Code-based Key Encapsulation Mechanism Preserving Short Ciphertext and Secret Key

Jayashree Dey and Ratna Dutta
Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur-721302, India

Keywords: Public Key Encryption, Key Encapsulation Mechanism, MDS Code, Companion Matrix.

Abstract: Post-quantum cryptography has recently drawn considerable attention from both industry and academia due to the impending threat by quantum computers. Developing *key encapsulation mechanism* (KEM) that resists attacks equipped with quantum computers has become relevant as KEM is used in practice quite heavily. Coding theory is an attractive option to guarantee secure communication in the post-quantum world. Motivated by the goal of improving efficiency, we revisit code-based KEM in this article. We present basicPKE, a *public key encryption* (PKE) scheme using a parity check matrix of *maximum distance separable* (MDS) code. Our construction is built on top of a companion matrix in deriving an MDS code. This significantly reduces the secret key size. We support the conjectured security of basicPKE by analysis and prove that the scheme achieves security against *indistinguishability under chosen plaintext attacks* (IND-CPA) in the random oracle model. Following the design framework of basicPKE, we construct fullPKE that leads to the design of fullKEM. We have shown that fullPKE is secure against *one-wayness under plaintext and validity checking attacks* (OW-PCVA) and fullKEM achieves security against *indistinguishability under chosen ciphertext attacks* (IND-CCA) in the random oracle model. An appealing feature of fullKEM is that it exhibits better performance guarantee in terms of communication bandwidth and secret key size when contrasted with existing similar approaches.

1 INTRODUCTION

With the proliferation of a wide range of Internet of Things (IoT), IoT devices are increasingly used to store secrets that are used to authenticate users or enable secure payments. Many IoT devices are not equipped with proper environments to store secret keys and provide developers with little programmability for their applications. It is therefore desirable to leverage the fact that users own multiple devices such as smart phone, smart watch, smart TV, etc. and enable multi-device cryptographic functionalities without making strong assumptions about a device's security features. Given the limited computation and communication power of IoT devices, it is essential to come up with cryptographic primitives that meet short communication bandwidth as well as short secret key.

Key encapsulation mechanism (KEM) is an important building block in cryptography which plays a vital role in transmitting symmetric key information securely utilizing asymmetric algorithms. Although public key cryptosystems are incompetent to communicate long messages in practice, KEM is beneficial to exchange relatively short symmetric key which is then used for encrypting a longer message. More precisely, one can encrypt a random symmetric key using the preferred public key algorithm. The receiver recovers the symmetric key by the decryption procedure of the public key algorithm.

Given the rapidly expanding set of KEM applications and *public key encryption* (PKE), there is a strong interest in making PKE and KEM post-quantum secure. Error-correcting codes play a significant role in designing quantum-safe alternatives while offering reasonable performance with solid security guarantees as code-based cryptographic schemes are usually very fast and can be implemented on several platforms.

Our Contribution. Since its introduction by McEliece (McEliece, 1978) in 1978, numerous proposals in constructing PKEs and KEMs based on error correcting codes have been presented yielding various improvements in security and efficiency. A line of recent work are offered to NIST call in 2016 for standardization of quantum safe cryptography (Barreto et al., 2017), (Bardet et al., 2017), (Yamada et al., 2017), (Bernstein et al., 2017), (Kim et al.,

Dey, J. and Dutta, R.
Code-based Key Encapsulation Mechanism Preserving Short Ciphertext and Secret Key.
DOI: 10.5220/0011273900003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 374-381
ISBN: 978-989-758-590-6; ISSN: 2184-7711

2018), (Aguilar-Melchor et al., 2018), (Albrecht et al., 2019), (Aragon et al., 2017a), (Banegas et al., 2018), (Baldi et al., 2018), (Szepieniec, 2017), (Wang, 2017), (Aragon et al., 2017b), (Melchor et al., 2017), (Aragon et al., 2017c), (Aragon et al., 2017b), (Melchor et al., 2017), (Aragon et al., 2017c), (Melchor et al., 2019), focusing on constructing KEM offering improvements in security and efficiency. The challenge lies in the requirement of low communication bandwidth and short key size while providing strong security guarantee. This motivates our search for a code-based KEM featuring *indistinguishability under chosen ciphertext attacks* (IND-CCA) security under the hardness of the syndrome decoding with relatively short ciphertext and short secret key. Using the structure of a companion matrix, we form an MDS code and integrate it to design basicPKE by coupling with the Niederreiter encryption (Nojima et al., 2008). We have shown that basicPKE achieves *indistinguishability against chosen plaintext attacks* (IND-CPA). More specifically, we use the parity check matrix of the MDS code as the public key matrix and keep the last row of the companion matrix as secret key. Our approach yields significant reduction in the secret key size and communication overhead, making the scheme useful in applications with limited communication bandwidth. Moreover, use of parity check matrix instead of generator matrix enables faster encapsulation. MDS codes satisfy the Singleton bound and have efficient decoding algorithm which is employed to decode ciphertext during decryption.

We extend basicPKE to fullPKE which is proven to be *one-wayness under plaintext and validity checking attacks* (OW-PCVA) secure in the random oracle model. Finally, we build *indistinguishability under chosen ciphertext attacks* (IND-CCA) secure fullKEM from fullPKE. We briefly discuss our fullKEM in reference to the previous similar works.

• As exhibited in Table 1, our fullKEM offers several strong advantages over the existing approaches ((Albrecht et al., 2019), (Aragon et al., 2019), (Bernstein et al., 2017), (Baldi et al., 2018), (Bardet et al., 2017), (Banegas et al., 2018), (Dey and Dutta, 2019)) that are constructed over finite fields with characteristic 2. The most appealing feature of our fullKEM is that we use MDS code in our work owning the binary structure and the use of the companion matrix helps to reduce the secret key size. A bit more precisely, the secret key size of our construction is comparatively shorter than the schemes ((Albrecht et al., 2019), (Bernstein et al., 2017), (Bardet et al., 2017), (Baldi et al., 2018),(Banegas et al., 2018), (Dey and Dutta, 2019)) although the size of public key remains large. The BIKE variants and LEDAkem are efficient in terms of public key sizes and achieve IND-CCA security. However, they experience a small decoding failure rate unlike our candidate.

• More interestingly, our fullKEM supports low communication overhead and performs better in terms of ciphertext size over DAGS (Banegas et al., 2018) and (Dey and Dutta, 2019). It uses parity check matrix during encapsulation instead of generator matrix which leads faster encapsulation in contrast to DAGS and NTS-KEM. In fact, the encapsulation procedure of fullKEM is closest to the work in (Bardet et al., 2017).

• To prove the security of fullKEM, we follow the generic transformations by Hofheinz et al. (Hofheinz et al., 2017). More concretely, our basicPKE supports IND-CPA security in random oracle model under the hardness of syndrome decoding problem and the indistinguishability of the public key matrix from a random matrix. We prove that breaking OW-PCVA security of our fullPKE would lead to breaking the IND-CPA security of basicPKE in the random oracle model. Achieving IND-CCA security of fullKEM seems quite tricky despite of its potential for comparative simplicity. As OW-PCVA security always implies *one-wayness under validity attacks* (OW-VA) security with zero queries to the plaintext checking oracle, the OW-PCVA security and consequently the OW-VA security of fullPKE follows from the IND-CPA security of basicPKE. We show that the OW-VA security of fullPKE implies the IND-CCA security of fullKEM in the random oracle model. We can further extend our security proof in the quantum random oracle following the work by Hofheinz et al. (Hofheinz et al., 2017).

2 PRELIMINARIES

In this section, we provide mathematical background and preliminaries that are necessary to follow the discussion in the paper.

Notation. We use the notation $\mathsf{x} \xleftarrow{U} X$ for choosing a random element from a set or distribution, $a \longleftarrow A$ for the sampling according to some distribution A, $\mathsf{wt}(\mathbf{x})$ to denote the weight of a vector $\mathbf{x}$, $(\mathbf{x}||\mathbf{y})$ for the concatenation of the two vectors $\mathbf{x}$ and $\mathbf{y}$. The matrix I_n is the $n \times n$ identity matrix. We let $\mathsf{GF}(q)$ to denote the Galois field of cardinality q and $\mathbb{Z}^+$ to represent the set $\{a \in \mathbb{Z} | a \geq 0\}$ where $\mathbb{Z}$ is the set of integers. We denote the transpose of a matrix A by A^T and concatenation of two matrices A and B by $[A|B]$. The uniform distribution over $c \times d$ random q-ary matrices is de-

Table 1: Comparative summary of IND-CCA secure KEMs.

Scheme	pk size (in bits)	sk size (in bits)	CT size (in bits)	Code used	Cyclic/Dyadic	Correctness error
NTS-KEM (Albrecht et al., 2019)	$(n-k)k$	$2(n-k+r)m$ $+nm+r$	$(n-k+r)$	Binary Goppa	–	No
BIKE-1 (Aragon et al., 2019)	n	$n+w\cdot\lceil\log_2 k\rceil$	n	MDPC	Quasi-cyclic	Yes
BIKE-2 (Aragon et al., 2019)	k	$n+w\cdot\lceil\log_2 k\rceil$	k	MDPC	Quasi-cyclic	Yes
BIKE-3 (Aragon et al., 2019)	n	$n+w\cdot\lceil\log_2 k\rceil$	n	MDPC	Quasi-cyclic	Yes
LEDAkem (Baldi et al., 2018)	$n-\ell$	$\frac{n}{\ell}w\log_2\ell$	ℓ	LDPC	Quasi-cyclic	Yes
Classic McEliece (Bernstein et al., 2017)	$k(n-k)$	$n+mt+mn$	$(n-k)+r$	Binary Goppa	–	No
BIG QUAKE (Bardet et al., 2017)	$\frac{k}{\ell}(n-k)$	$mt+mn$	$(n-k)+2r$	Binary Goppa	Quasi-cyclic	No
DAGS (Banegas et al., 2018)	$\frac{k}{s}(n-k)\log_2 q$	$2mn\log_2 q$	$[n+k']\log_2 q$	GS	Quasi-dyadic	No
(Dey and Dutta, 2019)	$\frac{k}{s}(n-k)\log_2 q$	$2mn\log_2 q$	$[k'+(n-k)]\log_2 q$	GS	Quasi-dyadic	No
fullKEM	k^2m^2	km	$2k'+km$	MDS	–	No

pk=public key, sk=secret key, CT=ciphertext, k=dimension of the code, n=length of the code, ℓ=length of each blocks, t=error correcting capacity, $k'<k,s,r,w,p_1,p_2$ are positive integers ($\ell << s$), $s=2^{p_2}$, $q=2^{p_1}$, λ=security parameter, m= the degree of field extension, r=the desired key length, GS=Generalized Srivastava, MDPC=Moderate Density Parity Check, LDPC=Low Density Parity Check

noted by $U_{c,d}$.

2.1 MDS Codes

Definition 1. (**MDS Code (MacWilliams and Sloane, 1977)**). An $[n,k,d]$ linear code with length n, dimension k and minimum distance d is said to be a *maximum distance separable* (MDS) code if $k=n-d+1$.

Definition 2. (**MDS Matrix (Gupta and Ray, 2013)**). Let $\mathsf{GF}(q)$ be a finite field and m, n be two integers. Let $\mathbf{x}\rightarrow M\times\mathbf{x}$ be a mapping from $(\mathsf{GF}(q))^m$ to $(\mathsf{GF}(q))^n$ defined by the $n\times m$ matrix M. We say that M is an *MDS matrix* if the set of all pairs $(\mathbf{x},M\times\mathbf{x})$ is an MDS code, i.e. a linear code of dimension m, length $m+n$ and minimum distance $n+1$.

Theorem 1. **(MacWilliams and Sloane, 1977)** *An $[n,k,d]$ linear code with generator matrix $G=[I|M]\in(\mathsf{GF}(q))^{k\times n}$ is MDS code if and only if every square submatrix of M is nonsingular where M is a $k\times(n-k)$ matrix over $\mathsf{GF}(q)$. We say M is an MDS matrix if the corresponding code is MDS.*

Definition 3. (**Companion Matrix (Gupta et al., 2017a)**). Let $g(X)=z_0+z_1X+\cdots+z_{k-1}X^{k-1}+X^k$ be a monic polynomial over $\mathsf{GF}(q)$ of degree k. The $k\times k$ *companion matrix* C_g associated to the polynomial g is given by

$$C_g=\begin{bmatrix}0 & 1 & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & \cdots & 1\\ -z_0 & -z_1 & \cdots & -z_{k-1}\end{bmatrix}$$

Theorem 2. **(Blaum and Roth, 1999)** *Let $\mathsf{GF}(q)$ be the finite field containing q elements with characteristic 2, $\mathsf{Mat}(m,\mathsf{GF}(q))$ be the ring of $m\times m$ matrices over $\mathsf{GF}(q)$ and $\mathsf{Mat}(n,m)$ be the set of $n\times n$ block matrices over $\mathsf{Mat}(m,\mathsf{GF}(2))$. A matrix $M\in\mathsf{Mat}(n,\mathsf{GF}(q))$ is MDS if and only if every square submatrix of M is nonsingular. Similarly, a block matrix $M\in\mathsf{Mat}(n,m)$ is MDS if and only if every square block submatrix of M is nonsingular.*

The following result follows easily from the above theorem.

Lemma 1. *A block matrix $M\in\mathsf{Mat}(n,m)$ is MDS if and only if its transpose M^T is MDS.*

The order of a polynomial $g(X)\in\mathsf{GF}(q)[X]$ ($g(0)\neq 0$), denoted by $\mathsf{ord}(g)$, is the least positive integer n such that $g(X)$ divides X^n-1. The weight of a polynomial is the number of its coefficients that are nonzero.

Theorem 3. **(Gupta et al., 2017a)** *Let $g(X)\in\mathsf{GF}(q)[X]$ be a monic polynomial of degree k with $\mathsf{ord}(g)\geq 2k$. Then the matrix $M=(C_g)^k$ is MDS if and only if the weight of any nonzero multiple of degree $\leq 2k-1$ of the polynomial $g(X)$ is greater than k.*

Definition 4. (**Permutation Equivalent Matrices (Kesarwani et al., 2019)**) Two matrices M and M' are said to be *permutation equivalent*, denoted by $M\sim_{pe}M'$, if there exist two permutation matrices P,Q such that $M'=PMQ$.

Lemma 2. **(Kesarwani et al., 2019)** *Suppose that two matrices M and M' are permutation equivalent. Then M is MDS if and only if M' is MDS.*

Definition 5. (**Expanded Codes (Khathuria et al., 2019)**). Let n,k be positive integers with $k\leq n$, q be a prime power and m be an integer. Let C be a linear code of length n and dimension k over $\mathsf{GF}(q^m)$. The *expanded code* of C with respect to a primitive element $\gamma\in\mathsf{GF}(q^m)$ is a linear code over the base field $\mathsf{GF}(q)$ defined as $\widehat{\mathsf{C}}=\{\phi_n(c):c\in\mathsf{C}\}$ where $\phi_n:(\mathsf{GF}(q^m))^n\longrightarrow(\mathsf{GF}(q))^{mn}$ is the $\mathsf{GF}(q)$-linear isomorphism defined by γ as

$$\phi_n(\alpha_0,\alpha_1,\ldots,\alpha_{n-1})=(\phi(\alpha_0),\phi(\alpha_1),\ldots,\phi(\alpha_{n-1}))$$

and $\phi : \mathsf{GF}(q^m) \longrightarrow (\mathsf{GF}(q))^m$ is given by

$$\phi(a_0 + a_1\gamma + \cdots + a_{m-1}\gamma^{m-1}) = (a_0, a_1, \ldots, a_{m-1}).$$

Lemma 3. (Khathuria et al., 2019). *Let* C *be a linear code in* $(GF(q^m))^n$, $\gamma \in GF(q^m)$ *be a primitive element and* $\phi_n : (GF(q^m))^n \longrightarrow (GF(q))^{mn}$ *be the* $GF(q)$*-linear isomorphism defined by* γ *as in Definition 5.*

(i) If $G = [g_1, g_2, \ldots, g_k]^T$ *is a generator matrix of* C *where* $g_1, g_2, \ldots, g_k$ *are vectors in* $(GF(q^m))^n$*, then the expanded code* $\widehat{\mathsf{C}}$ *of* C *over* $GF(q)$ *with respect to the primitive element* $\gamma \in GF(q^m)$ *has the expanded generator matrix*
$\widehat{G} = [\phi_n(g_1), \phi_n(\gamma g_1), \ldots, \phi_n(\gamma^{n-1} g_1), \phi_n(g_2), \phi_n(\gamma g_2), \ldots, \phi_n(\gamma^{n-1} g_2), \ldots, \phi_n(g_k), \phi_n(\gamma g_k), \ldots, \phi_n(\gamma^{n-1} g_k)]^T$.

(ii) If $H = [h_1^T, h_2^T, \ldots, h_n^T]$ *is a parity check matrix of* C *where* $h_1, h_2, \ldots, h_n$ *are vectors in* $(GF(q^m))^{n-k}$*, then the expanded code* $\widehat{\mathsf{C}}$ *of* C *over* $GF(q)$ *with respect to the primitive element* $\gamma \in GF(q^m)$ *has the expanded parity check matrix*
$\widehat{H} = [\phi_{n-k}(h_1)^T, \phi_{n-k}(\gamma h_1)^T, \ldots, \phi_{n-k}(\gamma^{n-1} h_1)^T, \phi_{n-k}(h_2)^T, \phi_{n-k}(\gamma h_2)^T, \ldots, \phi_{n-k}(\gamma^{n-1} h_2)^T, \ldots, \phi_{n-k}(h_n)^T, \phi_{n-k}(\gamma h_n)^T, \ldots, \phi_{n-k}(\gamma^{n-1} h_n)^T]$.

(iii) $\phi_n(xG) = \phi_k(x)\widehat{G}$ *for all* $x \in (GF(q^m))^k$,

(iv) $\phi_{n-k}(Hy^T) = \widehat{H}(\phi_n(y))^T$ *for all* $y \in (GF(q^m))^n$.

2.2 Hardness Assumptions

Definition 6. ((Search) (q-ary) Syndrome Decoding (SD) Problem (Barg, 1997)). Given a full-rank matrix $H_{(n-k)\times n}$ over $\mathsf{GF}(q)$, a vector $\mathbf{c} \in (\mathsf{GF}(q))^{n-k}$ and a non-negative integer w, the search version of q-ary SD problem is to find a vector $\mathbf{e} \in (\mathsf{GF}(q))^n$ of weight w such that the syndrome $H\mathbf{e}^T$ of $\mathbf{e}$ satisfies $H\mathbf{e}^T = \mathbf{c}$.

Assumption 1. The public key matrix, output by the key generation algorithm of a code-based PKE scheme, is computationally indistinguishable from a uniformly chosen matrix of the same size.

3 basicPKE: AN IND-CPA SECURE PUBLIC KEY ENCRYPTION

We now present the details of our public key encryption scheme basicPKE = (Setup, KeyGen, Enc, Dec).

- basicPKE.Setup(λ) $\longrightarrow$ $\mathsf{pp}_{\mathsf{basicPKE}}$: Taking security parameter λ as input, a trusted authority proceeds as follows to generate the global public parameters $\mathsf{pp}_{\mathsf{basicPKE}}$.

(i) Sample $k\ (\geq 2), m \in \mathbb{Z}^+$, set $q = 2^m$. Let $\gamma \in \mathsf{GF}(q)$ be a primitive element of $\mathsf{GF}(q)$.

(ii) Set $w \leq k/2$ and sample $k' \in \mathbb{Z}^+$.

(iii) Choose a cryptographic hash function $\mathcal{H}_1 : \{0,1\}^* \longrightarrow \{0,1\}^{k'}$.

(iv) Publish the global parameters $\mathsf{pp}_{\mathsf{basicPKE}} = (k, k', w, q, m, \gamma, \mathcal{H}_1)$.

- basicPKE.KeyGen($\mathsf{pp}_{\mathsf{basicPKE}}$) $\longrightarrow$ (pk, sk): A user on input $\mathsf{pp}_{\mathsf{basicPKE}}$, performs the following steps to generate public key pk and secret key sk.

(i) Select $z_0, z_1, \ldots, z_{k-1} \in \mathsf{GF}(q)$ where $q = 2^m$. Let $g(X) \in \mathsf{GF}(q)[X]$ be a monic polynomial of degree $k \geq 2$ given by $g(X) = z_0 + z_1X + z_2X^2 + \cdots + z_{k-1}X^{k-1} + X^k$ with $\mathsf{ord}(g) \geq 2k$ such that $g(X)$ has no nonzero multiple of degree $\leq 2k-1$ with weight $\leq k$. Such polynomials can be constructed using the approaches proposed by Gupta et al. ((Gupta et al., 2017b), (Gupta et al., 2019)).

(ii) The companion matrix associated with the polynomial $g(X)$ is

$$C_g = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ z_0 & z_1 & z_2 & \cdots & z_{k-1} \end{bmatrix} \in (\mathsf{GF}(q))^{k\times k}$$

(iii) Compute $\widetilde{M} = (C_g)^k$ which is MDS by Theorem 3 as $g(X)$ satisfies the conditions stated in this theorem. Therefore, every square submatrix of $\widetilde{M}$ is non-singular by Theorem 2. Hence by Theorem 1, the matrix $G = [I|\widetilde{M}] \in (\mathsf{GF}(q))^{k\times n}$ is a generator matrix of an MDS code C having code length $n = 2k$, dimension k and minimum distance $k+1$. Then the parity check matrix of the code C is $H = [\widetilde{M}^T|I_{n-k}] \in (\mathsf{GF}(q))^{(n-k)\times n}$.

(iv) Let $\widehat{H}$ be the expanded parity check matrix of the expanded code $\widehat{\mathsf{C}}$ of C with respect to the primitive element γ of $\mathsf{GF}(q)$ where $q = 2^m$ and the isomorphism ϕ_n as in Definition 5. Here $\widehat{H}$ is an $(n-k)m \times nm$ matrix over $\mathsf{GF}(2)$ by Lemma 3 (ii).

(v) Write $\widehat{H} \in (\mathsf{GF}(2))^{(n-k)m\times nm}$ in systematic form $[\widehat{M}|I_{(n-k)m}]$ where $\widehat{M}$ is an $(n-k)m \times km$ matrix and $n-k = k$.

(vi) Publish the public key $\mathsf{pk} = \widehat{M}$ and keep the secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$ secret to itself.

- basicPKE.Enc($\mathsf{pp}_{\mathsf{basicPKE}}$, pk, $\mathbf{m}$; $\mathbf{r}$) $\longrightarrow$ $\mathbf{c}$: Given system parameters $\mathsf{pp}_{\mathsf{basicPKE}}$, public key $\mathsf{pk} = \widehat{M}$ and a message $\mathbf{m} \in (\mathsf{GF}(2))^{k'}$, an encryptor proceeds as follows to generate a ciphertext $\mathbf{c} \in (\mathsf{GF}(2))^{km+k'}$.

Algorithm 1: Function $\mathcal{G}$: Error vector derivation.

Input: A binary seed vector σ of any length, integers nm, t.
Output: A binary error vector $\mathbf{e} = (e_0, e_1, \ldots, e_{nm-1})$ of length nm, weight t.

1: Set $\mathbf{e} \longleftarrow 1^t || 0^{nm-t}$; $\mathbf{b} \leftarrow \sigma$;
2: **for** $(i = 0$ **to** $t-1)$ **do**
3: $\quad j \leftarrow \mathcal{F}(\mathbf{b}) \bmod (nm - i - 1)$; // see Remark 1
4: $\quad$ Swap entries e_i and e_{i+j} in $\mathbf{e}$; $\mathbf{b} \leftarrow \mathsf{Hsh}(\mathbf{b})$; // Hsh is a hash function
5: **end for**
6: **return** $\mathbf{e} = (e_0, e_1, \ldots, e_{nm-1})$

(i) Select a random vector σ.

(ii) Run Algorithm 1 to generate a unique binary error vector $\mathbf{e}$ of length nm and weight w using σ as a seed, i.e, $\mathbf{e} = \mathcal{G}(\sigma)$. Here, $\mathbf{r} = \mathbf{e}$ is the randomness used in the procedure.

(iii) Using the public key $\widehat{M}$, construct the parity check matrix $\widehat{H} = (\widehat{M}|I_{(n-k)m})$ for the MDS code where $n - k = k$.

(iv) Compute the syndrome $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{m} \oplus \mathcal{H}_1(\mathbf{e}), \widehat{H}\mathbf{e}^T) \in (\mathsf{GF}(2))^{(n-k)m+k'}$.

(v) Publish the ciphertext $\mathbf{c}$.

- $\mathsf{basicPKE.Dec}(\mathsf{pp}_{\mathsf{basicPKE}}, \mathsf{sk}, \mathbf{c}) \longrightarrow \mathbf{m}'$: On receiving a ciphertext $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2)$, a decryptor executes the following steps using public parameters $\mathsf{pp}_{\mathsf{basicPKE}}$ and its secret key $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$.

(i) First proceed as follows to decode $\mathbf{c}_2$ and find binary error vector $\mathbf{e}'$ of length nm and weight w :

(a) Use $\mathsf{sk} = (z_0, z_1, \ldots, z_{k-1})$ to form $k \times k$ companion matrix

$$C_g = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ z_0 & z_1 & z_2 & \cdots & z_{k-1} \end{bmatrix}$$

associated with the monic polynomial $g(X) = z_0 + z_1 X + z_2 X^2 + \cdots + z_{k-1} X^{k-1} + X^k \in \mathsf{GF}(q)[X]$ of degree $k \geq 2$ and $\mathsf{ord}(g) \geq 2k$ that has no nonzero multiple of degree $\leq 2k - 1$ with weight $\leq k$. Then compute $\widetilde{M} = (C_g)^k$ and the parity check matrix $H = [\widetilde{M}^T | I_{n-k}] \in (\mathsf{GF}(q))^{(n-k)\times n}$ for $n = 2k$.

(b) Compute $\mathbf{c}_2' = \phi_{n-k}^{-1}(\mathbf{c}_2)$ where $\mathbf{c}_2$ is a column vector of length $(n-k)m$ over $\mathsf{GF}(2)$, $\mathbf{c}_2'$ is a column vector of length $(n-k)$ over $\mathsf{GF}(q)$ and ϕ_{n-k} is the $\mathsf{GF}(2)$-linear isomorphism defined by γ (Definition 5). The $(n-k) \times n$ parity check matrix H is used to decode $\mathbf{c}_2$ by first computing the syndrome $S = H(\mathbf{c}_2'||\mathbf{0})^T$ where $\mathbf{0}$ is a vector consisting of k zeros and then by running the decoding algorithm for MDS codes to find the vector $\widetilde{\mathbf{e}} \in (\mathsf{GF}(q))^n$.

(c) Apply ϕ_n to get $\mathbf{e}' = \phi_n(\widetilde{\mathbf{e}}) \in (\mathsf{GF}(2))^{nm}$.

(ii) Compute $\mathbf{m}' = \mathbf{c}_1 \oplus \mathcal{H}_1(\mathbf{e}')$.

(iii) Return $\mathbf{m}'$.

Remark 1. The function $\mathcal{G}$ in Algorithm 1 uses a hash function Hsh in line 4 and in $\mathcal{F}$ in line 3. Note that, $\mathcal{G}$ returns a binary vector of length nm and weight t on input an arbitrary binary string and integers nm, t. For simplicity, we use the notation $\mathcal{G}(\sigma)$ to denote the output of the error vector generation algorithm instead of $\mathcal{G}(\sigma, nm, t)$. The subroutine $\mathcal{F}(b) \bmod (nm - i - 1) \longrightarrow j$ outputs an integer j on input a binary vector $\mathbf{b}$ of length k as follows.
Step 1. Truncate $\mathsf{Hsh}(\mathbf{b})$ to a string of s bytes where s is larger than the byte size of nm.
Step 2. Convert this s–bytes string to an integer A.
(a) If $A > 2^{8s} - (2^{8s} \bmod (nm - i - 1))$ then go to Step 1.
(b) else set $j = A \mod (nm - i - 1)$.

Correctness. While decoding $\mathbf{c}_2$, we form an $(n-k) \times n$ parity check matrix H over $\mathsf{GF}(q)$ using the secret key sk and find the syndrome $H(\mathbf{c}_2'||\mathbf{0})^T$ to estimate the error vector $\mathbf{e}' \in (\mathsf{GF}(2))^{nm}$ with $\mathsf{wt}(\mathbf{e}') = w$. Note that, the ciphertext component $\mathbf{c}_2 = \widehat{H}(\mathbf{e})^T$ is the syndrome of $\mathbf{e}$ where the matrix $\widehat{H}$ is a parity check matrix in the systematic form over $\mathsf{GF}(2)$ which is indistinguishable from a random matrix over $\mathsf{GF}(2)$. At the time of decoding $\mathbf{c}_2$, we need a parity check matrix over $\mathsf{GF}(q)$. The matrix $\widehat{H}$, a parity check matrix of MDS code in the systematic form derived from the public key pk, does not help to decode $\mathbf{c}_2$ as the SD problem is hard over $\mathsf{GF}(2)$. The decoding algorithm in our decryption procedure uses the parity check matrix H derived from the secret key sk. This procedure can correct upto $k/2$ errors. In our scheme, the error vector $\mathbf{e}$ used in the procedure $\mathsf{basicPKE.Enc}$ satisfies $\mathsf{wt}(\mathbf{e}) = w \leq k/2$. Consequently, the decoding procedure will recover the correct $\mathbf{e}$ by Lemma 3 (iv).

The method for achieving an indistinguishable public key matrix by mixing secret key matrix by two matrices is rather old and shows vulnerability to the scheme according to Strenzke et al.(Strenzke, 2010). The most reliable method suggested by Biswas and Sendrier (Biswas and Sendrier, 2008) to obtain the public key matrix is to compute the systematic form of the secret key matrix. In our scheme, we start with choosing $z_i, i = 0, 1, \ldots, n-1$ randomly to construct a companion matrix and then proceed to obtain a parity check matrix of an MDS code. After that, we find the expanded parity check matrix over base field $\mathsf{GF}(2)$ and write it in the systematic form to obtain public key.

Theorem 4. *If* SD *problem (Definition 6 in Subsection 2.2) is hard and the public key matrix* $\widehat{H}$ *(derived from the public key* pk *which is generated by running* basicPKE.KeyGen($\mathsf{pp}_{\mathsf{basicPKE}}$) *where* $\mathsf{pp}_{\mathsf{basicPKE}} \longleftarrow$ basicPKE.Setup(λ)) *is indistinguishable, then the public key encryption scheme* basicPKE = (Setup, KeyGen, Enc, Dec) *described above is* IND-CPA *secure in the random oracle model.*

4 fullPKE: AN OW-PCVA SECURE PUBLIC KEY ENCRYPTION

We now discuss a public key encryption fullPKE = (Setup, KeyGen, Enc, Dec) that is constructed from the framework of basicPKE.

- fullPKE.Setup(λ) $\longrightarrow \mathsf{pp}_{\mathsf{fullPKE}}$: A trusted authority runs basicPKE.Setup(λ) and sets global parameters $\mathsf{pp}_{\mathsf{fullPKE}} = (k, k', w, q, m, \gamma, \mathcal{H}_1)$ taking security parameter λ as input.
- fullPKE.KeyGen($\mathsf{pp}_{\mathsf{fullPKE}}$) $\longrightarrow$ (pk, sk) : A user generates public-secret key pair (pk, sk) by running basicPKE.KeyGen($\mathsf{pp}_{\mathsf{fullPKE}}$) where pk $= \widehat{M}$ and sk $= (z_0, z_1, \ldots, z_{k-1})$.
- fullPKE.Enc($\mathsf{pp}_{\mathsf{fullPKE}}$, pk, $\mathbf{m}; \mathbf{r}$) $\longrightarrow$ CT : An encryptor encrypts a message $\mathbf{m} \in \mathcal{M} = (\mathsf{GF}(2))^{k'}$ using public parameters $\mathsf{pp}_{\mathsf{fullPKE}}$ and its public key pk as input and produces a ciphertext CT as follows.

(i) Run Algorithm 1 using $\mathbf{m}$ as a seed to obtain an error vector $\mathbf{e}$ of length nm and weight w i.e. $\mathbf{e} = \mathcal{G}(\mathbf{m})$. Let $\mathbf{r} = \mathbf{e} = \mathcal{G}(\mathbf{m})$. Compute $\mathbf{d} = \mathcal{H}_1(\mathbf{m}) \in (\mathsf{GF}(2))^{k'}$.

(ii) Use the public key pk $= \widehat{M}$ to construct the matrix $\widehat{H} = (\widehat{M} | I_{(n-k)m})$.

(iii) Compute $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{m} \oplus \mathcal{H}_1(\mathbf{e}), \widehat{H}\mathbf{e}^T) \in (\mathsf{GF}(2))^{(n-k)m+k'}$. Here, $n = 2k$.

(iv) Return the ciphertext CT $= (\mathbf{c}, \mathbf{d}) \in \mathcal{C} = (\mathsf{GF}(2))^{km+2k'}$.

- fullPKE.Dec($\mathsf{pp}_{\mathsf{fullPKE}}$, sk, CT) $\longrightarrow \mathbf{m}'$: On receiving the ciphertext CT, the decryptor executes the following steps using public parameters $\mathsf{pp}_{\mathsf{fullPKE}}$ and its secret key sk $= (z_0, z_1, \ldots, z_{k-1})$.

(i) Use the secret key sk $= (z_0, z_1, \ldots, z_{k-1})$ to form a parity check matrix H and then find error vector $\mathbf{e}'$ of weight w and length nm as in the procedure basicPKE.Dec.

(ii) Compute $\mathbf{m}' = \mathbf{c}_1 \oplus \mathcal{H}_1(\mathbf{e}') \in (\mathsf{GF}(2))^{k'}$, $\mathbf{d}' = \mathcal{H}_1(\mathbf{m}') \in (\mathsf{GF}(2))^{k'}$ and $\mathbf{e}'' = \mathcal{G}(\mathbf{m}') \in (\mathsf{GF}(2))^{nm}$.

(iii) If $(\mathbf{e}' \neq \mathbf{e}'') \vee (\mathbf{d} \neq \mathbf{d}')$, output $\perp$ that indicates decryption failure. Otherwise, return $\mathbf{m}'$.

Correctness. The decoding algorithm in fullPKE.Dec uses the parity check matrix H (derived from the secret key sk) and can correct upto $k/2$ errors. In our scheme, the error vector $\mathbf{e}$ used in the procedure fullPKE.Enc satisfies wt($\mathbf{e}$) $= w \leq k/2$. Consequently, the decoding procedure will recover the correct $\mathbf{e}$ as Lemma 3 (iv) holds. We regenerate $\mathbf{e}''$ and compare it with $\mathbf{e}'$ obtained after decoding. Since the error vector generation algorithm $\mathcal{G}$ uses a deterministic function to obtain a fixed low weight error vector, $\mathbf{e}' = \mathbf{e}''$ occurs.

Theorem 5. *If the public key encryption scheme* basicPKE =(Setup, KeyGen, Enc, Dec) *as described in Section 3 is* IND-CPA *secure, then the scheme* fullPKE = (Setup, KeyGen, Enc, Dec) *as described above achieves* OW-PCVA *security considering the hash function* $\mathcal{G}$ *as a random oracle.*

As the OW-PCVA security for an encryption scheme trivially implies the OW-VA security of the scheme considering zero queries to the PCO($\cdot,\cdot$) oracle, we can obtain the following corollary as an immediate consequence.

Corollary 1. *If the PKE scheme* basicPKE =(Setup, KeyGen, Enc, Dec) *as described in Section 3 is* IND-CPA *secure, then the scheme* fullPKE = (Setup, KeyGen, Enc, Dec) *as described in Section 4 is* OW-VA *secure considering the hash function* $\mathcal{G}$ *as a random oracle.*

5 fullKEM: AN IND-CCA SECURE KEY ENCAPSULATION MECHANISM

We now present the details of our key encapsulation mechanism fullKEM = (Setup, KeyGen, Encaps, Decaps).

- fullKEM.Setup(λ)$\longrightarrow \mathsf{pp}_{\mathsf{fullKEM}}$: A trusted authority runs fullPKE.Setup(λ), chooses another cryptographic hash function $\mathcal{H}_2 : \{0,1\}^* \longrightarrow \{0,1\}^r$ and sets public parameters $\mathsf{pp}_{\mathsf{fullKEM}} = (k, k', w, r, q, m, \gamma, \mathcal{H}_1, \mathcal{H}_2)$ taking security parameter λ as input.
- fullKEM.KeyGen($\mathsf{pp}_{\mathsf{fullKEM}}$) $\longrightarrow$ (pk, sk): A user generates public-secret key pair (pk, sk) by running fullPKE.KeyGen($\mathsf{pp}_{\mathsf{fullKEM}}$) where pk $= \widehat{M}$ and sk $= (z_0, z_1, \ldots, z_{k-1})$.

- fullKEM.Encaps($\mathsf{pp}_{\mathsf{fullKEM}}$, pk)$\longrightarrow$ (CT, K): Given system parameters $\mathsf{pp}_{\mathsf{fullKEM}}$ and public key pk $= \widehat{M}$, an encapsulator proceeds as follows to generate a ciphertext header CT $\in (\mathsf{GF}(2))^{km+2k'}$ and an encapsulation key $K \in \{0,1\}^r$.

(i) Sample $\mathbf{m} \xleftarrow{U} (\mathsf{GF}(2))^{k'}$.

(ii) Run Algorithm 1 using $\mathbf{m}$ as a seed to obtain an error vector $\mathbf{e}$ of length nm and weight w i.e. $\mathbf{e} = \mathcal{G}(\mathbf{m})$. Compute $\mathbf{d} = \mathcal{H}_1(\mathbf{m}) \in (\mathsf{GF}(2))^{k'}$.

(iii) Use the public key pk $= \widehat{M}$ to construct the matrix $\widehat{H} = (\widehat{M}|I_{(n-k)m})$.

(iv) Compute $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{m} \oplus \mathcal{H}_1(\mathbf{e}), \widehat{H}\mathbf{e}^T) \in (\mathsf{GF}(2))^{(n-k)m+k'}$ where $n = 2k$.

(v) Set the ciphertext header CT $= (\mathbf{c}, \mathbf{d}) \in \mathcal{C} = (\mathsf{GF}(2))^{km+2k'}$ and the encapsulation key $K = \mathcal{H}_2(\mathbf{m}) \in \{0,1\}^r$.

(vi) Publish the ciphertext header CT $= (\mathbf{c}, \mathbf{d})$ and keep K as secret.

- fullKEM.Decaps($\mathsf{pp}_{\mathsf{fullKEM}}$, sk, CT)$\longrightarrow K$: On receiving a ciphertext header CT $= (\mathbf{c}, \mathbf{d})$, a decapsulator executes the following steps using public parameters $\mathsf{pp}_{\mathsf{fullKEM}} = (k, k', w, r, q, m, \gamma, \mathcal{H}_1, \mathcal{H}_2)$ and its secret key sk $= (z_0, z_1, \ldots, z_{k-1})$.

(i) Using the secret key sk, form a parity check matrix H and then proceed to find error vector $\mathbf{e}'$ of weight w and length nm as in the procedure fullPKE.Dec (i.e as in basicPKE.Dec).

(ii) Compute $\mathbf{m}' = \mathbf{c}_1 \oplus \mathcal{H}_1(\mathbf{e}') \in (\mathsf{GF}(2))^{k'}$, $\mathbf{d}' = \mathcal{H}_1(\mathbf{m}') \in (\mathsf{GF}(2))^{k'}$ and $\mathbf{e}'' = \mathcal{G}(\mathbf{m}') \in (\mathsf{GF}(2))^{nm}$.

(iii) If $(\mathbf{e}' \neq \mathbf{e}'') \vee (\mathbf{d} \neq \mathbf{d}')$, output $\perp$ indicating decapsulation failure. Otherwise, compute the encapsulation key $K = \mathcal{H}_2(\mathbf{m}')$.

Correctness. Correctness of fullKEM follows from that of fullPKE.

Theorem 6. *If the scheme* fullPKE $=$ (Setup, KeyGen, Enc, Dec) *as described in Section 4 is OW-VA secure, then the scheme* fullKEM $=$ (Setup, KeyGen, Encaps, Decaps) *as described above provides* IND-CCA *security modelling hash function $\mathcal{H}_2$ as a random oracle.*

Theorem 7. *Depending on the hardness of* SD *problem (Definition 6 in Subsection 2.2) and indistinguishability of the public key matrix $\widehat{H}$ that is derived from the public key* pk *by running key generation algorithm of* fullKEM*, the scheme* fullKEM *described in Section 5 provides* IND-CCA *security considering $\mathcal{G}$ and $\mathcal{H}_2$ as random oracles.*

Proof. This is an immediate consequence of Theorem 4, Corollary 1 and Theorem 6. Due to limited space, proofs of Theorem 4, Theorem 5 and Theorem 6 will appear in the full version of the paper.

Remark 2. To prove security in quantum random oracle model, it is necessary to show post-quantum security of a scheme where the adversary can submit queries to the random oracle having quantum access. In this scenario, the adversary has quantum access to the random oracles besides having classical access to some other oracles like plaintext checking oracles, ciphertext validity checking oracles, decapsulation oracles, etc. The scheme basicPKE provides IND-CPA security in random oracle model as the SD problem is hard and the public key matrix is indistinguishable (see Theorem 4). Note that IND-CPA security always implies OW-CPA security. Following the work of (Hofheinz et al., 2017), we can show that OW-CPA security of the encryption scheme basicPKE indicates OW-PCA security of fullPKE considering $\mathcal{G}$ as a quantum random oracle. Then, we can prove that OW-PCA security of fullPKE implies the IND-CCA security of fullKEM modeling $\mathcal{H}_1, \mathcal{H}_2$ as quantum random oracles.

Thus we can get the following theorem.

Theorem 8. *Depending on the hardness of* SD *problem (Definition 6 in Subsection 2.2) and indistinguishability of the public key matrix $\widehat{H}$ derived from the public key* pk *by running key generation algorithm of* fullKEM*, our scheme* fullKEM *described in Section 5 achieves* IND-CCA *security considering the hash functions $\mathcal{G}, \mathcal{H}_1$ and $\mathcal{H}_2$ as quantum random oracles.*

6 CONCLUSION

In this work, we give a proposal to design a key encapsulation mechanism exploiting the structure of a companion matrix. We have shown that our KEM protocol provides IND-CCA security in the random oracle model and quantum random oracle model. However, the issue regarding the public key size needs to be explored in near future. In terms of secret key size and ciphertext size, our work seems promising. Therefore, we believe that our proposal to design a KEM will offer an attractive approach in the area of code-based cryptography.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Ayineedi Venkateswarlu for his valuable opinions that helped to improve the manuscript.

REFERENCES

Aguilar-Melchor, C., Blazy, O., Deneuville, J.-C., Gaborit, P., and Zémor, G. (2018). Efficient encryption from random quasi-cyclic codes. *IEEE Transactions on Information Theory*, 64(5):3927–3943.

Albrecht, M., Cid, C., Paterson, K. G., Tjhai, C. J., and Tomlinson, M. (2019). Nts-kem. *NIST submissions*.

Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C. A., et al. (2017a). Bike: Bit flipping key encapsulation. *NIST submissions*.

Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C. A., et al. (2019). Bike: Bit flipping key encapsulation. *NIST submissions*.

Aragon, N., Blazy, O., Deneuville, J.-C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.-P., and Zémor, G. (2017b). Lake-low rank parity check codes key exchange.

Aragon, N., Blazy, O., Deneuville, J.-C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.-P., and Zémor, G. (2017c). Locker-low rank parity check codes encryption.

Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., and Santini, P. (2018). Ledakem: a post-quantum key encapsulation mechanism based on qc-ldpc codes. In *International Conference on Post-Quantum Cryptography*, pages 3–24. Springer.

Banegas, G., Barreto, P. S., Boidje, B. O., Cayrel, P.-L., Dione, G. N., Gaj, K., Gueye, C. T., Haeussler, R., Klamti, J. B., N'diaye, O., et al. (2018). Dags: Key encapsulation using dyadic gs codes. *Journal of Mathematical Cryptology*, 12(4):221–239.

Bardet, M., Barelli, E., Blazy, O., Canto-Torres, R., Couvreur, A., Gaborit, P., Otmani, A., Sendrier, N., and Tillich, J.-P. (2017). Big quake. *NIST submissions*.

Barg, A. (1997). Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(46).

Barreto, P. S., Gueron, S., Güneysu, T., Misoczki, R., Persichetti, E., Sendrier, N., and Tillich, J.-P. (2017). Cake: Code-based algorithm for key encapsulation. In *IMA International Conference on Cryptography and Coding*, pages 207–226. Springer.

Bernstein, D. J., Chou, T., Lange, T., von Maurich, I., Misoczki, R., Niederhagen, R., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., et al. (2017). Classic mceliece: conservative code-based cryptography. *NIST submissions*.

Biswas, B. and Sendrier, N. (2008). Mceliece cryptosystem implementation: Theory and practice. In *International Workshop on Post-Quantum Cryptography*, pages 47–62. Springer.

Blaum, M. and Roth, R. M. (1999). On lowest density mds codes. *IEEE Transactions on Information Theory*, 45(1):46–59.

Dey, J. and Dutta, R. (2019). Secure key encapsulation mechanism with compact ciphertext and public key from generalized srivastava code. In *International Conference on Information Security and Cryptology*, pages 175–193. Springer.

Gupta, K. C., Pandey, S. K., and Venkateswarlu, A. (2017a). On the direct construction of recursive mds matrices. *Designs, Codes and Cryptography*, 82(1-2):77–94.

Gupta, K. C., Pandey, S. K., and Venkateswarlu, A. (2017b). Towards a general construction of recursive mds diffusion layers. *Designs, Codes and Cryptography*, 82(1-2):179–195.

Gupta, K. C., Pandey, S. K., and Venkateswarlu, A. (2019). Almost involutory recursive mds diffusion layers. *Designs, Codes and Cryptography*, 87(2-3):609–626.

Gupta, K. C. and Ray, I. G. (2013). On constructions of mds matrices from companion matrices for lightweight cryptography. In *International Conference on Availability, Reliability, and Security*, pages 29–43. Springer.

Hofheinz, D., Hövelmanns, K., and Kiltz, E. (2017). A modular analysis of the fujisaki-okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer.

Kesarwani, A., Sarkar, S., and Venkateswarlu, A. (2019). Exhaustive search for various types of mds matrices. *IACR Transactions on Symmetric Cryptology*, pages 231–256.

Khathuria, K., Rosenthal, J., and Weger, V. (2019). Encryption scheme based on expanded reed-solomon codes. *arXiv preprint arXiv:1906.00745*.

Kim, J.-L., Kim, Y.-S., Galvez, L., Kim, M. J., and Lee, N. (2018). Mcnie: A code-based public-key cryptosystem. *arXiv preprint arXiv:1812.05008*.

MacWilliams, F. J. and Sloane, N. J. A. (1977). *The theory of error-correcting codes*, volume 16. Elsevier.

McEliece, R. J. (1978). A public-key cryptosystem based on algebraic coding theory. *Coding Thv*, 4244:114–116.

Melchor, C. A., Aragon, N., Bardet, M., Bettaieb, S., Bidoux, L., Blazy, O., and Deneuville, J.-C. (2019). Rollo-rank-ouroboros, lake & locker.

Melchor, C. A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Hauteville, A., Zémor, G., and Bourges, I.-C. (2017). Ouroboros-r.

Nojima, R., Imai, H., Kobara, K., and Morozov, K. (2008). Semantic security for the mceliece cryptosystem without random oracles. *Designs, Codes and Cryptography*, 49(1-3):289–305.

Strenzke, F. (2010). A timing attack against the secret permutation in the mceliece pkc. In *International Workshop on Post-Quantum Cryptography*, pages 95–107. Springer.

Szepieniec, A. (2017). Ramstake. Technical report, Technical report, National Institute of Standards and Technology.

Wang, Y. (2017). Rlcekey encapsulation mechanism (rlce-kem) specifcation. *NIST Submission*.

Yamada, A., Eaton, E., Kalach, K., Lafrance, P., and Parent, A. (2017). Qc-mdpc kem: A key encapsulation mechanism based on the qc-mdpc mceliece encryption scheme. *NIST Submission*.