ be column vectors drawn uniformly at random and let

   $$G_1 = [g_0, \cdots, g_{n-w}, r_0, \cdots, g_{n-1}, r_{w-1}]$$

   be the $k \times (n + w)$ matrix obtained by inserting column vectors $r_i$ into $G_s$.

2. Let $A_0, \cdots, A_{w-1} \in GF(q)^{2 \times 2}$ be dense nonsingular $2 \times 2$ 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**($G, m, e$). For a row vector message $m \in GF(q)^k$, choose a random row vector $e = [e_0, \cdots, e_{n+w-1}] \in GF(q)^{n+w}$ such that the Hamming weight of $e$ is at most $t$. The cipher text is $c = mG + e$.  

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

$$cP_2^{-1}A^{-1} = mS G_1 + eP_2^{-1}A^{-1} = [c'_0, \cdots, c'_{n+w-1}].$$

Let $c' = [c'_0, c'_1, \cdots, c'_{n-w}, c'_{n-w+2}, \cdots, c'_{n+w-2}]$ be the row vector of length $n$ selected from the length $n + w$ row vector $cP_2^{-1}A^{-1}$. Then $cP_2^{-1} = mS G_s + e'$ for some error vector $e' \in GF(q)^n$ where the Hamming weight of $e' \in GF(q)^n$ is at most $t$. Using the efficient decoding algorithm, one can compute $m' = mS$ and $e' = mS^{-1}$. Finally, calculate the Hamming weight $w = \text{wt}(c - mG)$. If $w \leq t$ then output $m$ as the decrypted plaintext. Otherwise, output error.
Remark 1. If \( w = n \), 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.

Remark 2. If the \((n + w) \times (n + w)\) matrix \( A \) is taken as the identity matrix \( I_{(n+w)\times(n+w)} \), then the revised RLCE scheme is the same as the Wieschebrink’s encryption scheme [33].

Remark 3. It is sufficient to only include the first column of \( A_i^{-1} \) for each \( i = 0, \ldots, w \) in the private key.

5 Systematic Decoding RLCE schemes

In the revised RLCE encryption scheme, one first recovers \( m' = mS \) and then calculates the message \( m = m'S^{-1} \). To simplify the message recovering process, one can use echelon format public keys and restrict the permutation \( P_2 \) to the last \( n + w - k \) columns of \( SG_1A \). In case that \( w \leq n - k \), this implies that the matrix \( SG_1P_1 \) is in echelon format. Thus \( S \) is the matrix that converts \( G,P_1 \) to echelon format. By requiring \( w \leq n - k \), the RLCE scheme remains unchanged except the following revised decryption process.

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

\[
eP_2^{-1}A^{-1} = mSG_1 + eP_2^{-1}A^{-1} = [c'_0, \ldots, c'_{n+w-1}] = [c_0, \ldots, c_{k-1}, c'_k, \ldots, c'_{n+w-1}].
\]

Let \( c' = [c_0, c_1, \ldots, c_{k-1}, \ldots, c'_{n-w}, c'_{n-w+2}, \ldots, c'_{n+w-2}] \) be the row vector of length \( n \) selected from the length \( n + w \) row vector \( eP_2^{-1}A^{-1} \). Then \( eP_2^{-1} = mSG_1 + e' \) for some error vector \( e' \in GF(q)^n \) where the Hamming weight of \( e' \in GF(q)^n \) is at most \( t \). Using the efficient decoding algorithm, one recovers \( e' \) from \( eP_2^{-1} \).

Since \( e' = mSG_1P_1 + e'P_1 \) and \( SG_1P_1 \) is in echelon format, the message \( m \) equals to the first \( k \) elements in the vector \( c' - e'P_1 \). Finally, calculate the Hamming weight \( w = \text{wt}(c - mG) \). If \( w \leq t \) then output \( m \) as the decrypted plaintext. Otherwise, output error.

Remark. For systematic decoding RLCE scheme, one does not need to include \( S^{-1} \) within the private key.

6 Security analysis

Loidreau and Sendrier [21] pointed out some weak keys for binary Goppa code based McEliece schemes and similar weak keys for RLCE schemes should not be used. For RLCE schemes, one can obtain a valid ciphertext for a message \( m + m' \) by letting \( c' = c + m'G \) without knowing the message \( m \). This kind of attacks could be defeated by using IND-CCA2-secure message padding schemes which will be discussed in this paper. Faugère, Otmani, Perret, and Tillich [9] developed an algebraic attack against quasi-cyclic and dyadic structure based compact variants of McEliece encryption scheme. Wang [31] showed that the algebraic attacks will not work against the RLCE encryption scheme. A straightforward modification of the analysis in [31] can be used to show that the algebraic attacks will not work against the revised RLCE scheme either. In the following sections, we carry out heuristic security analyses on the revised RLCE scheme. We first show how to choose appropriate parameters \( n, k, w \) to defeat Sidelnikov-Shestakov attacks and filtration attacks against RLCE schemes.

6.1 Sidelnikov-Shestakov’s attack

Niederreiter’s scheme [24] replaces the binary Goppa codes in McEliece scheme by GRS codes. Sidelnikov and Shestakov [29] broke Niederreiter scheme by recovering an equivalent private key \((x', y')\) from a public key \( G \) for the code \( \text{GRS}_k(x, y) \). For the given public key \( G \), one computes the echelon form \( E(G) = [I|G'] \)
using Gaussian elimination.

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

(4)

Assume the \(i\)th row codeword \(b_i\) of \(E(G)\) encodes a message \(p_i(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}\). Then

\[
y_0p_i(x_0) = 0, \ldots, y_i p_i(x_j) = 1, \ldots, y_{n-1} p_i(x_{n-1}) = b_{i,n-1}
\]

(5)

Since the only non-zero elements are \(b_{i,j}, b_{i,k+1}, \ldots, b_{i,n-1}, p_i\) can be written as

\[
p_i(x) = c_i \cdot \prod_{j=1, j\neq i}^k (x - x_j)
\]

(6)

for some \(c_i \neq 0\). By the fact that \(\text{GRS}_k(x, y) = \text{GRS}_k(ax + b, cy)\) for all \(a, b, c \in GF(q)\) with \(ab \neq 0\), we may assume that \(x_0 = 0\) and \(x_1 = 1\). In the following, we try to recover \(x_2, \ldots, x_{n-1}\). Using equation (6), one can divide the row entries in (4) 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

\[
\frac{b_{i_0,j}}{b_{i_1,j}} = \frac{y_j p_{i_0}(x_j)}{y_j p_{i_1}(x_j)} = \frac{c_{i_0}(x_j - x_{i_0})}{c_{i_1}(x_j - x_{i_1})}
\]

(7)

for \(j = k, \ldots, n-1\). First, by taking \(i_0 = 0\) and \(i_1 = 1\), equation (7) could be used to recover \(x_k, \ldots, x_{n-1}\) by guessing the value of \(\frac{c_0}{c_i}\) which is possible when \(q\) is small. By letting \(i_0 = 0\) and \(i_1 = 2, \ldots, k-1\) respectively, equation (7) could be used to recover \(x_{i_1}\). Sidelnikov and Shestakov [29] showed that the values of \(y\) can then be recovered by solving some linear equation systems based on \(x_0, \ldots, x_{n-1}\).

In the RLCE scheme, \(2w\) columns of the public key matrix \(G\) are randomized. In case that the filtration attack in the next Section can identify the \(n - w\) non-randomized columns, one can permute the columns of \(G\) to obtain a new matrix \(G_N\) such that the first \(n - w\) columns are the non-randomized columns. Then one can compute an echelon form \(E(G_N)\) for \(G_N\). Since the last \(2w\) columns are randomized, they could not be used to establish any of the equations in Sidelnikov and Shestakov attack. We distinguish the following two cases:

1. If \(w \geq n - k\), then one cannot establish enough equations within (5) to obtain the equation (6). Thus no equations in (7) could be established and Sidelnikov and Shestakov attack could not continue.

2. If \(n - k > w\), equations (7) may only be used to recover the values of \(x_0, \ldots, x_{n-w-1}\). If it has a negligible probability for one to guess the remaining values \(x_{n-w}, \ldots, x_{n-1}\), then Sidelnikov and Shestakov attack will not be successful. The probability for one to guess the remaining values \(x_{n-w}, \ldots, x_{n-1}\) correctly is bounded by \(1/(q^{n-w+1})w!\).

Thus for a security parameter \(\kappa\), the RLCE parameters should be chosen in such a way that

\[
w \geq n - k \text{ or } \left(\frac{q-n+w+1}{w}\right)w! \geq 2^\kappa
\]

(8)
6.2 Filtration attacks

Couvreur et al. [7] designed a filtration technique to attack GRS code based McEliece scheme. For two codes \( C_1 \) and \( C_2 \) of length \( n \), the star product code \( C_1 \ast C_2 \) is the vector space spanned by \( a \ast b \) for all pairs \( (a, b) \) in \( C_1 \times C_2 \) where \( a \ast b = [a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}] \). For the square code \( C^2 = C \ast C \) of \( C \), we have \( \dim C^2 \leq \min \{n, \binom{\dim C + 1}{2} \} \). For an \([n, k] \) GRS code \( C \), let \( a, b \in \text{GRS}_k(x, y) \) where \( a = (y_0p_1(x_0), \cdots, y_{k-1}p_1(x_{n-1})) \) and \( a = (y_0p_2(x_0), \cdots, y_{k-1}p_2(x_{n-1})) \). Then \( a \ast b = (y_0^2p_1(x_0)p_2(x_0), \cdots, y_{k-1}^2p_1(x_{n-1})p_2(x_{n-1})) \). Thus GRS\(_2\) \((x, y)^2 \subseteq \text{GRS}_{2k-1}(x, y * y) \) where we assume \( 2k - 1 \leq n \). This property has been used in [7] to recover the non-random columns in Wieschebrink’s public key [33].

Let \( G \) be the generator matrix for an \([n, k, d, t, w] \) RLCE encryption scheme based on a GRS code. Let \( C \) be the code generated by the rows of \( G \). Let \( D_1 \) be the code with a generator matrix \( D_1 \) obtained from \( G \) by replacing the randomized \( 2w \) columns with all-zero columns and let \( D_2 \) be the code with a generator matrix \( D_2 \) obtained from \( G \) by replacing the \( n-w \) non-randomized columns with zero columns. Since \( C \subseteq D_1 + D_2 \) and the pair \( (D_1, D_2) \) is an orthogonal pair, we have \( C^2 \subseteq D_1^2 + D_2^2 \). It follows that

\[
2k - 1 \leq \dim C^2 \leq \min \{\lfloor 2k - 1, n-w \rfloor + 2w, n+w \}.
\]

In case that \( k \geq n-w \), no matter whether the filtration technique can recover the non-randomized \( n-w \) GRS columns or not, the recovered non-randomized GRS columns are useless for the adversary since the \( n-w \) GRS columns are equivalent to a basis of a \( n-w \) dimensional subspace. In the following, we consider filtration attacks against RLCE schemes with \( k < n-w \).

First assume that \( \min \{2k - 1, n-w \} + 2w \leq n+w \) and \( 2k \leq n-w \). That is, \( n \geq 2k - 1 + w \). Let \( C_i \) be the punctured \( C \) code at position \( i \). We distinguish the following two cases:

- Column \( i \) of \( G \) is a randomized column. In this case, the expected dimension for \( C_i^2 \) is \( 2k+2w-2 \).
- Column \( i \) of \( G \) is a non-randomized column. In this case, the expected dimension for \( C_i^2 \) is \( 2k+2w-1 \).

This shows that if \( n \geq 2k-1+w \), then the filtration techniques could be used to identify the randomized columns within the public key \( G \). For the case of \( n < 2k-1+w \), one can use the shortened code based filtration techniques in Couvreur et al. [7] to recover the non-randomized \( n-w \) GRS columns.

Once non-randomized \( n-w \) GRS columns are recovered, one can use Sidelnikov-Shestakov attack to compute an equivalent private key for the underlying \([n, k] \) GRS\(_k\) code. The condition (8) in the preceding section shows how to choose RLCE parameters in such a way that it is computationally infeasible to use Sidelnikov-Shestakov attacks to calculate an equivalent private key for the underlying \([n, k] \) GRS\(_k\) code.

Alternatively, one may use Sidelnikov-Shestakov attacks to calculate a private key for the punctured \([n-w, k] \) GRS\(_k\) code consisting of the recovered non-randomized GRS columns and then list-decode the punctured \([n-w, k] \) GRS\(_k\) code. We first review some results for GRS list-decoding. The error distance of a received word \( y \in GF(\mathbb{Q})^n \) to a code \( C \) is defined as \( \Delta(y, C) = \min \{\text{wt}(y-x) : x \in C \} \). For a vector \( y \in GF(\mathbb{Q})^n \), \( y \)’s Hamming ball of radius \( r \) is \( B(y; r) = \{y' : \text{wt}(y-y') \leq r \} \). For an MDS \([n, k, d] \) code \( C \) and a vector \( y \in GF(\mathbb{Q})^n \), \( B(y; r) \) contains at most one codeword from \( C \) if \( r \leq d/2 \). If \( d/2 < r \leq n - \sqrt{n(k-1)} \), \( B(y; r) \cap C \) contains at most polynomial many elements and the list-decoding algorithm by Guruswami and Sudan [13] can be used to efficiently output all elements in \( B(y; r) \cap C \). If the radius is stretched further, \( B(y; r) \cap C \) may contain exponentially many codewords.

For an RLCE ciphertext \( \mathbf{c} \), let \( \mathbf{c}' \) be the punctured ciphertext of length \( n-w \) by restricting \( \mathbf{c} \) to the punctured \([n-w, k] \) GRS\(_k\) code. In case that there are at most \( n-w - \sqrt{(n-w)(k-1)} \) errors in \( \mathbf{c}' \), one can decode the shortened \([n-w, k] \) GRS\(_k\) code using the list-decoding algorithm by Guruswami and Sudan [13]. Note that the probability for \( \mathbf{c}' \) to contain at most \( n-w - \sqrt{(n-w)(k-1)} \) errors is bounded by the
hypergeometric cumulative distribution function

\[ P_{n,w,t} = \sum_{i=0}^{n-w} \binom{n-w}{i} \binom{2w}{i-t} \binom{n+i}{t} \]  

Thus the parameters should be chosen in such a way that for the given security parameter \( \kappa \),

\[ w \geq n - k \text{ or } P_{n,w,t} \leq 2^{-\kappa}. \]  

Justesen and Hoholdt [15] showed the following theorem.

**Theorem 6.1** (Justesen and Hoholdt [15]) For an \([n,k]\) Reed-Solomon code \( C \) and an integer \( \delta < n - k \), the expected size of \( B(\mathbf{u}; \delta) \cap C \) is \( \binom{n-\delta}{n} / q^{n-\delta-k} \) for randomly chosen \( \mathbf{u} \in GF(q^n) \).

By theorem 6.1, we may further require that the RLCE parameters are chosen in such a way that the expected size of \( B(\mathbf{u}; \delta) \cap C \) is large than \( 2^\kappa \), where \( \delta \) is the number of expected errors that the ciphertext \( c' \) contains. However, this may not be necessary following Guruswami and Rudra’s result [12] that GRS list decoding algorithm that solves the general problem of polynomial reconstruction (like the Guruswami-Sudan algorithm) cannot go beyond the square root bound.

### 6.3 Information-Set Decoding

As mentioned in the introduction, the most powerful message 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 [25], which integrated optimized Lee-Brickell’s algorithm [19], Stern’s algorithm [30], and Leon’s minimum-weight-word-finding algorithm [20]. Peters’s attack [25] also integrated analysis techniques for information-set decoding attacks on binary McEliece scheme discussed in [3]. 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.

### 6.4 Known partial plaintext [6]

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

\[ c = mG + e = [m_l, m_r] \begin{bmatrix} G_l \\ G_r \end{bmatrix} + e = m_l G_l + m_r G_r + e. \]  

Thus if one knows the value of \( m_r \), the identity (12) becomes \( c - m_r G_l = m_l G_l + e \) which could be much easy to decode than the original codeword \( c \) since \( r < k \). The known-partial-plaintext-attack could be defeated using appropriate message padding for IND-CCA2-security that will be discussed in Section 7.
6.5 Related message attack [4]

Assume that \( c_1 = m_1G + e_1 \) and \( c_2 = m_2G + e_2 \). Furthermore, assume that the adversary knows the relation between \( m_1 \) and \( m_2 \). For example, assume that \( m = m_1 + m_2 \) and that the adversary knows the value of \( m \). Then we have \( c_1 + c_2 - mG = e_1 + e_2 \). Since \( e_1 \) and \( e_1 \) are different and both of them have low weight \( r \), it could be easy for the adversary to recover both \( e_1 \) and \( e_1 \) by trying all combinations. Even if one cannot enumerate all combinations to recover either \( e_1 \) or \( e_1 \), one can use the 0 entries within \( e_1 + e_2 \) as a hint to speed up the information set decoding algorithm for recovering \( m_1 \) from \( c_1 = m_1G + e_1 \). 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 appropriate message padding for IND-CCA2 security that will be discussed in Section 7.

6.6 Reaction attack [14]

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 \( c \), the adversary first randomly selects positions to add errors until the decryption oracle complains. That is, the adversary first obtains a ciphertext \( c' \) 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 \( c' \) 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 \( c \). These error-free positions could be used to recover the plaintext message for the ciphertext \( c \). The reaction-attack could be defeated using appropriate message padding for IND-CCA2 security that will be discussed in Section 7.

6.7 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 information within the vector \( m \in GF(q)^k \) and the ciphertext is \( c = mG + e \). 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 \( e \). That is, let \( e_1, \ldots, e_i \in GF(q) \setminus \{0\} \) be the non-zero elements within \( e \) and encode further information within \( e_1, \ldots, e_i \). 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 \( e_j \) cannot be zeros.

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

The basicEncoding approach is straightforward. For the mediumEncoding, after one recovers the vector \( m \), one needs to compute \( mG - c \) to obtain the values of \( e_1, \ldots, e_i \). For the advancedEncoding approach, we need to compute an invertible function

\[
\varphi : W_{n+w,f} \leftrightarrow \left\{ i : 1 \leq i \leq \binom{n+w}{t} \right\}
\]

where \( W_{n+w,f} \subseteq GF(2)^{n+w} \) is the set of all \((n+w)\)-bit binary string of weight \( t \). For the invertible function \( \varphi \) in (13), one may use the enumerative source encoding construction in Cover [8]:

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

where \( \varphi(i_1, \ldots, i_t) = \binom{i_t}{1} + \cdots + \binom{i_1}{1} \) and \( 0 \leq i_1 < i_2 < \cdots < i_t < n + w \) are the positions of ones. The function \( \varphi \) could be evaluated with the cost of \( O \left( \left( \log_2 \left( \binom{n+w}{t} \right) \right)^2 \right) \) operations (see, e.g., Sendrier [27]).

### 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 [18] recommended the use of Pointcheval’s generic conversion [26] or Fujisak-Okamoto’s generic conversion [10] 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**: \( c = \text{Mc.Enc}(G, r_1, H_1(m||r_2))|| (H_2(r_1) \oplus (m||r_2)) \).
- **Fujisak-Okamoto padding**: \( c = \text{Mc.Enc}(G, r_1, H_1(m||r_1))|| (H_2(r_1) \oplus m) \).
- **Kobara-Imai’s \( \alpha \)-padding**: \( c = \text{Mc.Enc}(G, y_1, H_1(m||r_1))|| y_2 \) where \( y_1||y_2 = H_2(H_1(m||r_1)) \oplus (m||r_1) \).
- **Kobara-Imai’s \( \beta \)-padding**: \( c = y_1||\text{Mc.Enc}(G, y_2, H_1(r_1)) \) where \( y_1||y_2 = (r \oplus H_1(H_2(r) \oplus m))|| (H_2(r) \oplus m) \).
Kobara-Imai’s γ-padding: \( c = y_3 || \text{Mc.Enc}(G, y_1, y_2) \) where \( y_3 || y_2 || y_1 = (r || \text{H}_1(H_2(r) || m || \text{const}))) || H_2(r) || (m || \text{const}) \).

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 RLCEpad and RLCEpad

In this section, we assume that the message bandwidth is \( m \text{Len} \)-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+ [28] and SAEP+ [5]. The first simple padding scheme RLCEpad is a one-round of a Feistel network that is similar to SAEP+. RLCEpad could be used to encrypt short messages (e.g., \( m \text{Len}/4 \)-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 \( m \text{Len} \)-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 \( e \) if mediumEncoding or advancedEncoding is used). For a RLCE setup process 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 RLCEpad proceeds as follows.

**RLCEpad(mLen, k1, k2, k3):** Let \( k_1, k_2, k_3 \) be parameters such that \( k_1 + k_2 + k_3 = \left\lceil \frac{m \text{Len}}{8} \right\rceil, k_1 + k_2 < k_3, \) and \( 8k_1 \leq m \text{Len}/4 \). Let \( \nu = 8(k_1 + k_2 + k_3) - m \text{Len}. \) Let \( H_1 \) be a random oracle that takes any-length inputs and outputs \( k_3 \)-bytes and let \( H_2 \) be a random oracle that takes any-length inputs and outputs \( (k_1 + k_2) \)-bytes. Let \( m \in \{0, 1\}^{8k_1} \) be a message to be encrypted, \( r_0 \in \{0, 1\}^{8k_3 - \nu} \) be a randomly selected sequence, and \( r = r_0 || 0^\nu \). We distinguish the following three cases:

- **basicEncoding:** Select a random \( e \in GF(q)^{n+w} \) of weight \( t \) and set
  \[
  y = ((m || H_1(m, r, e)) \oplus H_2(r, e)) || r. \tag{14}
  \]
  Convert \( y \) to an element \( y_1 \in GF(q)^K \). Let the ciphertext be \( c = y_1G + e \).

- **mediumEncoding:** Select random \( 0 \leq l_0 < l_1 < \cdots < l_{t-1} < n + w - 1 \) and let \( e_0 = l_0 || l_1 \cdots || l_{t-1} \in \{0, 1\}^{16t} \). Set
  \[
  y = ((m || H_1(m, r, e_0)) \oplus H_2(r, e_0)) || r. \tag{15}
  \]
  Convert \( y \) to an element \( (y_1, e_1) \in GF(q)^{K+t} \) where \( y_1 \in GF(q)^K \) and \( e_1 \in GF(q)^t \). Let \( e \in GF(q)^{n+w} \) such that \( e[l_i] = e_1[i] \) for \( 0 \leq i < t \) and \( e[j] = 0 \) for \( j \neq l_i \). Let the ciphertext be \( c = y_1G + e \).

- **advancedEncoding:** Set \( y = ((m || H_1(m, r)) \oplus H_2(r)) || r \). Convert \( y \) to an element \( y_1 \in GF(q)^K \) and a vector \( e \in GF(q)^{n+w} \) of weight \( t \). Let the ciphertext be \( c = y_1G + e \).

The mediumEncoding based RLCEpad is shown graphically in Figure 1.

Assuming the hardness of decoding RLCE ciphertexts, a similar proof as in [5] could be used to show that RLCE-RLCEpad scheme is secure against IND-CCA2 attacks. As an example with \( k_c = 128 \) 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-RLCEpad 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 \( k_1 + k_2 + k_3 = \lceil \frac{mLen}{8} \rceil \), \( \min \{ k_2, k_3 \} \geq \kappa_c \) where \( \kappa_c \) 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 \( m \in \{0, 1\}^{8k_1} \) be a message to be encrypted, \( r_0 \in \{0, 1\}^{8k_3-v} \) be a randomly selected sequence, and \( r = r_0 || 0^v \). We distinguish the following three cases:

- **basicEncoding**: Select a random \( e \in GF(q)^{t+w} \) of weight \( t \) and set \( y = ((m||H_1(m, r, e)) \oplus H_2(r, e)) || r \oplus H_3(((m||H_1(m, r, e)) \oplus H_2(r, e))) \) (16)
  Convert \( y \) to an element \( y_1 \in GF(q)^{k} \). Let the ciphertext be \( c = y_1G + e \).

- **mediumEncoding**: Select random \( 0 \leq l_0 < l_1 < \cdots < l_{t-1} \leq n + w - 1 \) and let \( e_0 = l_0||l_1|| \cdots ||l_{t-1} \in \{0, 1\}^{16t} \). Set
  \[ y = ((m||H_1(m, r, e_0)) \oplus H_2(r, e_0)) || r \oplus H_3(((m||H_1(m, r, e_0)) \oplus H_2(r, e_0))) \] (17)
  Convert \( y \) to an element \((y_1, e_1) \in GF(q)^{k+t} \) where \( y_1 \in GF(q)^{k} \) and \( e_1 \in GF(q)^{t} \). Let \( e \in GF(q)^{t+w} \) such that \( e[i] = e_1[i] \) for \( 0 \leq i < t \) and \( e[j] = 0 \) for \( j \neq i \). Let the ciphertext be \( c = y_1G + e \).

- **advancedEncoding**: Set
  \[ y = ((m||H_1(m, r)) \oplus H_2(r)) || r \oplus H_3(((m||H_1(m, r)) \oplus H_2(r))) \] (18)
  Convert \( y \) to an element \( y_1 \in GF(q)^{k} \) and a vector \( e \in GF(q)^{t+w} \) of weight \( t \). Let the ciphertext be \( c = y_1G + e \).

The mediumEncoding based RLCEpad is shown graphically in Figure 2.

Assuming the hardness of decoding RLCE ciphertexts, a similar proof as in [28] could be used to show that RLCE-RLCEpad scheme is secure against IND-CCA2 attacks. The proof in [28] shows that, for a given security parameter \( \kappa_c \), it is sufficient to choose \( k_2, k_3 \) with

\[
\max \left\{ \frac{1}{q^{k_2}}, \frac{1}{q^{k_3}} \right\} \leq \frac{1}{2\kappa_c}. \tag{19}
\]
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 $e_0$ or error vector $e$ is used in the RLCEs-pad/RLCEpad process and the message recipient needs to have the exact $e_0$ or $e$ 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 $r$) contain zero field elements, one revises the random value $r$ 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 $k_1 + k_2 + k_3 = \left\lceil \frac{m \cdot \text{len}}{8} \right\rceil$. Alternatively, one may use $k_1 + k_2 + k_3 = \left\lfloor \frac{m \cdot \text{len}}{8} \right\rfloor$ and adjust the schemes correspondingly.

## 8 Recommended parameters and performance evaluation

Taking into account of the condition (8) for avoiding Sidelnikov-Shestakov attacks, the condition (11) for avoiding filtration 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 (ISD) methods, we generated a recommended list of parameters for RLCE scheme in Table 1. In Table 1, $\kappa_c$ denotes the conventional security strength. For example, $\kappa_c = 128$ means an equivalent security of AES-128. For the naive ISD, 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 corresponding ciphertext values in these coordinates that corresponds to the $k$ columns of the public key. 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 $O(k^{2.807})$ field operations (though Coppersmith-Winograd algorithm takes $O(k^{2.376})$ 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 $2^{\kappa'}$ steps to find $k$-error free coordinates where, by Sterling’s approximation,

$$\kappa' = \log_2 \left( \frac{(n+w)k^2}{2^{\log_2^2 n}} \right) + O(1) \approx (n+w)I(\frac{k}{n+w}) - (n+w-t)I(\frac{k}{n+w-t}) + \log_2(2^{2.807}) + O(1) \quad (20)$$

and $I(x) = -x \log_2(x) - (1-x) \log_2(1-x)$ is the binary entropy of $x$. There are several improved ISD algorithms in the literature. These improved ISD algorithms allow a small number of error positions inside the ciphertext values corresponding to the selected $k$ coordinates or select $k' > k$ columns of the public key matrix for a small number $k' - k$ or both. The values of $\kappa_c$ in Table 1 are mainly calculated using the PARI/GP script from Peters [25]. 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., GRS code) over $GF(q)$ where $q = 2^{\log_2 n}$ or $q = 2^{12}$ (for convenient data conversion over 32 or 64 bit computers). For GRS codes, the natural construction requires $n = q - 1$. However, GRS codes 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 recommend $w = n - k$. For the purpose of comparison, we also list the recommended parameters from [3] for the binary Goppa code based McEliece encryption scheme.

To reduce the public key sizes, the authors in [3, 25] 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 while the binary Goppa code based McEliece encryption scheme has a public key size of 187.7KB.

The value $\kappa_q$ in Table 1 denotes the quantum security strength under quantum information-set decoding using Grover’s algorithm (see, e.g., Bernstein [2]). An RLCE scheme is said to have quantum security level $\kappa_q$ if the expected running time (or circuit depth) to decrypt an RLCE ciphertext using Grover’s algorithm based ISD is $2^{\kappa_q}$. For a function $f : \{0, 1\}^l \rightarrow \{0, 1\}$ with the property that there is an $x_0 \in \{0, 1\}^l$ such that $f(x_0) = 1$ and $f(x) = 0$ for all $x \neq x_0$, Grover’s algorithm finds the value $x_0$ using $\frac{\pi}{4}\sqrt{2^l}$ Grover iterations and $O(l)$ qubits. Specifically, Grover’s algorithm converts the function $f$ to a reversible circuit $C_f$ and calculates

$$|x\rangle \rightarrow (-1)^{f(x)}|x\rangle$$

in each of the Grover iterations, where $|x\rangle$ is an $l$-qubit register. Thus the total steps for Grover’s algorithm is bounded by $\frac{n|C_f|}{4}\sqrt{2^l}$.

For RLCE scheme, quantum information-set decoding could be carried out similarly as in Bernstein’s [2]. One first uniformly selects $k$ columns from the public key and checks whether it could be inverted. 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. Though Grover’s algorithm requires that the function $f$ evaluate to 1 on only one of the inputs, there are several approaches (see, e.g., Grassl et al [11]) to cope with cases that $f$ evaluates to 1 on multiple inputs.

For a randomly selected $k$ columns of the RLCE encryption scheme public key, the probability that the ciphertext contains no errors in these positions is approximately $\binom{n+w-1}{k}$. Thus the quantum ISD algorithm requires $\sqrt{\binom{n+w}{k}/\binom{n+w-t}{k}}$ Grover iterations. For each Grover iteration, the function $f$ needs to carry out the following computations:
1. Compute the inverse of a \( k \times k \) submatrix \( G_{\text{sub}} \) of the public key. This takes \( O(k^{2.807}) \) field operations if Strassen algorithm is used.

2. Check that the selected \( k \) positions contain no errors in the ciphertext. This takes \( O((n + w)k) \) field operations.

It is expensive for circuits to use look-up tables for field multiplications. Using Karatsuba algorithm, Kepley and Steinwandt [17] constructed a field element multiplication circuit with gate counts of \( 7 \cdot (\log_2 q)^{1.585} \). In a summary, the above function \( f \) for the RLCE quantum ISD algorithm could be evaluated using a reversible circuit \( C_f \) with gate counts of \( 7((n + w)k + k^{2.807})(\log_2 q)^{1.585} \) gates. To be conservative, we may assume that a randomly selected \( k \) columns from the public key is invertible. Thus Grover’s quantum algorithm requires approximately

\[
7((n + w)k + k^{2.807})(\log_2 q)^{1.585} \sqrt{n + w} \quad (21)
\]

steps for the simple ISD algorithm against RLCE encryption scheme. Advanced quantum ISD techniques may be developed based on improved ISD algorithms. However our analysis shows that the reduction on the quantum security is marginal. The reader is also referred to a recent report [16] for an analysis of quantum ISD based on improved ISD algorithms. In the proposed parameters \( \kappa_q \) in Table 1, we used conservative estimations by taking into these advanced quantum ISD attacks together with the estimate in (21). Parameters in Table 1 could be used for any MDS code based RLCE scheme.

**Table 1:** RLCE parameters: “600, 464, 68, 10, 154KB” represents \( n = 600, k = 464, t = 68, q = 2^{10} \)

<table>
<thead>
<tr>
<th>( \kappa_c )</th>
<th>( \kappa_q )</th>
<th>RLCE (( w = n - k ))</th>
<th>binary Goppa code [3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>85</td>
<td>600,464,68,10,154KB</td>
<td>2960, 2288, 57, 188KB</td>
</tr>
<tr>
<td>128</td>
<td>85</td>
<td>600,440,80,12,206KB</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>100</td>
<td>780,580,100,10,212KB</td>
<td>3100, 2300, 80, 302KB</td>
</tr>
<tr>
<td>160</td>
<td>100</td>
<td>760,540,110,12,348KB</td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>120</td>
<td>1000,790,105,10,405KB</td>
<td>4624, 3468, 97, 490KB</td>
</tr>
<tr>
<td>192</td>
<td>120</td>
<td>990,780,105,12,480KB</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>150</td>
<td>1300,800,250,11,1.05MB</td>
<td>6624, 5129, 117, 900KB</td>
</tr>
<tr>
<td>256</td>
<td>150</td>
<td>1300,800,250,12,1.14MB</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 lists the message bandwidth and message padding scheme parameters for the recommended schemes. In case that \( \nu = 8(k_1 + k_2 + k_3) - \text{mLen}_i > 0 \), the last \( \nu \)-bits of the \( k_3 \)-bytes random seed \( r \) should be set to zero and the last \( \nu \)-bit of the encoded string \( y \) is discarded. For RLCEpad with \( \nu > 0 \), the encoding and decoding process are straightforward. For RLCEpad with \( \nu > 0 \), the decoding process produces an encoded string \( y \) with last \( \nu \)-bits missing. After using \( H_3 \) to hash the first part of \( y \) resulting in \( k_3 \)-bytes hash output, one discards the last \( \nu \)-bits from the hash output and \( \oplus \) the remaining \( (8k_3 - \nu) \)-bits with the second half of \( y \) to obtain the \( (8k_3 - \nu) \)-bits of \( r \) without the \( \nu \)-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 2: Padding parameters: bE for basicEncoding, mE for mediumEncoding and aE for advancedEncoding

<table>
<thead>
<tr>
<th>ID</th>
<th>κc</th>
<th>κq</th>
<th>n</th>
<th>k</th>
<th>t</th>
<th>m</th>
<th>sys sk</th>
<th>sk</th>
<th>pk</th>
<th>nLen</th>
<th>RLCEspad</th>
<th>RLCEpad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>600</td>
<td>6</td>
<td>6</td>
<td>160</td>
<td>60767</td>
<td>430815</td>
<td>157761</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>128</td>
<td>85</td>
<td>600</td>
<td>6</td>
<td>6</td>
<td>160</td>
<td>60767</td>
<td>430815</td>
<td>157761</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>85</td>
<td>600</td>
<td>6</td>
<td>6</td>
<td>160</td>
<td>214663</td>
<td>505943</td>
<td>211201</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>160</td>
<td>780</td>
<td>600</td>
<td>100</td>
<td>100</td>
<td>160</td>
<td>294088</td>
<td>715748</td>
<td>290001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>160</td>
<td>760</td>
<td>540</td>
<td>10</td>
<td>10</td>
<td>160</td>
<td>360933</td>
<td>799413</td>
<td>356401</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>192</td>
<td>120</td>
<td>1000</td>
<td>790</td>
<td>105</td>
<td>192</td>
<td>419630</td>
<td>1201335</td>
<td>414751</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>192</td>
<td>990</td>
<td>780</td>
<td>105</td>
<td>12</td>
<td>192</td>
<td>496653</td>
<td>1410813</td>
<td>491401</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>150</td>
<td>1300</td>
<td>800</td>
<td>250</td>
<td>256</td>
<td>1108453</td>
<td>1990053</td>
<td>1100001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>256</td>
<td>1300</td>
<td>800</td>
<td>250</td>
<td>11</td>
<td>256</td>
<td>1208803</td>
<td>2170403</td>
<td>1200001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

9 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 GRS code, LDPC code, Turbo code, or Polar code. Heuristics and experiments encourage 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 [32].

References


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

<table>
<thead>
<tr>
<th>ID</th>
<th>sec/key</th>
<th>seconds/encryption</th>
<th>KB per sec</th>
<th>seconds/decryption</th>
<th>KB/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.038965</td>
<td>0.018558</td>
<td>0.018448</td>
<td>59.709</td>
<td>225.630</td>
</tr>
<tr>
<td>1</td>
<td>1.025386</td>
<td>0.019920</td>
<td>0.019850</td>
<td>45.856</td>
<td>171.230</td>
</tr>
<tr>
<td>2</td>
<td>2.369620</td>
<td>0.031716</td>
<td>0.031144</td>
<td>45.856</td>
<td>171.230</td>
</tr>
<tr>
<td>3</td>
<td>2.141296</td>
<td>0.029142</td>
<td>0.028558</td>
<td>57.037</td>
<td>217.050</td>
</tr>
<tr>
<td>4</td>
<td>3.370898</td>
<td>0.046863</td>
<td>0.047414</td>
<td>39.902</td>
<td>149.428</td>
</tr>
<tr>
<td>5</td>
<td>4.904205</td>
<td>0.044599</td>
<td>0.048186</td>
<td>51.245</td>
<td>177.071</td>
</tr>
<tr>
<td>6</td>
<td>9.316449</td>
<td>0.087165</td>
<td>0.086902</td>
<td>28.604</td>
<td>105.516</td>
</tr>
<tr>
<td>7</td>
<td>9.410073</td>
<td>0.086897</td>
<td>0.086147</td>
<td>31.083</td>
<td>116.972</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ID</th>
<th>sec/key</th>
<th>seconds/encryption</th>
<th>KB per sec</th>
<th>seconds/decryption</th>
<th>KB/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.812333</td>
<td>0.001981</td>
<td>0.002034</td>
<td>78.858</td>
<td>288.523</td>
</tr>
<tr>
<td>1</td>
<td>0.791667</td>
<td>0.002084</td>
<td>0.002156</td>
<td>89.017</td>
<td>294.253</td>
</tr>
<tr>
<td>2</td>
<td>1.828000</td>
<td>0.003275</td>
<td>0.003337</td>
<td>62.619</td>
<td>225.311</td>
</tr>
<tr>
<td>3</td>
<td>1.698000</td>
<td>0.003244</td>
<td>0.003372</td>
<td>72.258</td>
<td>267.138</td>
</tr>
<tr>
<td>4</td>
<td>3.968667</td>
<td>0.005044</td>
<td>0.005125</td>
<td>52.276</td>
<td>194.924</td>
</tr>
<tr>
<td>5</td>
<td>3.864667</td>
<td>0.005063</td>
<td>0.005072</td>
<td>63.656</td>
<td>237.209</td>
</tr>
<tr>
<td>6</td>
<td>7.229333</td>
<td>0.009769</td>
<td>0.009775</td>
<td>35.989</td>
<td>132.273</td>
</tr>
<tr>
<td>7</td>
<td>7.349000</td>
<td>0.009928</td>
<td>0.009931</td>
<td>38.362</td>
<td>143.074</td>
</tr>
</tbody>
</table>


