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

November 6, 2017

Abstract

Recently, Wang (2016) introduced a random linear code based quantum resistant public key 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 proposes message padding schemes for RLCE to achieve IND-CCA2 security. Practical RLCE parameters for the security levels of 128, 192, and 256 bits and for the quantum security levels of 80, 110, and 144 are recommended. The implementation of the RLCE encryption scheme and software packages for analyzing the security strength of RLCE parameters are 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.

Contents

1 Introduction

Since McEliece encryption scheme [17] 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 cryptography 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, 18], Wang's RLCE [24], and the original binary Goppa code based McEliece encryption scheme. Though no essential attacks have been identified for Goppa code and MDPC/LDPC based McEliece encryption schemes yet, the security of these schemes depends on certain structures of the underlying linear codes. The advantage of RLCE encryption scheme is that its security does not depends on any specific structure of underlying linear codes, instead its security is believed to depend on the NP-hardness of decoding random linear codes.

This paper proposes variants of the RLCE scheme with increased message communication bandwidth, reduced public key size, and improved encryption and decryption performance. This paper also systematically analyzes the security of RLCE schemes and investigates the security requirements for the RLCE scheme to have the smallest public key sizes. Practical message padding parameters for the RLCE scheme to be secure against adaptive chosen ciphertext attacks (IND-CCA2) are proposed and experimental results for different RLCE scheme parameter sizes are reported.

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 $k < n < q$. The generalized Reed-Solomon code $\mathrm{GRS}_k(\mathbf{x}, \mathbf{y})$ of dimension k is defined as

$$\mathrm{GRS}_k(\mathbf{x}, \mathbf{y}) = \{(y_0 p(x_0), \cdots, y_{n-1} p(x_{n-1})) : p(x) \in GF(q)[x], \deg(p) < k\}$$

where $\mathbf{x} = (x_0, \cdots, x_{n-1}) \in GF(q)^n$ is an n-tuple of distinct elements and $\mathbf{y} = (y_0, \cdots, y_{n-1}) \in GF(q)^n$ 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 [17] 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 $k \times k$ non-singular 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 [19] 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 $(n - k) \times (n - k)$ non-singular 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 $n - k$.

$\texttt{Nied.Dec}(S, H_s, P, \mathbf{c})$. 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 $\mathbf{m} = \mathbf{m}'(P^T)^{-1}$.

It is well known that McElience's scheme and Niederreiter's scheme are equivalent. Let G_s and H_s be the generator matrix and parity check matrix of an $(n, k, 2t+1)$ linear code C respectively. Let G be the public key of the corresponding McEliece encryption scheme. From G, one calculates a full rank $(n-k)\times n$ matrix H such that $GH^T = 0$. It is straightforward to verify that H is a public key for the Niederreiter scheme with corresponding parity check matrix H_s. For a McEliece's scheme ciphertext $\mathbf{c} = \mathbf{m}G + \mathbf{e}$, we have

$$\mathbf{c}H^T = \mathbf{m}GH^T + \mathbf{e}H^T = \mathbf{e}H^T.$$

Thus if one can break Niederreiter's scheme then one can break McEliece's scheme. On the other hand, from a given Niederreiter public key H, one can compute a full rank $k \times n$ matrix G such that $GH^T = 0$. It is straightforward to verify that G is a public key for the McEliece scheme with corresponding generator matrix G_s. For a ciphertext $\mathbf{c} = \mathbf{m}H^T$, one solves the equation $\mathbf{c} = \mathbf{a}H^T$ to obtain a vector $\mathbf{a} \in GF(q)^n$. By the fact that $(\mathbf{a} - \mathbf{m})H^T = 0$, there exists a vector $\mathbf{r} \in GF(q)^{n-k}$ such that $\mathbf{a} - \mathbf{m} = \mathbf{r}G$. That is, $\mathbf{a} = \mathbf{r}G + \mathbf{m}$. This shows that if one can break McEliece scheme, then one can break Niederreiter scheme.

The protocol for the RLCE Encryption scheme by Wang [24] consists of the following three processes: RLCE.KeySetup, RLCE.Enc, and RLCE.Dec.

$\texttt{RLCE.KeySetup}(n, k, d, t, r)$. 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 non-singular $(r+1) \times (r+1)$ matrices chosen uniformly at random and let $A = \text{diag}[A_0, \cdots, A_{n-1}]$ be an $n(r+1) \times n(r+1)$ non-singular matrix.

3. Let S be a random dense $k \times k$ non-singular 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).

$\texttt{RLCE.Enc}(G, \mathbf{m}, \mathbf{e})$. 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 $\mathbf{c} = \mathbf{m}G + \mathbf{e}$.

$\texttt{RLCE.Dec}(S, G_s, P, A, \mathbf{c})$. 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 $i \leq n-1$. 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 $wt = \texttt{wt}(\mathbf{c} - \mathbf{m}G)$. If $wt \leq t$ 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.

`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 non-singular $(r+1)\times(r+1)$ matrices chosen uniformly at random and let $A = \mathrm{diag}[A_0,\cdots,A_{n-1}]$ be an $n(r+1)\times n(r+1)$ non-singular matrix.

3. Let S be a random dense $(n-k)\times(n-k)$ non-singular 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).

`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 each non-zero element in $\mathbf{m}$ can be converted to at most $(t+1)$-nonzero elements in $\mathbf{m}P^TA^T$, the weight of $\mathbf{m}P^TA^T$ is at most $(r+1)t$. Thus we can decrypt the ciphertext $\mathbf{c}$ only if we had an efficient $(r+1)t$-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 a 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 show that it is hard to design an equivalent Niederreiter-type 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 [24] is that the revised scheme only inserts $w < n$ random columns after randomly selected number of 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 $G_s P_1 = [\mathbf{g}_0, \cdots, \mathbf{g}_{n-1}]$.

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 $A_0 = \begin{pmatrix} a_{0,00} & a_{0,01} \\ a_{0,10} & a_{0,11} \end{pmatrix}, \cdots, A_{w-1} = \begin{pmatrix} a_{w-1,00} & a_{w-1,01} \\ a_{w-1,10} & a_{w-1,11} \end{pmatrix} \in GF(q)^{2\times 2}$ be non-singular 2×2 matrices chosen uniformly at random such that $a_{i,00}a_{i,01}a_{i,10}a_{i,11} \neq 0$ for all $i = 0, \cdots, w - 1$. Let $A = \text{diag}[1, \cdots, 1, A_0, \cdots, A_{w-1}]$ be an $(n + w) \times (n + w)$ non-singular matrix.

3. Let S be a random dense $k \times k$ non-singular 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).

$\texttt{RLCE.Enc}(G, \mathbf{m}, \mathbf{e})$. 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 $\mathbf{c} = \mathbf{m}G + \mathbf{e}$.

$\texttt{RLCE.Dec}(S, G_s, P_1, P_2, A, \mathbf{c})$. 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 an efficient decoding algorithm, one can recover $\mathbf{m}SG_s$ from $\mathbf{c}'P_1^{-1}$. Let D be a $k \times k$ inverse matrix of SG'_s where G'_s is the first k columns of G_s. Then $\mathbf{m} = \mathbf{c}_1 D$ where $\mathbf{c}_1$ is the first k elements of $\mathbf{m}SG_s$. Finally, calculate the Hamming weight $wt = \texttt{wt}(\mathbf{c} - \mathbf{m}G)$. If $wt \leq t$ then output $\mathbf{m}$ as the decrypted plaintext. Otherwise, output error.

Remark. If $w = n$, then the revised RLCE scheme is the same as the original RLCE scheme with $r = 1$. If the $(n + w) \times (n + w)$ matrix A is taken as the identity matrix $\mathrm{I}_{(n+w)\times(n+w)}$, then the revised RLCE scheme is the same as the Wieschebrink's encryption scheme [26].

5 Systematic RLCE encryption scheme

To reduce RLCE scheme public key sizes, one can use a semantic secure message encoding approach (e.g., an IND-CCA2 padding scheme) so that the public key can be stored in a systematic matrix. For a McEliece encryption scheme over $GF(q)$, one needs to store $k(n - k)$ elements from $GF(q)$ for a systematic public key matrix instead of nk elements from $GF(q)$ for a non-systematic generator matrix public key.

In a systematic RLCE encryption scheme, the decryption could be done more efficiently. In the RLCE decryption process, one recovers $\mathbf{m}SG_s$ from $\mathbf{c}'P_1^{-1} = \mathbf{m}SG_s + \mathbf{e}'$ first. Let $\mathbf{m}SG_sP_1 = (d_0, \cdots, d_{n-1})$ and $\mathbf{c}_d = (d'_0, \cdots, d'_{n+w}) = (d_0, d_1, \cdots, d_{n-w}, \perp, d_{n-w+1}, \perp, \cdots, d_{n-1}, \perp)P_2$ be a length $n + w$ vector. For each $i < k$ such that $d'_i = d_j$ for some $j < n - w$, we have $m_i = d_j$. Let

$$I_R = \{i : m_i \text{ is recovered via } \mathbf{m}SG_s\} \text{ and } \bar{I}_R = \{0, \cdots, k-1\} \setminus I_R.$$

Assume that $|\bar{I}_R| = u$. It suffices to recover the remaining u message symbols m_i with $i \in \bar{I}_R$. In the following paragraphs, we present three approaches to recover these message symbols.

5.1 Decoding algorithm 0 for systematic RLCE encryption scheme

In the first approach, one recovers $\mathbf{m}S$ from $\mathbf{m}SG_s$ first. Then one multiplies $\mathbf{m}'$ with the corresponding u columns $S_{\bar{I}_R}$ of the matrix S^{-1} to get m_i with $i \in \bar{I}_R$.

5.2 Decoding algorithm 1 for systematic RLCE encryption scheme

Instead of recovering $\mathbf{m}S$ first, one may use public key to recover the remaining message symbols from $\mathbf{m}SG_s$ direcrtly. Let $i_0, \cdots, i_{u-1} \geq k$ be indices such that for each i_j, we have $d'_{i_j} = d_i$ for some $i < n - w$. The remaining message symbols with indices in $\bar{I}_R$ could be recovered by solving the linear equation system

$$\mathbf{m}\,[\mathbf{g}_{i_0}, \cdots, \mathbf{g}_{i_{u-1}}] = [d'_{i_0}, \cdots, d'_{i_{u-1}}]$$

where $\mathbf{g}_{i_0}, \cdots, \mathbf{g}_{i_{u-1}}$ are the corresponding columns in the public key. Let P be a permutation matrix so that the recovered message symbols m_i ($i \in I_R$) are the first $k - u$ elements in $\mathbf{m}P$. That is,

$$\mathbf{m}PP^{-1}\,[\mathbf{g}_{i_0}, \cdots, \mathbf{g}_{i_{u-1}}] = (\mathbf{m}_{I_R}, \mathbf{m}_{\bar{I}_R})P^{-1}\,[\mathbf{g}_{i_0}, \cdots, \mathbf{g}_{i_{u-1}}] = [d'_{i_0}, \cdots, d'_{i_{u-1}}]$$

where $\mathbf{m}_{I_R}$ is the list of message symbols with indices in I_R. Let

$$P^{-1}\,[\mathbf{g}_{i_0}, \cdots, \mathbf{g}_{i_{u-1}}] = \begin{pmatrix} V \\ W \end{pmatrix}$$

where V is a $(k-u) \times u$ matrix and W is a $u \times u$ matrix. Then we have

$$\mathbf{m}_{\bar{I}_R}W = [d'_{i_0}, \cdots, d'_{i_{u-1}}] - \mathbf{m}_{I_R}V.$$

Furthermore, one may pre-compute the inverse of W and include W^{-1} in the private key. Then one can recover the remaining message symbols

$$\mathbf{m}_{\bar{I}_R} = \left([d'_{i_0}, \cdots, d'_{i_{u-1}}] - \mathbf{m}_{I_R}V\right)W^{-1}.$$

5.3 Decoding algorithm 2 for systematic RLCE encryption scheme

In practice, one may use a larger I_R. Recall that in the RLCE decryption process, one recovers $\mathbf{m}SG_s$ from $\mathbf{c}'P_1^{-1} = \mathbf{m}SG_s + \mathbf{e}'$ first. Let $\mathbf{e}'P_1 = (e_0, \cdots, e_{n-1})$ and

$$\mathbf{e}_c = (e'_0, \cdots, e'_{n+w}) = (e_0, e_1, \cdots, e_{n-w}, \bar{e}_{n-w}, e_{n-w+1}, \bar{e}_{n-w+1}, \cdots, e_{n-1}, \bar{e}_{n-1})P_2$$

be a length $n+w$ vector. For each $e_{n-w+i_0} = 0$ $(0 \le i_0 < w)$, if $e'_i = e_{n-w+i_0}$ or $e'_i = \bar{e}_{n-w+i_0}$ for some $i < k$, then with high probability, we have $m_i = c_i$ since matrices A_i do not contain zero elements. Thus m_i might be recovered as c_i. Let

$$I_R^a = I_R \cup \{i < k : e'_i = e_{n-w+i_0} \text{ or } e'_i = \bar{e}_{n-w+i_0} \text{ for some } i_0 < w \text{ with } e_{n-w+i_0} = 0\}$$

and $\bar{I}_R^a = \{0, \cdots, k-1\} \setminus I_R^a$. Using the same algorithm as in Section 5.2 with $(I_R, \bar{I}_R)$ replaced by $(I_R^a, \bar{I}_R^a)$, one can then recover message symbols with indices in $\bar{I}_R^a$. With a small probability, the message recovered via $(I_R^a, \bar{I}_R^a)$ might be incorrect. If this happens, one restarts the decoding process using the pair $(I_R, \bar{I}_R)$.

5.4 Defeating side-channel attacks

The decoding algorithm 2 described in Section 5.3 might be vulnerable to side-channel attacks. The attacker may generate ciphertexts with appropriately chosen error locations and watch whether the decoding time is significantly long (which means that the message recovered via $(I_R^a, \bar{I}_R^a)$ might be incorrect). This information may be used to recover part of the private permutation P_2. If such kind of attacks needs to be defeated, then one should not use the decoding algorithm 2 described in Section 5.3.

For the decoding algorithms 0 and 1, the value u is dependent on the choice of the private permutation P_2. Though the leakage of the size of u is not sufficient for the adversary to recover P_2 or to carry out other attacks against RLCE scheme, this kind of side-channel information leakage could be easily defeated. Table 1 lists the values of u_0 such that, for each scheme, the value of u is smaller than u_0 for 90% of the choices of P_2 where the RLCE ID is the scheme ID described in Table 3. Thus one can select P_2 in such a way that u is smaller than the given u_0 of Table 1. Furthermore, during the decoding process, one can use dummy computations so that the decoding time is the same as the decoding time for $u = u_0$.

Table 1: The value u_0 for RLCE schemes

RLCE ID	0	1	2	3	4	5	6
u_0	200	123	303	190	482	309	7

6 Security analysis

Loidreau and Sendrier [16] pointed out some weak keys for binary Goppa code based McEliece schemes and similar weak keys for RLCE schemes should not be used. For an RLCE scheme ciphertext $\mathbf{c}$ of a message $\mathbf{m}$, one can obtain a valid ciphertext for a message $\mathbf{m} + \mathbf{m}'$ by letting $\mathbf{c}' = \mathbf{c} + \mathbf{m}'G$ without knowing the message $\mathbf{m}$. This kind of attacks could be defeated by using IND-CCA2-secure message padding schemes which will be discussed in this paper. Faugere, Otmani, Perret, and Tillich [9] developed an algebraic attack against quasi-cyclic and dyadic structure based compact variants of McEliece encryption scheme. Wang [24] showed that the algebraic attacks will not work against the RLCE encryption scheme. A straightforward modification of the analysis in [24] 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.

6.1 Classical and quantum Information-Set Decoding

Information-set decoding (ISD) is one of the most important message recovery attacks on McEliece encryption schemes. The state-of-the-art ISD attack for non-binary McEliece scheme is the one presented in Peters

[20], which is an improved version of Stern's algorithm [23]. Peters's attack [20] also integrated analysis techniques for ISD attacks on binary McEliece scheme discussed in [3]. For the RLCE encryption scheme, the ISD 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. The cost of ISD attack on an $[n, k, t; w]$-RLCE scheme is equivalent to the cost of ISD attack on an $[n + w, k; t]$-McEliece scheme.

For the naive ISD, one first uniformly selects k columns from the public key and checks whether it is invertible. If it is invertible, one multiplies the inverse with the corresponding ciphertext values in these coordinates that correspond 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). In a summary, the naive information-set decoding algorithm takes approximately $2^{\kappa'_c}$ steps to find k-error free coordinates where, by Sterling's approximation,

$$\kappa'_c = \log_2\left(\frac{\binom{n+w}{k}\left(k^{2.807}+k^2\right)}{\binom{n+w-t}{k}}\right) \simeq (n+w)I\left(\frac{k}{n+w}\right) - (n+w-t)I\left(\frac{k}{n+w-t}\right) + \log_2\left(k^{2.807}+k^2\right) \quad (4)$$

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 within the selected k ciphertext values or select $k + \delta$ columns of the public key matrix for a small number $\delta > 0$ or both. Peters provided a script [20][1] to calculate the security strength of a McEliece encryption scheme using the improved ISD algorithms. For the security strength $128 \le \kappa_c \le 256$, our experiment shows that generally we have $\kappa'_c - 10 \le \kappa_c \le \kappa'_c - 4$.

An RLCE scheme is said to have quantum security level κ_q if the expected running time (or circuit depth) to decrypt an RLCE ciphertext using Grover's algorithm based ISD is 2^{κ_q}. For a function $f : \{0,1\}^l \to \{0,1\}$ 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 $x \ne x_0$, 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

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

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 $\frac{\pi|C_f|}{4}\sqrt{2^l}$.

For the RLCE scheme, quantum ISD could be carried out similarly as in Bernstein's [2]. One first uniformly selects k columns from the public key and checks whether it is invertible. If it is invertible, 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 [10]) to cope with cases that f evaluates to 1 on multiple inputs.

For randomly selected k columns from a RLCE encryption scheme public key, the probability that the ciphertext contains no errors in these positions is $\frac{\binom{n+w-t}{k}}{\binom{n+w}{k}}$. 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] available from https://christianepeters.wordpress.com/publications/tools/.

1. Compute the inverse of a $k \times k$ sub-matrix G_{sub} of the public key and multiply it with the corresponding entries within the ciphertext. This takes $O\left(k^{2.807} + k^2\right)$ field operations if Strassen algorithm is used.
2. Check that the selected k positions contain no errors in the ciphertext. This can be done with one of the following methods:
   (a) Multiply the recovered message with the public key and compare the differences from the ciphertext. This takes $O((n+w)k)$ field operations.
   (b) Use the redundancy within message padding scheme to determine whether the recovered message has the correct padding information. The cost for this operation depends on the padding scheme.

It is expensive for circuits to use look-up tables for field multiplications. Using Karatsuba algorithm, Kepley and Steinwandt [15] 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 $O\left(7\left((n+w)k + k^{2.807} + k^2\right)(\log_2 q)^{1.585}\right)$ gates. To be conservative, we may assume that a randomly selected k-columns sub-matrix from the public key is invertible. Thus Grover's quantum algorithm requires approximately

$$7\left((n+w)k + k^{2.807} + k^2\right)(\log_2 q)^{1.585}\sqrt{\frac{\binom{n+w}{k}}{\binom{n+w-t}{k}}} \tag{5}$$

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 [14] for an analysis of quantum ISD based on improved ISD algorithms. For each of the recommended schemes in Table 3, the row (κ'_c, κ_q) in Table 2 shows the security strength under the classical ISD and classical quantum ISD attacks. For example, the RLCE scheme with ID = 1 in Table 3 has 139-bits security strength under classical ISD attacks and 89-bits security strength under quantum ISD attacks.

Table 2: Security strength for RLCE schemes in Table 3

Scheme ID (κ_c, κ_q)	**0 (128,80)**	**1 (128,80)**	**2 (192,110)**	**3 (192,110)**	**4 (256,144)**	**5 (256,144)**
(κ'_c, κ_q)	(139, 90)	(139, 89)	(205, 124)	(206, 124)	(269, 156)	(269,156)
(κ^s_c, κ^s_q)	(135, 86)	(135,85)	(202,120)	(202,120)	(266,154)	(266,153)
$(\kappa^{Stern}_c, \kappa^{Stern}_q)$	(130, 80)	(131, 80)	(195, 113)	(195, 113)	(257, 145)	(256,144)
insecure cipher prob.	$(7, 2^{-76})$	$(7, 2^{-76})$	$(11, 2^{-117})$	$(11, 2^{-117})$	$(14, 2^{-167})$	$(14, 2^{-165})$
κ_{SS}	$\perp$	4429	$\perp$	7328	$\perp$	11127
$(\kappa^f_{n,k,w}, \kappa^f_q)$	$\perp$	(128, 85)	$\perp$	(210, 127)	$\perp$	(260, 153)
known non-rand. pos.	459	301	741	506	772	576
Scheme ID (κ_c, κ_q)	**7 (128,80)**	**8 (128,80)**	**9 (192,110)**	**10 (192,110)**	**11 (256,144)**	**12 (256,144)**
(κ'_c, κ_q)	(139, 90)	(140, 40)	(207, 125)	(206, 124)	(268, 155)	(269,156)
(κ^s_c, κ^s_q)	(136, 86)	(136,86)	(202,120)	(201,119)	(266,153)	(267,153)
$(\kappa^{Stern}_c, \kappa^{Stern}_q)$	(130, 80)	(132, 81)	(196, 114)	(195, 113)	(256, 144)	(257,144)
insecure cipher prob.	$(7, 2^{-76})$	$(7, 2^{-77})$	$(11, 2^{-117})$	$(11, 2^{-117})$	$(14, 2^{-166})$	$(14, 2^{-166})$
κ_{SS}	$\perp$	4300	$\perp$	7147	$\perp$	10099
$(\kappa^f_{n,k,w}, \kappa^f_q)$	$\perp$	(138, 90)	$\perp$	(193, 119)	$\perp$	(288, 167)
known non-rand. pos.	454	304	766	500	671	478

6.2 Improved Information Set Decoding

In this section, we briefly review Stern's algorithm [23]. Let the $k \times (n+w)$ matrix G be the public key and $\mathbf{c}$ be an RLCE scheme ciphertext. Let $G_e = \left(\begin{array}{c}\mathbf{c}\\ G\end{array}\right)$ be a $(k+1)\times(n+w)$ matrix. Stern's algorithm will find the minimal weight code $\mathbf{e}$ that is generated by G_e. It is straightforward to show that $\mathbf{e}$ is the error vector for the ciphertext $\mathbf{c}$. Stern's information set decoding algorithm for finding the vector $\mathbf{e}$ is as follows.

1. Select two small numbers $p < k/2$ and $l < n+w-k$.

2. Select k columns $\mathbf{g}_{i_1}, \cdots, \mathbf{g}_{i_k}$ from G_e and l columns $\mathbf{g}_{j_1}, \cdots, \mathbf{g}_{j_l}$ from the remaining $n+w-k$ columns of G_e where $0 \le i_1, \cdots, i_k, j_1, \cdots, j_l \le n+w-1$ are distinct numbers. It is expected that the ciphertext $\mathbf{c}$ contains $2p$ errors within the locations $i_1, \cdots, i_k$ and no errors within the positions $j_1, \cdots, j_l$.

3. Let $P_{i_1,\cdots,i_k,j_1,\cdots,j_l}$ be a $(n+w)\times(n+w)$ permutation matrix so that
$$G_e P_{i_1,\cdots,i_k,j_1,\cdots,j_l} = (\mathbf{g}_{i_1}, \cdots, \mathbf{g}_{i_k}, \mathbf{g}_{j_1}, \cdots, \mathbf{g}_{j_l}, G_r),$$
where G_r is a $(k+1)\times(n+w-k-l)$ matrix.

4. Compute the echelon format
$$G_E = E(G_e P_{i_1,\cdots,i_k,j_1,\cdots,j_l}) = S G_e P_{i_1,\cdots,i_k,j_1,\cdots,j_l} = (I, L, G_r)$$
where S is a $(k+1)\times(k+1)$ matrix.

5. Find random vectors $\mathbf{u}, \mathbf{v} \in GF(q)^{(k+1)/2}$ of weight p such that $(\mathbf{u},\mathbf{v})L = \mathbf{0}$. If no such $\mathbf{u}, \mathbf{v}$ found, go to Step 2.

6. If $(\mathbf{u},\mathbf{v})L = \mathbf{0}$, then check whether $(\mathbf{u},\mathbf{v})G_r$ has weight $t-2p$. If it does not have weight $t-2p$, go to Step 2.

7. If $(\mathbf{u},\mathbf{v})G_r$ has weight $t-2p$, then $\mathbf{e} = (\mathbf{u},\mathbf{v})G_E P^{-1}_{i_1,\cdots,i_k,j_1,\cdots,j_l}$ is the error vector for the ciphertext $\mathbf{c}$.

It is noted that if we take $p = l = 0$, then Stern's algorithm is the naive ISD algorithm that we have discussed in the preceding section. For the convenience of analysis, we assume that $pl > 0$ in the following discussion. The algorithm takes approximately
$$S_I = \frac{\binom{n+w}{\lfloor k/2\rfloor}\binom{n+w-\lfloor k/2\rfloor}{k-\lfloor k/2\rfloor}\binom{n+w-k}{l}}{\binom{n+w-t}{\lfloor k/2\rfloor-p}\binom{t}{p}\binom{n+w-t-\lfloor k/2\rfloor-p}{k-\lfloor k/2\rfloor-p}\binom{t-p}{p}\binom{n+w-t-k+2p}{l}} \tag{6}$$
iterations. For each iteration, Step 4 takes $(2n+2w-k)k^2$ field operations, and Step 5 takes $2\binom{k/2}{p}(q-1)^p l(k+1)$ field operations. For each iteration, Step 6 runs $\binom{k/2}{p}^2(q-1)^{2p-l}$ times approximately and each runs takes $(n-k-l)(k+1)$ field operations. In a summary, Stern's ISD takes approximately 2^{κ_c} steps to find the error vector $\mathbf{e}$ where,
$$\kappa_c = \min_{p,l}\left\{\log_2\left(S_I\left((2n+2w-k)k^2 + 2\binom{k/2}{p}(q-1)^p l(k+1) + \binom{k/2}{p}^2(q-1)^{2p-l}(n-k-l)(k+1)\right)\right)\right\}. \tag{7}$$

Our experiments show that for RLCE schemes that we have interest in, the equation (7) is always achieved with $p = 1$ and $l = 3$. For quantum version of Stern's ISD algorithm, the Grover's algorithm could be used to reduce the iteration steps to $\sqrt{S_I}$. Thus the quantum security level under Stern attacks is approximately

$$\kappa_q = \min_{p,l}\left\{\log_2\left(\sqrt{S_I}\left((2n+2w-k)k^2 + 2\binom{k/2}{p}(q-1)^p l(k+1) + \binom{k/2}{p}^2 (q-1)^{2p-l}(n-k-l)(k+1)\right)\right)\right\}. \tag{8}$$

In order to speed up Stern's algorithm, Peters [20] considers the following improvement:

1. For each iteration, one does not randomly selects k columns from G_e in Step 2. Instead, one reuses $k - c$ columns from the previous iteration where c is a fixed constant.
2. For a small finite field, fix a parameter $r > 1$ for certain pre-computation of row sums. This will not provide any benefit for a large field size such as those used in RLCE schemes.
3. For a small finite field, fix a parameter $m > 1$ such that one can use m error-free sets of size l. This will not provide any benefit for a large field size such as those used in RLCE schemes.

Our experiments show that for $\kappa_c \leq 200$, Peters's improved version in [20] is at most 8 times fast than Stern's algorithm discussed in this section. That is, we generally have $\kappa_c - 3 \leq \kappa_c^{Peter} \leq \kappa_c$ where κ_c^{Peter} is the κ_c obtained from Peter's improved algorithm. For $\kappa_c \geq 250$, our experiments show that Peter's improved version has the same performance as Stern's algorithm discussed in this section. Furthermore, our experiments show that the optimal values for p, l in Peter's improved algorithm on all RLCE schemes are $p = 1$ and $l = 3$ also.

6.3 Information Set Decoding for systematic RLCE schemes

Canteaut and Sendrier [6] discussed a known-partial-plaintext-attack against McEliece encryption scheme where $\mathbf{c} = \mathbf{m}G + \mathbf{e}$. 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

$$\mathbf{c} = \mathbf{m}G + \mathbf{e} = [\mathbf{m}_l, \mathbf{m}_r]\begin{bmatrix} G_l \\ G_r \end{bmatrix} + \mathbf{e} = \mathbf{m}_l G_l + \mathbf{m}_r G_r + \mathbf{e}. \tag{9}$$

Thus if one knows the value of $\mathbf{m}_l$, the identity (9) 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 code-word $\mathbf{c}$ since $r < k$. Though this attack against RLCE could be defeated by using appropriate message padding for IND-CCA2-security that will be discussed in Section 7, this attack can be integrated into information set decoding to design more efficient attacks against systematic RLCE schemes.

For the ISD against a systematic RLCE scheme, one uniformly selects $k = k_1 + k_2$ columns from the public key where k_1 columns are from the first k columns of the public key. Instead of multiplying the inverse of the selected k columns with the corresponding ciphertext values in these coordinates, one uses the corresponding ciphertext values for the selected k_1 columns within the first k columns of the public key to determine k_1 entries of the plaintext. Using these "recovered" k_1 entries of the plaintext, one calculates a new ciphertext $\mathbf{c}'$ with k_2 unknown plaintext entries as in the known-partial-plaintext-attack. Next one uses the inverse of the $k_2 \times k_2$ matrix to recover the remaining k_2 entries of the plaintext. In a summary, for each guessed k columns, one needs $k_1 k_2$ field multiplications to compute the new ciphertext $\mathbf{c}'$, needs $k_2^{2.807}$ field multiplications to compute the matrix inverse, and additional k_2^2 steps to compute the remaining k_2

entries of the plaintext. If one selects the k columns uniformly at random, then the expected values for k_1, k_2 are $k_1 = \frac{k^2}{n+w}$ and $k_2 = \frac{k(n+w-k)}{n+w}$ respectively. Thus the above information-set decoding algorithm against systematic RLCE scheme takes approximately $2^{\kappa_c^s}$ steps to find k-error free coordinates where,

$$\kappa_c^s = \log_2\left(\frac{\binom{n+w}{k}\left(\frac{k^3(n+w-k)}{(n+w)^2} + \left(\frac{k(n+w-k)}{n+w}\right)^{2.807} + \left(\frac{k(n+w-k)}{n+w}\right)^2\right)}{\binom{n+w-t}{k}}\right). \tag{10}$$

Similarly, Grover's quantum algorithm based on the above ISD against systematic RLCE requires approximately

$$7\left(\frac{k^3(n+w-k)}{(n+w)^2} + \left(\frac{k(n+w-k)}{n+w}\right)^{2.807} + \left(\frac{k(n+w-k)}{n+w}\right)^2\right)(\log_2 q)^{1.585}\sqrt{\frac{\binom{n+w}{k}}{\binom{n+w-t}{k}}} \tag{11}$$

steps for the simple ISD algorithm against RLCE encryption scheme. For each of the recommended schemes in Table 3, the row (κ_c^s, κ_q^s) in Table 2 shows the security strength under the ISD and quantum ISD attacks against systematic RLCE schemes. For example, the RLCE scheme with ID = 1 in Table 3 has 135-bits security strength under ISD attacks and 85-bits security strength under quantum ISD attacks.

6.4 Insecure ciphertexts for systematic RLCE schemes

For a systematic RLCE encryption scheme, if a small number of errors were added to the first k components of the ciphertext, one may be able to exhaustively search these errors and recover the message. Given a ciphertext $\mathbf{c}$ with l errors within the first k components (note that the adversary does not know this value l), the adversary starts from $i = 1$, randomly select $k - i$ positions within the ciphertext, take these values as the uncorrupted message values, guess the remaining i values for the message. If these $k - i$ positions contain no errors, the adversary can use the redundant information within the padding scheme to check whether the guessed message is correct. Under the condition that there are l errors within the first k components of the ciphertext, the probability for this attack to be successful is bounded by

$$\gamma_l = \max_{l\le i\le t}\left\{\frac{\binom{k-l}{k-i}}{q^i\binom{k}{i}}\right\}$$

For each $i \le l$, the probability that there are at most l errors in the first k components of the ciphertext is bounded by

$$E_l = \frac{\sum_{i\le l}\binom{k}{i}\binom{n+w-k}{t-i}}{\binom{n+w}{t}}.$$

The RLCE scheme encryption process produces an insecure ciphertext in case that the ciphertext contains at most l errors within the first k components of the ciphertext and $\gamma_l > 2^{-\kappa_c}$ where κ_c is the security parameter.

In order to avoid producing insecure ciphertexts, RLCE encryption process should repeatedly encrypt the message until it produces a ciphertext with at least l errors in the first k components such that $\gamma_l \le 2^{-\kappa_c}$. If the error locations are chosen uniformly at random, then the RLCE scheme encryption process produces an insecure ciphertext with the probability of at most

$$\max\{E_l : l \le t, \gamma_l > 2^{-\kappa_c}\} \tag{12}$$

This probability is negligible for security parameters that we are interested in. Thus the RLCE scheme needs to repeat the encryption process for a second time only with a negligible probability. For each of the recommended schemes in Table 3, the row "insecure cipher prob." in Table 2 shows the number of errors that should be contained in the first k components of a secure ciphertext and the probability that a ciphertext is insecure. An an example, for the RLCE scheme with ID = 1 in Table 3, the first k components of an insecure ciphertext contain 7 or less errors and the probability for this to happen is smaller than 2^{-76}.

6.5 Sidelnikov-Shestakov's attack

Niederreiter's scheme [19] replaces the binary Goppa codes in McEliece scheme by GRS codes. Sidelnikov and Shestakov [22] broke Niederreiter's scheme by recovering an equivalent private key $(\mathbf{x}', \mathbf{y}')$ from a public key G for the code $\text{GRS}_k(\mathbf{x}, \mathbf{y})$. For the given public key G, one computes the echelon form $E(G) = [I|G']$ using Gaussian elimination.

$$E(G) = \begin{bmatrix} 1 & 0 & \cdots & 0 & b_{0,k} & \cdots & b_{0,n-1} \\ 0 & 1 & \cdots & 0 & b_{1,k} & \cdots & b_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{k-1,k} & \cdots & b_{k-1,n-1} \end{bmatrix} \tag{13}$$

Assume the ith row code-word $\mathbf{b}_i$ of $E(G)$ encodes a message $p_i(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}$. Then

$$y_0 p_i(x_0) = 0, \cdots, y_i p_i(x_i) = 1, \cdots, y_{n-1} p_i(x_{n-1}) = b_{i,n-1} \tag{14}$$

Since the only non-zero elements are $b_{i,i}, b_{i,k}, \cdots, b_{i,n-1}$, p_i can be written as

$$p_i(x) = c_i \cdot \prod_{j=1, j\neq i}^{k} (x - x_j) \tag{15}$$

for some $c_i \neq 0$. By the fact that $\text{GRS}_k(\mathbf{x}, \mathbf{y}) = \text{GRS}_k(a\mathbf{x} + b, c\mathbf{y})$ 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 $x_2, \cdots, x_{n-1}$. Using equation (15), one can divide the row entries in (13) 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

$$\frac{b_{i_0,j}}{b_{i_1,j}} = \frac{y_j p_{i_0}(x_j)}{y_j p_{i_1}(x_j)} = \frac{c_{i_0}(x_j - x_{i_1})}{c_{i_1}(x_j - x_{i_0})} \tag{16}$$

for $j = k, \cdots, n-1$. First, by taking $i_0 = 0$ and $i_1 = 1$, equation (16) could be used to recover $x_k, \cdots, x_{n-1}$ 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 (16) could be used to recover x_{i_1}. Sidelnikov and Shestakov [22] showed that the values of $\mathbf{y}$ can then be recovered by solving a linear equation system based on $x_0, \cdots, x_{n-1}$.

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 (14) to obtain the equation (15). Thus no equations in (16) could be established and Sidelnikov and Shestakov attack could not continue.

2. If $n-k>w$, equations (16) may only be used to recover the values of $x_0, \cdots, x_{n-w-1}$. If it has a negligible probability for one to guess the remaining values $x_{n-w}, \cdots, x_{n-1}$, then Sidelnikov and Shestakov attack will not be successful. The probability for one to guess the remaining values $x_{n-w}, \cdots, x_{n-1}$ correctly is bounded by $1/\binom{q-n+w+1}{w}w!$.

Thus for a security parameter κ_c, the RLCE parameters should be chosen in such a way that

$$w \geq n-k \text{ or } \binom{q-n+w+1}{w}w! \geq 2^{\kappa_c}. \tag{17}$$

For RLCE schemes that we are interested in, we generally have $w \geq n-k$ or $\binom{q-n+w+1}{w}w! > \sqrt{2^{\kappa_c}}$. For each of the recommended schemes in Table 3, the row κ_{SS} in Table 2 shows the security strength under the Sidelnikov-Shestakov attack. For example, the RLCE scheme with ID = 1 in Table 3 has 4429-bits security strength under the above Sidelnikov-Shestakov attack.

6.6 Known non-randomized column attack

In this section, we consider the security of RLCE schemes when the positions of non-randomized $n-w$ GRS columns are known to the adversary. In this scenario, the adversary has two ways to attack the RLCE scheme. In the first approach, the adversary may guess the remaining w columns of the GRS generator matrix. The probability for this attack to be successful is shown in (17) which is very small compared against the security parameters. Alternatively, the adversary may use Sidelnikov-Shestakov attack to calculate a private key for the punctured $[n-w,k]$ GRS_k code consisting of the 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 $\mathbf{y} \in GF(q)^n$ to a code C is defined as $\Delta(\mathbf{y}, C) = \min\{\mathtt{wt}(\mathbf{y}-\mathbf{x}) : \mathbf{x} \in C\}$. For a vector $\mathbf{y} \in GF(q)^n$, $\mathbf{y}$'s Hamming ball of radius r is $B(\mathbf{y}; r) = \{\mathbf{y}' : \mathtt{wt}(\mathbf{y}-\mathbf{y}') \leq r\}$. For an MDS $[n,k,d]$ code C and a vector $\mathbf{y} \in GF(q)^n$, $B(\mathbf{y}; r)$ contains at most one code-word from C if $r \leq d/2$. If $d/2 < r \leq n - \sqrt{n(k-1)}$, $B(\mathbf{y}; r) \cap C$ contains at most polynomial many elements and the list-decoding algorithm by Guruswami and Sudan [11] can be used to efficiently output all elements in $B(\mathbf{y}; r) \cap C$. If the radius is stretched further, $B(\mathbf{y}; r) \cap C$ may contain exponentially many code-words.

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 [11]. Note that the probability for $\mathbf{c}'$ to contain at most $n-w-\sqrt{(n-w)(k-1)}$ errors is bounded by the hyper-geometric cumulative distribution function

$$PK_{n,w,t} = \frac{\sum_{i=0}^{n-w-\sqrt{(n-w)(k-1)}} \binom{n-w}{i}\binom{2w}{t-i}}{\binom{n+w}{t}} \tag{18}$$

That is, with probability $PK_{n,w,t}$ the encryption process produces a ciphertext that could be list-decoded using the $[n-w,k]$ GRS_k code. Thus the parameters should be chosen in such a way that $P_{n,w,t}$ is negligible (e.g., $P_{n,w,t} \leq 2^{\kappa_c}$) or the encryption process should repeatedly encrypt the message until it produces a ciphertext with at least $n-w-\sqrt{(n-w)(k-1)}+1$ errors corresponding to the known non-randomized columns. Justesen and Hoholdt [13] showed the following theorem.

Theorem 6.1 *(Justesen and Hoholdt [13]) 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 $\binom{n}{n-\delta}/q^{n-\delta-k}$ for randomly chosen $\mathbf{u} \in GF(q)^n$.*

By theorem 6.1, we may further require that the RLCE scheme repeatedly encrypt the message until it produces a ciphertext such that the size of $B(\mathbf{u};\delta)\cap C$ is large than 2^{κ_c}, where δ is the number of errors that the ciphertext $\mathbf{c}'$ contains.

In order to avoid the attacks that we mentioned in this section, it is recommended that the encryption process should produce a ciphertext that avoids these attacks if the positions of non-randomized columns are publicly known. Alternatively, we may recommend that one select RLCE parameters in such a way that it is computationally infeasible to identify non-randomized columns from the public key.

6.7 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 $C_1 * C_2$ 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 the square code $C^2 = C * C$ of C, we have $\dim C^2 \leq \min\left\{n, \binom{\dim C+1}{2}\right\}$. For an $[n, k]$ GRS code C, let $\mathbf{a}, \mathbf{b} \in \text{GRS}_k(\mathbf{x}, \mathbf{y})$ where $\mathbf{a} = (y_0p_1(x_0), \cdots, y_{n-1}p_1(x_{n-1}))$ and $\mathbf{b} = (y_0p_2(x_0), \cdots, y_{n-1}p_2(x_{n-1}))$. Then $\mathbf{a} * \mathbf{b} = (y_0^2p_1(x_0)p_2(x_0), \cdots, y_{n-1}^2p_1(x_{n-1})p_2(x_{n-1}))$. Thus $\text{GRS}_k(\mathbf{x}, \mathbf{y})^2 \subseteq \text{GRS}_{2k-1}(\mathbf{x}, \mathbf{y} * \mathbf{y})$ where we assume $2k - 1 \leq n$. This property has been used in [7] to recover non-random columns in a Wieschebrink scheme's public key [26].

Let G be the public key 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 $\mathcal{D}_1$ be the code with a generator matrix D_1 obtained from G by replacing the randomized $2w$ columns with all-zero columns and let $\mathcal{D}_2$ be the code with a generator matrix D_2 obtained from G by replacing the $n-w$ non-randomized columns with zero columns. Since $C \subset \mathcal{D}_1 + \mathcal{D}_2$ and the pair $(\mathcal{D}_1, \mathcal{D}_2)$ is an orthogonal pair, we have $C^2 \subset \mathcal{D}_1^2 + \mathcal{D}_2^2$. It follows that

$$2k - 1 \leq \dim C^2 \leq \min\{2k - 1, n - w\} + 2w \tag{19}$$

where we assume that $2w \leq k^2$. In the following discussion, we assume that *the $2w$ randomized columns in $\mathcal{D}_2$ behave like random columns in the filtration attacks*. We first consider the simple case of $k \geq n - w$. In this case, we have $\dim C^2 = \mathcal{D}_1^2 + \mathcal{D}_2^2 = n - w + \mathcal{D}_2^2 = n + w$. Furthermore, for any code C' of length n' that is obtained from C using code puncturing and code shortening, we have $\dim C'^2 = n'$. Thus filtration techniques could not be used to recover any non-randomized columns in D_1.

Next we consider the case for $k < n - w$. For this case, we distinguish two sub-cases: $n - w \geq 2k$ and $n - w < 2k$. For the case $n - w \geq 2k$, 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: the expected dimension for C_i^2 is $2k + 2w - 2$.
- Column i of G is a non-randomized column: the expected dimension for C_i^2 is $2k + 2w - 1$.

This shows that if $n-w \geq 2k$, then the filtration techniques could be used to identify the randomized columns within the public key G. Thus it is recommended to have $n - w < 2k$ for RLCE scheme.

Now we consider the case of $k < n - w < 2k$. In order to carry out filtration attacks, we need to shorten the code C at certain locations. Assume that we shorten $l < k - 1$ columns from G. Among the $l = l_1 + l_2$ columns, l_1 columns are non-randomized columns from D_1 and l_2 columns are randomized columns from D_2. Then the shortened code has dimension

$$d_{l,l_1} = \min\left\{(k-l)^2, \min\left\{2(k-l_1) - 1, n - w - l_1, (k-l)^2\right\} + \min\left\{2w - l_2, (k-l)^2\right\}\right\}. \tag{20}$$

A necessary condition for the filtration attack to be observable is that, after the shortening of the l_1 columns in D_1, the following condition is satisfied

$$d_{l,l_1} = 2(k - l_1) - 1 + \min\left\{2w - l_2, (k-l)^2\right\}. \tag{21}$$

Thus for a given l, the probability for the filtration attack to be successful is bounded by the probability

$$\frac{\sum_{l_1=\max\{0,l-2w\}}^{l} \lambda(d_{l,l_1})\binom{n-w}{l_1}\binom{2w}{l-l_1}}{\binom{n+w}{l}}$$

where $\lambda(d_{l,l_1}) = 1$ if (21) holds and $\lambda(d_{l,l_1}) = 0$ otherwise. For a given l, one randomly selects l columns from G and shortens G from these locations. This process takes $O(kl(n+w))$ field operations. Then one calculates the dimension of the shortened code to see whether the equation (21) is achieved which takes $O((k-l)^4(n+w-l))$ field operations. In a summary, the expected time for one to carry out the filtration attack for a given l is

$$PF_{n,k,w,l} = \frac{\binom{n+w}{l}\left(O(kl(n+w)) + O((k-l)^4(n+w-l))\right)}{\sum_{l_1=\max\{0,l-2w\}}^{l} \lambda(d_{l,l_1})\binom{n-w}{l_1}\binom{2w}{l-l_1}}$$

Let

$$\kappa^f_{n,k,w} = \log_2 \min\left\{PF_{n,k,w,l} : 2k-n+w \le l \le k-2\right\}. \tag{22}$$

Then in order to guarantee that the RLCE scheme is secure against filtration attacks, the parameters should be chosen in such a way that "$n-w \le k$" or "$n-w < 2k$ and $\kappa_c \le \kappa^f_{n,k,w}$".

Filtration attacks could be combined with Grover's quantum search algorithm. The quantum Filtration attacks works in the same way as the filtration attack that we have discussed in the preceding paragraph except that one uses Grover's quantum computer to select l columns from the public key G. A similar analysis as in the Section 6.1 shows that, under quantum filtration attacks, the RLCE scheme has quantum security level κ^f_q where

$$\kappa^f_q = \log_2 \min_{2k-n+w\le l\le k-2} \left\{ \frac{7\cdot(\log_2 q)^{1.585}\cdot\left(O(kl(n+w)) + O((k-l)^4(n+w-l))\right)\sqrt{\binom{n+w}{l}}}{\sqrt{\sum\limits_{l_1=\max\{0,l-2w\}}^{l} \lambda(d_{l,l_1})\binom{n-w}{l_1}\binom{2w}{l-l_1}}} \right\}. \tag{23}$$

For each of the recommended schemes in Table 3, the row $(\kappa^f_{n,k,w}, \kappa^f_q)$ in Table 2 shows the security strength under the filtration attack and quantum filtration attacks. For example, the RLCE scheme with ID = 1 in Table 3 has 128-bits security strength under filtration attacks and 85-bits security strength under quantum filtration attacks.

6.8 Filtration with brute-force attack

In addition to the filtration attacks that we have discussed in the preceding section, the adversary may carry out a filtration attack by exhaustively searching some GRS columns. That is, the adversary randomly selects $u \le w$ pairs of columns from the public key G with the hope that u columns of the underlying GRS code generator matrix could be reconstructed using exhaustive search. The probability that the u pairs are correctly selected so that u columns of the underlying GRS code generator matrix could be exhaustively searched from these u pairs is bounded by $\frac{\binom{w}{u}}{\binom{n+w}{2u}}$. For each pair $(\mathbf{x}_i, \mathbf{y}_i)$ of columns, one randomly selects two elements $a_i, b_i \in GF(q)$ and computes a column vector $a_i\mathbf{x}_i + b_i\mathbf{y}_i$. In case that the u pairs of column selection is correct, then the probability that these calculated u column vectors are correct GRS code generator matrix columns is bounded by $\frac{1}{q^{2u}}$. In a summary, one can obtain u columns of GRS code generator matrix from the public key with a probability $\frac{\binom{w}{u}}{q^{2u}\binom{n+w}{2u}}$.

Assume that one has correctly guessed u columns of the GRS code generator matrix and $k < n - w + u$. Similar to the discussion in the preceding section, we can distinguish two cases: $n - w + u \geq 2k$ and $n - w + u < 2k$. In case that $n - w + u \geq 2k$, the filtration attack could be carried out straightforwardly. Thus it is recommended to have $n < 2k$ so that $n - w + u \leq n < 2k$. In the following, we consider the case that $n - w + u < 2k$. Let G' be the $k \times (n + w - u)$ matrix consisting of the guessed u columns and the remaining $n - 2u$ columns of the public key. Randomly select $l < k - 1$ columns from the non-guessed columns of G' and shorten G' from these locations. Among the $l = l_1 + l_2$ columns, l_1 columns are non-randomized columns and l_2 columns are from randomized columns. Then the shortened code has dimension

$$d'_{l,l_1} = \min\left\{(k-l)^2, \min\left\{2(k-l_1)-1, n-w+u-l_1, (k-l)^2\right\} + \min\left\{2(w-u)-l_2, (k-l)^2\right\}\right\}. \tag{24}$$

A necessary condition for the filtration attack to be observable is that, after the shortening of the l_1 columns in G', the following conditions are satisfied

$$d'_{l,l_1} = 2(k-l_1) - 1 + \min\left\{2(w-u)-l_2, (k-l)^2\right\}. \tag{25}$$

Thus for a given l, the probability for the filtration attack to be successful is bounded by the probability

$$\frac{\sum_{l_1=\max\{0,l-2(w-u)\}}^{l} \lambda(d'_{l,l_1})\binom{n-w}{l_1}\binom{2w-2u}{l-l_1}}{\binom{n+w-2u}{l}}$$

where $\lambda(d'_{l,l_1}) = 1$ if (25) holds and $\lambda(d'_{l,l_1}) = 0$ otherwise. A similar discussion as in the preceding section shows that the expected time for one to carry out the filtration attack for a given l and u is

$$PF_{n,k,w,l,u} = \frac{q^{2u}\binom{n+w}{2u}\binom{n+w-2u}{l}\left(O(kl(n+w-2u)) + O((k-l)^4(n+w-2u-l))\right)}{\binom{w}{u}\sum_{l_1=\max\{0,l-2(w-u)\}}^{l} \lambda(d_{l,l_1})\binom{n-w}{l_1}\binom{2w-2u}{l-l_1}}$$

Let

$$\kappa_{n,k,w}^{fb} = \log_2 \min\left\{PF_{n,k,w,l,u} : 2k-n+w-u \leq l \leq k-u-2, 0 \leq u \leq w\right\}. \tag{26}$$

Then in order to guarantee that the RLCE scheme is secure against filtration attacks, the parameters should be chosen in such a way that $\kappa_c \leq \kappa_{n,k,w}^{fb}$. Similarly, filtration attacks with brute-force could be combined with Grover's quantum search algorithm and, under quantum filtration attacks with brute-force, the RLCE scheme has quantum security level κ_q^{fb} as

$$\log_2 \min_{l,u}\left\{\frac{7\cdot(\log_2 q)^{1.585}\cdot q^{2u}\cdot\left(O(kl(n+w-2u)) + O((k-l)^4(n+w-2u-l))\right)\sqrt{\binom{n+w}{2u}\binom{n+w-2u}{l}}}{\sqrt{\binom{w}{u}\sum_{l_1=\max\{0,l-2(w-u)\}}^{l}\lambda(d_{l,l_1})\binom{n-w}{l_1}\binom{2w-2u}{l-l_1}}}\right\}. \tag{27}$$

Our experiments show that the values $(\kappa_{n,k,w}^{fb}, \kappa_q^{fb})$ always equal $(\kappa_{n,k,w}^{f}, \kappa_q^{f})$ with $u = 0$. That is, there is no improvement by using the exhaustive search for filtration attacks.

6.9 Known non-randomized column attack revisited

In Section 6.6, we showed that if $n-w>k$ and the positions of non-randomized columns are known to the adversary, then the adversary can decrypt ciphertexts that contains a small number of errors within the punctuated $[n-w,k]$ GRS_k code. In this section, we calculate the maximum number of non-randomized column positions that could be published so that the adversary still cannot recover the underlying GRS_k code. Assume that the adversary knows l positions of non-randomized columns within the public key G. The adversary can carry out the following attacks.

1. randomly selects $u\le w$ pairs of columns from the remaining $n+w-l$ columns of the public key matrix G with the hope that u columns of the underlying GRS code generator matrix could be reconstructed using an exhaustive search.
2. randomly selects $l_1\le n-w-l$ columns from the remaining $n+w-l-2u$ columns of the public key matrix G with the hope that these l_1 columns are non-randomized columns of the underlying GRS code generator matrix.

The probability that the u pairs are correctly selected so that u columns of the underlying GRS code generator matrix could be exhaustively searched from these u pairs and that these l_1 columns are non-randomized columns is bounded by

$$P_{l_1,u}=\frac{\binom{w}{u}\binom{n-w-l}{l_1}}{\binom{n+w-l}{2u}\binom{n+w-l-2u}{l_1}}.$$

A similar analysis as in Section 6.8 can be used to show that the probability for the adversary to obtain additional $u+l_1$ columns of GRS code generator matrix using the above process is bounded by $\frac{P_{l_1,u}}{q^{2u}}$. For each guessed $u+l_1$ columns for the underlying GRS code, the adversary can test whether the guessed $u+l_1$ columns are correct by mounting the following filtration attack in case that $l+u+l_1\ge k+2$:

1. Use the $l+u+l_1$ columns to form an $[l+u+l_1,k]$ GRS_k code C_1.
2. Shorten the GRS_k code C_1 in $k-2$ positions to obtain an $[l+u+l_1-k+2,2]$ GRS_2 code C_2. Note that this process takes $O((l+u+l_1-k+2)^2)$ steps.
3. Compute the dimension of the square code C_2^2. If the dimension of the square code is less than 4, then with high probability that these u columns are actual columns of the the underlying GRS code.

Thus in order to achieve κ_c bit security, the maximum number $l\le k+1$ of publicly known positions for the non-randomized GRS columns should satisfy the following condition.

$$\kappa_c\le\log_2\min\left\{\frac{q^{2u}(l+u+l_1-k+2)^2}{P_{l_1,u}}:l_1\le\min\{n-w-l,k+2-l\},k-l+2-l_1\le u\le w\right\}. \tag{28}$$

6.10 Filtration attacks with partially known non-randomized columns

In Section 6.9, we showed the maximum number of non-randomized column positions that one can release while still keeping the RLCE scheme secure (though there is no need to release the positions of non-randomized columns in practice) under an exhaustive search and a filtration attack on a dimension 2 code. In this section, we calculate the maximum number of non-randomized column positions that could be released under general filtration attacks.

Assume that the positions of u columns of the underlying GRS code generator matrix is known and $k<n-w<2k$. Randomly select $l<k-1$ columns from the unknown positions of the public key G and

shorten G from these locations. Among the $l = l_1 + l_2$ columns, l_1 columns are non-randomized columns and l_2 columns are from randomized columns. Then the shortened code has dimension

$$d'_{l,l_1} = \min\left\{(k-l)^2, \min\left\{2(k-l_1)-1, n-w-l_1, (k-l)^2\right\} + \min\left\{2w-l_2, (k-l)^2\right\}\right\}. \tag{29}$$

A necessary condition for the filtration attack to be observable is that, after the shortening of the l_1 columns in G, the following conditions are satisfied

$$d'_{l,l_1} = 2(k-l_1) - 1 + \min\left\{2w - l_2, (k-l)^2\right\}. \tag{30}$$

Thus for a given l, the probability for the filtration attack to be successful is bounded by the probability

$$\frac{\sum_{l_1=\max\{0,l-2w\}}^{l} \lambda(d'_{l,l_1})\binom{n-w-u}{l_1}\binom{2w}{l-l_1}}{\binom{n+w-u}{l}}$$

where $\lambda(d'_{l,l_1}) = 1$ if (30) holds and $\lambda(d'_{l,l_1}) = 0$ otherwise. Thus the expected time for one to carry out the filtration attack with u-known non-random positions is

$$PKF_{n,k,w,l,u} = \frac{\left(O(kl(n+w)) + O((k-l)^4(n+w-l))\right)\binom{n+w-u}{l}}{\sum_{l_1=\max\{0,l-2w\}}^{l} \lambda(d'_{l,l_1})\binom{n-w-u}{l_1}\binom{2w}{l-l_1}}$$

Thus in order to achieve κ_c bit security, the maximum number $u \le k+1$ of publicly known positions for the non-randomized GRS columns should satisfy the following condition.

$$\kappa_c \le \log_2 \min\left\{PKF_{n,k,w,l,u} : 2k - n + w \le l \le k-2, 0 \le u \le k\right\}. \tag{31}$$

For each of the recommended schemes in Table 3, the row "known non-rand. col." in Table 2 shows the maximum number of non-randomized column positions that could be made public to the adversary in the public key under the conditions (28) and (31). As an example, for the RLCE scheme with ID = 1 in Table 3, the adversary can learn at most 301 columns of non-randomized GRS columns within the public key. In other words, if the adversary learns the positions of more than 301 non-randomized GRS columns, the security strength of the scheme will be less than 128-bits.

6.11 Related message attack, reaction attack, and side channel attacks

Berson [4] discussed the following related message attack. Assume that $\mathbf{c}_1 = \mathbf{m}_1 G + \mathbf{e}_1$, $\mathbf{c}_2 = \mathbf{m}_2 G + \mathbf{e}_2$, and 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 appropriate message padding for IND-CCA2 security that will be discussed in Section 7.

Hall et al [12] discussed the following reaction attack. Assume 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 appropriate message padding for IND-CCA2 security that will be discussed in Section 7.

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 decrypt 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 $\mathbf{m} \in GF(q)^k$ 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 information that could be encoded is less than $2^{m(k+t)}$ since e_{i_j} cannot be zeros. That is, one can only encode information symbols $(x_1, \cdots, x_{k+t}) \in GF(q)^{k+t}$ with $x_{k+1} \cdot x_{k+1} \cdots x_{k-t} \neq 0$.

3. **advancedEncoding**: In addition to **mediumEncoding**, further information is encoded within the locations 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. It should be noted that for **advancedEncoding**, the adversary knows that the encoded locations within the interval $\left[2^{\lfloor \log_2 \binom{n+w}{t} \rfloor}, \binom{n+w}{t}\right]$ will not be the error location candidates.

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

$$\varphi : W_{n+w,t} \leftrightarrow \left\{ i : 1 \leq i \leq \binom{n+w}{t} \right\} \tag{32}$$

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 (32), one may use the enumerative source encoding construction in Cover [8]:

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

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.

7.2 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+ [21] and SAEP+ [5]. The first simple padding scheme RLCEspad is a one-round 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 `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 RLCE is over the finite field $GF(2^m)$. The RLCEspad proceeds as follows.

`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 $8k_1 \le \texttt{mLen}/4$. 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

$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e})) \oplus H_2(\mathbf{r}, \mathbf{e})) \| \mathbf{r}. \tag{33}$$

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

$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e}_0)) \oplus H_2(\mathbf{r}, \mathbf{e}_0)) \| \mathbf{r}. \tag{34}$$

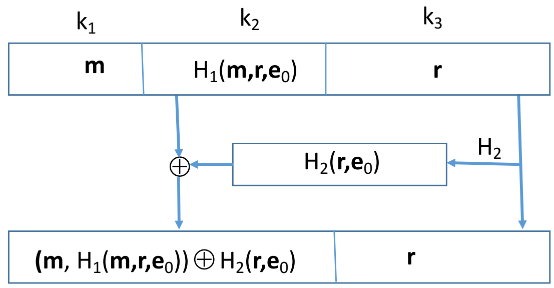
  Convert $\mathbf{y}$ to an element $(\mathbf{y}_1, \mathbf{e}_1) \in GF(q)^{k+t}$ where $\mathbf{y}_1 \in GF(q)^k$ and $\mathbf{e}_1 \in GF(q)^t$. Let $\mathbf{e} \in GF(q)^{n+w}$ such that $\mathbf{e}[l_i] = \mathbf{e}_1[i]$ for $0 \le i < t$ and $\mathbf{e}[j] = 0$ for $j \ne l_i$. Let the ciphertext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

Assuming the hardness of decoding RLCE ciphertexts, a similar proof as in [5] 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 $(630, 470, 80)$ over $GF(2^{10})$ in Table 3, we use $k_1 = k_2 = 171$-bytes for mediumEncoding and $k_1 = k_2 = 183$-bytes for advancedEncoding. Thus, we can encrypt $k_1 = 171$-bytes of information for

Figure 1: mediumEncoding based RLCEspad



mediumEncoding and $k_1 = 183$-bytes of information for advancedEncoding per RLCE-RLCEspad ciphertext.

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

$\texttt{RLCEpad}(\texttt{mLen}, k_1, k_2, k_3, t)$: Let k_1, k_2, k_3 be parameters such that $k_1 + k_2 + k_3 = \left\lceil \frac{\texttt{mLen}}{8} \right\rceil$ and $v = 8(k_1 + k_2 + k_3) - \texttt{mLen}$. Let H_1, H_2, and H_3 be random oracles that take arbitrary-length binary input strings and output k_2-bytes, $(k_1 + k_2)$-bytes, and k_3-bytes strings respectively. Let $\mathbf{m} \in \{0, 1\}^{8k_1}$ be a message to be padded and $\mathbf{r} = \mathbf{r}_0 \| 0^v$ where $\mathbf{r}_0 \in \{0, 1\}^{8k_3 - v}$ is a randomly selected binary string. Then the padding process proceeds as follows:

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

$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e})) \oplus H_2(\mathbf{r}, \mathbf{e})) \| \mathbf{r} \oplus H_3(((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e})) \oplus H_2(\mathbf{r}, \mathbf{e}))) \tag{35}$$

  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 $0 \le l_0 < l_1 < \cdots < l_{t-1} \le n + w - 1$ and let $\mathbf{e}_0 = l_0 \| l_1 \cdots \| l_{t-1} \in \{0, 1\}^{16t}$. Note that one may use $\mathbf{r}$ to compute $\mathbf{e}_0$. Set

$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e}_0)) \oplus H_2(\mathbf{r}, \mathbf{e}_0)) \| (\mathbf{r} \oplus H_3((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r}, \mathbf{e}_0)) \oplus H_2(\mathbf{r}, \mathbf{e}_0))) \tag{36}$$

  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 $0 \le i < t$ and $\mathbf{e}[j] = 0$ for $j \ne l_i$. Outputs $(\mathbf{y}_1, \mathbf{e})$.

- advancedEncoding: Set

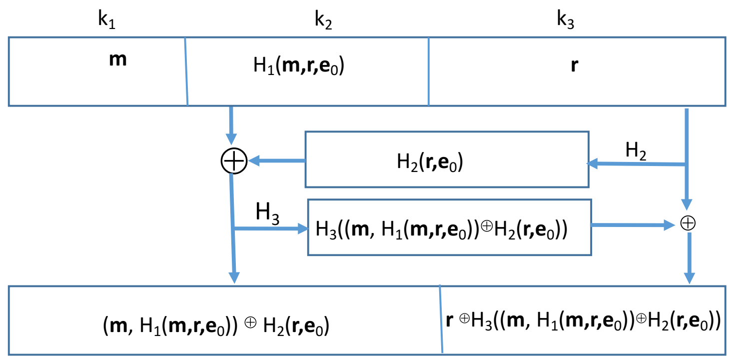
$$\mathbf{y} = ((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r})) \oplus H_2(\mathbf{r})) \| \mathbf{r} \oplus H_3(((\mathbf{m} \| H_1(\mathbf{m}, \mathbf{r})) \oplus H_2(\mathbf{r}))) \tag{37}$$

  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.

Shoup [21, Theorem 3] showed the following result for OAEP+: "*If the underlying trapdoor permutation scheme is one way, then OAEP+ is secure against adaptive chosen ciphertext attack in the random oracle model*". Our padding scheme RLCEpad is identical to OAEP+ with the following exceptions: In OAEP+, the function H_2 outputs a string of k_1-bytes which is $\oplus$-ed with $\mathbf{m}$. In RLCEpad, the function H_2 outputs a string of $(k_1 + k_2)$-bytes which is $\oplus$-ed with $\mathbf{m} \| H_2(\mathbf{r}, \mathbf{e_0})$. Since H_1, H_2, H_3 are random oracles,

Figure 2: mediumEncoding based RLCEpad



this revision requires no change in the security proof of [21, Theorem 3]. Thus, assuming the hardness of decoding RLCE ciphertexts, the proof in [21, Theorem 3] could be used to show that RLCE-RLCEpad is secure against IND-CCA2 attacks. The proof in [21] shows that the adversary A's advantage is bounded by

$$\mathtt{InvAdv}(A') + \frac{(q_{H_1} + q_D)}{2^{8k_3}} + \frac{(q_D + 1)q_{H_2}}{2^{8k_2}} \tag{38}$$

where q_D is the maximum number of decryption oracle queries, q_{H_1}, q_{H_2}, and q_{H_3} are the maximum number of queries made by A to the oracles H_1, H_2 and H_3 respectively, and $\mathtt{InvAdv}(A')$ is the success probability that a particular adversary A' has in breaking the one-way trapdoor permutation scheme. Furthermore, the time and space requirements of A' are related to those of A as follows:

$$\begin{aligned} T(A') &= O\left(T(A) + q_{H_2} q_{H_3} T_f + (q_{H_1} + q_{H_2} + q_{H_3} + q_D)\mathtt{mLen}\right) \\ S(A') &= O\left(S(A) + (q_{H_1} + q_{H_2} + q_{H_3})\mathtt{mLen}\right) \end{aligned}$$

where T_f is the time required to compute the one-way permutation f and space is measured in bits of storage.

The selection of RLCEpad parameters k_1, k_2, k_3 in Table 3 is based on the above reduction proof and bounds. As an example, for 128-bit secure RLCE scheme $(532, 376, 78)$, we use $k_2 = k_3 = 32$-bytes. Thus, we can encrypt $k_1 = 504$-bytes of information.

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 Recommended parameters

In Section 6, we carried out security analysis on the RLCE schemes. Based on these analysis, RLCE parameters for various security strength are recommended in Table 3. In particular, the recommendation takes

Table 3: Padding parameters: bE for basicEncoding, mE for mediumEncoding and aE for advancedEncoding

ID	κ_c	κ_q	LD	n	k	t	w	m	sk	cipher	pk		mLen	RLCEspad $k_1(k_2)$	RLCEspad k_3	RLCEpad k_1	RLCEpad $k_2(k_3)$
0	**128**	**80**	$\perp$	630	470	80	160	**10**	310116 192029	988	188001	bE	4700	146	296	524	32
												mE	5500	171	346	624	32
												aE	5869	183	368	670	32
1	**128**	**80**	$\perp$	532	376	78	96	**10**	179946 121666	785	118441	bE	3760	117	236	406	32
												mE	4540	141	286	504	32
												aE	4875	152	306	546	32
2	**192**	**110**	$\perp$	1000	764	118	236	**10**	747393 457073	1545	450761	bE	7640	238	479	859	48
												mE	8820	275	553	1007	48
												aE	9377	293	587	1077	48
3	**192**	**110**	$\perp$	846	618	114	144	**10**	440008 292461	1238	287371	bE	6180	193	387	677	48
												mE	7320	228	459	819	48
												aE	7825	244	491	883	48
4	**256**	**144**	$\perp$	1360	800	280	560	**11**	1773271 1241971	2640	1232001	bE	8800	275	550	980	60
												mE	11880	371	743	1365	60
												aE	13025	407	815	1509	60
5	**256**	**144**	$\perp$	1160	700	230	311	**11**	1048176 749801	2023	742089	bE	7700	240	483	843	60
												mE	10230	319	641	1159	60
												aE	11145	348	698	1274	60
6	22	22	$\perp$	40	20	10	5	**10**	1059 859	57	626	bE	200	6	13	17	4
												mE	300	9	20	30	4
												aE	331	10	22	34	4
7	**128**	**80**	(13,6663,14)	612	466	76	146	**10**	284636 173961	948	170091	bE	4660	145	293	519	32
												mE	5420	169	340	614	32
												aE	5771	180	362	658	32
8	**128**	**80**	(9,3767,10)	520	380	73	87	**10**	166998 110948	759	107826	bE	3800	118	239	411	32
												mE	4530	141	285	503	32
												aE	4847	151	304	542	32
9	**192**	**110**	(11,9317,12)	1000	790	108	210	**10**	703371 420946	1513	414751	bE	7900	246	496	892	48
												mE	8980	280	563	1027	48
												aE	9500	296	596	1092	48
10	**192**	**110**	(8,5358,9)	828	620	107	128	**10**	401724 265324	1195	260401	bE	6200	193	389	679	48
												mE	7270	227	455	813	48
												aE	7748	242	485	873	48
11	**256**	**144**	(26,23350,34)	1200	700	280	500	**11**	1382314 971326	2338	926501	bE	7700	240	483	843	60
												mE	10780	336	676	1228	60
												aE	11872	371	742	1364	60
12	**256**	**144**	(62,49149,82)	1050	590	262	330	**11**	888230 648100	1898	640889	bE	6460	202	408	692	60
												mE	9372	292	588	1052	60
												aE	10334	322	648	1172	60
13	24	24	(3, 68,4)	40	20	11	5	**10**	1059 859	57	626	bE	200	6	13	13	6
												mE	310	9	21	27	6
												aE	343	10	23	31	6
14	25	25	(10, 262,14)	40	20	12	5	**10**	1059 859	57	626	bE	200	6	13	13	6
												mE	320	10	20	28	6
												aE	354	11	23	33	6

into account of the conditions for avoiding improved classical and quantum information set decoding, the conditions for avoiding Sidelnikov-Shestakov attacks, the conditions for filtration attacks (with or without brute force), the cost of recovering McEliece encryption scheme secret keys from the public keys, and the cost of recovering plaintext messages from ciphertexts. In Table 3, κ_c denotes the conventional security strength and κ_q denotes the quantum security strength. For example, $\kappa_c = 128$ means an equivalent security of AES-128. The recommended parameters is based on any underlying MDS linear code (e.g., GRS code) over $GF(q)$ where $q = 2^{\lceil \log_2 n \rceil}$. For GRS codes, the BCH-style 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.

In Table 3, the schemes with ID $= 0, 1, 2, 3, 4, 5, 6$ are for MDS codes with $t = \frac{n-k}{2}$. The schemes with ID $= 7, 8, 9, 10, 11, 12, 13, 14$ are for MDS codes with $t > \frac{n-k}{2}$. The schemes with ID $= 6, 13, 14$ are for testing purpose only. The schemes with ID ≥ 7 require a list-decoding algorithm for the RLCE decryption process and have relatively smaller public key sizes. The column LD of Table 3 contains list-decoding parameters. For example, the scheme ID $= 7$ has LD parameter $(13, 6663, 14)$ which means that the interpolation process of the list decoding uses a zero multiplicity 13 and constructs a polynomial $Q(x, y)$ with a maximum x-degree of 6663 and a maximum y-degree of 14. List-decoding helps to reduce the public key size. For example, for 128-bit security, the scheme 1 without list-decoding has a public key of 118441 bytes and the scheme 8 with list-decoding has a public key of 107826 bytes. However, list decoding is extremely slow. For Kötter's iterative interpolation based list-decoding, it takes 1851 seconds (around 31 minutes) to decrypt a ciphertext for scheme 8 on a MacBook Pro with 2.9 GHz Intel Core i7 though it only takes 0.004884 seconds to decrypt a ciphertext for scheme 1 of the same security strength on the same machine. Note that it takes 0.034844 seconds to list-decrypt a ciphertext for the testing scheme 13 on the same MacBook Pro machine. For schemes in Table 3, the security strength under each specific attack discussed in this paper is listed in Table 2.

The following is a comparison of the parameters in Table 3 against binary Goppa code based McEliece encryption scheme parameters from [3]. Note that for RLCE encryption schemes over $GF(q)$, the systematic generator matrix public key is $k(n+w-k)\log q$ bits.

1. For the security strength 128, binary Goppa code uses $n = 2960, k = 2288, t = 57$ and the public key size is 188KB while RLCE has a public key size of 118001 bytes (that is, 115KB).

2. For the security strength 192, binary Goppa code uses $n = 4624, k = 3468, t = 97$ and the public key size is 490KB while RLCE has a public key size of 287371 bytes (that is, 280KB).

3. For the security strength 256, binary Goppa code uses $n = 6624, k = 5129, t = 117$ and the public key size is 900KB while RLCE has a public key size of 742089 bytes (that is, 724KB).

Table 3 also lists the message bandwidth and message padding scheme parameters for the recommended schemes. 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. In the column for `sk`, the first row is the private key size for RLCE scheme with decoding algorithm 1 and 3. The second row is the private key size for RLCE scheme with decoding algorithm 2.

9 Performance evaluation

9.1 Time cost

Table 4 lists the performance results for RLCE encryption scheme with decoding algorithm 0 on a MacBook Pro with 2.9 GHz Intel Core i7 and MacOS Sierra. The first column contains the encryption scheme ID from Table 3. 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 plaintext encryption. The fourth two-column group contains the time needed for one ciphertext decryption.

Table 4: Running times for RLCE with Decoding Algorithm 0 (in milliseconds)

ID	key	encryption		decryption	
		RLCEspad	RLCEpad	RLCEspad	RLCEpad
0	311.375	0.566	0.539	1.754	1.718
1	151.834	0.378	0.360	1.385	1.345
2	1151.206	1.229	1.166	3.474	3.432
3	637.988	0.814	0.776	2.717	2.676
4	2745.302	2.832	2.765	14.171	13.853
5	1587.330	2.216	1.745	9.324	9.383

Table 5 lists the performance results for RLCE encryption scheme with decoding algorithm 1 and pre-computation of the matrix W^{-1} (see Section 5.2 for details). It was tested with MacOS Sierra on a MacBook Pro with 2.9 GHz Intel Core i7. The pre-computation time for W^{-1} is included in the key generation process.

Table 5: Running times for RLCE with Decoding Algorithm 1 (in milliseconds)

ID	key	encryption		decryption	
		RLCEspad	RLCEpad	RLCEspad	RLCEpad
0	340.616	0.565	0.538	1.574	1.509
1	161.504	0.378	0.372	1.221	1.181
2	1253.926	1.255	1.166	3.034	2.937
3	667.239	0.815	0.791	2.396	2.340
4	3215.791	2.836	2.796	13.092	12.925
5	1678.032	2.242	1.763	8.560	8.572

Table 6 lists the performance results for RLCE encryption scheme with decoding algorithm 2. It was tested with MacOS Sierra on a MacBook Pro with 2.9 GHz Intel Core i7.

Table 6: Running times for RLCE with Decoding Algorithm 2 (in milliseconds)

ID	key	encryption		decryption	
		RLCEspad	RLCEpad	RLCEspad	RLCEpad
0	314.711	0.570	0.533	1.832	1.417
1	154.143	0.385	0.360	1.095	1.133
2	1169.991	1.267	1.176	3.199	2.946
3	635.208	0.814	0.788	2.547	2.300
4	2747.790	2.882	3.278	19.859	19.163
5	1561.936	2.263	1.772	10.939	10.932

For the list-decoding based RLCE encryption scheme, we only tested scheme with ID=7. For RLCE scheme 7, the key generation time is approximately 0.516495 seconds, the encryption time is approximately 0.001598 seconds, and the decryption time is approximately 1865 seconds (that is, approximately 31 minutes).

9.2 CPU cycles

Table 7 lists the CPU cycles for RLCE encryption scheme with decoding algorithms 0, 1, and 2 respectively. It was tested with MacOS Sierra on a MacBook Pro with 2.9 GHz Intel Core i7. The first column contains the encryption scheme ID from Table 3. The second column contains the padding scheme ID where 0 is for RLCEspad-mediumEncoding, 1 is for RLCEpad-mediumEncoding, 2 is for RLCEspad-basicEncoding, and 3 is for RLCEpad-basicEncoding. The third column group contains CPU cycles for a public/private key pair generation with algorithm 0, 1, and 2 respectively. The fourth column contains CPU cycles for encrypting a plaintext. The fifth column group contains CPU cycles for decrypting a ciphertext with algorithm 0, 1, and 2 respectively.

9.3 Memory requirements

Table 8 lists the memory requirements for RLCE encryption scheme with decoding algorithms 0, 1, and 2 respectively. It was tested on a Amazon AWS cloud computer running Ubuntu 16.10 with Intel(R) Xeon(R)

Table 7: RLCE CPU cycles

ID		key generation			encryption	decryption		
		algorithm 0	algorithm 1	algorithm 2		algorithm 0	algorithm 1	algorithm 2
0	0	965601110	11077817663	933934514	605316	5005712	5542671	5234518
0	1	934514130	1011071617	933987433	1805010	5355784	4646941	4162641
0	2	905419375	1011384425	930692646	1647873	6030799	4315597	5273153
0	3	916456332	1027954991	919679402	1502708	4724183	4457827	4079080
1	0	447781824	474300568	452928596	1099797	3936598	4559158	3330758
1	1	454712789	465481183	451876423	1040629	3765661	3589491	3308529
1	2	440842214	472911996	460746862	1072426	3930516	3594353	3328138
1	3	445231426	484260798	445410359	990231	3722669	3539037	3369610
2	0	3481345844	3778948523	3503531149	3488662	10061600	8946359	9563820
2	1	3450091135	3829675407	3501563776	3331234	9794176	8668186	8966646
2	2	3455930364	3896012515	3484052736	3461628	10119659	9733856	8816277
2	3	3478119945	3991809655	3557466677	3827084	9928669	9617728	8926914
3	0	1867717927	089254043	1876262340	2491667	8149425	7307906	7412701
3	1	1865554282	1962533052	1885644975	2361787	8048040	7160709	6993236
3	2	1847405744	1952723585	1876279636	2355121	9107857	6782841	6708294
3	3	1849732098	1962033778	1876339342	2339344	7832494	6987142	7166795
4	0	8114178839	9545637831	8332547491	8361371	39084326	38060629	57070221
4	1	8108201681	9612380645	8186025651	8184051	39099009	36705481	53669412
4	2	8081815282	9605949548	8138963149	8506474	40063190	38216918	59542579
4	3	8091168939	9590945573	8165136802	9428383	41171497	36879770	59958433
5	0	4696770782	5085862230	4722940209	6903975	26618836	24660121	30604712
5	1	4682712937	5057459034	4763826045	5362174	28191447	24174369	29967843
5	2	4706366223	5079303736	4741589960	5542775	26372021	24171293	31348693
5	3	4738340942	5046201517	4728263914	5474734	26915440	25084556	32945876

CPU E5-2630L v2 @ 2.40GHz. The first column is the RLCE scheme ID. The second column shows whether a finite field multiplication table is generated or not. These data shows that for schemes over $GF(2^{10})$ (that is, schemes 0, 1, 2, 3 for 128-bit and 192-bit security), there is around 2MB difference for the RAM requirement with a multiplication table and without a multiplication table. For schemes over $GF(2^{11})$ (that is, schemes 4, 5 for 256-bit security), there is around 7MB difference for the RAM requirement with multiplication table and without multiplication table. In practice, it is convenient to deploy hardware based multiplication tables.

9.4 Performance comparison with OpenSSL RSA

Table 9 shows the comparison of the RLCE performance against OpenSSL RSA performance. Both RSA and RLCE were tested with a MacOS Sierra on a MacBook Pro with 2.9 GHz Intel Core i7.

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 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 an implementation of RLCE over GRS codes, Wang [25] has a complete review and comparison on related algorithms.

Table 8: RLCE peak memory usage (bytes)

ID	Mul. Table	key generation			encryption	decryption		
		algorithm 0	algorithm 1	algorithm 2		algorithm 0	algorithm 1	algorithm 2
0	N	2,536,672	2,536,704	2,317,680	798,288	1,335,160	1,335,280	825,048
0	Y	4,648,624	4,648,656	4,447,144	2,437,320	2,832,216	2,856,584	2,629,128
1	N	1,571,912	1,571,944	1,467,064	507,632	779,616	779,616	525,920
1	Y	3,668,920	3,668,952	3,588,240	2,326,736	2,464,984	2,609,160	2,506,840
2	N	6,178,712	6,178,744	5,687,016	1,906,576	3,178,568	3,178,688	1,947,088
2	Y	8,287,280	8,287,312	7,803,048	2,865,400	3,443,432	3,825,112	3,144,688
3	N	3,896,376	3,896,408	3,657,112	1,222,944	1,881,600	1,881,728	1,250,864
3	Y	5,997,968	5,998,000	5,764,680	2,605,112	3,116,736	3,119,496	2,871,984
4	N	11,561,320	11,561,352	10,775,384	4,829,968	7,010,248	7,010,368	4,912,640
4	Y	19,975,008	19,975,040	19,223,704	10,258,112	12,227,368	12,227,384	11,433,720
5	N	7,345,376	7,345,408	6,909,152	2,920,256	4,152,096	4,152,096	2,971,680
5	Y	15,713,624	15,713,656	15,268,544	9,547,848	10,970,216	10,970,232	10,522,168

Table 9: Comparisson of RLCE and RSA performance (milliseconds)

κ_c	RSA modulus	key setup		encryption		decryption	
		RSA	RLCE	RSA	RLCE	RSA	RLCE
128	3072	433.607	151.834	0.135540	0.360	6.576281	1.345
192	7680	9346.846	637.988	0.672769	0.776	75.075443	2.676
256	15360	80790.751	1587.330	2.498523	1.745	560.225740	9.383

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] D.J. Bernstein. Grover vs. McEliece. In *Proc. Int. Workshop PQC*, pages 73–80. Springer, 2010.

[3] D.J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *Proc. Int. Workshop PQC*, pages 31–46. Springer, 2008.

[4] 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.

[5] D. Boneh. Simplified OAEP for the RSA and rabin functions. In *Proc. CRYPTO*, pages 275–291. Springer, 2001.

[6] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece cryptosystem. In *Proc. Asiacrypt*, pages 187–199. Springer, 1998.

[7] 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.

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

[9] 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.

[10] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt. Applying Grover's algorithm to AES: quantum resource estimates. In *Proc. Int. Workshop PQC*, pages 29–43. Springer, 2016.

[11] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Tran. Information Theory*, 45:1757–1767, 1999.

[12] C. Hall, I. Goldberg, and B. Schneier. Reaction attacks against several public-key cryptosystem. In *Proc. ICICS*, pages 2–12. Springer, 1999.

[13] J. Justesen and T. Hoholdt. Bounds on list decoding of mds codes. *Information Theory, IEEE Tran.*, 47(4):1604–1609, 2001.

[14] G. Kachigar and J.-P. Tillich. Quantum information set decoding algorithms. Cryptology ePrint Archive, Report 2017/213, 2017. http://eprint.iacr.org/2017/213.

[15] S. Kepley and R. Steinwandt. Quantum circuits for F_{2^m}-multiplication with subquadratic gate count. *Quantum Information Processing*, 14(7):2373–2386, 2015.

[16] P. Loidreau and N. Sendrier. Some weak keys in mceliece public-key cryptosystem. In *Proc. IEEE ISIT*, pages 382–382, 1998.

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

[18] 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.

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

[20] C. Peters. Information-set decoding for linear codes over F_q. In *Proc. Int. Workshop PQC*, pages 81–94. Springer, 2010.

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

[22] 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.

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

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

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

[26] C. Wieschebrink. Two NP-complete problems in coding theory with an application in code based cryptography. In *Proc. IEEE ISIT*, pages 1733–1737. IEEE Press, 2006.